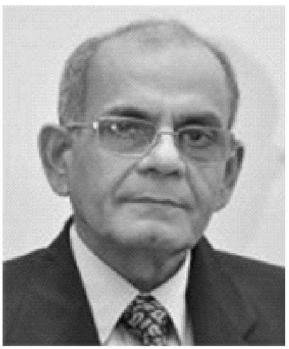
Digital Control and State Variable Methods

Conventional and Intelligent Control Systems

FOURTH EDITION

About the Author



M.Gopal, a Professor in Electrical Engineering at Indian Institute of Technology Delhi, is at present the Director of School of Engineering, Shiv Nadar University, Gautam Budh Nagar (U.P.) India. His teaching and research stints span three decades at the IITs.

Dr. Gopal is a globally known academician with excellent credentials as author, teacher, researcher, and administrator. He is the author/co-author of six books on *Control Engineering*. His books are used worldwide, and some of them have been translated into Chinese and Spanish as well. McGraw-Hill, Singapore has published his books for the Asia Pacific market and McGraw-Hill, USA for the US market. In India, his books have been serving as the major source of learning for more than three decades.

As a teacher, his potential is being used globally through a video course (http://www.youtube.com/iit), which is one of the most popular courses on You Tube by the IIT faculty across India. He has been conducting Faculty Development Programs on active learning and effective teaching.

A recognized researcher in the area of Machine Learning, Dr. Gopal has been a key-note speaker in many international conferences. He periodically runs executive programs as tutorial sessions in conferences, and short-term workshops, to empower the participants with the state-of-the-art techniques in pattern recognition and machine learning.

He is the author/co-author of over 135 research papers and his key contributions have been published in high impact factor journals. He has supervised 16 doctoral research projects. His current research interests are in the areas of Machine Learning, Soft-Computing Technologies, Pattern Recognition, and Intelligent Control.

In administrative capacity, he contributed to the growth of a large department at IIT Delhi with electrical (power), electronics, communications, computer technology, and information technology as the areas of activities. The department has acquired a respectable international standing in both teaching and research.

M.Gopal holds B.Tech (Electrical), M.Tech (Control Systems), and Ph.D. degrees form BITS, Pilani.

Digital Control and State Variable Methods

Conventional and Intelligent Control Systems

FOURTH EDITION

M GOPAL
Professor
Department of Electrical Engineering
Indian Institute of Technology Delhi
New Delhi

Tata McGraw Hill Education Private Limited
NEW DELHI

McGraw-Hill Offices

New Delhi New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto

Tata McGraw-Hill

Published by the Tata McGraw Hill Education Private Limited,
7 West Patel Nagar, New Delhi 110 008.

Digital Control and State Variable Methods: Conventional and Intelligent Control Systems, 4e

ISBN (13): 978-0-07-133327-6
ISBN (10): 0-07-133327-4

Vice President and Managing Director—McGraw-Hill Education: *Ajay Shukla*

Head—Higher Education Publishing and Marketing: *Vibha Mahajan*
Publishing Manager—SEM & Tech Ed.: *Shalini Jha*
Editorial Executive: *Koyel Ghosh*
Sr Copy Editor: *Nimisha Kapoor*
Sr Production Manager: *Satinder Singh Baveja*
Proof Reader: *Yukti Sharma*

Marketing Manager—Higher Education: *Vijay Sarathi*
Sr Product Specialist—SEM & Tech Ed.: *Tina Jajoriya*
Graphic Designer (Cover): *Meenu Raghav*

General Manager—Production: *Rajender P Ghansela*
Production Manager: *Reji Kumar*

Typeset at Tej Composers, WZ-391, Madipur, New Delhi 110063, and printed at

Dedicated
with all my love to my
son, Ashwani
and
daughter, Anshu

Contents

Preface

Control Engineering is an active field of research and hence there is a steady influx of new concepts, ideas and techniques. In time, some of these elements develop to the point where they join the list of things every control engineer must know. To grasp the significance of modern developments, a strong foundation is necessary in analysis, design and stability procedures applied to continuous-time linear and nonlinear feedback control systems. Simultaneously, knowledge of the corresponding methods in the digital version of control systems is also required because of the use of microprocessors, programmable logic devices and DSP chips as controllers in modern systems. This book aims at presenting the vital theories required for appreciating the past and present status of control engineering.

When compiling the material for the first edition of the book, decisions had to be made as to what should be included and what should not. It was decided to place the emphasis on the control of continuous-time and discrete-time linear systems, based on frequency-domain and state-space methods of design. In the subsequent editions, we continue to emphasize solid mastery of the underlying techniques for linear systems; in addition, the subject of nonlinear control has occupied an important place in our presentation. The availability of powerful low-cost microprocessors has spurred great interest in nonlinear control. Many practical nonlinear control systems based on conventional nonlinear control theory have been developed. The emerging trends are to employ intelligent control technology for nonlinear systems. As a result, the subject of nonlinear control (based on conventional as well as intelligent control methodologies) has become a necessary part of the fundamental background of control engineers.

The vast array of systems to which feedback control is applied and the growing variety of techniques available for the solution of control problems means that today's student of control engineering needs to manage a great deal of information. To help the students in this task and to keep their perspective as they plow through a variety of techniques, a user-friendly format has been devised for the book. We have divided the contents in three parts. **Part I** deals with digital control principles and design in transform domain, assuming that the reader has had an introductory course in control engineering concentrating on the basic principles of feedback control and covering the various classical analog methods of control system design. The material presented in this part of the book is closely related to the material a student may already be familiar with, but towards the end a direction to wider horizons is indicated. Basic principles of feedback control and classical analog methods of design have been elaborately covered in another book: M Gopal, *Control Systems: Principles and Design*, 4th edition, Tata McGraw-Hill, 2012.

Part II of the book deals with state variable methods in automatic control. State variable analysis and design methods are usually not covered in an introductory course. It is assumed that the reader is not exposed to the so-called modern control theory. Our approach is to first discuss the state variable methods

for continuous-time systems and then give a compact presentation for discrete-time systems using the analogy with the continuous-time systems. This formatting is a little different from the conventional one. Typically, a book on digital control systems starts with transform-domain design and then carries over to state space. These books give a detailed account of state variable methods for discrete-time systems. Since the state variable methods for discrete-time systems run quite parallel to those for continuous-time systems, a full-blown repetition is not appreciated by readers conversant with state variable methods for continuous-time systems. And for readers with no background of this type, a natural way of introducing state variable methods is to give the treatment for continuous-time systems, followed by a brief parallel presentation for discrete-time systems. This sequence of presentation is natural because it evolves from the sequence of steps in a design procedure. The systems to be controlled (plants) are continuous-time systems; we, therefore, investigate the properties of these systems using continuous-time models. Sampling is introduced only to insert a microprocessor in the feedback loop.

Part III of the book deals with nonlinear control schemes. The choice and emphasis of the schemes is guided by the basic objective of making an engineer or a student gain insights into the current nonlinear techniques in use for the solution of practical control problems in the industry. Some results of mostly theoretical interest are not included. Instead, emerging trends in nonlinear control are introduced. The *conventional* nonlinear control structures like Feedback Linearization, Model-Reference Adaptive Control, Self-Tuning Control, Generalized Model Predictive Control, Sliding Mode Control, etc., fall well short of the requirements of modern complex systems. While extensions and modifications to these conventional methods of control design based on mathematical models continue to be made, intelligent control technology is emerging as an alternative to solve complex control problems. This technology is slowly gaining wider acceptance in both academics and industry. The scientific community and industry are converging to the fact that there is something fundamentally significant about this technology. Rigorous characterization of theoretical properties of intelligent control methodology is not our aim; rather we focus on the development of systematic design procedures, which will guide the design of a controller for a specific problem.

The fundamental aim in preparing the book has been to work from basic principles and to present control theory in a way that can be easily understood and applied. Solved examples are provided as and when a new concept is introduced. The section on review examples briefly reiterates the key concepts of the chapter. A supplement of problems, with final answers, is also made available for pen-and-paper practice. MATLAB/Simulink tools are introduced in appendices to train the students in computer-aided-design. All the solved examples, review examples and problems can be done using software tools. Some problems specifically designed with a focus on MATLAB solutions, are given in appendices. A rich collection of references, classified to topics, has been given for more enthusiastic readers.

ORGANIZATION OF THE BOOK

The contents of the book are organized into fourteen chapters and two appendices. The appendices are given in the web supplements of the book. Appendix A deals with MATLAB/Simulink support for Conventional Control (Chapters 1−10) and Appendix B deals with MATLAB/Simulink support for Intelligent Control (Chapters 11−14).

The fourteen chapters of the book are classified into three parts, with each part serving a clearly defined objective. **Part I** (Chapters 1−4) deals with digital control principles and classical digital methods of design, paralleling and extending considerably the similar topics in analog control.

In **Chapter 1**, introduction to the digital control problem is given. A rich variety of practical problems are placed as examples. A rapid review of the classical procedures used for analog control is also provided.

For the study of classical procedures for digital control, the required mathematical background includes *z*-transforms. A review of *z*-transformation is presented in **Chapter 2**. With this background, the concepts of transfer function models and frequency-response models are introduced; and then dynamic response, steady-state response and stability issues are covered. After taking the student gradually through mathematical domain of digital control systems, Chapter 2 introduces the sampling theorem and the phenomenon of aliasing. Methods to generate discrete-time models which approximate continuous-time dynamics are also introduced in this chapter.

Chapter 3 briefly describes the digital control hardware including microprocessors, shaft-angle encoders, stepping motors, programmable logic controllers, etc. Transform-domain models of digital control loops are developed, with examples of some of the widely used digital control systems. Digital PID controllers, their implementation and tuning are also included in this chapter.

Chapter 4 establishes a toolkit of design-oriented techniques. It puts forward alternative design methods based on root locus and Bode plots. Design of digital controllers using *z*-plane synthesis is also included in this chapter.

Part II (Chapters 5−8) of the book deals with state variable methods in automatic control. The manner of presentation followed here is to first discuss state variable methods for continuous-time systems and then give a compact presentation of the methods for discrete-time systems, using the analogy with the continuous-time case.

Chapter 5 is on state variable analysis. It exposes the problems of state variable representation, diagonalization, solution, controllability and observability. The relationship between transfer function and state variable models is also given. Although it is assumed that the reader has the necessary background on vector-matrix analysis, a reasonably detailed account of vector-matrix analysis is provided in this chapter for convenient reference.

State variable analysis concepts, developed in continuous-time format in Chapter 5, are extended to digital control systems in **Chapter 6**.

The techniques of achieving desired system characteristics by pole-placement using complete state variable feedback are developed in **Chapter 7**. Also included is the method of using the system output to form estimates of the states for use in state feedback. Results are given for both continuous-time and discrete-time systems.

Lyapunov stability analysis is introduced in **Chapter 8**. In addition to stability analysis, Lyapunov functions are useful in solving some optimization problems. We discuss in this chapter, the solution of linear quadratic optimal control problem through Lyapunov synthesis. Results are given for both continuous-time and discrete-time systems.

Parts I and **II** exclusivelydeal with linear systems. In **Part III** (Chapters 9−14), the focus is on nonlinear systems. We begin with conventional methods of analysis and design (Chapters 9−10) which are of current importance in terms of industrial practice. Results of mostly theoretical interest are not included. Instead, the emerging trends in nonlinear control based on intelligent control technology are presented in reasonable details (Chapters 11−14).

In **Chapter 9**, considerable attention is paid to describing function and phase plane methods, which have demonstrated great utility in analysis of nonlinear systems. Also included is stability analysis of nonlinear systems using Lyapunov functions.

Chapter 10 introduces the concepts of feedback linearization, model reference adaptive control, system identification and self-tuning control and variable structure control. In terms of theory, major strides have been made in these areas. In terms of applications, many practical nonlinear control systems have been developed.

Neural networks are widely used in intelligent control systems. An informative description of neural networks is presented in **Chapter 11**. This chapter contains architectures and algorithms associated with multi-layer perceptron networks, radial basis function networks and support vector machines. Application examples from the perspectives of system identification and control are given.

Chapter 12 introduces the concepts of fuzzy sets and knowledge representation using fuzzy rules-based learning. Conceptual paradigms of fuzzy controllers are presented, with a discussion on Mamdani architecture for design. The approach to system identification as linguistic rules using the popular Takagi-Sugeno fuzzy representation is discussed. A brief description of system identification and control using neuro-fuzzy systems is also included in this chapter.

The focus in **Chapter 13** is on genetic algorithm for optimization. The applications of this algorithm to the learning of neural networks, as well as to the structural and parameter adaptations of fuzzy systems are also described.

Chapter 14 presents a new control architecture that is based on reinforcement learning. Several recent developments in reinforcement learning have substantially increased its viability as a general approach to intelligent control.

WEB SUPPLEMENTS

The book includes a wealth of supplements available in the dedicated website:

http://www.mhhe.com/gopal/dc4e

It includes:

For Students

- The source codes of the MATLAB problems given in Appendix A and Appendix B of the book.

For Instructors

- The Solution Manual for the exercise problems given at the end of all the chapters in the book. This part of the website is password protected and will be available to the instructors who adopt this text. This request can be sent to a local TMH sales representative.
- Also included are downloadable files of figures from the book to be used in presentation materials.

READERSHIP

The book is intended to be a comprehensive treatment of advanced control engineering for courses at senior undergraduate level and postgraduate (Masters degrees) level. It is also intended to be a reference source for PhD research students and practicing engineers.

For the purpose of organizing different courses for students with different backgrounds, the sequencing of chapters and dependence of each chapter on previous chapters has been properly designed in the text. A typical engineering curriculum at the second-degree level includes core courses on 'digital control systems' and 'linear system theory'. Parts I and II of the book have been designed to fully meet the requirements of the two courses. In Part III of the book, a reasonably detailed account of nonlinear control schemes, both the conventional and the intelligent, is given. The requirements of elective courses on 'nonlinear control systems' and 'intelligent control', will be partially or fully (depending on the depth of coverage of the courses) served by Part III of the book.

A typical engineering curriculum at the first-degree level includes a core course on feedback control systems, with one or two elective courses on the subject. This book meets the requirements of elective courses at the first-degree level.

ACKNOWLEDGEMENTS

I would like to acknowledge the contributions of faculty, students and practicing engineers across the country, whose suggestions through previous editions have made a positive impact on this new edition. In particular, the considerable help and education I have received from my students and colleagues at Indian Institute of Technology, Delhi, deserves sincere appreciation.

I also express my appreciation to the reviewers who offered valuable suggestions for this fourth and previous editions. The reviews have had a great impact on the project.

Finally, I would like to thank The Tata McGraw-Hill Publishing Company, and its executives for providing professional support for this project through all phases of its development.

Generous participation of instructors, students, and practicing engineers to eliminate errors in the text (if any), and to refine the presentation will be gratefully acknowledged.

M.Gopal

mgopal@ee.iitd.ac.in

PUBLISHER'S NOTE

Remember to write to us. We look forward to receiving your feedback, comments and ideas to enhance the quality of this book. You can reach us at *tmh.elefeedback@gmail.com*. Please mention the title and author's name as the subject.

In case you spot piracy of this book, please do let us know.

Part I

Digital Control: Principles and Design in Transform Domain

Automatic control systems play a vital role in the (technological) progress of human civilization. These control systems range from the very simple to the fairly complex in nature. Automatic washing machines, refrigerators, and ovens are examples of some of the simpler systems used in homes. Aircraft automatic pilots, robots used in manufacturing, and electric power generation and distribution systems represent complex control systems. Even such problems as inventory control, and socio-economic systems control, may be approached from the theory of feedback control.

Our world is one of *continuous-time* variables type. Quantities like flow, temperature, voltage, position, and velocity are not *discrete-time* variables but continuous-time ones. If we look back at the development of automatic control, we find that mass-produced analog (electronic) controllers have been available since about the 1940s. A first-level introduction to control engineering, provided in the companion book **'Control Systems: Principles and Design'**, deals with the basics of control, and covers sufficient material to enable us to design analog (op amp based) controllers for many simple control loops found in the industry.

From the 1980s onwards, we find microprocessor digital technology to be a dominant industrial phenomenon. Today, the most complex industrial processes are under computer control. A *microprocessor* determines the input to manipulate the physical system, or *plant*; and this requires facilities to apply this input to the physical world. In addition, the control strategy typically relies on measured values of the plant behavior; and this requires a mechanism to make these measured values available to the computing resources. The plant can be viewed as changing continuously with time. The controller, however, has a *discrete* clock that governs its behavior and so its values change only at discrete points in time. To obtain deterministic behavior and ensure data integrity, the sensor must include a mechanism to sample continuous data at discrete points in time, while the actuators need to produce a continuous value between the time points with discrete-time data.

Computer interfacing for data acquisition, consists of analog-to-digital (A/D) conversion of the input (to controller) analog signals. Prior to the conversion, the analog signal has to be conditioned to meet the input requirements of the A/D converter. Signal conditioning consists of amplification (for sensors generating very low power signals), filtering (to limit the amount of noise on the signal), and isolation (to protect the sensors from interacting with one another and/or to protect the signals from possibly damaging inputs). Conversion of a digital signal to an analog signal (D/A) at the output (of controller), is to be carried out to send this signal to an actuator which requires an analog signal. The signal has to be amplified by a transistor or solid state relay or power amplifier. Most manufacturers of electronic instrumentation devices are producing signal conditioners as modules.

The immersion of computing power into the physical world has changed the scene of control system design. A comprehensive theory of digital 'sampled' control has been developed. This theory requires a sophisticated use of new concepts such as *z-transform*. It is, however, quite straightforward to translate analog design concepts into digital equivalents. After taking a guided tour through the analog design concepts and op amp technology, the reader will find in Part I of this book sufficient material to enable him/her to design digital controllers for many simple control loops, and interfacing the controllers to other subsystems in the loop; thereby building complete feedback control systems.

The broad space of digital control applications can be roughly divided into two categories: *industrial control* and *embedded control*. Industrial control applications are those in which control is used as part of the process of creating or producing an end product. The control system is not a part of the actual end product itself. Examples include the manufacture of pharmaceuticals and the refining of oil. In the case of industrial control, the control system must be *robust* and *reliable*, since the processes typically run continuously for days, weeks or years.

Embedded control applications are those in which the control system is a component of the end product itself. For example, Electronic Control Units (ECUs) are found in a wide variety of products including automobiles, airplanes, and home applications. Most of these ECUs implement different feedback control tasks. For instance, engine control, traction control, anti-lock braking, active stability control, cruise control, and climate control. While embedded control systems must also be reliable, cost is a more significant factor, since the components of the control system contribute to the overall cost of manufacturing the product. In this case, much more time and effort is usually spent in the design phase of the control system to ensure reliable performance without requiring any unnecessary excess of processing power, memory, sensors, actuators, etc., in the digital control system. Our focus in this book will be on industrial control applications.

Perhaps more than any other factor, the development of microprocessors has been responsible for the explosive growth of the computer industry. While early microprocessors required many additional components in order to perform any useful task, the increasing use of *Large-Scale Integration* (LSI) or *Very Large-Scale Integration* (VLSI) semiconductor fabrication techniques has led to the production of *microcomputers*, where all of the required circuitry is embedded on one or a small number of integrated circuits. A further extension of the integration is the single-chip *microcontroller*, which adds analog and binary I/O, timers, and counters so as to be able to carry out real-time control functions with almost no additional hardware. Examples of such microcontrollers are Intel 8051, 8096 and Motorola MCH 68HC11. These chips were developed largely, in response to the automotive industries' desire for computer-controlled ignition, emission control and anti-skid systems. They are now widely used in process industries. This *digital control practice*, along with the theory of *sampled-data systems* is covered in Chapters 2–4 of the book.

Chapter 1

Introduction

1.1 CONTROL SYSTEM TERMINOLOGY

A *Control System* is an interconnection of components to provide a desired function. The portion of the system to be controlled is given various names: *process*, *plant*, and *controlled system* being perhaps the most common. The portion of the system that does the controlling is the *controller*. Often, a control system designer has little or no design freedom with the plant; it is usually fixed. The designer's task is, therefore, to develop a controller that will control the given plant acceptably. When measurements of the plant response are available to the controller (which, in turn, generates signals affecting the plant), the configuration is a *feedback control system*.

A *digital control system* uses digital hardware, usually in the form of a programmed digital computer, as the heart of the controller. In contrast, the controller in an *analog control system* is composed of analog hardware; an electronic controller made of resistors, capacitors, and operational amplifiers is a typical example. Digital controllers normally have analog devices at their periphery to interface with the plant; it is the internal working of the controller that distinguishes digital from analog control.

The signals used in the description of control systems are classified as continuous-time and discrete-time. *Continuous-time signals* are defined for all time, whereas *discrete-time signals* are defined only at discrete instants of time, usually evenly spaced steps. The signals for which both time and amplitude are discrete, are called *digital signals*. Because of the complexity of dealing with quantized (discrete-amplitude) signals, digital control system design proceeds as if computer-generated signals were not of discrete amplitude. If necessary, further analysis is then done, to determine if a proposed level of quantization is acceptable.

Systems and system components are termed *continuous-time* or *discrete-time* according to the type of signals they involve. They are classified as being *linear* if signal components in them can be superimposed—any linear combination of signal components, applied to a system, produces the same linear combination of corresponding output components; otherwise the system is *nonlinear*. A system or component is *time-invariant* if its properties do not change with time—any time shift of the inputs produces an equal time shift of every corresponding signal. If a system is not time-invariant, then it is *time-varying*.

A typical topology of a computer-controlled system is sketched schematically in Fig. 1.1. In most cases, the measuring transducer (sensor) and the actuator (final control element) are analog devices, requiring,

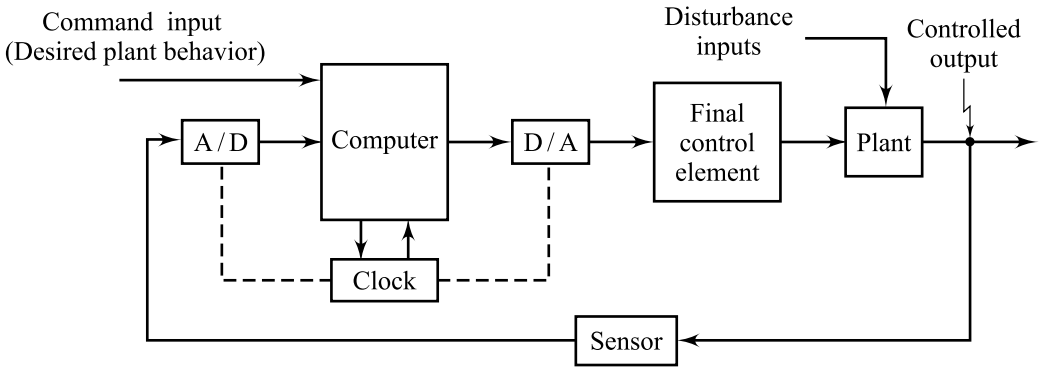


Fig. 1.1 Basic structure of a computer-controlled system

respectively, *analog-to-digital* (A/D) and *digital-to-analog* (D/A) conversion at the computer input and output. There are, of course, exceptions; sensors which combine the functions of the transducer and the A/D converter, and actuators which combine the functions of the D/A converter and the final control element are available. In most cases, however, our sensors will provide an analog voltage output, and our final control elements will accept an analog voltage input.

In the control scheme of Fig. 1.1, the A/D converter performs the sampling of the sensor signal (analog feedback signal) and produces its binary representation. The digital computer (*control algorithm*) generates a digital control signal using the information on desired and actual plant behavior. The digital control signal is then converted to analog control signal via the D/A converter. A real-time clock synchronizes the actions of the A/D and D/A converters, and the *shift registers*. The analog control signal is applied to the plant actuator to control the plant's behavior.

The overall system in Fig. 1.1 is *hybrid* in nature; the signals are in the *sampled* form (discrete-time signals) in the computer, and in a continuous form in the plant. Such systems have traditionally been called *sampled-data systems*; we will use this term as a synonym for *computer control systems*/*digital control systems*.

The word '*servomechanism*' (or *servo system*) is used for a *command-following system,* wherein the controlled output of the system is required to follow a given command. When the desired value of the controlled outputs is more or less fixed, and the main problem is to reject disturbance effects, the control system is sometimes called a *regulator*. The command input for a regulator becomes a constant and is called *set-point*, which corresponds to the desired value of the controlled output. The set-point may however be changed in time, from one constant value to another. In a *tracking system*, the controlled output is required to follow, or track, a time-varying command input.

To make these definitions more concrete, let us consider some familiar examples of control systems.

Example 1.1 *Servomechanism for Steering of Antenna*

One of the earliest applications of radar tracking was for anti-aircraft fire control; first with guns and later with missiles. Today, many civilian applications exist as well, such as satellite-tracking radars, navigation-aiding radars, etc.

The radar scene includes the radar itself, a target, and the transmitted waveform that travels to the target and back. Information about the target's spatial position is first obtained by measuring the changes in the back-scattered waveform relative to the transmitted waveform. The time shift provides information about the target's range, the frequency shift provides information about the target's radial velocity, and the received voltage magnitude and phase provide information about the target's angle[1][1].

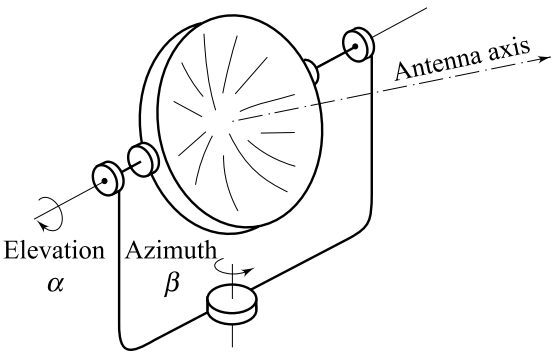


Fig. 1.2 Antenna configuration

In a typical radar application, it is necessary to point the radar antenna towards the target and follow its movements. The radar sensor detects the error between the antenna axis and the target, and directs the antenna to follow the target. The servomechanism for steering the antenna in response to commands from the radar sensor, is considered here. The antenna is designed for two independent angular motions; one about the vertical axis in which the azimuth angle is varied, and the other about the horizontal axis in which the elevation angle is varied (Fig. 1.2). The servomechanism for steering the antenna is described by two controlled variables—*azimuth angle* β and *elevation angle* α. The desired values or commands are the azimuth angle β_r and the elevation angle α_r of the target. The feedback control problem involves error self-nulling, under conditions of disturbances beyond our control (such as wind power).

The control system for steering antenna can be treated as two independent systems—the *azimuth-angle servomechanism*, and the *elevation-angle servomechanism*. This is because the interaction effects are usually small. The operational diagram of the azimuth-angle servomechanism is shown in Fig. 1.3.

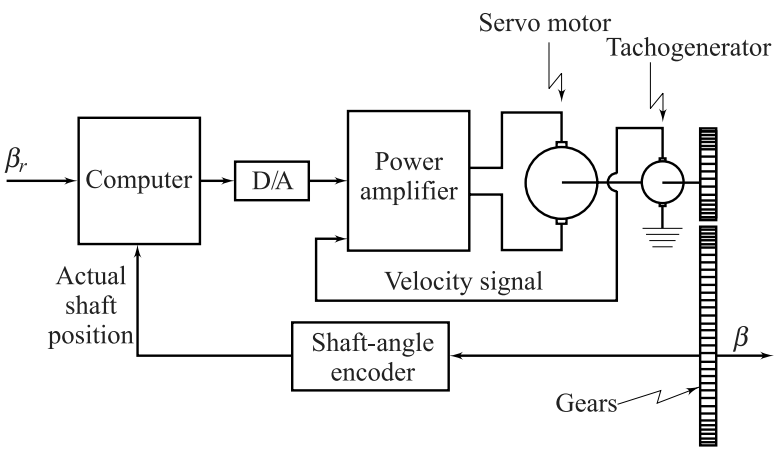


Fig. 1.3 Azimuthal servomechanism for steering of antenna

[1] The bracketed numbers coincide with the list of references given at the end of the book.

The steering command from the radar sensor, which corresponds to target the azimuth angle, is compared with the azimuth angle of the antenna axis. The occurrence of the azimuth-angle error causes an error signal to pass through the amplifier, which increases the angular velocity of the servo motor in a direction towards an error reduction. In the scheme of Fig. 1.3, the measurement and processing of signals (calculation of control signal) is digital in nature. The shaft-angle encoder combines the functions of transducer and A/D converter.

Figure 1.4 gives the functional block diagrams of the control system. A simple model of the load (antenna) on the motor is shown in Fig. 1.4b. The moment of inertia J and the viscous friction coefficient B are the parameters of the assumed model. *Nominal* load is included in the plant model for the control design. The main disturbance inputs are the deviations of the load from the nominal estimated value as a result of uncertainties in our estimate, effect of wind power, etc.

In the tracking system of Fig. 1.4a, the occurrence of error causes the motor to rotate in a direction favoring the dissolution of error. The processing of the error signal (calculation of the control signal) is based on the *proportional control logic*. Note that the components of our system cannot respond instantaneously, since any real-world system cannot go from one energy level to another in zero time. Thus, in any real-world system, there is some kind of dynamic *lagging* behavior between input and output. In the servo system of Fig. 1.4a, the control action, on occurrence of the deviation of the controlled output from the desired value (the occurrence of error), will be delayed by the cumulative dynamic lags of the shaft-angle encoder, digital computer and digital-to-analog converter, power amplifier, and the servo motor with load. Eventually, however, the trend of the controlled variable deviation from the desired value, will be reversed by the action of the amplifier output on the rotation of the motor,

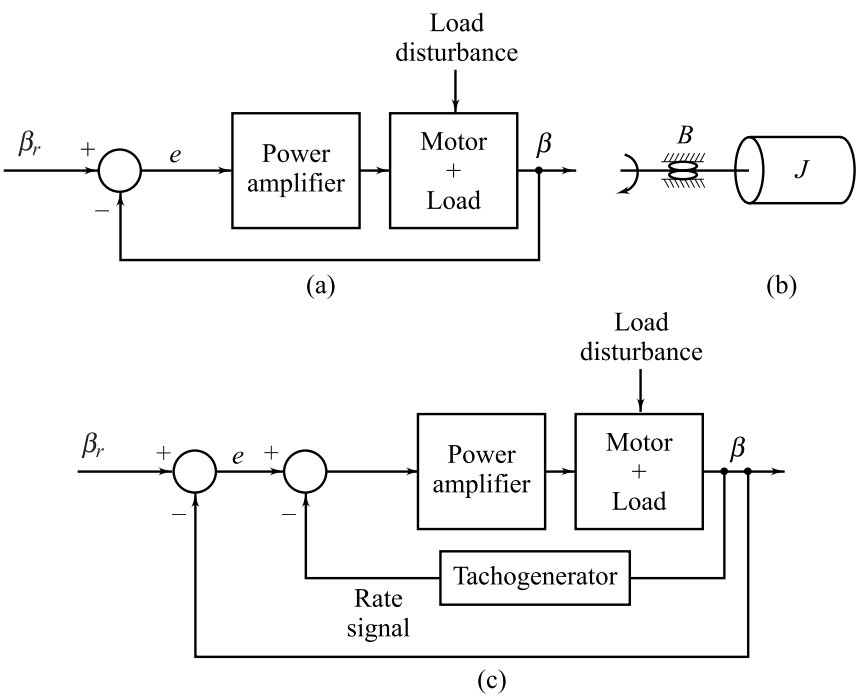


Fig. 1.4 Functional block diagrams of azimuthal servomechanism

returning the controlled variable towards the desired value. Now, if a strong correction (high amplifier gain) is applied (which is desirable from the point of view of control system performance, e.g., strong correction improves the speed of response), the controlled variable overshoots the desired value (the 'run-out' of the motor towards an error with the opposite rotation), causing a reversal in the algebraic sign of the system error. Unfortunately, because of system dynamic lags, a reversal of correction does not occur immediately, and the amplifier output (acting on 'old' information) is now actually driving the controlled variable in the direction it was already heading, instead of opposing its excursions, thus leading to a larger deviation. Eventually, the reversed error does cause a reversed correction, but the controlled variable overshoots the desired value in the opposite direction and the correction is again in the wrong direction. The controlled variable is thus driven, alternatively, in opposite directions before it settles to an equilibrium condition. This oscillatory state is unacceptable as the behavior of antenna-steering servomechanism. The considerable amplifier gain, which is necessary if high accuracies are to be obtained, aggravates the described unfavorable phenomenon.

The occurrence of these oscillatory effects can be controlled by the application of special compensation feedback. When a signal proportional to the motor's angular velocity (called the rate signal) is subtracted from the error signal (Fig. 1.4c), the braking process starts sooner than the error reaches a zero value.

The 'loop within a loop' (*velocity* feedback system embedded within a *position* feedback system) configuration utilized in this application, is a classical scheme called *minor-loop feedback* scheme.

Example 1.2 *Variable Speed dc Drive*

Many industrial applications require variable speed drives. For example, variable speed drives are used for pumping duty to vary the flow rate or the pumping pressure, rolling mills, harbor cranes, rail traction, etc. [2–4].

The variable speed dc drive is the most versatile drive available. Silicon Controlled Rectifiers (SCR) are almost universally used to control the speed of dc motors, because of considerable benefits that accrue from the compact static controllers supplied directly from the ac mains.

Basically, all dc systems involving SCR controllers are similar but, with different configurations of the devices, different characteristics may be obtained from the controller. Figure 1.5 shows a dc motor driven by a full-wave rectified supply. Armature current of the dc motor is controlled by an SCR, which is, in turn, controlled by the pulses applied by the SCR trigger control circuit. The SCR controller thus combines the functions of a D/A converter and a final control element.

Firing angle of the SCR controls the average armature current, which, in turn, controls the speed of the dc motor. The average armature current (speed) increases as the trigger circuit reduces the delay angle of firing of the SCR, and the average armature current (speed) reduces as the delay angle of firing of the SCR is increased.

In the regulator system of Fig. 1.5, the reference voltage which corresponds to the desired speed of the dc motor, is compared with the output voltage of tachogenerator, corresponding to the actual speed of the motor. The occurrence of the error in speed, causes an error signal to pass through the trigger circuit, which controls the firing angle of the SCR in a direction towards an error reduction. When the processing of the error signal (calculation of the control signal) is based on the proportional control logic, a steady-

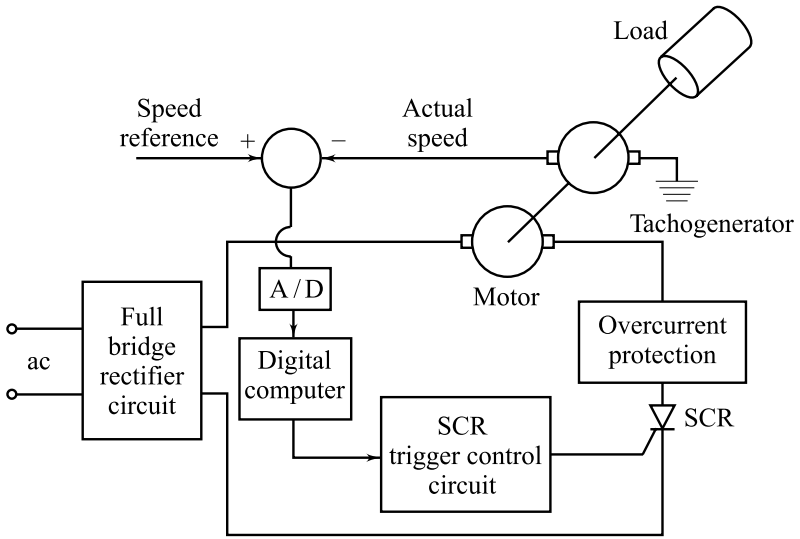


Fig. 1.5 Variable speed dc drive

state error between the actual speed and the desired speed exists. The occurrence of steady-state error can be eliminated by generating the control signal with two components: one component proportional to the error signal, and the other proportional to the integral of the error signal.

Example 1.3 *Liquid-level Control System*

This example describes the hardware features of the design of a PC-based liquid-level control system. The plant of our control system is a cylindrical tank. Liquid is pumped into the tank from the sump (Fig. 1.6). The inflow to the tank can be controlled by adjusting valve V_1. The outflow from the tank goes back into the sump.

Valve V_1 of our plant is a rotary valve; a *stepping motor* has been used to control the valve. The stepping motor controller card, interfaced to the PC, converts the digital control signals into a series of pulses which are fed to the stepping motor using a driver circuit. Three signals are generated from the digital control signal at each sampling instant, namely, number of steps, speed of rotation, and direction of rotation. The stepping motor driver circuit converts this information into a single pulse train, which is fed to the stepping motor. The valve characteristics between the number of steps of the stepping motor and the outflow from the valve, are nonlinear.

The probe used for measurement of liquid level, consists of two concentric cylinders connected to a bridge circuit, to provide an analog voltage. The liquid partially occupies the space between the cylinders, with air in the remaining part. This device acts like two capacitors in parallel; one with dielectric constant of air ($\simeq$1) and the other with that of the liquid. Thus, the variation of the liquid level causes variation of the electrical capacity, measured between the cylinders. The change in the capacitance causes a change in the bridge output voltage which is fed to the PC through an amplifier circuit. The characteristics of the sensor between the level and the voltage are approximately linear.

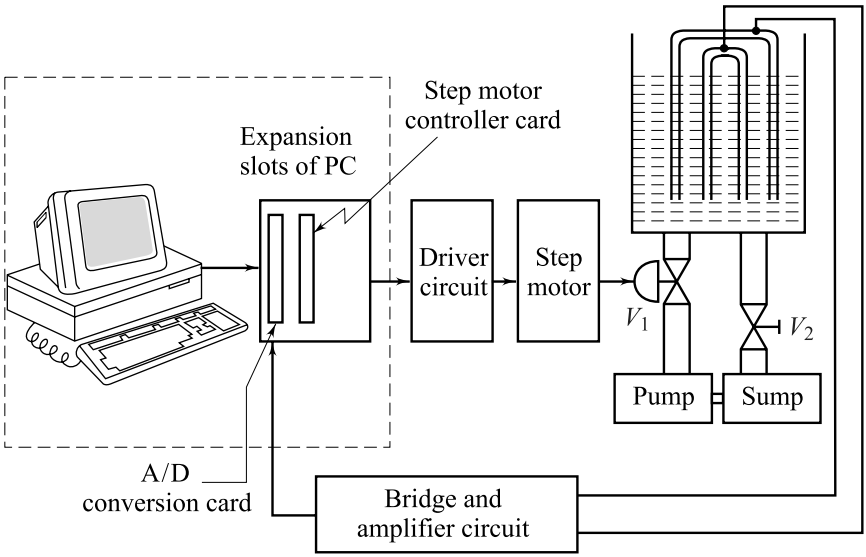


Fig. 1.6 Liquid-level control system

In the liquid-level control system of Fig. 1.6, the *command* signal (which corresponds to the desired level of the liquid in the cylinder) is fed through the keyboard; the actual level signal is received through the A/D conversion card. The digital computer compares the two signals at each sampling instant, and generates a control signal which is the sum of two components: one proportional to the error signal, and the other, proportional to the integral of the error signal.

1.2 COMPUTER-BASED CONTROL: HISTORY AND TRENDS

Digital computers were first applied to industrial process control in the late 1950s. The machines were generally large-scale 'main frames' and were used in a so-called *supervisory control mode*; the individual temperature, pressure, flow and the like, feedback loops were locally controlled by electronic or pneumatic analog controllers. The main function of the computer was to gather information on how the overall process was operating, feed this into a technical-economic model of the process (programmed into computer memory), and then, periodically, send signals to the set-points of all the analog controllers, so that each individual loop operated in such a way as to optimize the overall operation.

In 1962, Imperial Chemical Industries in England made a drastic departure from this approach—a digital computer was installed, which measured 224 variables and manipulated 129 valves directly. The name *Direct Digital Control* (DDC) was coined to emphasize that the computer controlled the process directly. In DDC systems, analog controllers were no longer used. The central computer served as a single, time-shared controller for all the individual feedback loops. Conventional control laws were still used for each loop, but the digital versions of control laws for each loop resided in the software in the central computer. Though digital computers were very expensive, one expected DDC systems to have economic advantage

for processes with many (50 or more) loops. Unfortunately, this did not often materialize. As failures in the central computer of a DDC system shut down the entire system, it was necessary to provide a 'fail-safe' backup system, which usually turned out to be a complete system of individual loop analog controllers, thus negating the expected hardware savings.

There was a substantial development of digital computer technology in the 1960s. By the early 1970s, smaller, faster, more reliable, and cheaper computers became available. The term *minicomputers* was coined for the new computers that emerged. DEC PDP11 is by far, the best-known example. There were, however, many related machines from other vendors.

The minicomputer was still a fairly large system. Even as performance continued to increase and prices to decrease, the price of a minicomputer main frame in 1975, was still about $10,000. Computer control was still out of reach for a large number of control problems. However, with the development of *microcomputer*, the price of a card computer, with the performance of a 1975 minicomputer, dropped to $500 in 1980. Another consequence was that digital computing power in 1980 came *in quanta* as small as $50. This meant that computer control could now be considered as an alternative, no matter how small the application [54–57].

Microcomputers have already made a great impact on the process control field. They are replacing analog hardware even as *single-loop* controllers. Small DDC systems have been made using microcomputers. Operator communication has vastly improved with the introduction of color video-graphics displays.

The variety of commercially available industrial controllers ranges from single-loop controllers through multiloop single computer systems to multiloop distributed computers. Although the range of equipment available is large, there are a number of identifiable trends which are apparent.

Single-loop microprocessor-based controllers, though descendants of single-loop analog controllers, have greater degree of flexibility. Control actions which are permitted, include on/off control, proportional action, integral action, derivative action, and the lag effect. Many controllers have self-tuning option. During the self-tune sequence, the controller introduces a number of step commands, within the tolerances allowed by the operator, in order to characterize the system response. From this response, values for proportional gain, reset time, and rate time are developed. This feature of online tuning in industrial controllers is interesting, and permits the concept of the computer automatically adjusting to changing process conditions [11–12].

Multiloop single computer systems have variability in available interface and software design. Both single-loop and multiloop controllers may be used in stand-alone mode, or may be interfaced to a host computer for distributed operation. The reducing costs and increasing power of computing systems, has tended to make distributed computing systems for larger installations, far more cost effective than those built around one large computer. However, the smaller installation may be best catered for by a single multiloop controller, or even a few single-loop devices.

Control of large and complex processes using *Distributed Computer Control Systems* (DCCS), is facilitated by adopting a multilevel or hierarchical view point of control strategy. The multilevel approach subdivides the system into a hierarchy of simpler control design problems. *On the lowest level of control* (*direct process control level*), the following tasks are handled: acquisition of process data, i.e., collection of instantaneous values of individual process variables, and status messages of plant control facilities (valves, pumps, motors, etc.) needed for efficient direct digital control; processing of collected data; plant hardware monitoring, system check and diagnosis; closed-loop control and logic control functions, based on directives from the next 'higher' level.

Supervisory level copes with the problems of determination of optimal plant work conditions, and generation of relevant instructions to be transferred to the next 'lower' level. Adaptive control, optimal control, plant performance monitoring, plant coordination and failure detections are the functions performed at this level.

Production scheduling and control level is responsible for production dispatching, inventory control, production supervision, production rescheduling, production reporting, etc.

Plant(s) management level, the 'highest' hierarchical level of the plant automation system, is in charge of the wide spectrum of engineering, economic, commercial, personnel, and other functions.

It is, of course, not to be expected that in all available distributed computer control systems, all four hierarchical levels are already implemented. For automation of small-scale plants, any DCCS having at least two hierarchical levels, can be used. One system level can be used as a direct process control level, and the second one as a combined plant supervisory, and production scheduling and control level. Production planning and other enterprise-level activities, can be managed by the separate mainframe computer or the computer center. For instance, in a LAN (Local Area Network)-based system structure, shown in Fig. 1.7a, the 'higher' automation levels are implemented by simply attaching the additional 'higher' level computers to the LAN of the system [89].

For complex process plant monitoring, SCADA (*Supervisory Control And Data Acquisition*) systems are available. The basic functions carried out by a SCADA system are as follows:

- Data acquisition and communication
- Events and alarms reporting
- Data processing
- Partial process control

The full process control functions are delegated to the special control units, connected to the SCADA system, and are capable of handling emergency shut down situations.

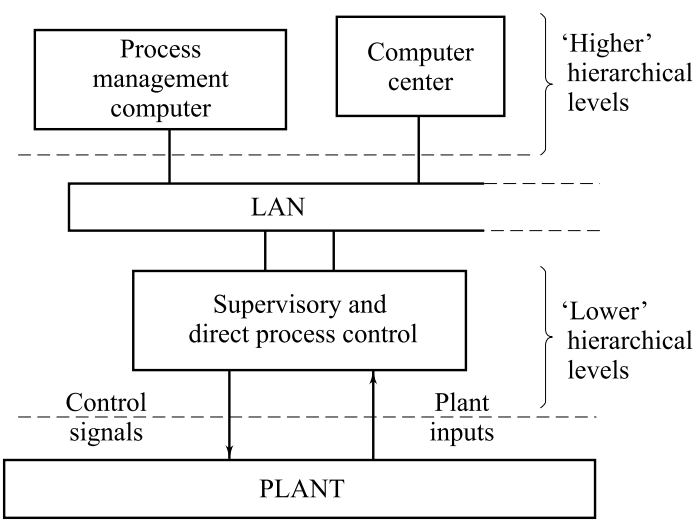


Fig. 1.7a Hierarchical levels in Computer Integrated Process Systems (CIPS)

The separation of SCADA and DCCS is slowly vanishing and the SCADA systems are being brought within the field of DCCS; the hierarchical, distributed, flexible and extremely powerful *Computer Integrated Process Systems* (CIPS), is now a technical reality.

The other main and early application area of digital methods was *machine tool numerical control*, which developed at about the same time as computer control in process industries. Earlier, numerically controlled (NC) machines used 'hard-wired' digital techniques. As the price and performance of microcomputers improved, it became feasible to replace the hard-wired functions with their software-implemented equivalents, using a microcomputer as a built-in component of the machine tool. This approach has been called *Computerized Numerical Control* (CNC) [20]. Industrial *robots* were developed simultaneously with CNC systems.

A quiet revolution is ongoing in the manufacturing world, which is changing the look of factories. Computers are controlling and monitoring the manufacturing processes [21–22]. The high degree of automation that, until recently, was reserved for mass production only, is also applied now to small batches. This requires a change from hard automation in the production line, to a *Flexible Manufacturing System* (FMS), which can be more readily rearranged to handle new market requirements.

Flexible manufacturing systems, combined with automatic assembly and product inspection on one hand, and CAD/CAM systems on the other, are the basic components of the modern *Computer Integrated Manufacturing System* (CIMS). In a CIMS, the production flow, from the conceptual design to the finished product, is entirely under computer control and management.

Figure 1.7b illustrates the hierarchical structure of CIMS. The lowest level of this structure contains stand-alone computer control systems of manufacturing processes and industrial robots. The computer control of processes includes all types of CNC machine tools, welding, electrochemical machining, electrical discharge machining, and a high-power laser, as well as the adaptive control of these processes.

When a battery of NC or CNC machine tools is placed under the control of a single computer, the result is a system known as *Direct Numerical Control* (DNC).

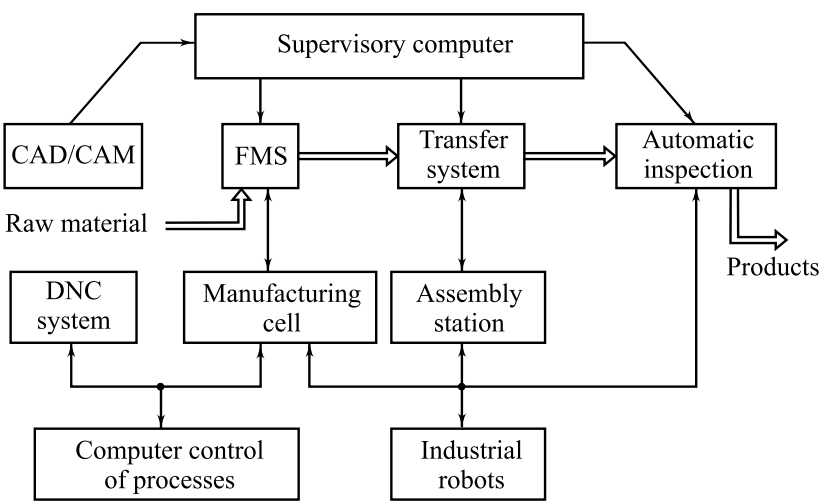


Fig. 1.7b Hierarchical levels in Computer Integrated Manufacturing Systems (CIMS)

The operation of several CNC machines and industrial robots, can be coordinated by systems called manufacturing cells. The computer of the cell is interfaced with the computer of the robot and CNC machines. It receives 'completion of job' signals from the machines and issues instructions to the robot to load and unload the machines, and change their tools. The software includes strategies permitting the handling of machine breakdown, tool breakage, and other special situations.

The operation of many manufacturing cells can be coordinated by Flexible Manufacturing System (FMS). The FMS accepts incoming workpieces and processes them under computer control, into finished parts.

The parts produced by the FMS must be assembled into the final product. They are routed on a transfer system to assembly stations. In each station, a robot will assemble parts, either into a sub-assembly or (for simple units), into the final product. The sub-assemblies will be further assembled by robots located in other stations. The final product will be tested by an automatic inspection system.

The FMS uses CAD/CAM systems to integrate the design and manufacturing of parts. At the highest hierarchical level, there will be a supervisory computer, which coordinates participation of computers in all phases of a manufacturing enterprise: the design of the product, the planning of its manufacture, the automatic production of parts, automatic assembly, automatic testing, and, of course, computer-controlled flow of materials and parts through the plant.

In a LAN-based system, the 'higher' automation levels (production planning and other enterprise-level activities), can be implemented by simply attaching the additional 'higher' level computers to the LAN of the system.

One of the most ingenious devices ever devised to advance the field of industrial automation, is the *Programmable Logic Controller* (PLC). The PLC, a microprocessor-based general-purpose device, provides a 'menu' of basic operations that can be configured by programming to create logic control system for any application [23–25]. So versatile are these devices, that they are employed in the automation of almost every type of industry. CIPS and CIMS provide interfaces to PLCs for handling high-speed logic (and other) control functions. Thousands of these devices go unrecognized in process plants and factory environments—quietly monitoring security, manipulating valves, and controlling machines and automatic production lines.

Thus, we see that the recent appearance of powerful and inexpensive microcomputers, has made digital control practical for a wide variety of applications. In fact, now every process is a candidate for digital control. The flourishing of digital control is just beginning for most industries, and there is much to be gained by exploiting the full potential of new technology. There is every indication that a high rate of growth in the capability and application of digital computers, will continue far into the future.

1.3 CONTROL THEORY: HISTORY AND TRENDS

The development of control system analysis and design can be divided into three eras. In the first era, we have the *classical control theory*, which deals with techniques developed during the 1940s and 1950s. Classical control methods—Routh–Hurwitz, Root Locus, Nyquist, Bode, Nichols—have in common the use of transfer functions in the complex frequency (Laplace variable s) domain, and the emphasis

on the graphical techniques. Since computers were not available at that time, a great deal of emphasis was placed on developing methods that were amenable to manual computation and graphics. A major limitation of the classical control methods was the use of Single-Input, Single-Output (SISO) control configurations. Also, the use of the transfer function and frequency domain limited one to linear time-invariant systems. Important results of this era have been discussed in Part I of this book.

In the second era, we have *modern control* (which is not so modern any longer), which refers to *state-space*-based methods developed in the late 1950s and early 1960s. In modern control, system models are directly written in the time domain. Analysis and design are also carried out in the time domain. It should be noted that before Laplace transforms and transfer functions became popular in the 1920s, engineers were studying systems in the time domain. Therefore, the resurgence of time-domain analysis was not unusual, but it was triggered by the development of computers and advances in numerical analysis. As computers were available, it was no longer necessary to develop analysis and design methods that were strictly manual. *Multivariable* (Multi-Input, Multi-Output (MIMO)) control configurations could be analyzed and designed. An engineer could use computers to numerically solve or simulate large systems that were nonlinear and/or time-varying. Important results of this era—*Lyapunov stability criterion*, *pole-placement by state feedback*, *state observers, optimal control*—are discussed in Part II of this book.

Modern control methods initially enjoyed a great deal of success in academic circles, but they did not perform very well in many areas of application. Modern control provided a lot of insight into system structure and properties, but it masked other important feedback properties that could be studied and manipulated using the classical control theory. A basic requirement in control engineering is to design control systems that will work properly when the plant model is uncertain. This issue is tackled in the classical control theory using gain and phase margins. Most modern control design methods, however, inherently require a precise model of the plant. In the years since these methods were developed, there have been few significant implementations and most of them have been in a single application area—the aerospace industry. The classical control theory, on the other hand, is going strong. It provides an efficient framework for the design of feedback controls in all areas of application. The classical design methods have been greatly enhanced by the availability of low-cost computers for system analysis and simulation. The graphical tools of classical design can now be more easily used with computer graphics for SISO as well as MIMO systems.

During the past three decades, the control theory has experienced a rapid expansion, as a result of the challenges of the stringent requirements posed by modern systems, such as flight vehicles, weapon control systems, robots, and chemical processes; and the availability of low-cost computing power. A body of methods emerged during this third era of control-theory development, which tried to provide answers to the problems of plant uncertainty. These techniques, commonly known as *robust control*, are a combination of modern state-space and classical frequency-domain techniques. For a thorough understanding of these new methods, we need to have adequate knowledge of state-space methods, in addition to the frequency-domain methods. This has guided the preparation of this text.

Robust control system design has been dominated by linear control techniques, which rely on the key assumption of availability of the *uncertainty model*. When the required operation range is large, and a reliable uncertainty model cannot be developed, a linear controller is likely to perform very poorly. Nonlinear controllers, on the other hand, may handle the nonlinearities in large-range operations, directly. Also, nonlinearities can be intentionally introduced into the controller part of a control system, so that the model uncertainties can be tolerated. Advances in computer technology have made the implementation

of nonlinear control schemes—*feedback linearization*, *variable structure sliding mode control*, *adaptive control*, *gain scheduling*—a relatively simpler task.

The third era of control-theory development has also given an alternative to model-based design methods: the *knowledge-based control method*. In this approach, we look for a control solution that exhibits *intelligent behavior*, rather than using purely mathematical methods to keep the system under control.

Model-based control techniques have many advantages. When the underlying assumptions are satisfied, many of these methods provide good stability, robustness to model uncertainties and disturbances, and speed of response. However, there are many practical deficiencies of these 'crisp' ('hard' or 'inflexible') control algorithms. It is generally difficult to accurately represent a complex process by a mathematical model. If the process model has parameters whose values are partially known, ambiguous or vague, then crisp control algorithms, that are based on such *incomplete information*, will not usually give satisfactory results. The environment with which the process interacts, may not be completely predictable and it is normally not possible for a crisp algorithm, to accurately respond to a condition that it did not anticipate, and that it could not 'understand'.

Intelligent control is the name introduced to describe control systems in which control strategies are based on AI (Artificial Intelligence) techniques. In this control approach, which is an alternative to *model-based control* approach, a *behavioral* (and not mathematical) description of the process is used, which is based on qualitative expressions and experience of people working with the process. Actions can be performed either as a result of evaluating rules (reasoning), or as unconscious actions based on presented process behavior after a learning phase. Intelligence becomes a measure of the capability to reason about facts and rules, and to learn about presented behavior. It opens up the possibility of applying the experience gathered by operators and process engineers. Uncertainty about facts and rules along with ignorance about the structure of the system can then be handled easily.

Fuzzy logic, and *neural networks* are very good methods to model real processes which cannot be described mathematically. Fuzzy logic deals with linguistic and imprecise rules based on an expert's knowledge. Neural networks are applied in the case where we do not have any rules but several data.

The main feature of *fuzzy logic control* is that a control engineering knowledge base (typically in terms of a set of rules), created using an expert's knowledge of process behavior, is available within the controller and the control actions are generated by applying existing process conditions to the knowledge base, making use of an *inference mechanism*. The knowledge base and the inference mechanism can handle noncrisp and incomplete information, and the knowledge itself will improve and evolve through learning and past experience.

In *neural network based control*, the goal of *artificial neural network* is to emulate the mechanism of human brain function and reasoning, and to achieve the same intelligence level as the human brain in learning, abstraction, generalization and making decisions under uncertainty.

In conventional design exercises, the system is modeled analytically by a set of differential equations, and their solution tells the controller how to adjust the system's control activities for each type of behavior. In a typical intelligent control scheme, these adjustments are handled by an intelligent controller, a logical model of thinking processes that a person might go through in the course of manipulating the system. This shift in focus from the process to the person involved, changes the entire approach to automatic control problems. It provides a new design paradigm such that a controller can be designed for complex,

ill-defined processes without knowing quantitative input-output relations, which are otherwise required by conventional methods.

The ever-increasing demands of the complex control systems being built today, and planned for the future, dictate the use of novel and more powerful methods in control. The potential for intelligent control techniques in solving many of the problems involved is great, and this research area is evolving rapidly. The emerging viewpoint is that model-based control techniques should be augmented with intelligent control techniques in order to enhance the performance of control systems. The developments in intelligent control methods should be based on firm theoretical foundations (as is the case with model-based control methods), but this is still at its early stages. Strong theoretical results guaranteeing control system properties such as stability are still to come, although promising results reporting progress in special cases have been reported recently. The potential of intelligent control systems clearly needs to be further explored and both theory and applications need to be further developed. A brief account of nonlinear control schemes, both the conventional and the intelligent, is given in Part III of this book.

1.4 AN OVERVIEW OF THE CLASSICAL APPROACH TO ANALOG CONTROLLER DESIGN

The tools of classical linear control system design are the Laplace transform, stability testing, root locus, and frequency response. Laplace transformation is used to convert system descriptions in terms of integro-differential equations to equivalent algebraic relations involving rational functions. These are conveniently manipulated in the form of transfer functions with block diagrams and signal flow graphs [155].

The block diagram of Fig. 1.8 represents the basic structure of feedback control systems. Not all systems can be forced into this format, but it serves as a reference for discussion.

In Fig. 1.8, the variable $y(t)$ is the *controlled variable* of the system. The desired value of the controlled variable is $y_r(t)$, the command input. $y_r(t)$ and $y(t)$ have the same units. The *feedback elements* with transfer function $H(s)$ are system components that act on the controlled variable $y(t)$ to produce the *feedback signal* $b(t)$. $H(s)$ typically represents the sensor action to convert the controlled variable $y(t)$ to an electrical sensor output signal $b(t)$.

The *reference input elements* with transfer function $A(s)$ convert the command signal $y_r(t)$ into a form compatible with the feedback signal $b(t)$. The transformed command signal is the actual physical input to the system. This actual signal input is defined as the *reference input*.

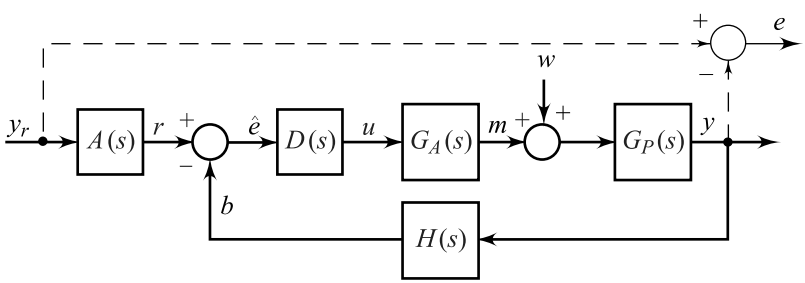


Fig. 1.8 Generalised operational block diagram of a feedback system

The comparison device (*error detector*) of the system compares the reference input $r(t)$ with the feedback signal $b(t)$ and generates the *actuating error signal* $\hat{e}(t)$. The signals $r(t)$, $b(t)$, and $\hat{e}(t)$ have the same units. The *controller* with transfer function $D(s)$ acts on the actuating error signal to produce the *control signal* $u(t)$.

The control signal $u(t)$ has the knowledge about the desired control action. The power level of this signal is relatively low. The *actuator elements* with transfer function $G_A(s)$, are the system components that act on the control signal $u(t)$ and develop enough torque, pressure, heat, etc. (*manipulated variable* $m(t)$), to influence the *controlled system*. $G_P(s)$ is the transfer function of the controlled system.

The *disturbance* $w(t)$ represents the undesired signals that tend to affect the controlled system. The disturbance may be introduced into the system at more than one location.

The dashed-line portion of Fig. 1.8 shows the *system error* $e(t) = y_r - y(t)$. Note that the actuating error signal $\hat{e}(t)$ and the system error $e(t)$ are two different variables.

The basic feedback system block diagram of Fig. 1.8 is shown in an abridged form in Fig. 1.9. The output $Y(s)$ is influenced by the control signal $U(s)$ and the disturbance signal $W(s)$ as per the following relation:

$$Y(s) = G_P(s)\, G_A(s)\, U(s) + G_P(s)\, W(s) \tag{1.1a}$$

$$= G(s)\, U(s) + N(s)\, W(s) \tag{1.1b}$$

where $G(s)$ is the transfer function from the control signal $U(s)$ to the output $Y(s)$, and $N(s)$ is the transfer function from the disturbance input $W(s)$ to the output $Y(s)$. Using Eqns (1.1), we can modify the block diagram of Fig. 1.9 to the form shown in Fig. 1.10. Note that in the block diagram model of Fig. 1.10, the plant includes the actuator elements.

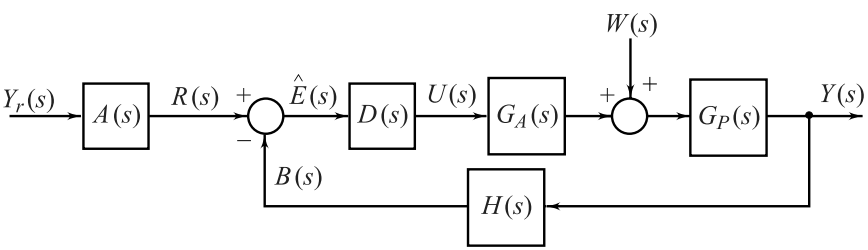


Fig. 1.9 A general linear feedback system

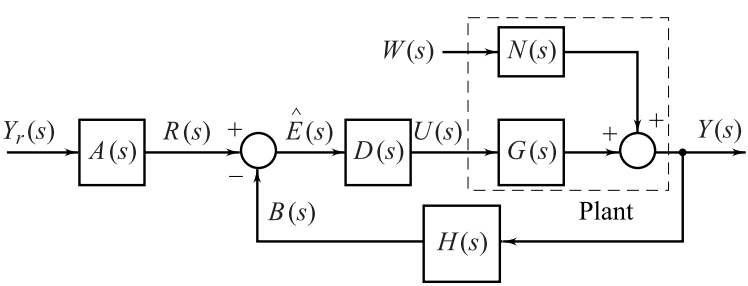


Fig. 1.10 Equivalent representation of the block diagram of Fig. 1.9

The actuating error signal

$$\hat{E}(s) = R(s) - B(s) = A(s)\ Y_r(s) - H(s)\ Y(s)$$

The control signal

$$U(s) = D(s)\ A(s)\ Y_r(s) - D(s)\ H(s)\ Y(s) \tag{1.2a}$$

$$= D(s)\ H(s) \left[\frac{A(s)}{H(s)}\ Y_r(s) - Y(s) \right] \tag{1.2b}$$

Using Eqns (1.2a) and (1.2b), we can simplify Fig. 1.10 to obtain the structure shown in Fig. 1.11.

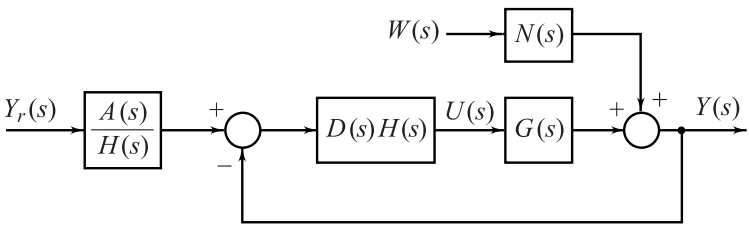


Fig. 1.11 Simplification of the block diagram of Fig. 1.10

Further simplification of Fig. 1.11 is possible if $H = A$. In this case, which is quite common, we can model the system as a *unity-feedback system* shown in Fig. 1.12, and take advantage of the fact that now the actuating signal is the system error $e(t)$.

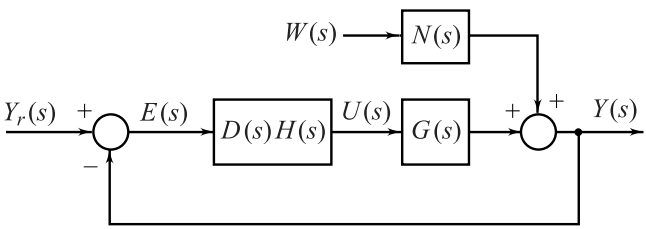


Fig. 1.12 Unity feedback system

The block diagrams in Figs 1.10–1.12 are very useful for the purpose of system design. However, it should be clear that these block diagrams have lost physical significance. For example, the block in Fig. 1.11 with transfer function $A(s)/H(s)$, does not refer to any physical portion of the original system. Rather, it represents the result of manipulating Eqn. (1.2a) into the form given by Eqn. (1.2b).

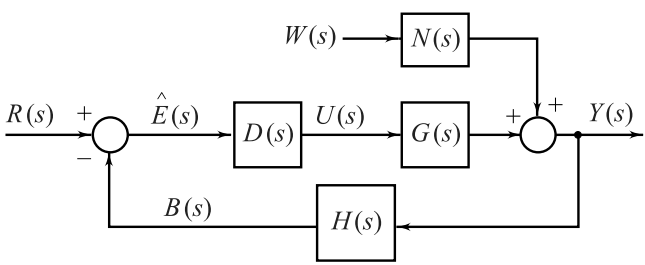


Fig. 1.13 A typical feedback systen with two inputs

Thus, the reader is advised to think in terms of the equations that the block diagrams represent, rather than attach any special significance to the block diagrams themselves. The only role played by a block diagram is that it is a convenient means of representing the

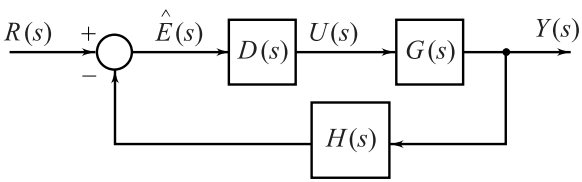


Fig. 1.14 Block diagram without disturbance input

various system equations, rather than writing them out explicitly. *Block diagram manipulation* is nothing more than the manipulation of a set of algebraic transform equations.

For the analysis of a feedback system, we require the transfer function between the input—either reference or disturbance—and the output. We can use block diagram manipulations to eliminate all the signals except the input and the output. The reduced block diagram leads to the desired result.

Consider the block diagram of Fig. 1.13. The feedback system has two inputs. We shall use superposition to treat each input separately.

When disturbance input is set to zero, the single-input system of Fig. 1.14 results. The transfer function between the input $R(s)$ and the output $Y(s)$ is referred to as the *reference transfer function* and will be denoted by $M(s)$. To solve for $M(s)$, we write the pair of transform equations

$$\hat{E}(s) = R(s) - H(s)\ Y(s);\ Y(s) = G(s)\ U(s) = G(s)\ D(s)\ \hat{E}(s)$$

and then eliminate $\hat{E}(s)$ to obtain

$$[1 + D(s)\ G(s)\ H(s)]\ Y(s) = D(s)\ G(s)\ R(s)$$

which leads to the desired result

$$M(s) = \left.\frac{Y(s)}{R(s)}\right|_{W(s)=0} = \frac{D(s)G(s)}{1 + D(s)G(s)H(s)} \tag{1.3}$$

Similarly, we obtain the *disturbance transfer function* $M_w(s)$ by setting the reference input to zero in Fig. 1.13 yielding Fig. 1.15, and then solving for $Y(s)/W(s)$. From the revised block diagram,

$$\hat{E}(s) = -H(s)Y(s);\ Y(s) = G(s)D(s)\ \hat{E}(s) + N(s)W(s)$$

from which $\hat{E}(s)$ can be eliminated to give

$$M_w(s) = \left.\frac{Y(s)}{W(s)}\right|_{R(s)=0} = \frac{N(s)}{1 + D(s)G(s)H(s)} \tag{1.4}$$

The response to the simultaneous application of $R(s)$ and $W(s)$ is given by

$$Y(s) = M(s)\ R(s) + M_w(s)\ W(s) \tag{1.5}$$

Figure 1.16 shows the reduced block diagram model of the given feedback system.

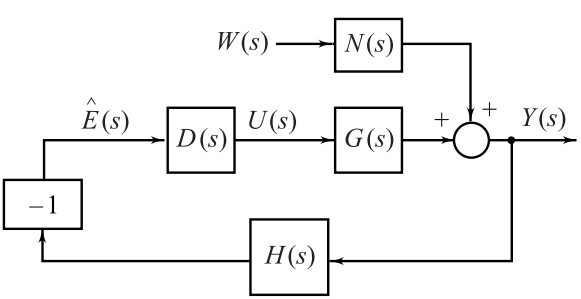


Fig. 1.15 Block diagram without reference input

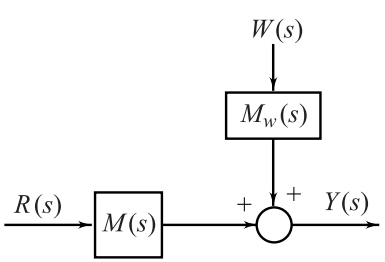


Fig. 1.16 Reduced block diagram model for system of Fig. 1.13

The transfer functions given by Eqns (1.3) and (1.4) are referred to as *closed-loop transfer functions*. The denominator of these transfer functions has the term $D(s)G(s)H(s)$ which is the multiplication of all the transfer functions in the feedback loop. It may be viewed as the transfer function between the variables $R(s)$ and $B(s)$ if the loop is broken at the summing point. $D(s)G(s)H(s)$ may, therefore, be given the name *open-loop transfer function*. The roots of denominator polynomial of $D(s)G(s)H(s)$ are the *open-loop poles*, and the roots of numerator polynomial of $D(s)G(s)H(s)$ are the *open-loop zeros*.

The roots of the characteristic equation

$$1 + D(s)G(s)H(s) = 0 \tag{1.6}$$

are the *closed-loop poles* of the system. These poles indicate whether or not the system is Bounded-Input Bounded-Output (BIBO) stable, according to whether or not all the poles are in the left half of the complex plane. Stability may be tested by the Routh stability criterion.

A root locus plot consists of a pole-zero plot of the open-loop transfer function of a feedback system, upon which is superimposed the locus of the poles of the closed-loop transfer function, as some parameter is varied. Design of the *controller* (*compensator*) $D(s)$ can be carried out using the root locus plot. One begins with simple compensators, increasing their complexity until the performance requirements can be met. Principal measures of transient performance are peak overshoot, settling time, and rise time. The compensator poles, zeros, and multiplying constant are selected to give feedback system pole locations, that result in acceptable transient response to step inputs. At the same time, the parameters are constrained so that the resulting system has acceptable *steady-state* response to important inputs, such as steps and ramps.

Frequency response characterizations of systems have long been popular because of the ease and practicality of steady-state sinusoidal response measurements. These methods also apply to systems in which rational transfer function models are not adequate, such as those involving time delays. They do not require explicit knowledge of system transfer function models; experimentally obtained open-loop sinusoidal response data can directly be used for stability analysis and compensator design. A stability test, the Nyquist criterion, is available. Principal measures of transient performance are gain margin, phase margin, and bandwidth. The design of the compensator is conveniently carried out using the Bode plot and the Nichols chart. One begins with simple compensators, increasing their complexity until the transient and steady-state performance requirements are met.

There are two approaches to carry out the digital controller (compensator) design. The first approach uses the methods discussed above to design an analog compensator, and then transform it into a digital one. The second approach, first transforms analog plants into digital plants, and then carries out the design using digital techniques. The first approach performs discretization *after* design; the second approach performs discretization *before* design. The classical approach to designing a digital compensator directly using an equivalent digital plant for a given analog plant, parallels the classical approach to analog compensator design. The concepts and tools of the classical digital design procedures are given in Chapters 2–4. This background will also be useful in understanding and applying the state variable methods to follow.

Chapter 2

Signal Processing in Digital Control

2.1 WHY USE DIGITAL CONTROL?

Digital control systems offer many advantages over their analog counterparts. Of course, there are possible problems also. Let us first look at the advantages of digital control over the corresponding analog control before we talk of the price one has to pay for the digital option.

2.1.1 Advantages Offered by Digital Control

Flexibility An important advantage offered by digital control is in the flexibility of its modifying controller characteristics, or in other words, in adaptability of the controller if plant dynamics change with operating conditions. The ability to 'redesign' the controller by changing software (rather than hardware) is an important feature of digital control against analog control.

Wide Selection of Control Algorithms Implementation of advanced control techniques was earlier constrained by the limitations of analog controllers and the high costs of digital computers. However, with the advent of inexpensive digital computers with virtually limitless computing power, the techniques of modern control theory may now be put to practice. For example, in multivariable control systems with more than one input and one output, modern techniques for optimizing system performance or reducing interactions between feedback loops can now be implemented.

Integrated Control of Industrial Systems Feedback control is only one of the functions of a computer. In fact, most of the information transfer between the process and the computer exploits the logical decision-making capability of the computer. Real-time applications of information processing and decision-making, e.g., production planning, scheduling, optimization, operations control, etc., may now be integrated with the traditional process control functions.

To enable the computer to meet a variety of demands imposed on it, its tasks are *time-shared*.

Future Generation Control Systems The study of emerging applications shows that Artificial Intelligence (AI) will affect the design and application of control systems, as profoundly as the impact of microprocessors in the last two decades. It is clear that future generation control systems will have a significant AI component; the list of applications of computer-based control will continue to expand.

2.1.2 Implementation Problems in Digital Control

The main problems associated with the implementation of digital control are related to the effects of *sampling* and *quantization*.

Most processes that we are called upon to control, operate in continuous-time. This implies, that we are dealing largely with an analog environment. To this environment, we need to interface digital computers through which we seek to influence the process.

The interface is accomplished by a system of the form shown in Fig. 2.1. It is a cascade of *analog-to-digital* (A/D) conversion system followed by a computer which is, in turn, followed by a *digital-to-analog* (D/A) conversion system. The A/D conversion process involves deriving samples of the analog signal at discrete instants of time separated by *sampling period T* sec. The D/A conversion process involves reconstructing continuous-time signals from the samples given by the digital computer.

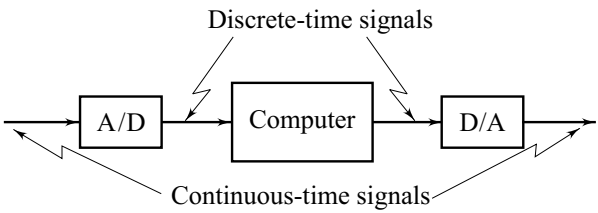


Fig. 2.1 Discrete-time processing of continuous-time signals

Quantization Effects

The conversion of signals from analog into digital form and vice versa is performed by electronic devices (A/D and D/A converters) of *finite* resolution. A device of n-bit resolution has 2^n quantization levels. Here, the analog signal gets tied to these finite number of quantization levels in the process of conversion to digital form. Therefore, by the sheer act of conversion, a valuable part of information about the signal, is lost.

Furthermore, any computer employed as a real-time controller must perform all the necessary calculations with limited precision, thus introduction of a *truncation error* after each arithmetic operation has been performed. As computational accuracy is normally much higher than the resolution of real converters, a further truncation must take place before the computed data are converted into the analog form. The repetitive process of approximate conversion–computation–conversion may be costly, if not disastrous, in terms of control system performance.

The process of quantization in signal conversion systems is discussed ahead.

Sampling Effects

The selection of a sampling period is a fundamental problem in digital control systems. Later in this chapter, we will discuss the *sampling theorem* which states that the sampling period T should be chosen such that

$$T < \pi/\omega_m$$

where ω_m is the strict bandwidth of the signal being sampled. This condition ensures that there is no loss of information due to sampling and the continuous-time signal can be completely recovered from its samples using an ideal low-pass filter.

There are, however, two problems associated with the use of this theorem in practical control systems:

(i) Real signals are not band-limited and hence strict bandwidth limits are not defined.

(ii) An ideal low-pass filter, needed for the distortionless reconstruction of continuous-time signals from its samples, is not physically realizable. Practical devices, such as the D/A converter, introduce distortions.

Thus, the process of sampling and reconstruction also affects the amount of information available to the control computer, and degrades control system performance. For example, converting a given continuous-time control system into a digital control system, without changing the system parameters, degrades the system stability margin.

The ill-effects of sampling can be reduced, if not eliminated completely, by sampling at a very high rate. However, excessively fast sampling ($T \rightarrow 0$) may result in *numerical ill-conditioning* in the implementation of recursive control algorithms (described later in this chapter).

With the availability of low-cost, high-performance digital computers and interfacing hardware, the implementation problems in digital control do not pose a serious threat to its usefulness. The advantages of digital control outweigh its implementation problems for most of the applications.

This book attempts to provide a modest coverage of digital control theory and practice. In the present chapter, we focus on digital computers and their interface with signal conversion systems (Fig. 2.1). The goal is to formulate tools of analysis necessary to understand and guide the design of programs for a computer acting as a control logic component. Needless to say, digital computers can do many things other than control dynamic systems; our purpose is to examine their characteristics while executing the elementary control task.

2.2 CONFIGURATION OF THE BASIC DIGITAL CONTROL SCHEME

Figure 2.2 depicts a block diagram of a digital control system showing a configuration of the basic control scheme. The basic elements of the system are shown by the blocks.

The analog feedback signal coming from the sensor is usually of low frequency. It may often include high frequency 'noise'. Such noise signals are too fast for the control system to correct; low-pass filtering is often needed to allow good control performance. The anti-aliasing filter shown in Fig. 2.2 serves this purpose. In digital systems, a phenomenon called *aliasing* introduces some new aspects of noise problems. We will study this phenomenon later in this chapter.

The analog signal, after anti-aliasing processing, is converted into digital form by the A/D conversion system. The conversion system usually consists of an A/D converter preceded by a sample-and-hold (S/H) device. The A/D converter converts a voltage (or current) amplitude, at its input, into a binary code representing a quantized amplitude value closest to the amplitude of the input. However, the conversion is not instantaneous. Input signal variation, during the conversion time of the A/D converter, can lead to erroneous results. For this reason, high performance A/D conversion systems include an S/H device, which keeps the input to the A/D converter, constant during its conversion time.

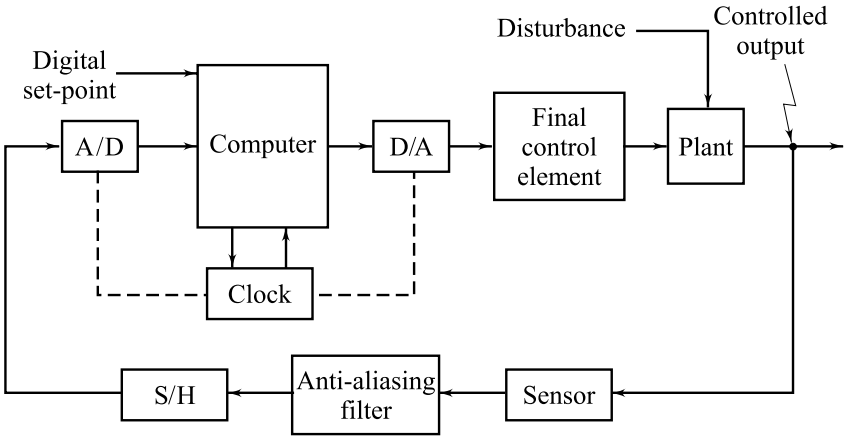


Fig. 2.2 Configuration of the basic digital control scheme

The digital computer processes the sequence of numbers by means of an algorithm and produces a new sequence of numbers. Since data conversions and computations take time, there will always be a delay when a control law is implemented using a digital computer. The delay, which is called *computational delay*, degrades the control system performance. It should be minimized by the proper choice of hardware and by the proper design of software for the control algorithm. *Floating-point operations* take a considerably longer time to perform (even when carried out by an arithmetic co-processor) than the *fixed-point* ones. We, therefore, try to execute fixed-point operations whenever possible. Alternative realization schemes for a control algorithm are given in the next chapter.

The D/A conversion system in Fig. 2.2 converts the sequence of numbers in numerical code into a piecewise *continuous-time signal*. The output of the D/A converter is fed to the plant through the *actuator* (final control element) to control its dynamics.

The basic control scheme of Fig. 2.2 assumes a *uniform sampling* operation, i.e., only one sampling rate exists in the system and the sampling period is constant. The real-time clock in the computer, synchronizes all the events of A/D conversion–computation–D/A conversion.

The control scheme of Fig. 2.2 shows a single feedback loop. In a control system having multiple loops, the largest time constant involved in one loop may be quite different from that in other loops. Hence, it may be advisable to sample slowly in a loop involving a large time constant, while in a loop involving only small time constants, the sampling rate must be fast. Thus, a digital control system may have different sampling periods in different feedback paths, i.e., it may have *multiple-rate sampling*. Although digital control systems with multirate sampling are important in practical situations, we shall concentrate on *single-rate sampling*. (The reader interested in multirate digital control systems may refer to Kuo [87]).

The overall system in Fig. 2.2 is hybrid in nature; the signals are in a *sampled* form (*discrete-time signals*/*digital signals*) in the computer and in *continuous-time* form in the plant. Such systems have traditionally been called *sampled-data control systems*. We will use this term as a synonym of *computer control systems*/*digital control systems*.

In the present chapter, we focus on digital computers and their analog interfacing. For the time being, we delink the digital computer from the plant. The link will be re-established in the next chapter.

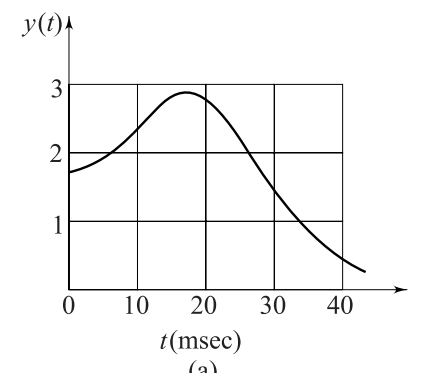
2.3 PRINCIPLES OF SIGNAL CONVERSION

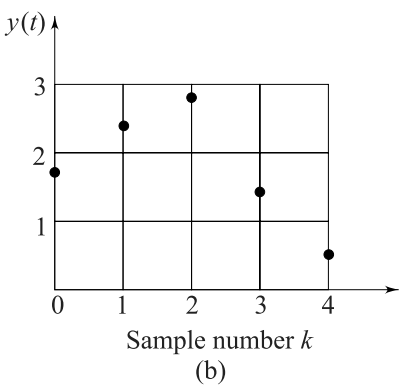
Figure 2.3a shows an analog signal $y(t)$—it is defined at the continuum of times, and its amplitudes assume a continuous range of values. Such a signal cannot be stored in digital computers. The signal, therefore, must be converted to a form that will be accepted by digital computers. One very common method to do this is to record sample values of this signal at equally spaced instants. For example, we sample the signal every 10 msec, we would obtain the *discrete-time signal* sketched in Fig. 2.3b. The *sampling interval* of 10 msec corresponds to a *sampling rate* of 100 samples/sec. The choice of sampling rate is important, since it determines how accurately the discrete-time signal can represent the original signal.

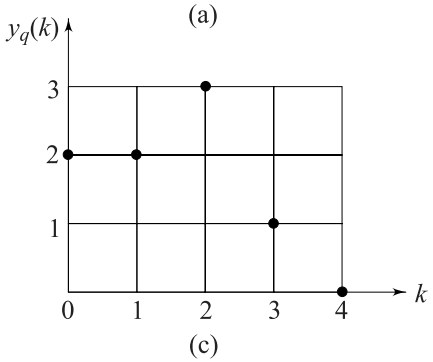
In a practical situation, the sampling rate is determined by the range of frequencies present in the original signal. Detailed analysis of uniform sampling process, and the related problem of *aliasing* will appear later in this chapter.

Notice that the time axis of the discrete-time signal in Fig. 2.3b, is labeled simply 'sample number' and index k has been used to denote this number ($k = 0, 1, 2, ...$). Corresponding to different values of sample number k, the discrete-time signal assumes the same continuous range of values assumed by the analog signal $y(t)$. We can represent the sample values by a sequence of numbers y_s (refer to Fig. 2.3b):

$$y_s = \{1.7, 2.4, 2.8, 1.4, 0.4, ...\}$$







k	Digital word
0	10
1	10
2	11
3	01
4	00

(d)

Fig. 2.3 Sampling, quantization and coding of an analog signal

In general,

$$y_s = \{y(k)\},\ 0 \le k < \infty$$

where $y(k)$ denotes the kth number in the sequence.

The sequence defined above is a *one-sided sequence*; $y_s = 0$ for $k < 0$. In digital control applications, we normally encounter one-sided sequences.

Although, strictly speaking, $y(k)$ denotes the kth number in the sequence, the notation given above is often unnecessarily cumbersome, and it is convenient and unambiguous to refer to $y(k)$ itself as a sequence.

Throughout our discussion on digital control, we will assume *uniform sampling*, i.e., sample values of the analog signal are extracted at equally spaced sampling instants. If the physical time, corresponding to the sampling interval is T seconds, then the kth sample $y(k)$, gives the value of the discrete-time signal at $t = kT$ seconds. We may, therefore, use $y(kT)$ to denote a sequence wherein the independent variable is the physical time.

The signal of Fig. 2.3b is defined at discrete instants of time. The sample values are, however, tied to a continuous range of numbers. Such a signal, in principle, can be stored in an *infinite-bit* machine because a *finite-bit* machine can store only a finite set of numbers.

Binary number	*Decimal equivalent*
00	0
01	1
10	2
11	3

A simplified hypothetical two-bit machine can store four numbers as given adjacent in the table.

The signal of Fig. 2.3b can be stored in such a machine if the sample values are quantified to four *quantization levels*. Figure 2.3c shows a quantized discrete-time signal for our hypothetical machine. We have assumed that any value in the interval [0.5, 1.5) is rounded to 1, and so forth. The signals for which both *time* and *amplitude* are discrete, are called *digital signals*.

After sampling and quantization, the final step required in converting an analog signal to a form acceptable to digital computers is *coding* (or *encoding*). The encoder maps each quantized sample value into a digital word. Figure 2.3d gives the coded digital signal, corresponding to the analog signal of Fig. 2.3a for our hypothetical two-bit machine.

The device that performs the sampling, quantization, and coding is an A/D *converter*. Figure 2.4 is a block diagram representation of the operations performed by an A/D converter.

It may be noted that the quantized discrete-time signal of Fig. 2.3c and the coded signal of Fig. 2.3d carry exactly the same information. For the purpose of analytical study of digital systems, we will use the quantized discrete-time form for digital signals.

The number of binary digits carried by a device is its *word length*, and this is obviously an important characteristic related to the *resolution* of the device—the smallest change in the input signal that will produce a change in the output signal. The A/D converter that generates signals of Fig. 2.3 has two binary digits and thus four quantization levels. Any change, therefore, in the input over the interval [0.5, 1.5) produces no change in the output. With three binary digits, 2^3 quantization levels can be obtained, and the resolution of the converter could be improved.

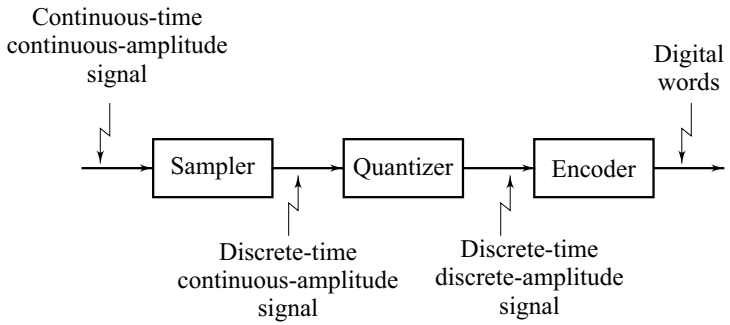


Fig. 2.4 Operations performed by an A/D converter

The A/D converters in common use have word lengths of 8 to 16 bits. For an A/D converter with a word length of 8 bits, an input signal can be resolved to one part in 2^8, or 1 in 256. If the input signal has a range of 10 V, the resolution is 10/256, or approximately 0.04 V. Thus, the input signal must change by at least 0.04 V, in order to produce a change in the output.

With the availability of converters with resolution ranging from 8 to 16 bits, the quantization errors do not pose a serious threat to computer control of industrial processes. In our treatment of the subject, we assume quantization errors to be zero. This is equivalent to assuming *infinite-bit* digital devices. Thus we treat digital signals as if they are discrete-time signals with amplitudes assuming a continuous range of values. In other words, we make no distinction between the words 'discrete-time' and 'digital.'

A typical topology of a single-loop digital control system is shown in Fig. 2.2. It has been assumed that the measuring transducer and the actuator (final control element) are analog devices, requiring respectively A/D and D/A conversion at the computer input and output. The D/A conversion is a process of producing an analog signal from a digital signal and is, in some sense, the reverse of the sampling process discussed above.

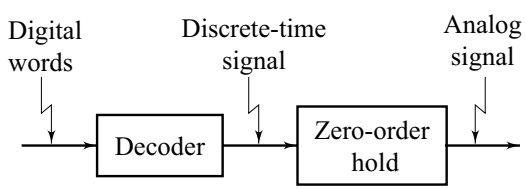


Fig. 2.5 Operations performed by a D/A converter

The D/A converter performs two functions: first, generation of output samples from the binary-form digital signals produced by the machine, and second, conversion of these samples to analog form. Figure 2.5 is a block diagram representation of the operations performed by a D/A converter. The *decoder* maps each digital word into a sample value of the signal in discrete-time form. It is usually not possible to drive a load, such as a motor, with these samples. In order to deliver sufficient energy, the sample amplitude might have to be so large that it may become infeasible to realistically generate it. Also large-amplitude signals might saturate the system being driven.

The solution to this problem is to smooth the output samples to produce a signal in analog form. The simplest way of converting a sample sequence into a continuous-time signal is to hold the value of the sample until the next one arrives. The net effect is to convert a sample to a pulse of duration T—the sample period. This function of a D/A converter is referred to as a *Zero-Order Hold* (ZOH) operation. The term *zero-order* refers to the zero-order polynomial used to extrapolate between the sampling times (detailed

discussion will appear later in this chapter). Figure 2.6 shows a typical sample sequence produced by the decoder, and the analog signal[1] resulting from the zero-order hold operation.

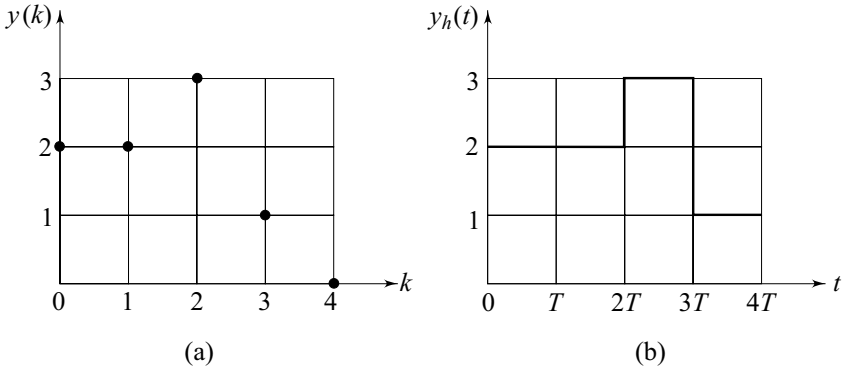


Fig. 2.6 (a) Sampled sequence (b) Analog output from ZOH

2.3.1 D/A Converter Circuits

Most D/A converters use the principle shown in the three-bit form in Fig. 2.7 to convert the HI/LO digital signals at the computer output to a single analog voltage. The circuit of Fig. 2.7 is an '*R*–2*R*' ladder; the value of *R* typically ranges from 2.5 to 10K ohms.

Suppose a binary number $b_2b_1b_0$ is given. The switch (actually, electronic gates) positions in Fig. 2.7 correspond to the digital word 100, i.e., $b_2 = 1$ and $b_1 = b_0 = 0$. The circuit can be simplified to the equivalent form shown in Fig. 2.8a. The currents in the resistor branches are easily calculated and are indicated in the circuit (for the high gain amplifier, the voltage at point *A* is practically zero [155]). The output voltage is

$$V_0 = 3R\,\frac{i_2}{2} = \frac{1}{2}\,V_{\text{ref}}$$

If $b_1 = 1$ and $b_2 = b_0 = 0$, then the equivalent circuit is as shown in Fig. 2.8b. The output voltage is

$$V_0 = 3R\,\frac{i_1}{4} = \frac{1}{4}\,V_{\text{ref}}$$

Similarly, if $b_0 = 1$ and $b_2 = b_1 = 0$, then the equivalent circuit is as shown in Fig. 2.8c. The output voltage is

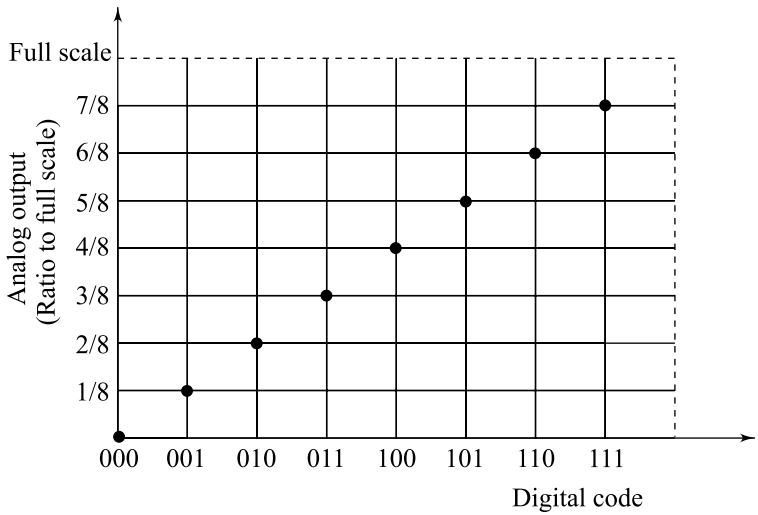
$$V_0 = 3R\,\frac{i_0}{8} = \frac{1}{8}\,V_{\text{ref}}$$

In this way, we find that when the input data is $b_2b_1b_0$ (where the b_i's are either 0 or 1), then the output voltage is

$$V_0 = (b_2 2^{-1} + b_1 2^{-2} + b_0 2^{-3})V_{FS} \tag{2.1}$$

where $V_{FS} = V_{\text{ref}}$ = full scale output voltage.

[1] In the literature, including this book, the terms 'continuous-time signal' and 'analog signal' are frequently interchanged.



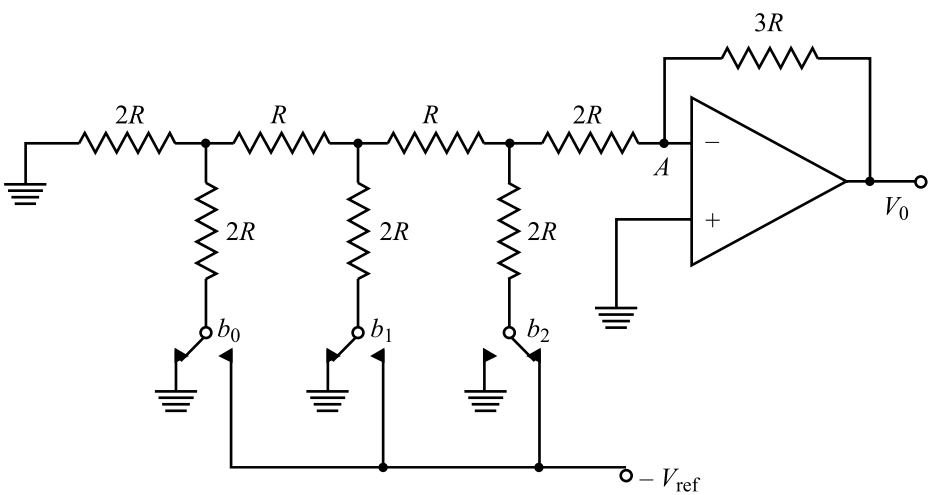


Fig. 2.7 Three-bit D/A converter

The circuit and the defining equation for an *n*-bit D/A converter easily follow from Fig. 2.7 and Eqn. (2.1), respectively.

2.3.2 A/D Converter Circuits

Most A/D converters use the principle of successive approximation. Figure 2.9 shows the organization of an A/D converter that uses this method. Its principal components are a D/A converter, a comparator, a Successive Approximation Register (SAR), a clock, and control and status logic.

On receiving the (Start-Of-Conversion) SOC command, the SAR is cleared to 0s and its most significant bit is set to 1. This results in a V_0 value that is one half of the full scale (refer to Eqn. (2.1)). The output of

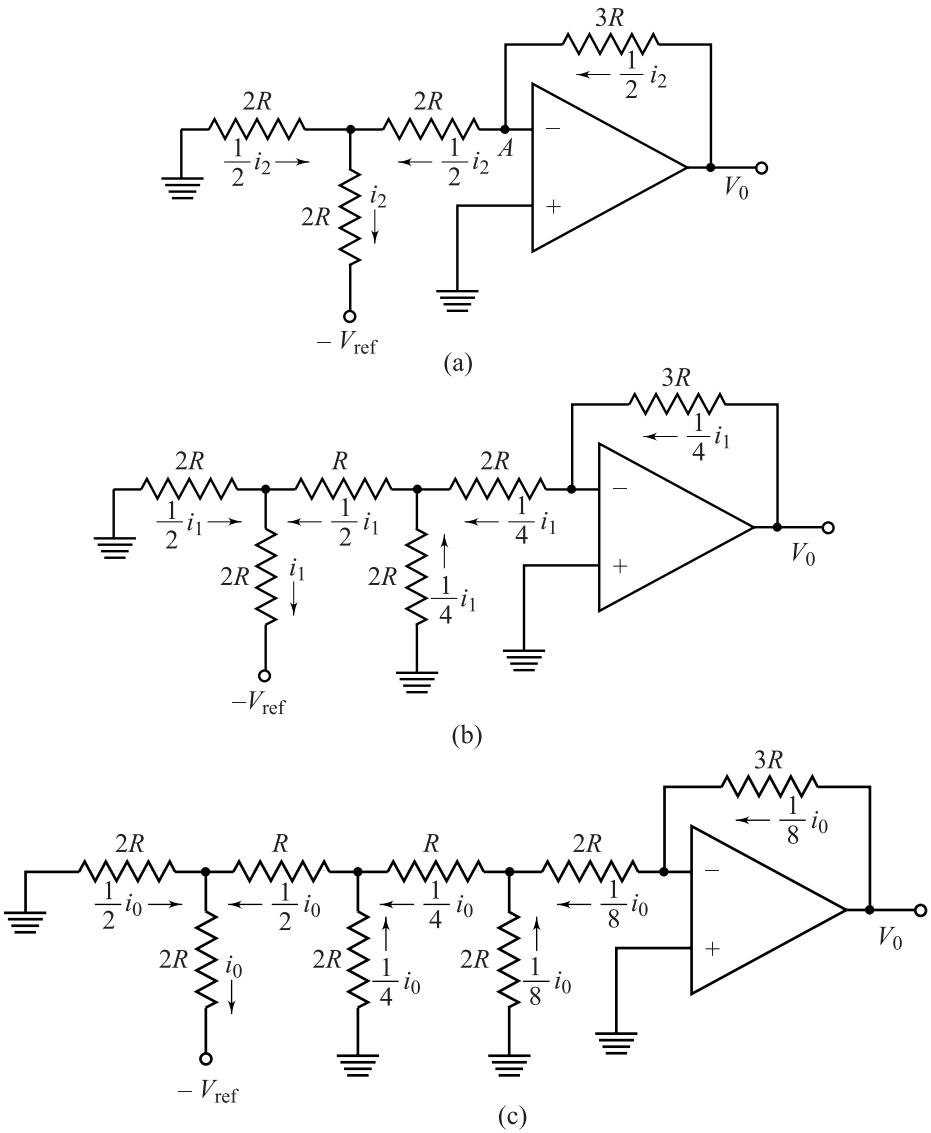


Fig. 2.8 Equivalent circuits of the D/A converter shown in Fig. 2.7: (a) $b_0 = b_1 = 0$, $b_2 = 1$ (b) $b_0 = 0$, $b_1 = 1$, $b_2 = 0$ (c) $b_0 = 1$, $b_1 = b_2 = 0$

the comparator is then tested to see whether V_{IN} is greater than or less than V_0. If V_{IN} is greater, the most significant bit is left on; otherwise it is turned off (complemented).

In the next step, the next most significant bit of the SAR is turned on. At this stage, V_0 will become either three quarters or one quarter of the full scale, depending on whether V_{IN} was, respectively, greater than or less than V_0 in the first step. Again, the comparator is tested and if V_{IN} is greater than the new V_0, the next most significant bit is left on. Otherwise it is turned off.

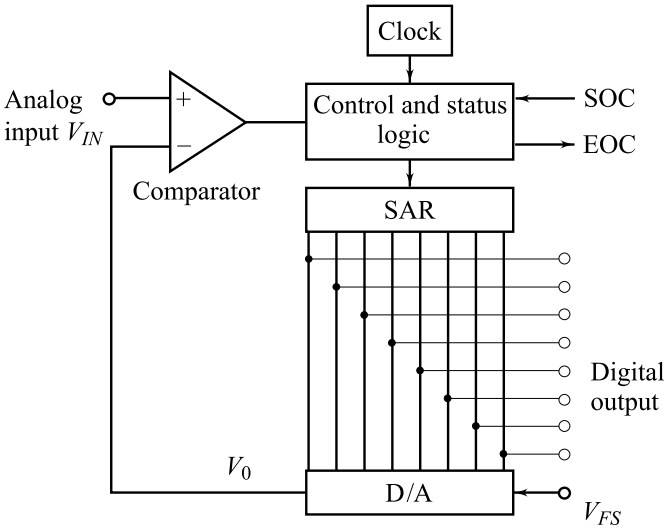


Fig. 2.9 Organization of a successive approximation A/D converter

The process is repeated for each remaining SAR bit. When the process has been carried out for each bit, the SAR contains the binary number that is proportional to V_{IN}, and the (End-Of-Conversion) EOC line indicates that comparison has been completed and digital output is available for transmission. Figure 2.10 gives the code sequence for a three-bit successive approximation.

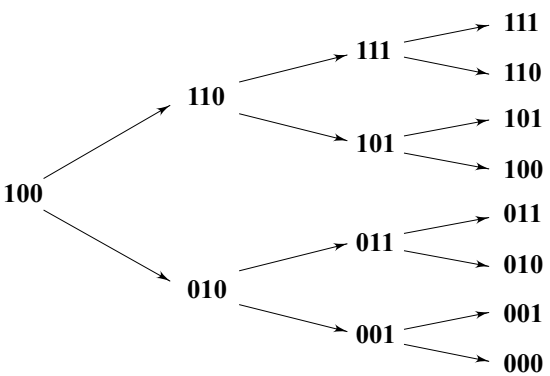


Fig. 2.10 Code sequence for a three-bit successive approximation

Typical conversion times of commercial A/D units range from 10 $n$sec to 200 μsec. The input V_{IN} in Fig. 2.9 should remain constant during the conversion time of the A/D converter. For this reason, a high performance A/D conversion system includes an S/H device which keeps the input to the A/D converter, constant during its conversion time. The S/H operation is described in Section 2.10.

2.4 BASIC DISCRETE-TIME SIGNALS

There are a number of basic discrete-time signals which play an important role in the analysis of signals and systems. These signals are direct counterparts of the basic continuous-time signals.[2] As we shall see, many characteristics of basic discrete-time signals are directly analogous to the properties of basic continuous-time signals. There are, however, several important differences in discrete-time, and we will point these out as we examine the properties of these signals.

[2] Chapter 2 of the companion book [155].

Unit-Sample Sequence

The unit sample sequence contains only one *nonzero* element and is defined by (Fig. 2.11a)

$$\delta(k) = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2a}$$

The *delayed* unit-sample sequence, denoted by $\delta(k - n)$, has its nonzero element at sample time n (Fig. 2.11b):

$$\delta(k-n) = \begin{cases} 1 & \text{for } k = n \\ 0 & \text{otherwise} \end{cases} \tag{2.2b}$$

One of the important aspects of the unit-sample sequence is that an arbitrary sequence can be represented as a sum of scaled, delayed unit samples. For example, the sequence $r(k)$ in Fig. 2.11c can be expressed as

$$\begin{aligned} r(k) &= r(0)\,\delta(k) + r(1)\,\delta(k-1) + r(2)\,\delta(k-2) + \cdots \\ &= \sum_{n=0}^{\infty} r(n)\,\delta(k-n) \end{aligned} \tag{2.3}$$

$r(0), r(1), \ldots$, are the sample values of the sequence $r(k)$. This representation of a discrete-time signal is found useful in the analysis of linear systems through the principle of superposition.

As we will see, the unit-sample sequence plays the same role for discrete-time signals and systems, that the unit-impulse function does for continuous-time signals and systems. For this reason, the unit-sample sequence is often referred to as the *discrete-time impulse*. It is important to note that a discrete-time impulse does not suffer from the same mathematical complexity as a continuous-time impulse. Its definition is simple and precise.

Unit-Step Sequence

The unit-step sequence is defined as[3] (Fig. 2.11d)

$$\mu(k) = \begin{cases} 1 & \text{for } k \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

The delayed unit-step sequence, denoted by $\mu(k - n)$, has its first nonzero element at sample time n (Fig. 2.11e):

$$\mu(k-n) = \begin{cases} 1 & \text{for } k \geq n \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

An arbitrary discrete-time signal $r(k)$ switched on to a system at $k = 0$ is represented as $r(k)\mu(k)$.

Sinusoidal Sequence

A one-sided sinusoidal sequence has the general form (Fig. 2.11f)

$$r(k) = A\cos(\Omega k + \phi)\,\mu(k) \tag{2.6}$$

[3] In discrete-time system theory, the unit-step sequence is generally denoted by $u(k)$. In control theory, $u(k)$ is used to represent the control signal. In this book, $\mu(k)$ has been used to represent the unit-step sequence while $u(k)$ denotes the control signal.

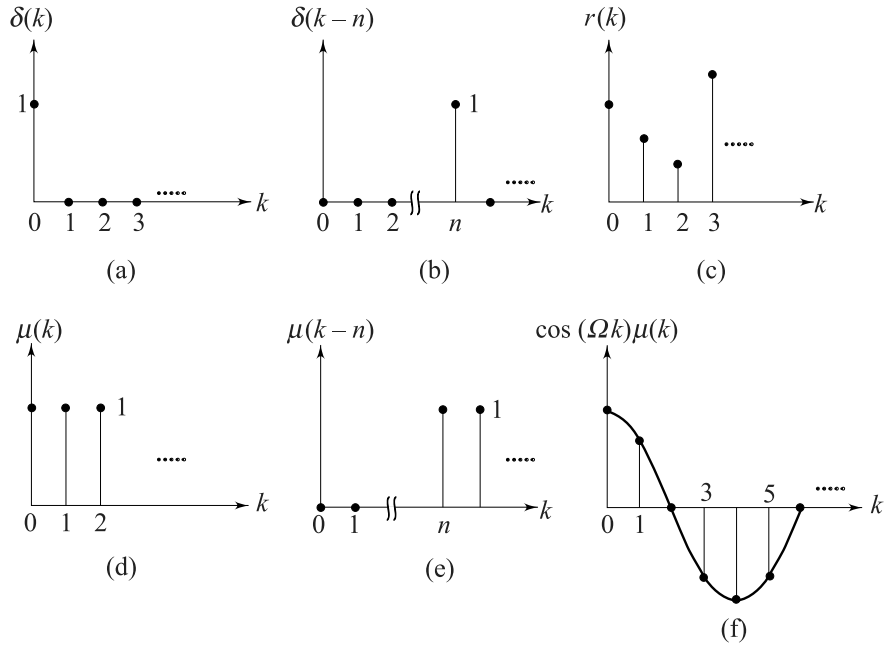


Fig. 2.11 Basic discrete-time signals

The quantity Ω is called the *frequency* of the discrete-time sinusoid and ϕ is called the *phase*. Since k is a *dimensionless* integer, the dimension of Ω must be *radians* (we may specify the units of Ω to be radians/sample, and units of k to be samples).

The fact that k is always an integer in Eqn. (2.6) leads to some differences between the properties of discrete-time and continuous-time sinusoidal signals. An important difference lies in the *range of values* the frequency variable can take on. We know that for the continuous-time signal $r(t) = A \cos \omega t = \text{real}\ \{Ae^{j\omega t}\}$, ω can take on values in the range $(-\infty, \infty)$. In contrast, for the discrete-time sinusoid $r(k) = A \cos \Omega k = \text{real}\ \{Ae^{j\Omega k}\}$, Ω can take on values in the range $[-\pi, \pi]$.

To illustrate the property of discrete-time sinusoids, consider $\Omega = \pi + x$, where x is a small number compared with π. Since

$$e^{j\Omega k} = e^{j(\pi + x)k} = e^{j(2\pi - \pi + x)k} = e^{j(-\pi + x)k}$$

a frequency of $(\pi + x)$ results in a sinusoid of frequency $(-\pi + x)$. Suppose now, that Ω is increased to 2π. Since $e^{j2\pi k} = e^{j0}$, the observed frequency is 0. Thus, the observed frequency is always between $-\pi$ and π, and is obtained by adding (or subtracting) multiples of 2π to Ω until a number in that range is obtained.

The highest frequency that can be represented by a digital signal is, therefore, π radians/sample interval. The implications of this property for sequences obtained by sampling sinusoids and other signals is discussed in Section 2.11.

2.5 TIME-DOMAIN MODELS FOR DISCRETE-TIME SYSTEMS

A discrete-time system is defined mathematically, as a transformation, or an operator, that maps an *input* sequence $r(k)$ into an *output* sequence $y(k)$. Classes of discrete-time systems are defined by placing constraints on the transformation. As they are relatively easy to characterize mathematically, and as they can be designed to perform useful signal processing functions, the class of *linear time-invariant* systems will be studied in this book. In the control structure of Fig. 2.2, the digital computer transforms an input sequence into a form which is in some sense more desirable. Therefore, the discrete-time systems we consider here are, in fact, *computer programs*.

As we shall see, there is a similarity in the structure of models of continuous-time and discrete-time systems. This has resulted in the development of similar methods of analysis. For example, the simulation diagrams of discrete-time systems are similar to those for continuous-time systems, with only the dynamic element changed from an *integrator* to a *delayer*. The *convolution summation* is similar to *convolution integral*, and the *z-transform method*, tailored especially for linear discrete-time systems, bears many similarities to the *Laplace transform*. There are differences also between the properties of discrete-time and continuous-time systems. In this chapter, we are concerned with the analysis tools for discrete-time systems. Similarities with the tools for continuous-time systems will be obvious. The differences are pointed out specifically.

For linear time-invariant discrete-time systems, four different ways of mathematical representation are discussed. *Time-domain models* are described in this section, and a *transform-domain model* is given in Section 2.7.

2.5.1 State Variable Models

Consider a simple computer program expressed in MATLAB:

$$\begin{aligned} & y(1) = 0 \\ \text{for} \quad & i = 2, N \\ & y(i) = r(i-1) + 0.1 * y(i-1) \\ \text{end} & \end{aligned} \tag{2.7}$$

where $r(i)$ is the ith sample of the input sequence, $y(i)$ is the ith sample of the output sequence, and N is the total length of the signal record. We must define the value $y(1)$ in order to start signal processing. This value is the *initial condition* of the signal processor. For the signal processor, represented by the computer program (2.7), the initial condition has been taken as zero.

In the computer program (2.7), the initial condition is represented by $y(1)$ and not by $y(0)$ because MATLAB does not allow *arrays* to be indexed starting with zero. For the analytical study of discrete-time systems, starting a sequence $y(k)$ with $y(0)$ is more convenient.

Simulation Diagrams

It is obvious that the computer program (2.7) is characterized by the three basic operations:

(i) multiplication of a machine variable by a constant,
(ii) addition of several machine variables, and
(iii) storage of past values of machine variables.

These basic operations are diagrammatically represented in Fig. 2.12. The *unit delayer* represents a means for storing previous values of a sequence. If the signal $x_1(k)$; $k \geq 0$ is the input to the unit delayer, its output sequence $x_2(k)$ has the sample values:

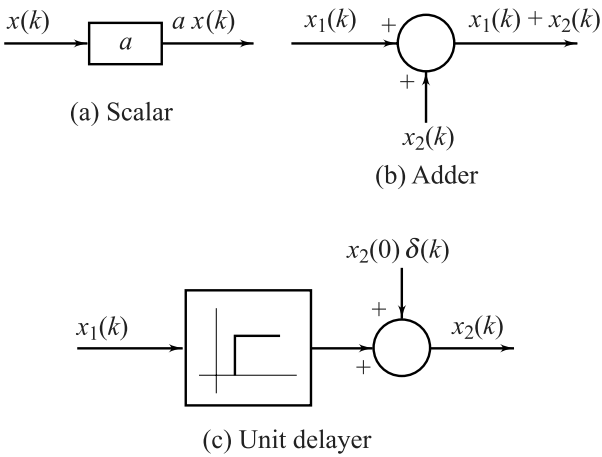


Fig. 2.12 Basic building blocks of linear time-invariant discrete-time systems

$$
\begin{aligned}
x_2(0) &= \text{specified initial condition} \\
x_2(1) &= x_1(0) \\
x_2(2) &= x_1(1) \\
&\vdots
\end{aligned}
$$

A specified initial condition is stored before the commencement of the algorithm, in the appropriate register (of the digital computer) containing $x_2(\cdot)$. This can be diagrammatically represented by adding a signal $x_2(0)\delta(k)$ to the output of the delayer, where $\delta(k)$ is the unit-sample sequence defined by Eqn. (2.2a).

The signal processing function performed by the computer program (2.7) can be represented by a block diagram shown in Fig. 2.13. Various blocks in this figure represent the basic computing operations of a digital computer. The unit delayer is the only *dynamic element* involved. The signal processing configuration of Fig. 2.13, thus, represents a *first-order* discrete-time system.

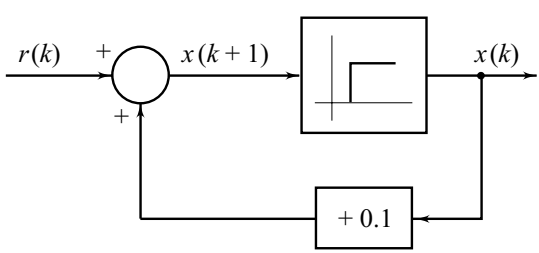


Fig. 2.13 A first-order linear discrete-time system

The output $x(k)$ of the dynamic element gives the *state* of the system at any k. If the signal $r(k)$ is switched on to the system at $k = 0$ ($r(k) = 0$ for $k < 0$), the sample value $x(0)$ of the output sequence $x(k)$, represents the initial state of the system. Since the initial state in the computer program (2.7) is zero, a signal of the form $x(0)\,\delta(k)$ does not appear in Fig. 2.13.

The defining equation for the computer program (2.7), obtained by forming an equation of the *summing junction* in Fig. 2.13, is

$$x(k+1) = 0.1\,x(k) + r(k);\;\; x(0) = 0 \tag{2.8}$$

The solution of this *first-order linear difference equation* for given input $r(k)$ applied at $k = 0$, and given initial state $x(0)$, yields the state $x(k)$; $k > 0$. Equation (2.8) is thus the *state equation* of the discrete-time system of Fig. 2.13. Conversely, Fig. 2.13 is the *simulation diagram* for the mathematical model (2.8).

To solve an equation of the form (2.8) is an elementary matter. If k is incremented to take on values $k = 0, 1, 2, ...$, etc., the state $x(k)$; $k = 1, 2, ...$, can easily be generated by an iterative procedure. The iterative method, however, generates only a sequence of numbers and not a *closed-form solution*.

Example 2.1

In order to introduce discrete-time systems, we study the signal processing algorithm given by the difference equation:

$$x(k+1) = -\alpha\, x(k) + r(k);\ \ x(0) = 0 \tag{2.9}$$

where α is a real constant.

We shall obtain a closed-form solution of this equation by using a so-called brute force method (z-transform method of solving linear difference equations is given in Section 2.7). When solved repetitively, Eqn. (2.9) yields

$$\begin{aligned} x(0) &= 0;\ x(1) = r(0) \\ x(2) &= -\alpha\, r(0) + r(1);\ x(3) = (-\alpha)^2\, r(0) - \alpha\, r(1) + r(2) \\ &\ \ \vdots \end{aligned}$$

The general term becomes ($r(k) = 0$ for $k < 0$),

$$x(k) = (-\alpha)^{k-1}\, r(0) + (-\alpha)^{k-2}\, r(1) + \cdots + r(k-1) \tag{2.10}$$

Examining this equation, we note that the response $x(k)$ is a linear combination of the input samples $r(0)$, $r(1)$, ..., $r(k-1)$, and there appears to be a definite structure of the various weights.

The response of linear discrete-time systems to an impulse input $\delta(k)$ (defined in Eqn. (2.2a)) will be of special interest to us. Let us denote this response, called the *impulse response*, by $g(k)$.

For the system described by Eqn. (2.9), the impulse response obtained from Eqn. (2.10) is given by

$$g(k) = \begin{cases} 0 & \text{for } k = 0 \\ (-\alpha)^{k-1} & \text{for } k \geq 1 \end{cases} \tag{2.11}$$

The question of whether or not the solution decays, is more closely related to the *magnitude* of α than to its *sign*. In particular, for $|\alpha| > 1$, $g(k)$ grows with increasing k while it decays when $|\alpha| < 1$. The nature of time functions of the form (2.11) for different values of α is examined in Section 2.9.

A discrete-time system is completely characterized by the output variables of independent dynamic elements of the system. The outputs of independent dynamic elements, thus, constitute a set of *characterizing variables* of the system. The values of the characterizing variables at instant k describe the *state* of the system at that instant. These variables are, therefore, the *state variables* of the system.

The discrete-time system shown in Fig. 2.14 has two dynamic elements; the outputs $x_1(k)$ and $x_2(k)$ of these elements are, therefore, the state variables of the system. The following dynamical equations for

the state variables easily follow from Fig. 2.14:

$$x_1(k+1) = x_2(k);\ x_1(0) = x_1^0 \tag{2.12a}$$
$$x_2(k+1) = \alpha_1\, x_1(k) + \alpha_2\, x_2(k) + r(k);\ x_2(0) = x_2^0$$

The solution of these equations for a given input $r(k)$ applied at $k = 0$, and given initial state $\{x_1^0, x_2^0\}$, yields the state $\{x_1(k), x_2(k)\}$, $k > 0$.

If $y(k)$ shown in Fig. 2.14 is the desired output information, we have the following algebraic relation to obtain $y(k)$:

$$y(k) = c_1\, x_1(k) + c_2\, x_2(k) \tag{2.12b}$$

Equations (2.12a) are the *state equations*, and Eqn. (2.12b) is the *output equation* of the discrete-time system of Fig. 2.14.

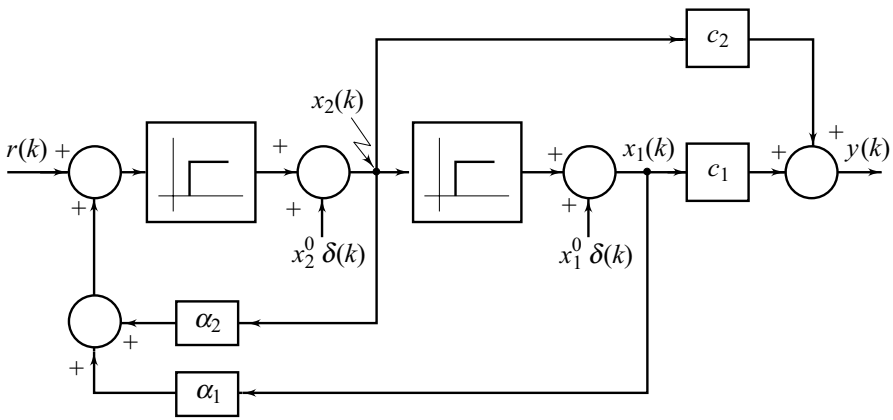


Fig. 2.14 A second-order discrete-time system

In general, the state variable formulation may be visualized in block diagram form as shown in Fig. 2.15. We have depicted a *Multi-Input*, *Multi-Output* (MIMO) system which has p inputs, q outputs, and n state variables; the different variables are represented by the *input vector* $\mathbf{r}(k)$, the *output vector* $\mathbf{y}(k)$ and the *state vector* $\mathbf{x}(k)$, where

$$\mathbf{r}(k) \triangleq \begin{bmatrix} r_1(k) \\ r_2(k) \\ \vdots \\ r_p(k) \end{bmatrix};\ \mathbf{y}(k) \triangleq \begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_q(k) \end{bmatrix};\ \mathbf{x}(k) \triangleq \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix}$$

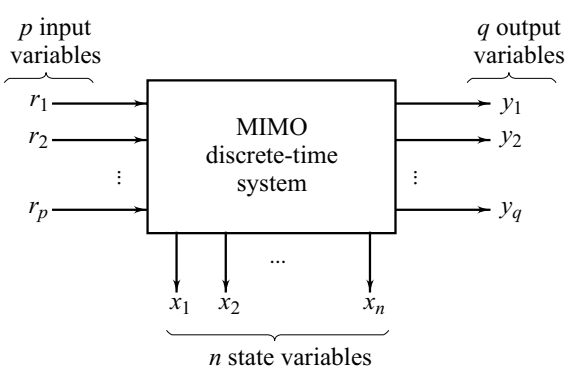


Fig. 2.15 Structure of a general discrete-time system

Assuming that the input is switched on to the system at $k = 0$ ($\mathrm{r}(k) = \mathbf{0}$ for $k < 0$), the initial state is given by

$$\mathbf{x}(0) \triangleq \mathbf{x}^0, \text{ a specified } n \times 1 \text{ vector}$$

The dimension of the state vector defines the *order* of the system. The dynamics of an *n*th-order linear time-invariant system are described by equations of the form

$$\begin{aligned} x_1(k+1) &= f_{11}\,x_1(k) + f_{12}\,x_2(k) + \cdots + f_{1n}\,x_n(k) + g_{11}\,r_1(k) \\ &\quad + g_{12}\,r_2(k) + \cdots + g_{1p}\,r_p(k) \\ x_2(k+1) &= f_{21}\,x_1(k) + f_{22}\,x_2(k) + \cdots + f_{2n}\,x_n(k) + g_{21}\,r_1(k) \\ &\quad + g_{22}\,r_2(k) + \cdots + g_{2p}\,r_p(k) \\ &\vdots \\ x_n(k+1) &= f_{n1}\,x_1(k) + f_{n2}\,x_2(k) + \cdots + f_{nn}\,x_n(k) + g_{n1}\,r_1(k) \\ &\quad + g_{n2}\,r_2(k) + \cdots + g_{np}\,r_p(k) \end{aligned} \tag{2.13}$$

where the coefficients f_{ij} and g_{ij} are constants.

In the *vector-matrix* form, Eqns (2.13) may be written as

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{r}(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{2.14}$$

where

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{bmatrix} \quad \text{and} \quad \mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1p} \\ g_{21} & g_{22} & \cdots & g_{2p} \\ \vdots & \vdots & & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{np} \end{bmatrix}$$

are, respectively, $n \times n$ and $n \times p$ constant matrices. Equation (2.14) is called the *state equation* of the system.

The output variables at $t = kT$ are linear combinations of the values of the state variables and input variables at that time, i.e.,

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{r}(k) \tag{2.15}$$

where

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & & \vdots \\ c_{q1} & c_{q2} & \cdots & c_{qn} \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1p} \\ d_{21} & d_{22} & \cdots & d_{2p} \\ \vdots & \vdots & & \vdots \\ d_{q1} & d_{q2} & \cdots & d_{qp} \end{bmatrix}$$

are, respectively, $q \times n$ and $q \times p$ constant matrices. Equation (2.15) is called the *output equation* of the system.

The state equation (2.14) and the output equation (2.15) together give the *state variable model* of the MIMO system[4]:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{r}(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{2.16a}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{r}(k) \tag{2.16b}$$

For single-input ($p = 1$) and single-output ($q = 1$) system, the state variable model takes the form

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}r(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{2.17a}$$

$$y(k) = \mathbf{c}\mathbf{x}(k) + dr(k) \tag{2.17b}$$

[4] We have used lower case bold letters to represent vectors and upper case bold letters to represent matrices.

where **g** is $n \times 1$ column vector, **c** is $1 \times n$ row vector and d is a scalar:

$$\mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}; \mathbf{c} = [c_1\ c_2 \cdots c_n]$$

Example 2.2

The discrete-time system of Fig. 2.16 has one dynamic element (unit delayer); it is, therefore, a first-order system. The state of the system at any k is described by $x(k)$—the output of the dynamic element.

The equation of the input summing junction is

$$x(k+1) = 0.95\, x(k) + r(k);\ x(0) = 0 \tag{2.18a}$$

This is the state equation of the first-order system.

The output $y(k)$ is given by the following output equation:

$$y(k) = 0.0475\, x(k) + 0.05\, r(k) \tag{2.18b}$$

Equations (2.18a) and (2.18b) together constitute the state variable model of the first-order system.

Let us study the response of the system of Fig. 2.16 to the *unit-step sequence,*

$$\mu(k) = \begin{cases} 1 & \text{for } k \geq 0 \\ 0 & \text{for } k < 0 \end{cases} \tag{2.19a}$$

and the *unit-alternating sequence,*

$$r(k) = \begin{cases} (-1)^k & \text{for } k \geq 0 \\ 0 & \text{for } k < 0 \end{cases} \tag{2.19b}$$

We will first solve Eqn. (2.18a) for $x(k)$ and then use Eqn. (2.18b) to obtain $y(k)$.

The solution of Eqn. (2.18a) directly follows from Eqn. (2.10):

$$\begin{aligned} x(k) &= (0.95)^{k-1}\, r(0) + (0.95)^{k-2}\, r(1) + \cdots + r(k-1) \\ &= \sum_{i=0}^{k-1} (0.95)^{k-1-i}\, r(i) \end{aligned} \tag{2.20}$$

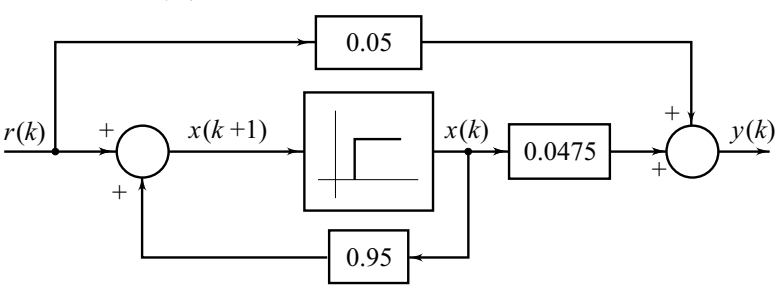


Fig. 2.16 A first-order linear discrete-time system

For the unit-step input given by Eqns (2.19a), we have[5]

$$x(k) = \sum_{i=0}^{k-1} (0.95)^{k-1-i} = (0.95)^{k-1} \sum_{i=0}^{k-1} \left(\frac{1}{0.95}\right)^i$$

$$= (0.95)^{k-1} \left[\frac{1-\left(\frac{1}{0.95}\right)^k}{1-\frac{1}{0.95}}\right] = \frac{1}{0.05} [1 - (0.95)^k]$$

The output

$$y_1(k) = \frac{0.0475}{0.05} [1 - (0.95)^k] + 0.05; k \geq 0$$

$$= 1 - (0.95)^{k+1}; k \geq 0 \tag{2.21}$$

Consider now the system excited by the unit-alternating input given by Eqn. (2.19b). It follows from Eqn. (2.20) that for this input, the state

$$x(k) = \sum_{i=0}^{k-1} (0.95)^{k-1-i} (-1)^i = \frac{1}{1.95} [(0.95)^k - (-1)^k]$$

The output

$$y_2(k) = 0.0475\, x(k) + 0.05\, (-1)^k$$

$$= \frac{0.05}{1.95} [(-1)^k + (0.95)^{k+1}]; k \geq 0 \tag{2.22}$$

From Eqns (2.21) and (2.22), we observe that the steady-state values of $y_1(k)$ and $y_2(k)$ are

$$y_1(k) = 1 \text{ for large } k; y_2(k) = \frac{1}{39} (-1)^k \text{ for large } k$$

Thus, the discrete-time system of Fig. 2.16 readily transmits a *unit step* and rejects a *unit-alternating input* (reduces its magnitude by a factor of 39). Since the unit-alternating signal is a rapidly fluctuating sequence of numbers, while the unit step can be viewed as a slowly fluctuating signal, the discrete-time system of Fig. 2.16 represents a *low-pass digital filter*. In Example 2.11, we will study the frequency-domain characteristics of this filter.

2.5.2 Difference Equation Models

Consider the *single-input*, *single-output* (SISO) system represented by the state model (2.17). The system has two types of inputs; the *external* input $r(k)$, and the *initial state* $\mathbf{x}(0)$ representing initial storage in the appropriate registers (of the digital computer) containing $x_i(\cdot)$.

If the dynamic evolution of the state $\mathbf{x}(k)$ is not required, i.e., we are interested only in the input-output relation for $k \geq 0$, a linear time-invariant discrete-time system composed of n dynamic elements can be

[5] $\sum_{j=0}^{k} a^j = \frac{1-a^{k+1}}{1-a}; a \neq 1$

analyzed using a single *n*th-order difference equation as its model. A general form of *n*th-order linear difference equation relating output $y(k)$ to input $r(k)$ is given below.

$$y(k) + a_1 y(k-1) + \cdots + a_n y(k-n) = b_0 r(k) + b_1 r(k-1) + \cdots + b_m r(k-m)$$

The coefficients a_i and b_j are real constants; m and n are integers with $m \leq n$.

We will consider the general linear difference equation in the following form:

$$y(k) + a_1 y(k-1) + \cdots + a_n y(k-n) = b_0 r(k) + b_1 r(k-1) + \cdots + b_n r(k-n) \tag{2.23}$$

There is no loss of generality in this assumption; the results for $m = n$ can be used for the case of $m < n$ by setting appropriate b_j coefficients to zero.

If the input is assumed to be switched on at $k = 0$ ($r(k) = 0$ for $k < 0$), then the difference equation model (2.23) gives the output at instant '0' in terms of the past values of the output; $y(-1), y(-2), ..., y(-n)$, and the present input $r(0)$. Thus the initial conditions of the model (2.23) are $\{y(-1), y(-2), ..., y(-n)\}$.

Since the difference equation model (2.23) represents a time-invariant system, the choice of the initial point on the time scale is simply a matter of convenience in analysis. Shifting the origin from $k = 0$ to $k = n$, we get the equivalent difference equation model:

$$y(k+n) + a_1 y(k+n-1) + \cdots + a_n y(k) = b_0 r(k+n) + b_1 r(k+n-1) + \cdots + b_0 r(k) \tag{2.24}$$

Substituting $k = 0$ in Eqn. (2.24), we observe that the output at instant 'n' is expressed in terms of n values of the past outputs: $y(0), y(1), ..., y(n-1)$, and in terms of inputs: $r(0), r(1), ..., r(n)$. If k is incremented to take on values $k = 0, 1, 2, ...$, etc., the $y(k)$; $k = n, n+1, ...$, can easily be generated by the iterative procedure. Given $\{y(-1), y(-2), ..., y(-n)\}$, the initial conditions $\{y(0), y(1), ..., y(n-1)\}$ of the model (2.24) can be determined by successively substituting $k = -n, -n+1, ..., -2, -1$ in Eqn. (2.24).

In this book, we have not accommodated the classical methods of solution of linear difference equations of the form (2.23) for given initial conditions and/or external inputs. Our approach is to transform the model (2.23) to other forms which are more convenient for analysis and design of digital control systems. Our emphasis is on the *state variable models* and *transfer functions*.

In Chapter 6, we will present methods of conversion of difference equation models of the form (2.23), to state variable models. We will use state variable models to obtain the system response to given initial conditions and external inputs, to construct digital computer simulation diagrams, and to design digital control algorithms using modern methods of design.

Later in this chapter, the *z-transform technique* for transforming difference equation model (2.23) to transfer function form has been presented. Here we will use transfer function models to study input-output behavior of discrete-time systems; and to design digital control algorithms using classical methods of design.

2.5.3 Impulse Response Models

Consider the SISO system represented by the state model (2.17) or the difference equation model (2.23). The system has two types of inputs: the external input $r(k)$; $k \geq 0$, and initial state $\mathbf{x}(0)$.

A system is said to be *relaxed* at $k = 0$ if the initial state $\mathbf{x}(0) = \mathbf{0}$. In terms of the representation (2.23), a system is relaxed if $y(k) = 0$ for $k < 0$.

We have earlier seen in Eqn. (2.3) that an arbitrary sequence $r(k)$ can be represented as a sum of scaled, delayed impulse sequences. It follows from this result, that a linear time-invariant, initially relaxed system can be completely characterized by its impulse response. This can be easily established.

Let $g(k)$ be the response of an initially relaxed, linear time-invariant discrete-time system to an impulse $\delta(k)$. Due to *time-invariance property*, the response to $\delta(k-n)$ will be $g(k-n)$. By *linearity property*, the response to an input signal $r(k)$ given by Eqn. (2.3) will be

$$\begin{aligned} y(k) &= r(0)\, g(k) + r(1)\, g(k-1) + r(2)\, g(k-2) + \cdots \\ &= \sum_{j=0}^{\infty} r(j)\, g(k-j);\ k \geq 0 \end{aligned} \tag{2.25}$$

As a consequence of Eqn. (2.25), a linear time-invariant system is completely characterized by its impulse response $g(k)$ in the sense that given $g(k)$, it is possible to use Eqn. (2.25) to compute the output to any input $r(k)$. Equation (2.25) is commonly called the *convolution sum*.

It should be pointed out that the summation in Eqn. (2.25) is not really infinite in a practical situation, since for *causal systems*, $g(k-j)$ resulting from the input $\delta(k-j)$ is zero for $k < j$. This is because a causal system cannot respond until the input is applied. Thus, for a causal system with input $r(k)\mu(k)$, Eqn. (2.25) modifies to

$$y(k) = \sum_{j=0}^{k} r(j)\, g(k-j);\ k \geq 0 \tag{2.26}$$

Another important observation concerns the *symmetry* of the situation. If we let $k - j = m$ in Eqn. (2.26), we get

$$y(k) = \sum_{m=k}^{0} r(k-m)\, g(m)$$

Reversing the order of summation,

$$y(k) = \sum_{m=0}^{k} g(m)\, r(k-m) \tag{2.27}$$

The symmetry shows that we may reverse the roles of $r(\cdot)$ and $g(\cdot)$ in the convolution formula.

We may remind the reader here, that whenever impulse response models are used to describe a system, the system is always implicitly assumed to be linear, time-invariant, and initially relaxed.

We now *transform* $r(k)$ and $g(k)$ using the mapping

$$f(k) \rightarrow F(z)$$

where

$$F(z) \triangleq \sum_{k=0}^{\infty} f(k) z^{-k};\ z \text{ is a complex variable} \tag{2.28}$$

The application of this mapping to Eqn. (2.25) yields

$$Y(z) = \sum_{k=0}^{\infty} y(k) z^{-k} = \sum_{k=0}^{\infty} \left[\sum_{j=0}^{\infty} r(j)\, g(k-j) \right] z^{-k}$$

Changing the order of summations gives

$$Y(z) = \sum_{j=0}^{\infty} r(j)z^{-j} \sum_{k=0}^{\infty} g(k-j)z^{-(k-j)}$$

Since $g(k-j) = 0$ for $k < j$, we can start the second summation at $k = j$. Then, defining the index $m = k - j$, we can write

$$\begin{aligned} Y(z) &= \sum_{j=0}^{\infty} r(j)z^{-j} \sum_{m=0}^{\infty} g(m)z^{-m} \\ &= R(z)\, G(z) \end{aligned} \tag{2.29}$$

where

$$R(z) \triangleq \sum_{k=0}^{\infty} r(k)z^{-k} \quad \text{and} \quad G(z) \triangleq \sum_{k=0}^{\infty} g(k)z^{-k}$$

We see that by applying the mapping (2.28), a convolution sum is transformed into an algebraic equation. The mapping (2.28) is, in fact, the definition of *z-transform*.

The use of *z*-transform technique for the analysis of discrete-time systems runs parallel to that of Laplace transform technique for continuous-time systems. The brief introduction to the theory of *z*-transform given in this chapter provides the working tools adequate for the purposes of this text.

2.6 THE *z*-TRANSFORM

There are basically two ways to approach the *z*-transform. One way is to think in terms of systems that are *intrinsically discrete*. Signal processing by a digital computer, as we have seen in the previous section, is an example of such systems. In fact, intrinsically discrete-time systems arise in a number of ways. A model of the growth of cancer is discrete, because the cancer cells divide at discrete points in time. A macroeconomic model is usually discrete, because most economic data is usually reported monthly and quarterly. Representing the discrete instants of time by the integer variable k ($k = 0, 1, 2, ...$), we denote the output of a SISO system by the sequence $y(k)$; $k \geq 0$, and the input by the sequence $r(k)$; $k \geq 0$.

The alternative approach to the *z*-transform is in terms of *sampled-data systems*. This is the approach we will adopt because it best fits the problem we intend to solve, namely, the control of *continuous-time systems* by a *digital signal processor* (refer to Fig. 2.2). Sampling a continuous-time signal defines the discrete instants of time. Interestingly, we will see that the *z*-transform (2.28) defined for analyzing systems that are intrinsically discrete, is equally useful for sampled-data systems.

2.6.1 The Ideal Sampler

Consider an analog signal $x_a(t)$; $t \geq 0$. By substituting $t = kT$; $k = 0, 1, 2, ...$, a sequence $x_a(kT)$ is said to be derived from $x_a(t)$ by *periodic sampling*, and T is called the fixed *sampling period*. The reciprocal of T is called the *sampling frequency* or *sampling rate*. In a typical digital control scheme (refer to Fig. 2.2), the operation of deriving a sequence from a continuous-time signal is performed by an analog-to-digital (A/D) converter. A simple ideal sampler representation of the sampling operation is shown in Fig. 2.17.

The ideal sampler consists of a switch that closes and reopens instantaneously. For our purposes, time will be in seconds, although this is not always the case. In chemical processes, for instance, the unit of time could very well be minutes.

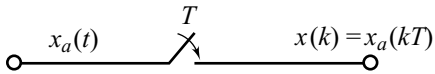


Fig. 2.17 Ideal-sampler representation of the sampling operation

Of course, no switch opens and closes instantaneously, but the modern A/D converter comes very close. A/D converters with a conversion rate of 100 kHz are fairly inexpensive, and conversion rates in the megahertz range are available. In control applications, the sampling rate is usually less than 100 Hz. If the conversion rate is 100 kHz, the conversion is completed in less than *one thousandth* of the typical sample period. Thus the conversion rate of A/D converter is close enough to instantaneous.

For intrinsically discrete-time systems, the sequence $x(k)$ represents the values the variable x takes at discrete instants $k = 0, 1, 2, \ldots$. When the sampling operation is involved, the sequence $x_a(kT)$ represents the values of continuous-time signal $x_a(t)$ derived at $t = kT$; $k = 0, 1, 2, \ldots$; and T is a fixed sampling period. Since T remains fixed, there is no loss in information if variable $x(k)$ is used to represent $x_a(kT)$; the advantage is in terms of *notational* convenience. We will follow this notation.

To establish a relationship of the sequence $x(k)$ to the continuous-time function $x_a(t)$ from which this sequence is derived, we take the following approach. We treat each sample of the sequence $x(k)$ as an *impulse function* of strength equal to the value of the sample (Impulse function $A\delta(t - t_0)$ is an impulse of strength A occurring at $t = t_0$). The idea is to give a *mathematical* description to periodic samples of a continuous-time function in such a way, that we can analyze the samples and the function simultaneously, using the same tool (Laplace transform). The sequence $x(k)$ can be viewed as a train of impulses represented by the continuous-time function $x^*(t)$:

$$x^*(t) = x(0)\,\delta(t) + x(1)\,\delta(t - T) + x(2)\,\delta(t - 2T) + \cdots$$

$$= \sum_{k=0}^{\infty} x(k)\,\delta(t - kT) \tag{2.30}$$

Typical signals $x_a(t)$, $x(k)$ and $x^*(t)$ are shown in Fig. 2.18. The sampler of Fig. 2.17 can thus be viewed as an 'impulse modulator' with the *carrier signal,*

$$\delta_T(t) = \sum_{k=0}^{\infty} \delta(t - kT) \tag{2.31}$$

and *modulating signal* $x_a(t)$. The modulation process is schematically represented in Fig. 2.19a, and the *impulse train* $\delta_T(t)$ in Fig. 2.19b:

$$x^*(t) = x_a(t)\,\delta_T(t) \tag{2.32a}$$

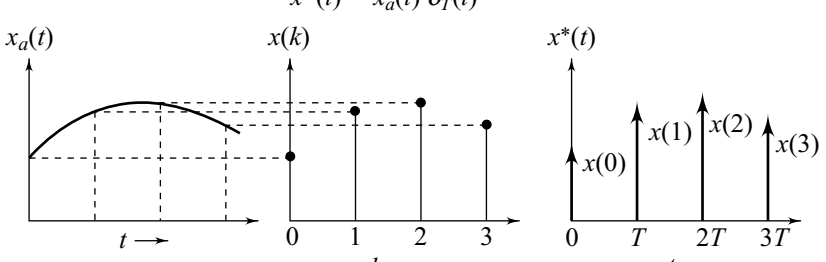
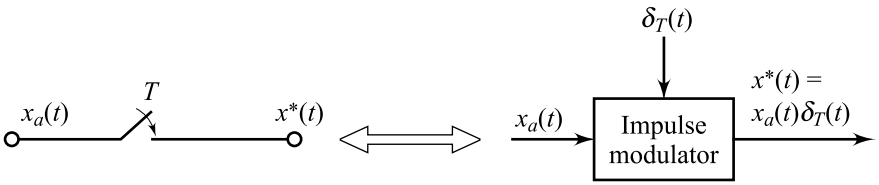
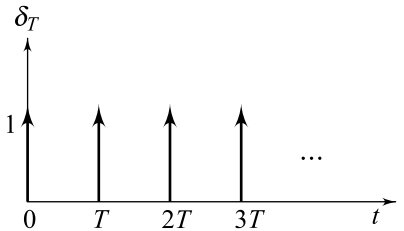


Fig. 2.18 Conversion of a continuous-time signal to a sequence



(a) Representation of sampling as impulse modulator



(b) Representation of impulse train

Fig. 2.19

We will eliminate the impulse function by simply taking the Laplace transform of $x^*(t)$ to obtain (refer to Chapter 2 of the companion book [155] for definitions and properties of impulse function, and Laplace transform)

$$\mathscr{L}[x^*(t)] \triangleq X^*(s) = \int_0^\infty x_a(t)\,\delta_T(t)e^{-st}\,dt$$

$$= \sum_{k=0}^{\infty} \int_0^\infty x_a(t)\delta(t-kT)e^{-st}\,dt$$

$$= \sum_{k=0}^{\infty} x_a(kT)e^{-skT} \tag{2.32b}$$

This expression for $X^*(s)$ represents a Laplace transform, but it is not a transform that is easy to use because the complex variable s occurs in the exponent of the *transcendental function e*. By contrast, the Laplace transforms that we have used previously in the companion book [155], have mostly been *ratios of polynomials* in the Laplace variable s, with real coefficients. These latter transforms are easy to manipulate and interpret.

Ultimately, we will be able to achieve these same ratios of polynomials in a new complex variable z by transforming $X^*(s)$ to reach what we will call the *z-plane*.

We remind the reader here that $X^*(s)$ is the notation used for Laplace transform of *impulse modulated signal* $x^*(t)$; the 'star' distinguishes it from $X(s)$—the *conventional* Laplace transform of the *unsampled continuous function* $x_a(t)$. We have used the same complex plane (s-plane) for Laplace transform of 'starred' functions and conventional functions. This is the most compact approach used almost universally.

2.6.2 Definition of the *z*-Transform

The expression $X^*(s)$, given by (2.32b), contains the term e^{-Ts}; T is the fixed sampling period. To transform the irrational function $X^*(s)$ into a rational function, we use a transformation from the complex variable s to another complex variable, say, z. An obvious choice for this transformation is

$$z = e^{Ts} \tag{2.33a}$$

although $z = e^{-Ts}$ would be just as acceptable.

Solving for s in Eqn. (2.33a), we obtain

$$s = \frac{1}{T} \ln z \tag{2.33b}$$

The relationship between s and z in Eqns (2.33) may be defined as the *z-transformation*. In these two equations, z is a complex variable; its relation with real and imaginary parts of complex variable s is given by (with $s = \sigma + j\omega$):

$$z = e^{T(\sigma + j\omega)} = e^{T\sigma}\, e^{j\omega T} = re^{j\Omega} \tag{2.34}$$

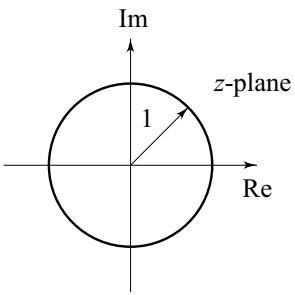


Fig. 2.20 Unit circle in *z*-plane

For a fixed value of r, the locus of z is a circle in the complex z-plane. Circle of radius unity in the complex z-plane will be of specific interest to us. This circle is called a *unit circle* (Fig. 2.20).

When Eqns (2.33) are substituted in Eqn. (2.32b), we have

$$X^*\left(s = \frac{1}{T} \ln z\right) = X(z) = \sum_{k=0}^{\infty} x_a(kT) z^{-k} \tag{2.35}$$

Thus, the z-transformation given by Eqns (2.33) is same as defined earlier in Eqn. (2.28) for intrinsically discrete-time systems.

Since T remains fixed, there is no loss of information if variable $x(k)$ is used to represent $x_a(kT)$. The expression

$$X(z) = \sum_{k=0}^{\infty} x(k) z^{-k}$$

is often used as the definition of the z-transform of sequence $x(k)$ (intrinsically discrete-time sequence or derived from continuous-time signal $x_a(t)$; $x(k) \triangleq x_a(kT)$), denoted symbolically as $\mathcal{Z}[x(k)]$:

$$X(z) \triangleq \mathcal{Z}[x(k)] = \sum_{k=0}^{\infty} x(k) z^{-k} \tag{2.36}$$

The summation in Eqn. (2.36) does not converge for all functions; and when it does, it does so for restricted values of z in the z-plane. Therefore, for each sequence for which the summation converges, there is an associated *convergence region* in the complex z-plane (Examples of z-plane convergence regions will shortly follow).

We now have a transform $X(z)$ defined in terms of a complex variable z. However, our expression for $X(z)$ is an infinite sum in the complex variable z. What we would like to achieve is something of the form:

$$X(z) = \frac{b_0 z^m + b_1 z^{m-1} + \cdots + b_m}{z^n + a_1 z^{n-1} + \cdots + a_n}\ ;\ m \le n \tag{2.37}$$

where all the a_i and b_j are real.

Under these conditions, we can write

$$X(z) = \frac{K\prod_{i=1}^{m}(z-\beta_i)}{\prod_{j=1}^{n}(z-\alpha_j)} \tag{2.38}$$

where the β_i and α_j are either real or complex-conjugate pairs.

The *z*-transform *zeros* are the values of z for which the transform is zero. $z = -\beta_i$ are the zeros of $X(z)$ in Eqn. (2.38).

The *z*-transform *poles* are the values of z for which the transform is infinite. $z = -\alpha_j$ are the poles of $X(z)$ in Eqn. (2.38).

Actually, we are not very far from our goal. We will now show that we can find closed-form expressions for the *z*-transforms of all the functions that we need to study the control of sampled-data systems.

2.6.3 *z*-Transforms of Functions Useful for Control System Analysis

In this subsection, our goal is to find the *z*-transform of the functions we will subsequently need for analysis of control systems.

Unit Sample Sequence

The unit-sample sequence contains only one nonzero element and is defined by

$$\delta(k) = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

The *z*-transform of the elementary signal is

$$\mathcal{Z}[\delta(k)] = \sum_{k=0}^{\infty} \delta(k)]\, z^{-k} = z^0 = 1;\ |z| > 0 \tag{2.39}$$

Unit Step Sequence

The unit-step sequence is defined as

$$\mu(k) = \begin{cases} 1 & \text{for } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

The *z*-transform is

$$\mathcal{Z}[\mu(k)] = \sum_{k=0}^{\infty} \mu(k)]\, z^{-k} = \sum_{k=0}^{\infty} z^{-k}$$

Using the geometric series formula

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x};\ |x| < 1,$$

this becomes

$$\mathcal{Z}[\mu(k)] = \frac{1}{1-z^{-1}} = \frac{z}{z-1} \tag{2.40}$$

Note that this equation holds only if the infinite sum converges, that is, if $|z^{-1}| < 1$ or $|z| > 1$. Thus, the region of convergence is the area outside the unit circle in the z-plane.

Sampled Ramp Function

Discrete form of $y(t) = t\mu(t)$ is of the form

$$y(k) = \begin{cases} kT & \text{for } k \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

From the basic definition (2.36),

$$\frac{dX(z)}{dz} = \sum_{k=0}^{\infty} \frac{d}{dz}[x(k)z^{-k}] = \sum_{k=0}^{\infty} (-k)\, x(k) z^{-k-1}$$

Consequently,

$$-z\frac{dX(z)}{dz} = \sum_{k=0}^{\infty} k\ x(k) z^{-k}$$

or

$$\mathcal{Z}[kT\, x(k)] = -\ Tz\, \frac{dX(z)}{dz} \tag{2.41}$$

For $x(k) = \mu(k)$, we obtain

$$\mathcal{Z}[kT\,\mu(k)] = \mathcal{Z}[y(k)] = Y(z) = -Tz\,\frac{d}{dz}\left(\frac{z}{z-1}\right)$$

$$= -Tz\left[\frac{-z}{(z-1)^2} + \frac{1}{z-1}\right]$$

$$= \frac{Tz}{(z-1)^2}\ ;\ |z| > 1 \tag{2.42}$$

Sampled Exponential Function

Discrete form of $x(t) = e^{-at}$ is of the form

$$x(k) = \begin{cases} e^{-akT} & \text{for } k \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Then

$$X(z) = \sum_{k=0}^{\infty} e^{-akT} z^{-k} = \sum_{k=0}^{\infty} (e^{aT} z)^{-k}$$

$$= \sum_{k=0}^{\infty} \alpha^{-k};\ \alpha = e^{aT} z$$

$$= \frac{1}{1-\alpha^{-1}} = \frac{e^{aT} z}{e^{aT} z - 1} = \frac{z}{z - e^{-aT}}$$

This equation holds only if the infinite sum converges, that is, if $|e^{-aT} z^{-1}| < 1$ or $|z| > |e^{-aT}|$. The result holds for both *real* and *complex a*.

Sampled Sinusoids

Discrete form of $x(t) = A \cos(\omega t + \phi)$ is of the form

$$x(k) = \begin{cases} A \cos(\omega kT + \phi) \text{ for } k \geq 0 \\ 0 \qquad\qquad\qquad \text{otherwise} \end{cases}$$

The transform of sampled sinusoids can be found by expanding the sequence into complex exponential components.

$$x(k) = A\left(\frac{e^{j(\omega kT + \phi)} + e^{-j(\omega kT + \phi)}}{2}\right)$$

Then

$$X(z) = \frac{A}{2} \sum_{k=0}^{\infty} e^{j\phi} e^{j\omega kT} z^{-k} + \frac{A}{2} \sum_{k=0}^{\infty} e^{-j\phi} e^{j\omega kT} z^{-k}$$

$$= \frac{A}{2} \frac{ze^{j\phi}}{z - e^{-j\omega T}} + \frac{A}{2} \frac{ze^{-j\phi}}{z - e^{j\omega T}}$$

$$= \frac{Az\left[\dfrac{z(e^{j\phi} + e^{-j\phi})}{2} - \dfrac{e^{j(\omega T + \phi)} + e^{-j(\omega T + \phi)}}{2}\right]}{z^2 - 2\left(\dfrac{e^{j\omega T} + e^{-j\omega T}}{2}\right)z + 1}$$

$$= \frac{Az\left[z\cos\phi - \cos(\omega T + \phi)\right]}{z^2 - 2z\cos\omega T + 1};|z| > 1$$

Given the z-transform of $A\cos(\omega kT + \phi)$, we can obtain the z-transform of $Ae^{-akT}\cos(\omega kT + \phi)$ as follows:

$$\mathcal{Z}[e^{-akT} x(k)] = \sum_{k=0}^{\infty} x(k) e^{-akT} z^{-k}$$

$$= \sum_{k=0}^{\infty} x(k)(ze^{aT})^{-k} = X(ze^{aT}) \tag{2.43}$$

Noting that

$$\mathcal{Z}[A\cos(\omega kT + \phi)] = \frac{Az[z\cos\phi - \cos(\omega T + \phi)]}{z^2 - 2z\cos\omega T + 1}$$

we substitute ze^{aT} for z to obtain the z-transform of $Ae^{-akT}\cos(\omega kT+\phi)$ as follows:

$$\mathcal{Z}[Ae^{-akT}\cos(\omega kT+\phi)] = \frac{Aze^{aT}[ze^{aT}\cos\phi-\cos(\omega T+\phi)]}{z^2e^{2aT}-2ze^{aT}\cos\omega T+1};\ |z|>e^{-aT} \tag{2.44}$$

Example 2.3

Let us find the z-transform of

$$X(s) = \frac{1}{s(s+1)}$$

Whenever a function in s is given, one approach for finding the corresponding z-transform is to convert $X(s)$ into $x(t)$ and then find the z-transform of $x(kT) = x(k)$; T is the sampling interval.

The inverse Laplace transform of $X(s)$ is

$$x(t) = 1-e^{-t};\ t \geq 0$$

Hence

$$x(kT) \triangleq x(k) = 1-e^{-kT};\ k \geq 0$$

$$X(z) = \mathcal{Z}[x(k)] = \frac{z}{z-1}-\frac{z}{z-e^{-T}} = \frac{(1-e^{-T})z}{(z-1)(z-e^{-T})};|z|>1$$

We summarize the z-transforms we have derived up to this point, plus some additional transforms in Table 2.1. The table lists commonly encountered functions $x(t)$; $t \geq 0$ and z-transforms of sampled version of these functions, given by $x(kT)$. We also include in this table, the Laplace transforms $X(s)$ corresponding to the selected $x(t)$. We have seen in Example 2.3, that whenever a function in s is given, one approach for finding the corresponding z-transform is to convert $X(s)$ into $x(t)$, and then find its z-transform. Another approach is to expand $X(s)$ into partial functions and use z-transform table to find the z-transforms of the expanded terms. Table 2.1 will be helpful for this second approach.

All the transforms listed in the table, can easily be derived from first principles. It may be noted that in this transform table, regions of convergence have not been specified. In our applications of systems analysis, which involve transformation from *time domain* to *z-domain* and *inverse transformation*, the variable z acts as a dummy operator. If transform pairs for sequences of interest to us are available, we are not concerned with the region of convergence.

2.6.4 Useful Operations with *z*-Transforms

Shifting Theorem: Advance (Forward Shift)

z-transformation of *difference equations* written in terms of advanced versions of the input and output variables (refer to Eqn. (2.24)) requires the following results:

$$\mathcal{Z}[y(k+1)] = \sum_{k=0}^{\infty} y(k+1)z^{-k} = z\sum_{k=0}^{\infty} y(k+1)z^{-(k+1)}$$

Table 2.1 Abbreviated table of *z*-transforms

$X(s)$	$x(t);\ t \geq 0$	$x(k);\ k = 0, 1, 2, \ldots$	$X(z)$
–	–	$\delta(k)$	1
$\frac{1}{s}$	$\mu(t)$	$\mu(k)$	$\frac{z}{z-1}$
$\frac{1}{s+a}$	e^{-at}	e^{-akT}	$\frac{z}{z-e^{-aT}}$
$\frac{1}{s^2}$	t	kT	$\frac{Tz}{(z-1)^2}$
$\frac{1}{(s+a)^2}$	$t\,e^{-at}$	kTe^{-akT}	$\frac{Te^{-aT}z}{(z-e^{-aT})^2}$
$\frac{a}{s(s+a)}$	$1-e^{-at}$	$1-e^{-akT}$	$\frac{(1-e^{-aT})z}{(z-1)(z-e^{-aT})}$
$\frac{\omega}{s^2+\omega^2}$	$\sin \omega t$	$\sin \omega kT$	$\frac{(\sin \omega T)z}{z^2-(2\cos \omega T)z+1}$
$\frac{s}{s^2+\omega^2}$	$\cos \omega t$	$\cos \omega kT$	$\frac{z^2-(\cos \omega T)z}{z^2-(2\cos \omega T)z+1}$
$\frac{\omega}{(s+a)^2+\omega^2}$	$e^{-at}\sin \omega t$	$e^{-akT}\sin \omega kT$	$\frac{(e^{-aT}\sin \omega T)z}{z^2-(2e^{-aT}\cos \omega T)z+e^{-2aT}}$
$\frac{s+a}{(s+a)^2+\omega^2}$	$e^{-at}\cos \omega t$	$e^{-akT}\cos \omega kT$	$\frac{z^2-(e^{-aT}\cos \omega T)z}{z^2-(2e^{-aT}\cos \omega T)z+e^{-2aT}}$

Letting $k + 1 = m$, yields

$$\mathcal{Z}[y(k+1)] = z\sum_{k=0}^{\infty} y(m)z^{-m} = z\left[\sum_{k=0}^{\infty} y(m)z^{-m} - y(0)\right]$$
$$= zY(z) - zy(0) \tag{2.45a}$$

Use of this result in a recursive manner leads to the following general result:

$$\mathcal{Z}[y(k+n)] = z^n\,Y(z) - z^n\,y(0) - z^{n-1}\,y(1) - \cdots - z^2\,y(n-2) - zy(n-1) \tag{2.45b}$$

Shifting Theorem: Delay (Backward Shift)

z-transformation of difference equations written in terms of delayed versions of input and output variables (refer to Eqn. (2.23)) requires the following results:

$$\begin{aligned}\mathcal{Z}[y(k-1)] &= \sum_{k=0}^{\infty}(k-1)z^{-k}\\ &= y(-1)z^{0} + y(0)z^{-1} + y(1)z^{-2} + \cdots\\ &= z^{-1}\left[\sum_{k=0}^{\infty} y(k)z^{-k}\right] + y(-1)\\ &= z^{-1}\,Y(z) + y(-1)\end{aligned} \tag{2.46a}$$

Use of this result in a recursive manner leads to the following general result:

$$\begin{aligned}\mathcal{Z}[y(k-n)] &= z^{-n}\,Y(z) + z^{-(n-1)}\,y(-1) + z^{-(n-2)}\,y(-2) + \cdots\\ &\quad + z^{-1}\,y(-n+1) + y(-n)\end{aligned} \tag{2.46b}$$

If $y(k) = 0$ for $k < 0$, we have

$$\mathcal{Z}[y(k-n)] = z^{-n}\,Y(z) \tag{2.47}$$

Example 2.4

Let us find the z-transforms of unit-step functions that are delayed by one sampling period, and n sampling periods, respectively.

Using the shifting theorem given by Eqn. (2.47), we have

$$\mathcal{Z}[\mu(k-1)] = z^{-1}\,\mathcal{Z}[\mu(k)] = z^{-1}\left(\frac{1}{1-z^{-1}}\right) = \frac{z^{-1}}{1-z^{-1}}$$

Also

$$\mathcal{Z}[\mu(k-n)] = z^{-n}\,\mathcal{Z}[\mu(k)] = z^{-n}\left(\frac{1}{1-z^{-1}}\right) = \frac{z^{-n}}{1-z^{-1}}$$

Remember that multiplication of the z-transform $X(z)$ by z has the effect of *advancing* the signal $x(k)$ by one sampling period, and that multiplication of the z-transform $X(z)$ by z^{-1} has the effect of *delaying* the signal $x(k)$ by one sampling period. In control engineering and signal processing, $X(z)$ is frequently expressed as a ratio of polynomials in z^{-1} as follows (refer to Table 2.1):

$$\mathcal{Z}[e^{-akT}\cos\omega kT] = \frac{z^2 - (e^{-aT}\cos\omega T)z}{z^2 - (2e^{-aT}\cos\omega T)z + e^{-2aT}} \tag{2.48a}$$

$$= \frac{1 - (e^{-aT}\cos\omega T)z^{-1}}{1 - (2e^{-aT}\cos\omega T)z^{-1} + e^{-2aT}z^{-2}} \tag{2.48b}$$

z^{-1} is interpreted as the *unit-delay operator*.

In finding the *poles* and *zeros* of $X(z)$, it is convenient to express $X(z)$ as a ratio of polynomials z, as is done in Eqn. (2.48a). In this and the next chapter, $X(z)$ will be expressed in terms of powers of z, as given by Eqn. (2.48a), or in terms of powers of z^{-1}, as given by Eqn. (2.48b), depending on the circumstances.

Example 2.5

Analogous to the operation of integration, we can define the summation operation

$$x(k) = \sum_{i=0}^{k} y(i) \tag{2.49}$$

In the course of deriving an expression for $X(z)$ in terms of $Y(z)$, we shall need the infinite series sum:

$$\sum_{k=0}^{\infty} (az^{-1})^k = \frac{1}{1-az^{-1}} = \frac{z}{z-a} \tag{2.50}$$

which converges, provided that $|az^{-1}| < 1$, or $|z| > a$.

Successive values of $x(k)$ are as follows:

$$\begin{aligned} x(0) &= y(0) \\ x(1) &= y(0) + y(1) \\ &\vdots \\ x(k) &= y(0) + y(1) + \cdots + y(k) \end{aligned}$$

Thus, $X(z)$ is the infinite sum given below.

$$\begin{aligned} X(z) = \sum_{k=0}^{\infty} x(k)z^{-k} &= x(0) + z^{-1}x(1) + \cdots + z^{-k}x(k) + \cdots \\ &= y(0) + z^{-1}[y(0) + y(1)] + \cdots + z^{-k}[y(0) + \cdots + y(k)] + \cdots \\ &= y(0)[1 + z^{-1} + z^{-2} + \cdots] + y(1)[z^{-1} + z^{-2} + \cdots] + \cdots \\ &\quad + y(k)[z^{-k} + z^{-k-1} + \cdots] + \cdots \\ &= \left(\frac{z}{z-1}\right)[y(0) + z^{-1}y(1) + z^{-2}y(2) + \cdots] \\ &= \left(\frac{z}{z-1}\right)\left[\sum_{k=0}^{\infty} y(k)z^{-k}\right] \end{aligned}$$

Therefore,

$$X(z) = \frac{z}{z-1}\ Y(z) \tag{2.51}$$

2.6.5 *z*-Transform Inversion

We will obtain the inverse z-transform in exactly the same way that we obtained the inverse Laplace transform (Chapter 2 [155]), namely, by *partial fraction expansion*. The reason that the partial fraction expansion method works is that we frequently encounter transforms that are *rational functions*, i.e., ratio of two polynomials in z with real coefficients (refer to Eqn. (2.37)). The fact that the coefficients are real is crucial, because it guarantees that the roots of the numerator and denominator polynomials will be either *real*, or *complex-conjugate pairs*. This, in turn, means that the individual terms in the partial fraction expansion of the transform will be simple in form and we will be able to do the inverse transformation by inspection.

The transform pairs encountered using the partial fraction expansion technique will usually be those found in Table 2.1. Those not found in the table can easily be derived by using basic properties of z-transformation. The partial fraction expansion method for z-transforms is very straightforward, and similar, in most respects, to the partial fraction expansion in Laplace transforms. We first illustrate the method with some examples from which we can then abstract some general guidelines.

Example 2.6

We observe that the transforms of the elementary functions (see Table 2.1) contain a factor of z in the numerator, e.g.,

$$\mathcal{Z}[\mu(k)] = \frac{z}{z-1}$$

where $\mu(k)$ is a unit-step sequence.

To ensure that the partial fraction expansion will yield terms corresponding to those tabulated, it is customary to first expand the function $Y(z)/z$, if $Y(z)$ has one or more roots at the origin, and then multiply the resulting expansion by z.

For instance, if $Y(z)$ is given as

$$Y(z) = \frac{2z^2 - 1.5z}{z^2 - 1.5z + 0.5} = \frac{z(2z-1.5)}{(z-0.5)(z-1)},$$

we are justified in writing

$$\frac{Y(z)}{z} = \frac{2z-1.5}{(z-0.5)(z-1)} = \frac{A_1}{z-0.5} + \frac{A_2}{z-1}$$

Constants A_1 and A_2 can be evaluated by applying the conventional partial fraction expansion rules.

$$A_1 = (z-0.5)\left[\frac{Y(z)}{z}\right]\Bigg|_{z=0.5} = 1; A_2 = (z-1)\left[\frac{Y(z)}{z}\right]\Bigg|_{z=1} = 1$$

$$\frac{Y(z)}{z} = \frac{1}{z-0.5} + \frac{1}{z-1}$$

or

$$Y(z) = \frac{z}{z-0.5} + \frac{z}{z-1}$$

Using the transform pairs from Table 2.1,

$$y(k) = (0.5)^{kT} + 1; k \geq 0$$

Example 2.7

When $Y(z)$ does not have one or more zeros at the origin, we expand $Y(z)$, instead of $Y(z)|/z$, into partial fractions and utilize *shifting theorem* given by Eqn. (2.47) to obtain inverse z-transform.

In applying the shifting theorem, notice that if

$$\mathcal{Z}^{-1}[Y(z)] = y(k) \quad \text{then} \quad \mathcal{Z}^{-1}[z^{-1}Y(z)] = y(k-1)$$

Let us consider an example:

$$Y(z) = \frac{10}{(z-1)(z-0.2)}$$

We first expand $Y(z)$ into partial fractions:

$$Y(z) = \frac{12.5}{z-1} - \frac{12.5}{z-0.2}$$

Notice that the inverse z-transform of $1/(z–1)$ is not available in Table 2.1. However, using the shifting theorem, we find that

$$\mathcal{Z}^{-1}\left[\frac{1}{z-1}\right] = \mathcal{Z}^{-1}\left[z^{-1}\left(\frac{z}{z-1}\right)\right] = \begin{cases} 1; & k = 1, 2, \ldots \\ 0; & k \le 0 \end{cases}$$

Also,

$$\mathcal{Z}^{-1}\left[\frac{1}{z-0.2}\right] = \mathcal{Z}^{-1}\left[z^{-1}\left(\frac{z}{z-0.2}\right)\right] = \begin{cases} (0.2)^{k-1}; & k = 1, 2, \ldots \\ 0 & ; k \le 0 \end{cases}$$

Therefore,

$$y(k) = \mathcal{Z}^{-1}[Y(z)] = \begin{cases} 12.5\left[1-(0.2)^{k-1}\right]; & k = 1, 2, \ldots \\ 0 & ; k \le 0 \end{cases}$$

which can be written as

$$y(k) = 12.5[1 - (0.2)^{k-1}]\mu(k-1)$$

2.6.6 Final Value Theorem

The final value theorem is concerned with the evaluation of $y(k)$ as $k \to \infty$ assuming, of course, that $y(k)$ does approach a limit. Using partial fraction expansion for inverting z-transforms, it is a simple matter to show that $y(k)$ approaches a limit as $k \to \infty$, if all the poles of $Y(z)$ lie inside the *unit circle* ($|z| < 1$) in the complex z-plane. The unit-circle boundary is, however, excluded except for a single pole at $z = 1$. This is because purely sinusoidal signals whose transforms will have poles on the unit circle, do not settle to a constant value as $k \to \infty$. Multiple poles at $z = 1$ are also excluded because, as we have already seen in the table of z-transform, these correspond to unbounded signals like ramps. A more compact way of phrasing these conditions is to say that $(z - 1)Y(z)$ must be *analytic* on the boundary, and outside the unit circle in the complex z-plane. The final value theorem states that when this condition on $(z - 1)Y(z)$ is satisfied, then

$$\lim_{k \to \infty} y(k) = \lim_{z \to 1} (z-1)Y(z) \tag{2.52}$$

The proof is as follows:

$$\mathcal{Z}[y(k+1) - y(k)] = \lim_{m \to \infty} \sum_{k=0}^{m} [y(k+1) - y(k)]z^{-k}$$

or

$$zY(z) - z\,y(0) - Y(z) = \lim_{m \to \infty} \sum_{k=0}^{m} [y(k+1) - y(k)]z^{-k}$$

Letting $z \to 1$ on both sides,

$$\lim_{z\to 1}[(z-1)Y(z)] = y(0) + \lim_{z\to 1}\lim_{m\to\infty}\sum_{k=0}^{m}[y(k+1)-y(k)]z^{-k}$$

Interchanging the order of limits on the right-hand side, we have

$$\lim_{z\to 1}[(z-1)Y(z)] = y(0) + \lim_{m\to\infty}\sum_{k=0}^{m}[y(k+1)-y(k)] = y(\infty)$$

Example 2.8

Given

$$X(z) = \frac{z}{z-1} - \frac{z}{z-e^{-aT}}; a > 0$$

By applying the final value theorem to the given $X(z)$, we obtain

$$\lim_{k\to\infty} x(k) = \lim_{z\to 1}\left[(z-1)\left(\frac{z}{z-1} - \frac{z}{z-e^{-aT}}\right)\right] = \lim_{z\to 1}\left[1 - \frac{z-1}{z-e^{-aT}}\right]z = 1$$

It is noted that the given $X(z)$ is actually the z-transform of

$$x(k) = 1 - e^{-akT}$$

By substituting $k = \infty$ in this equation, we have

$$x(\infty) = \lim_{k\to\infty}\left(1 - e^{-akT}\right) = 1$$

2.7 TRANSFER FUNCTION MODELS

The very first step in the analytical study of a system is to set up mathematical equations to describe the system. Because of different analytical methods used, or because of different questions asked, we may often set up different mathematical equations to describe the same system.

We have seen earlier in Section 2.5 (Eqn. (2.26)) that the input $r(k)$ and output $y(k)$ of any linear time-invariant discrete-time system that is initially relaxed at $k = 0$, can be described by an equation of the form

$$y(k) = \sum_{j=0}^{k} r(j)\, g(k-j);\ k \geq 0 \tag{2.53}$$

This is called the *convolution sum*. The function $g(k)$ is defined for $k \geq 0$ and is called the *impulse response* of the system.

The application of z-transform to Eqn. (2.53) gives us an extremely useful mathematical description, of a linear time-invariant discrete-time system (refer to Eqn. (2.29))

$$Y(z) = \mathcal{Z}[(y(k)] = G(z)R(z) \tag{2.54}$$

where

$$R(z) \triangleq \mathcal{Z}[(r(k)] \quad \text{and} \quad G(z) \triangleq \mathcal{Z}[(g(k)]$$

We see that by applying the z-transform, a convolution sum is transformed into an algebraic equation. The function $G(z)$ is called the *transfer function* of the discrete-time system.

The transfer function of a linear time-invariant discrete-time system is, by definition, the z-transform of the impulse response of the system.

An alternative definition of transfer function follows from Eqn. (2.54).

$$G(z) = \left.\frac{\mathcal{Z}[y(k)]}{\mathcal{Z}[r(k)]}\right|_{\substack{\text{System} \\ \text{initially relaxed}}} = \left.\frac{Y(z)}{R(z)}\right|_{\substack{\text{System} \\ \text{initially relaxed}}} \tag{2.55}$$

Thus, *the transfer function of a linear time-invariant discrete-time system is the ratio of the z-transforms of its output and input sequences, assuming that the system is initially relaxed.*

Figure 2.21 gives the block diagram of a discrete-time system in transform domain.

Let us use the definition given by Eqn. (2.55) to obtain transfer function model of a discrete-time system, represented by a difference equation of the form (2.23), relating its output $y(k)$ to the input $r(k)$. We assume that the discrete-time system is initially relaxed:

$$y(k) = 0 \text{ for } k < 0$$

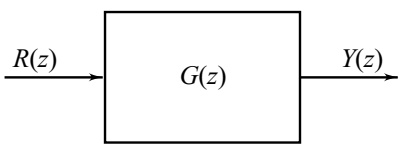


Fig. 2.21 Block diagram of a discrete-time system

and is excited by an input

$$r(k);\ k \geq 0$$

Taking z-transform of all the terms of Eqn. (2.23), under the assumption of zero initial conditions, we obtain

$$Y(z) + a_1 z^{-1} Y(z) + \cdots + a_n z^{-n} Y(z) = b_0 R(z) + b_1 z^{-1} R(z) + \cdots + b_n z^{-n} R(z)$$

where

$$Y(z) \triangleq \mathcal{Z}[y(k)] \text{ and } R(z) \triangleq \mathcal{Z}[r(k)]$$

Solving for $Y(z)$,

$$Y(z) = \frac{(b_0 + b_1 z^{-1} + \cdots + b_n z^{-n}) R(z)}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}}$$

Therefore, the transfer function $G(z)$ of the discrete-time system represented by difference equation (2.23) is

$$G(z) = \frac{Y(z)}{R(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \tag{2.56}$$

The same result can be obtained by taking the z-transformation of the shifted difference equation (2.24). We first consider the case with $n = 2$, and then present the general result.

$$y(k+2) + a_1 y(k+1) + a_2 y(k) = b_0 r(k+2) + b_1 r(k+1) + b_2 r(k) \tag{2.57}$$

The z-transform of Eqn. (2.57) gives (using the shifting theorem given by Eqns (2.45)):

$$[z^2Y(z) - z^2y(0) - zy(1)] + a_1[zY(z) - zy(0)] + a_2Y(z) = b_0[z^2R(z) - z^2r(0) - zr(1)] + b_1[zR(z) - zr(0)] + b_2R(z)$$

or

$$(z^2 + a_1z + a_2)\ Y(z) = (b_0z^2 + b_1z + b_2)\ R(z) + z^2\ [y(0) - b_0r(0)] + z[y(1) + a_1y(0) - b_0r(1) - b_1r(0)] \tag{2.58}$$

Since the system is initially at rest, and switched on at $k = 0$, we have $y(k) = 0$ for $k < 0$, and $r(k) = 0$ for $k < 0$. To determine the initial conditions $y(0)$ and $y(1)$, we substitute $k = -2$ and $k = -1$, respectively, into Eqn. (2.57).

$$y(0) + a_1y(-1) + a_2y(-2) = b_0r(0) + b_1r(-1) + b_2r(-2)$$

which simplifies to

$$y(0) = b_0r(0) \tag{2.59a}$$

and

$$y(1) + a_1y(0) + a_2y(-1) = b_0r(1) + b_1r(0) + b_2r(-1)$$

or

$$y(1) = -a_1y(0) + b_0r(1) + b_1r(0) \tag{2.59b}$$

By substituting Eqns (2.59) into Eqn. (2.58), we get

$$(z^2 + a_1z + a_2)Y(z) = (b_0z^2 + b_1z + b_2)\ R(z)$$

Therefore,

$$G(z) = \frac{Y(z)}{R(z)} = \frac{b_0z^2 + b_1z + b_2}{z^2 + a_1z + a_2} = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}}$$

Therefore, we can express the general transfer function model (2.56) as

$$G(z) = \frac{Y(z)}{R(z)} = \frac{b_0z^n + b_1z^{n-1} + \cdots + b_n}{z^n + a_1z^{n-1} + \cdots + a_n} \tag{2.60a}$$

We will represent the *numerator polynomial* of $G(z)$ by $N(z)$, and the *denominator polynomial* by $\Delta(z)$:

$$G(z) = \frac{N(z)}{\Delta(z)} \tag{2.60b}$$

where

$$N(z) = b_0\,z^n + b_1\,z^{n-1} + \cdots + b_n; \quad \Delta(z) = z^n + a_1\,z^{n-1} + \cdots + a_n$$

The terminology used in connection with $G(s)$—the transfer function of continuous-time systems[6]—is directly applicable in the case of $G(z)$.

The highest power of the complex variable z in the denominator polynomial $\Delta(z)$ of the transfer function $G(z)$ determines the *order of the transfer function model*. The denominator polynomial $\Delta(z)$ is called the *characteristic polynomial*.

[6] Chapter 2 of reference [155].

The roots of the equation

$$\Delta(z) = 0 \tag{2.61a}$$

are called the *poles* of the transfer function $G(z)$, and roots of the equation

$$N(z) = 0 \tag{2.61b}$$

are called the *zeros*.

Equation (2.61a) is called the *characteristic equation*; the poles are the *characteristic roots*.

Section 2.9 shows that if the poles of the transfer function $G(z)$ of a discrete-time system lie inside the *unit circle* in the complex z-plane, the discrete-time system is stable.

2.7.1 Transfer Function of Unit Delayer

We now give a simple example of transfer function description of discrete-time systems. Figure 2.12 describes the basic operations characterizing a computer program. The *unit delayer* shown in this figure is a dynamic system with input $x_1(k)$ and output $x_2(k)$; $x_2(0)$ represents the initial storage in the shift register.

We assume that the discrete-time system (unit delayer) is initially relaxed:

$$x_2(0) = 0 \tag{2.62a}$$

and is excited by an input sequence

$$x_1(k);\ k \geq 0 \tag{2.62b}$$

The following state variable model gives the output of the unit delayer at $k = 0, 1, 2, \ldots$

$$x_2(k+1) = x_1(k) \tag{2.63}$$

The z-transformation of Eqn. (2.63) yields

$$X_2(z) = z^{-1} X_1(z)$$

where

$$X_2(z) \triangleq \mathcal{Z}[x_2(k)];\ X_1(z) \triangleq \mathcal{Z}[x_1(k)]$$

Therefore, the transfer function of the unit delayer represented by Eqn. (2.63) is

$$\frac{X_2(z)}{X_1(z)} = z^{-1} \tag{2.64}$$

The discrete-time system of Fig. 2.16 may be equivalently represented by Fig. 2.22a using the transfer function description of the unit delayer. Use of the block-diagram analysis results in Fig. 2.22b, which gives

$$Y(z) = \left[\frac{0.0475}{z - 0.95} + 0.05\right] R(z)$$

Therefore, the transfer function $G(z)$ of the discrete-time system of Fig. 2.22 is

$$G(z) = \frac{Y(z)}{R(z)} = \frac{0.05z}{z - 0.95} \tag{2.65}$$

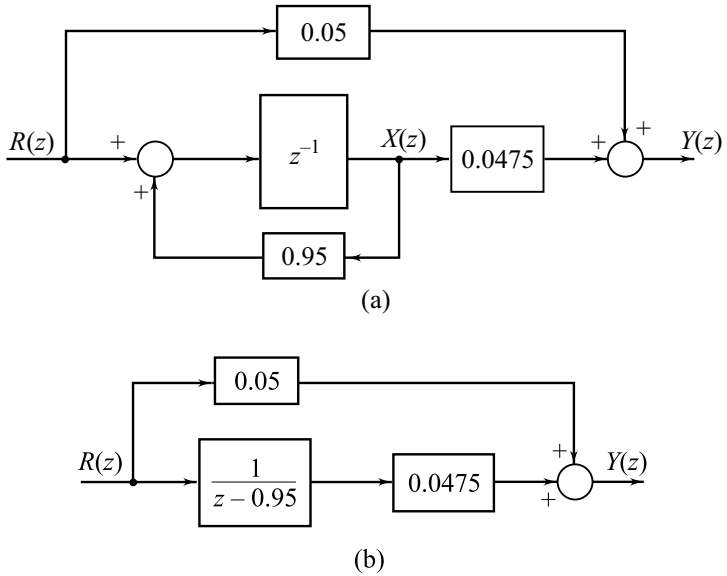


Fig. 2.22 Equivalent representations of the discrete-time system of Fig. 2.16

2.7.2 Dynamic Response

A standard problem in control engineering is to find the *response*, $y(k)$, of a system given the *input*, $r(k)$, and a *model* of the system. With z-transforms, we have a means for easily computing the response of linear time-invariant systems to quite general inputs.

Given a general relaxed, linear discrete-time system and an input signal $r(k)$, the procedure for determining the output $y(k)$ is given by the following steps:

Step 1 Determine the transfer function $G(z)$ by taking z-transform of equations of motion.

Step 2 Determine the z-transform of the input signal; $R(z) = \mathcal{Z}[r(k)]$.

Step 3 Determine the z-transform of the output; $Y(z) = G(z)R(z)$.

Step 4 Break-up $Y(z)$ by partial fraction expansion.

Step 5 Invert $Y(z)$ to get $y(k)$; find the components of $y(k)$ in a table of transform pairs and combine the components to get the total solution in the desired form.

Example 2.9

A discrete-time system is described by the transfer function

$$G(z) = \frac{Y(z)}{R(z)} = \frac{1}{z^2 + a_1 z + a_2};\ a_1 = -\frac{3}{4},\ a_2 = \frac{1}{8}$$

Find the response $y(k)$ to the input (i) $r(k) = \delta(k)$, (ii) $r(k) = \mu(k)$.

Solution The transfer function $G(z)$ expressed as a ratio of polynomials in z^{-1}:

$$G(z) = \frac{z^{-2}}{1 - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2}}$$

Since z^{-1} is a unit-delay operator, we can immediately write the corresponding difference equation:

$$y(k) - \frac{3}{4}\, y(k-1) + \frac{1}{8}\, y(k-2) = r(k-2)$$

The difference equations can be solved by means of recursion procedure. The recursion procedure is quite simple and convenient for digital computations.

In the following, we obtain the *closed-form solutions*.

(i) The z-transform of $\delta(k)$ is (refer to Table 2.1)

$$\mathcal{Z}[\delta(k)] = 1$$

Letting $R(z) = 1$, we obtain

$$Y(z) = G(z) = \frac{1}{z^2 - \frac{3}{4}z + \frac{1}{8}} = \frac{1}{\left(z - \frac{1}{4}\right)\left(z - \frac{1}{2}\right)} = \frac{4}{z - \frac{1}{2}} - \frac{4}{z - \frac{1}{4}}$$

The impulse response $g(k)$ is, therefore, (refer to Table 2.1)

$$g(k) = \mathcal{Z}^{-1}\left[z^{-1}\left(\frac{4z}{z - \frac{1}{2}}\right)\right] - \mathcal{Z}^{-1}\left[z^{-1}\left(\frac{4z}{z - \frac{1}{4}}\right)\right]$$

$$= 4\left(\frac{1}{2}\right)^{k-1} - 4\left(\frac{1}{4}\right)^{k-1}; k \geq 1$$

The impulse input thus *excites* the *system poles* without creating any additional response term.

(ii) The z-transform of $\mu(k)$ is (refer to Table 2.1)

$$\mathcal{Z}[\mu(k)] = \frac{z}{z-1}$$

Letting $R(z) = \frac{z}{z-1}$, we obtain

$$Y(z) = \frac{1}{\left(z - \frac{1}{4}\right)\left(z - \frac{1}{2}\right)}\left[\frac{z}{z-1}\right] = \frac{z}{\left(z - \frac{1}{4}\right)\left(z - \frac{1}{2}\right)(z-1)}$$

$$= \underbrace{\frac{\frac{16}{3}z}{z - \frac{1}{4}} + \frac{-8z}{z - \frac{1}{2}}}_{\text{System poles}} + \underbrace{\frac{\frac{8}{3}z}{z-1}}_{\text{Excitation pole}}$$

The inverse transform operation gives (refer to Table 2.1)

$$y(k) = \left[\underbrace{\frac{16}{3}\left(\frac{1}{4}\right)^k - 8\left(\frac{1}{2}\right)^k}_{\text{Transient response}} + \underbrace{\frac{8}{2}(1)^k}_{\text{Steady-state response}} \right]$$

The *transient response* terms correspond to system poles excited by the input $\mu(k)$. These terms vanish as $k \to \infty$.

The second response term arises due to the *excitation pole*, and has the same nature as the input itself except for a modification in magnitude caused by the system's behavior to the specified input. Since the input exists as $k \to \infty$, the second response term does not vanish and is called the *steady-state response* of the system.

The steady-state response can be quickly obtained without doing the complete inverse transform operation by use of the *final value theorem* (Eqn. (2.52)):

$$\lim_{k\to\infty} y(k) = \lim_{z\to 1} (z-1)Y(z)$$

if $(z-1)Y(z)$ has no poles on the boundary and outside of the *unit circle* in the complex z-plane.

Example 2.10

Consider a discrete-time system described by the difference equation

$$y(k+2) + \frac{1}{4}y(k+1) - \frac{1}{8}y(k) = 3r(k+1) - r(k)$$

The system is initially relaxed ($y(k) = 0$ for $k < 0$) and is excited by the input

$$r(k) = (-1)^k \mu(k)$$

Obtain the transfer function model of the discrete-time system, and therefrom, find the output $y(k)$; $k \geq 0$.

Solution The given difference equation is first converted to the equivalent form:

$$y(k) + \frac{1}{4}y(k-1) - \frac{1}{8}y(k-2) = 3r(k-1) - r(k-2)$$

z-transformation of each term in this equation yields (using *shifting theorem* (2.47))

$$Y(z) + \frac{1}{4}z^{-1}Y(z) - \frac{1}{8}z^{-2} = 3z^{-1}R(z) - z^{-2}R(z)$$

or

$$\left(1 + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}\right) Y(z) = (3z^{-1} - z^{-2})\, R(z)$$

or

$$\left(z^2 + \frac{1}{4}z - \frac{1}{8}\right) Y(z) = (3z - 1)\, R(z)$$

The transfer function of the given discrete-time system is

$$G(z) = \frac{Y(z)}{R(z)} = \frac{3z-1}{z^2+\frac{1}{4}z-\frac{1}{8}}$$

For (refer to Table 2.1; $e^{-aT} = -1$)

$$R(z) = \mathcal{Z}[(-1)^k] = \frac{z}{z+1},$$

$$Y(z) = G(z)R(z) = \frac{z(3z-1)}{\left(z^2+\frac{1}{4}z-\frac{1}{8}\right)(z+1)}$$

$$= \frac{z(3z-1)}{\left(z+\frac{1}{2}\right)\left(z-\frac{1}{4}\right)(z+1)} = \frac{\frac{20}{3}z}{z+\frac{1}{2}} + \frac{\frac{-4}{15}z}{z-\frac{1}{4}} + \frac{\frac{-32}{5}z}{z+1}$$

Then

$$y(k) = \left[\frac{20}{3}\left(-\frac{1}{2}\right)^k - \frac{4}{15}\left(\frac{1}{4}\right)^k - \frac{32}{5}(-1)^k\right]\mu(k)$$

2.8 FREQUENCY RESPONSE

Using the Laplace transform, we were able to show that if we applied the input $r(t) = R_0 \cos \omega t$ to a linear time-invariant system with transfer function $G(s)$, the *steady-state* output y_{ss} was of the form (refer to [155]),

$$y_{ss} = \lim_{t\to\infty} y(t) = R_0|G(j\omega)| \cos(\omega t + \phi)$$

where $\phi = \angle G(j\omega)$.

A similar result can be obtained for discrete-time systems. Let $G(z)$ be the z-transform of a discrete linear time-invariant system, and let the input to this system be $r(kT) = R_0 \cos(\omega kT)$, with T sampling period. Then

$$\mathcal{Z}[r(kT)] = \frac{R_0 z(z-\cos\omega T)}{z^2 - 2z\cos\omega T + 1} = \frac{R_0 z(z-\cos\omega T)}{\left(z-e^{j\omega T}\right)\left(z-e^{-j\omega T}\right)}$$

Suppose

$$G(z) = \frac{k\prod_{i=1}^{m}(z-p_i)}{\prod_{j=1}^{n}(z-\alpha_j)}; |\alpha_j| < 1 \tag{2.66}$$

For simplicity, we assume no repeated poles. Then

$$Y(z) = G(z)R(z)$$

can be expressed as

$$Y(z) = \frac{Az}{z - e^{j\omega T}} + \frac{A^* z}{z - e^{-j\omega T}} + \sum_{j=1}^{n} \frac{B_j z}{z - \alpha_j}$$

Now

$$A = \left(z - e^{j\omega T}\right) \frac{R_0(z - \cos\omega T)}{\left(z - e^{j\omega T}\right)\left(z - e^{-j\omega T}\right)} G(z)\Big|_{z=e^{j\omega T}}$$

$$= \frac{R_0\left[e^{j\omega T} - 0.5\left(e^{j\omega T} + e^{-j\omega T}\right)\right]}{e^{j\omega T} - e^{-j\omega T}} G\left(e^{j\omega T}\right) = \frac{R_0}{2} G\left(e^{j\omega T}\right) = |A| e^{j\theta}$$

The other residue A^* is the complex conjugate of A. Taking the inverse transform of $Y(z)$, we obtain

$$y(k) = \underbrace{Ae^{j\omega kT} + A^* e^{-j\omega kT}}_{\text{Steady-state component}} + \underbrace{\sum_{j=1}^{n} B_j (\alpha_j)^k}_{\text{Transient component}}$$

If $|a_j| < 1$ for $j = 1, 2, \ldots, n$, then

$$\lim_{k\to\infty} \sum_{j=1}^{n} B_j \left(\alpha_j\right)^{\prime} = 0$$

Thus,

$$y_{ss} \triangleq \lim_{k\to\infty} y(kT) = Ae^{j\omega kT} + A^* e^{-j\omega kT} = \frac{2\,|A|\left[e^{j(\omega kT+\theta)} + e^{-j(\omega kT+\theta)}\right]}{2}$$

$$= 2|A| \cos(\omega kT + \theta)$$

$$= R_0 \left|G\left(e^{j\omega T}\right)\right| \cos(\omega kT + \theta) \tag{2.67}$$

We have obtained a result that is analogous to that for continuous-time systems. For a sinusoidal input, the steady-state output is also sinusoidal; scaled by the gain factor $|G(e^{j\omega T})|$, and shifted in phase by $\theta = \angle G(e^{j\omega T})$. An important difference is that in the continuous-time case we have $|G(j\omega)|$, while in the discrete-time case we have $|G(e^{j\omega T})|$. The implications of this difference are discussed later in this chapter.

The steady-state response is also given by (2.67) when the poles of $G(z)$ in Eqn.(2.66) are repeated with $|a_j| < 1$. This can easily be verified.

Example 2.11

The discrete-time system of Fig. 2.22 is described by the transfer function (refer to Eqn.(2.65))

$$G(z) = \frac{Y(z)}{R(z)} = \frac{0.05z}{z - 0.95}$$

The given input sequence is

$$r(k) = R_0 \cos(\Omega k) = \text{Re}\ \{R_0 e^{j\Omega k}\}$$

The output is then given by

$$y(k) = \text{Re}\left\{\mathcal{Z}^{-1}\left[G(z)\frac{R_0\ z}{z - e^{j\Omega}}\right]\right\} = \text{Re}\left\{\mathcal{Z}^{-1}\left[\frac{0.05\ R_0 z^2}{(z-0.95)\ (z - e^{j\Omega})}\right]\right\}$$

$$= \text{Re}\left\{\mathcal{Z}^{-1}\left[\frac{0.0475\ R_0}{0.95 - e^{j\Omega}}\left(\frac{z}{z-0.95}\right) + G(e^{j\Omega})\frac{R_0 z}{(z - e^{j\Omega})}\right]\right\}$$

where

$$G(e^{j\Omega}) = G(z)\Big|_{z = e^{j\Omega}} = \frac{0.05\ e^{j\Omega}}{e^{j\Omega} - 0.95}$$

The inverse z-transform operation yields

$$y(k) = \text{Re}\left[\underbrace{\frac{0.0475\ R_0}{0.95 - e^{j\Omega}}(0.95)^k}_{\substack{\text{Transient}\\ \text{component}}} + \underbrace{G(e^{j\Omega})R_0 e^{j\Omega k}}_{\substack{\text{Steady-state}\\ \text{component}}}\right]$$

This equation shows that as k increases, the transient component dies out. When this happens, the output expression becomes

$$y(k)\Big|_{k \text{ very large}} = y_{ss}(k) = \text{Re}\ \{G(e^{j\Omega})\ R_0\ e^{j\Omega k}\}$$

Let $$G(e^{j\Omega}) = |G(e^{j\Omega})|\ e^{j\phi};\ \phi = \angle G(e^{j\Omega})$$

Then

$$y_{ss}(k) = \text{Re}\{R_0\ |G(e^{j\Omega})| e^{j(\Omega k + \phi)}\} = R_0\ |G(e^{j\Omega})|\ \cos(\Omega k + \phi)$$

The steady-state response has the same form as the input (discrete sinusoidal), but is modified in *amplitude* by $|G(e^{j\Omega})|$ and in *phase* by $\angle G(e^{j\Omega})$.

The graphs of $|G(e^{j\Omega})|$ and $\angle G(e^{j\Omega})$ as the frequency Ω is varied, are the *frequency-response curves* of the given discrete-time system. The graphs are shown in Fig. 2.23. It is obvious from these curves, that the given system is a *low-pass digital filter*.

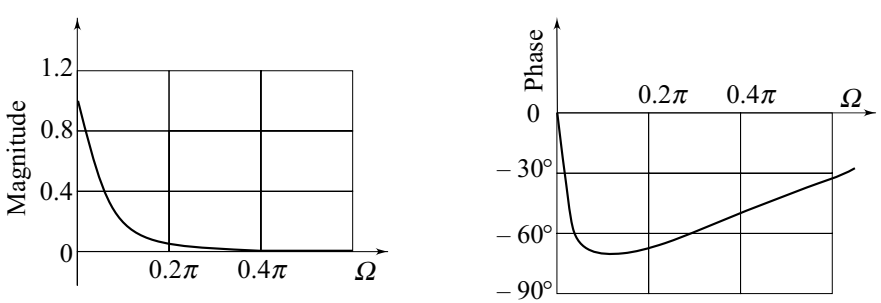


Fig. 2.23 Frequency-response curves of the discrete-time system of Fig. 2.22

2.9 STABILITY ON THE *z*-PLANE AND THE JURY STABILITY CRITERION

Stability is concerned with the qualitative analysis of the dynamic response of a system. This section is devoted to the stability analysis of linear time-invariant discrete-time systems. Stability concepts and definitions used in connection with continuous-time systems[7] are directly applicable here.

A linear time-invariant discrete-time system described by the state variable model (refer to Eqns (2.17)),

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}r(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0$$
$$y(k) = \mathbf{c}\mathbf{x}(k) + d\,r(k)$$

has the following two sources of excitation:

(i) the initial state $\mathbf{x}^0$ representing *initial internal energy storage*; and (ii) the *external input* $r(k)$.

The system is said to be in *equilibrium state* $\mathbf{x}^e = \mathbf{0}$, when both the initial internal energy storage and the external input are zero.

In the stability study, we are generally concerned with the questions listed below.

(i) If the system with zero input ($r(k) = 0;\ k \geq 0$) is *perturbed* from its equilibrium state $\mathbf{x}^e = \mathbf{0}$ at $k = 0$, will the state $\mathbf{x}(k)$ return to $\mathbf{x}^e$, remain 'close' to $\mathbf{x}^e$, or diverge from $\mathbf{x}^e$?

(ii) If the system is relaxed, will a *bounded* input $r(k)$; $k \geq 0$, produce a *bounded* output $y(k)$ for all k?

The first notion of stability is concerned with the 'boundedness' of the state of an *unforced system* in response to arbitrary initial state, and is called *zero-input stability*. The second notion is concerned with the boundedness of the output of a relaxed system in response to the bounded input, and is called *Bounded-Input, Bounded-Output* (BIBO) *stability*.

2.9.1 BIBO Stability

A relaxed system (zero initial conditions) is said to be BIBO stable if for every bounded input $r(k)$; $k \geq 0$, the output $y(k)$ is bounded for all k.

For a linear time-invariant system to satisfy this condition, it is necessary and sufficient that

$$\sum_{k=0}^{\infty} |g(k)| < \infty \tag{2.68}$$

where $g(k)$ is the impulse response of the system.

To prove that condition (2.68) guarantees BIBO stability—i.e., sufficiency—we first establish an upper bound on $|y(k)|$.

From Eqn. (2.25), we can express the output as a convolution sum:

$$y(k) = \sum_{j=0}^{\infty} g(j)\, r(k-j)$$

[7] Chapter 5 of reference [155].

If we consider the magnitude of the response $y(k)$, it is easy to see that

$$|y(k)| = \left| \sum_{j=0}^{\infty} g(j) r(k-j) \right|$$

which is surely less than the sum of the magnitudes as given by

$$|y(k)| \leq \sum_{j=0}^{\infty} |g(j)|\,|r(k-j)|$$

Now take a bounded input, i.e.,

$$|r(k)| < M;\ 0 \leq k < \infty$$

where M is an arbitrary but finite positive constant. With this input,

$$|y(k)| \leq M \sum_{j=0}^{\infty} |g(j)|$$

and if condition (2.68) holds true, $|y(k)|$ is finite; hence the output is bounded and the system is BIBO stable.

The condition (2.68) is also necessary, for if we consider the bounded input

$$r(k-j) = \begin{cases} +1 & \text{if} \quad g(j) > 0 \\ 0 & \text{if} \quad g(j) = 0 \\ -1 & \text{if} \quad g(j) < 0 \end{cases}$$

then the output at any fixed value of k is given by

$$|y(k)| = \left| \sum_{j=0}^{\infty} g(j) r(k-j) \right| = \sum_{j=0}^{\infty} |g(j)|$$

Thus, unless the condition given by (2.68) is true, the system is not BIBO stable.

The condition (2.68) for BIBO stability can be translated into a set of restrictions on the location of poles of the transfer function $G(z)$ in the z-plane. Consider the discrete-time system shown in Fig. 2.24. The block-diagram analysis gives the following input-output relation for this system.

$$\frac{Y(z)}{R(z)} = G(z) = \frac{z}{z^2 + a_1 z + a_2}$$

or

$$Y(z) = \frac{z}{z^2 + a_1 z + a_2} R(z)$$

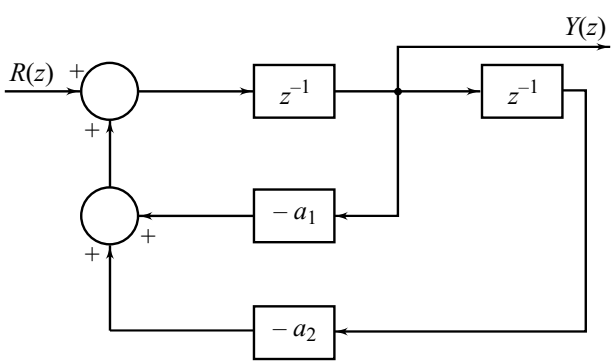


Fig. 2.24 A simple discrete-time system

For the impulse input, $R(z) = 1$. Therefore, response transform

$$Y(z) = \frac{z}{z^2 + a_1 z + a_2}$$

The impulse response of the system is given by

$$y(k) = g(k) = \mathcal{Z}^{-1}\left[\frac{z}{z^2 + a_1 z + a_2}\right]$$

Assume that the poles of the response transform $Y(z)$ are real and distinct:

$$z^2 + a_1 z + a_2 = (z - \alpha_1)(z - \alpha_2)$$

Partial fraction expansion of $Y(z)/z$ is then of the form

$$\frac{Y(z)}{z} = \frac{A_1}{z - \alpha_1} + \frac{A_2}{z - \alpha_2}$$

where A_1 and A_2 are real constants.

This gives $$Y(z) = \frac{A_1 z}{z - \alpha_1} + \frac{A_2 z}{z - \alpha_2};\ y(k) = A_1(\alpha_1)^k + A_2(\alpha_2)^k$$

The time functions $(\alpha_1)^k$ and $(\alpha_2)^k$ are the response functions contributed by the system poles at $z = \alpha_1$ and $z = \alpha_2$, respectively. These time functions dictate the qualitative nature of the impulse response of the system.

A time function $(\alpha)^k$ either *grows* or *decays* depending on $|\alpha| > 1$ or $|\alpha| < 1$, respectively. The growth or decay is monotonic when α is positive and alternates in sign when α is negative. $(\alpha)^k$ remains constant for $\alpha = 1$ and alternates in sign with constant amplitude for $\alpha = -1$ (Fig. 2.25).

Consider now the situation wherein the poles of response transform $Y(z)$ are complex.

For the complex-conjugate pole pair at $z = p = R_0 e^{j\Omega}$ and $z = p^* = R_0 e^{-j\Omega}$ of $Y(z)$, the response $y(k)$ is obtained as follows:

$$\frac{Y(z)}{z} = \frac{A}{z - p} + \frac{A^*}{z - p^*}$$

where $A = |A| \angle\phi$ and A^* is complex-conjugate of A.

The impulse response

$$\begin{aligned} y(k) &= A(p)^k + A^*(p^*)^k = A(p)^k + [A(p)^k]^* \\ &= 2\text{Re}\,[A\,(p)^k] = 2\text{Re}\,[|A|\, e^{j\phi}\, R_0^k\, e^{j\Omega k}] \\ &= 2|A|\, R_0^k\ \text{Re}[e^{j(\Omega k + \phi)}] = 2|A|\, R_0^k \cos(\Omega k + \phi) \end{aligned}$$

Therefore, the complex-conjugate pair of poles of the response transform $Y(z)$ gives rise to a sinusoidal or oscillatory response function $R_0^k \cos(\Omega k + \phi)$, whose envelope R_0^k can be constant, growing or decaying depending on whether $R_0 = 1$, $R_0 > 1$, or $R_0 < 1$, respectively (Fig. 2.26).

For an nth-order linear discrete-time system, the response transform $Y(z)$ has an nth-order characteristic polynomial. Assume that $Y(z)$ has a real pole at $z = \alpha$ of multiplicity m, and partial fraction expansion of

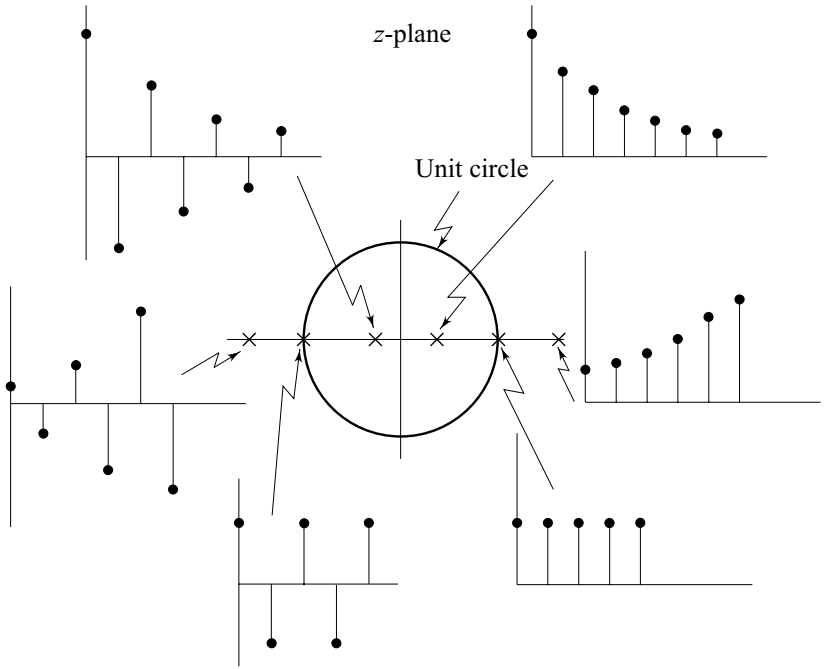


Fig. 2.25 The nature of response function $(\alpha)^k$ for different values of α

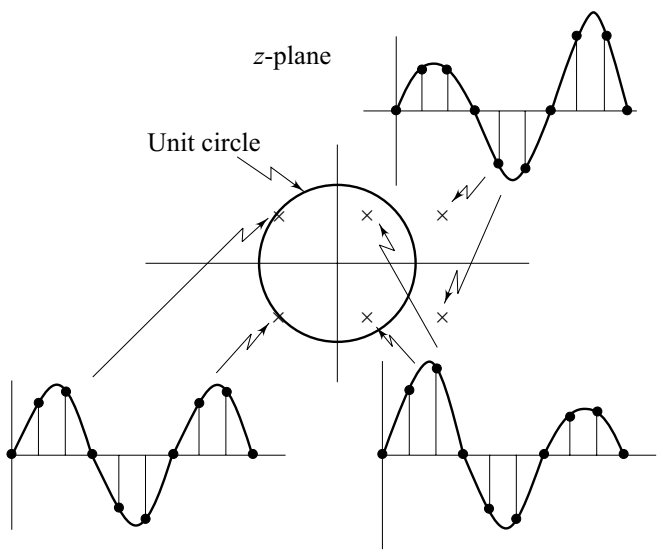


Fig. 2.26 The nature of response function $(R_0)^k \cos(\Omega k + \phi)$ for different values of R_0

$Y(z)$ is of the form

$$Y(z) = \frac{A_{1(m)}z}{(z-\alpha)^m} + \frac{A_{1(m-1)}z}{(z-\alpha)^{m-1}} + \cdots + \frac{A_{12}z}{(z-\alpha)^2} + \frac{A_{11}z}{(z-\alpha)} \tag{2.69}$$

where $A_{1(m)}$, ..., A_{12}, A_{11} are real constants.

Response functions contributed by the real pole of multiplicity m can be evaluated as follows:

Consider the transform pair

$$\frac{z}{z-\alpha} \leftrightarrow (\alpha)^k$$

Application of the shifting theorem (Eqn. (2.47)) gives

$$\frac{1}{z-\alpha} \leftrightarrow \alpha^{k-1}\mu(k-1)$$

Using Eqn. (2.41),

$$-z\frac{d}{dz}\left(\frac{1}{z-\alpha}\right) \leftrightarrow k\alpha^{k-1}\mu(k-1) \quad \text{or} \quad \frac{z}{(z-\alpha)^2} \leftrightarrow k\,\alpha^{k-1}\mu(k-1)$$

From this pair, we may write (Eqn. (2.47)),

$$\frac{1}{(z-\alpha)^2} \leftrightarrow (k-1)\alpha^{k-2}\mu(k-2)$$

Performing differentiation operation once again, we have

$$-z\frac{d}{dz}\left[\frac{1}{(z-\alpha)^2}\right] \leftrightarrow k(k-1)\alpha^{k-2}\mu(k-2)$$

or

$$\frac{z}{(z-\alpha)^3} \leftrightarrow \frac{1}{2!}k(k-1)\alpha^{k-2}\mu(k-2)$$

In general,

$$\frac{z}{(z-\alpha)^m} \leftrightarrow \frac{k!(\alpha)^{k-m+1}}{(k-m+1)!\,(m-1)!}\mu(k-m+1) \tag{2.70}$$

It can easily be established using final value theorem (Eqn. (2.52)) that each response function in Eqn. (2.69) equals zero as $k \to \infty$ if $|\alpha| < 1$. However, the response functions in Eqn. (2.69) grow without bound for $|\alpha| \geq 1$.

Similar conclusions can be derived in case of response functions corresponding to complex-conjugate pair of poles ($R_0 e^{\pm j\Omega}$) of multiplicity m. The limit of each response function as $k \to \infty$ equals zero if $R_0 < 1$. The case of $R_0 \geq 1$ contributes growing response functions.

From the foregoing discussion it follows that the nature of the response terms contributed by the system poles (i.e., the poles of the transfer function $G(z)$), gives the nature of the impulse response $g(k)$ ($= \mathcal{Z}^{-1}[G(z)]$) of the system. This, therefore, answers the question of BIBO stability through condition (2.68), which says that for a system with transfer function $G(z)$ to be BIBO stable, it is necessary and

sufficient that

$$\sum_{k=0}^{\infty} |g(k)| < \infty$$

The nature of response terms contributed by various types of poles of $G(z) = \dfrac{N(z)}{\Delta(z)}$, i.e., the roots of the characteristic equation $\Delta(z) = 0$, has already been investigated. Observing the nature of response terms carefully, leads us to the following general conclusions on BIBO stability.

(i) If all the roots of the characteristic equation lie *inside* the unit circle in the z-plane, then the impulse response is bounded and eventually decays to zero. Therefore, $\sum_{k=0}^{\infty} |g(k)|$ is finite and the system is BIBO stable.

(ii) If any root of the characteristic equation lies *outside* the unit circle in the z-plane, $g(k)$ grows without bound and $\sum_{k=0}^{\infty} |g(k)|$ is infinite. The system is, therefore, unstable.

(iii) If the characteristic equation has repeated roots on the unit circle in the z-plane, $g(k)$ grows without bound and $\sum_{k=0}^{\infty} |g(k)|$ is infinite. The system is, therefore, unstable.

(iv) If one or more nonrepeated roots of the characteristic equation are on the unit circle in the z-plane, then $g(k)$ is bounded but $\sum_{k=0}^{\infty} |g(k)|$ is infinite. The system is, therefore, unstable.

An exception to the definition of BIBO stability is brought out by the following observations. Consider a system with transfer function

$$G(z) = \frac{N(z)}{(z-1)(z-e^{j\Omega})(z-e^{-j\Omega})}$$

The system has nonrepeated poles on the unit circle in the z-plane. The response functions contributed by the system poles at $z = 1$ and $z = e^{\pm j\Omega}$ are respectively $(1)^k$ and $\cos(\Omega k + \phi)$. The terms $(1)^k$ and $\cos(\Omega k + \phi)$ are bounded, $\sum_{k=0}^{\infty} |g(k)|$ is infinite and the system is unstable in the sense of our definition of BIBO stability.

Careful examination of the input-output relation

$$Y(z) = G(z)R(z) = \frac{N(z)}{(z-1)(z-e^{j\Omega})(z-e^{-j\Omega})}\ R(z)$$

shows that $y(k)$ is bounded for all bounded $r(k)$, unless the input has a pole matching one of the system poles on the unit circle. For example, for a unit-step input $r(k) = \mu(k)$,

$$R(z) = \frac{z}{z-1} \text{ and } Y(z) = \frac{zN(z)}{(z-1)^2(z-e^{j\Omega})(z-e^{-j\Omega})}$$

The response $y(k)$ is a linear combination of the terms $\cos(\Omega k + \phi)$, $(1)^k$, and $k(1)^k$, and therefore, $y(k) \to \infty$ as $k \to \infty$. Such a system, which has bounded output for all bounded inputs, except for the inputs having poles matching the system poles, may be treated as acceptable or non-acceptable. We will bring the situations where the system has nonrepeated poles on the unit circle under the class of *marginally stable systems*.

2.9.2 Zero-Input Stability

This concept of stability is based on the dynamic evolution of the system state in response to arbitrary initial state representing initial internal energy storage. State variable model (refer to Eqn. (2.17))

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) \tag{2.71}$$

is the most appropriate for the study of dynamic evolution of the state $\mathbf{x}(k)$ in response to the initial state $\mathbf{x}(0)$.

We may classify stability as follows:

(i) *Unstable:* There is at least one finite initial state $\mathbf{x}(0)$ such that $\mathbf{x}(k)$ grows thereafter without being bound as $k \to \infty$.

(ii) *Asymptotically stable:* For all possible initial states $\mathbf{x}(0)$, $\mathbf{x}(k)$ eventually decays to zero as $k \to \infty$.

(iii) *Marginally stable:* For all initial states $\mathbf{x}(0)$, $\mathbf{x}(k)$ remains thereafter within finite bounds for $k > 0$.

Taking z-transform on both the sides of Eqn. (2.71) yields

$$z\,\mathbf{X}(z) - z\,\mathbf{x}(0) = \mathbf{F}\mathbf{X}(z) \quad \text{where} \quad \mathbf{X}(z) \triangleq \mathscr{Z}[\mathbf{x}(k)]$$

Solving for $\mathbf{X}(z)$, we get

$$\mathbf{X}(z) = (z\mathbf{I} - \mathbf{F})^{-1} z\mathbf{x}(0) = \Phi(z)\,\mathbf{x}(0)$$

where

$$\Phi(z) = (z\mathbf{I} - \mathbf{F})^{-1} z = \frac{(z\mathbf{I} - \mathbf{F})^{+} z}{|z\mathbf{I} - \mathbf{F}|} \tag{2.72a}$$

The state vector $\mathbf{x}(k)$ can be obtained by inverse transforming $\mathbf{X}(z)$:

$$\mathbf{x}(k) = \mathscr{Z}^{-1}[\Phi(z)]\,\mathbf{x}(0) \tag{2.72b}$$

Note that for an $n \times n$ matrix $\mathbf{F}$, $|z\mathbf{I} - \mathbf{F}|$ is an nth-order polynomial in z. Also, each element of the *adjoint* matrix $(z\mathbf{I} - \mathbf{F})^{+}$ is a polynomial in z of order less than or equal to $(n - 1)$. Therefore, each element of $\Phi(z)/z$ is strictly a proper rational function, and can be expanded in a partial fraction expansion. Using the time-response analysis given earlier in this section, it is easy to establish that

$$\lim_{k \to \infty} \mathbf{x}(k) \to \mathbf{0}$$

if all the roots of the characteristic polynomial $|z\mathbf{I} - \mathbf{F}|$, lie strictly inside the unit circle of the complex plane. In Chapter 6 we will see that under mildly restrictive conditions (namely, the system must be both controllable and observable), the roots of the characteristic polynomial $|z\mathbf{I} - \mathbf{F}|$ are same as the poles of the corresponding transfer function, and asymptotic stability ensures BIBO stability and vice versa. This implies that stability analysis can be carried out using the BIBO stability test (or only the asymptotic stability test).

We will use the following terminology and tests for stability analysis of linear time-invariant systems described by the transfer function $G(z) = N(z)/\Delta(z)$, with the characteristic equation $\Delta(z) = 0$:

(i) If all the roots of the characteristic equation lie inside the unit circle in the z-plane, the system is *stable*.

(ii) If any root of the characteristic equation lies outside the unit circle in the z-plane, or if there is a repeated root on the unit circle, the system is *unstable*.

(iii) If condition (i) is satisfied except for the presence of one or more nonrepeated roots on the unit circle in the z-plane, the system is *marginally stable*.

It follows from the above discussion that stability can be established by determining the roots of the characteristic equations. All the commercially available CAD packages ([151–154]) include root-solving routines. However, there exist tests for determining the stability of a discrete-time system, without finding the actual numerical values of the roots of the characteristic equation.

A well-known criterion to test the location of zeros of the polynomial

$$\Delta(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$$

where a's are real coefficients, is the *Jury stability criterion*. The proof of this criterion is quite involved and is given in the literature (Jury and Blanchard [98]). The criterion gives the necessary and sufficient conditions for the roots to lie inside the unit circle. In the following, we present the Jury stability criterion without proof.

2.9.3 The Jury Stability Criterion

In applying the Jury stability criterion to a given characteristic equation $\Delta(z) = 0$, we construct a table whose elements are based on the coefficients of $\Delta(z)$.

Consider the general form of the characteristic polynomial $\Delta(z)$ (refer to Eqn. (2.60b)):

$$\Delta(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_k z^{n-k} + \cdots + a_{n-1} z + a_n;\ a_0 > 0 \tag{2.73}$$

The criterion uses the Jury stability table given in Table 2.2.

The Jury stability table is formed using the following rules:

(i) The first two rows of the table consist of the coefficients of $\Delta(z)$, arranged in ascending order of power of z in row 1, and in reverse order in row 2.

(ii) All even-numbered rows are simply the reverse of the immediately preceding odd-numbered rows.

(iii) The elements for rows 3 through $(2n - 3)$ are given by the following determinants:

$$\begin{aligned}
b_k &= \begin{vmatrix} a_n & a_{n-1-k} \\ a_0 & a_{k+1} \end{vmatrix};\ k = 0, 1, 2, \ldots, n-1 \\
c_k &= \begin{vmatrix} b_{n-1} & b_{n-2-k} \\ b_0 & b_{k+1} \end{vmatrix};\ k = 0, 1, 2, \ldots, n-2 \\
&\vdots \\
q_k &= \begin{vmatrix} p_3 & p_{2-k} \\ p_0 & p_{k+1} \end{vmatrix};\ k = 0, 1, 2
\end{aligned} \tag{2.74}$$

Table 2.2 General form of the Jury stability table

Row	z^0	z^1	z^2	z^3	⋯	z^k	⋯	z^{n-2}	z^{n-1}	z^n
1	a_n	a_{n-1}	a_{n-2}	a_{n-3}	⋯	a_{n-k}	⋯	a_2	a_1	a_0
2	a_0	a_1	a_2	a_3	⋯	a_k	⋯	a_{n-2}	a_{n-1}	a_n
3	b_{n-1}	b_{n-2}	b_{n-3}	b_{n-4}	⋯	b_{n-k-1}	⋯	b_1	b_0	
4	b_0	b_1	b_2	b_3	⋯	b_k	⋯	b_{n-2}	b_{n-1}	
5	c_{n-2}	c_{n-3}	c_{n-4}	c_{n-5}	⋯	c_{n-k-2}	⋯	c_0		
6	c_0	c_1	c_2	c_3	⋯	c_k	⋯	c_{n-2}		
⋮										
$2n-5$	p_3	p_2	p_1	p_0						
$2n-4$	p_0	p_1	p_2	p_3						
$2n-3$	q_2	q_1	q_0							

The procedure is continued until the $(2n-3)$rd row is reached which will contain exactly three elements.

The necessary and sufficient conditions for polynomial $\Delta(z)$ to have no roots on and outside the unit circle in the z-plane are:

$$\Delta(1) > 0$$

$$\Delta(-1) \begin{cases} > 0 & \text{for } n \text{ even} \\ < 0 & \text{for } n \text{ odd} \end{cases} \tag{2.75a}$$

$$|a_n| < |a_0|$$

$$\left.\begin{array}{l} |b_{n-1}| > |b_0| \\ |c_{n-2}| > |c_0| \\ \quad\vdots \\ |q_2| > |q_0| \end{array}\right\} (n-2) \text{ constraints} \tag{2.75b}$$

The conditions on $\Delta(1)$, $\Delta(-1)$, and between a_0 and a_n in (2.75a) form *necessary conditions* of stability that are very simple to check without carrying out the Jury tabulation.

It should be noted that the test of stability given in (2.75) is valid only if the inequality conditions provide conclusive results. Jury tabulation ends prematurely if, either the first and the last elements of a row are zero, or, a complete row is zero. These cases are referred to as *singular cases*. These problems can be resolved by expanding and contracting the unit circle infinitesimally, which is equivalent to moving the roots off the unit circle. The transformation for this purpose is

$$\hat{z} = (1 + \varepsilon)z$$

where ε is a very small real number.

This transformation can easily be applied since

$$(1 + \varepsilon)^n z^n \cong (1 + n\varepsilon)z^n$$

When ε is a positive number, the radius of the unit circle is expanded to $(1 + \varepsilon)$, and when ε is negative, the radius of the unit circle is reduced to $(1 - |\varepsilon|)$. This is equivalent to moving the roots slightly. The difference between the number of roots found inside (or outside) the unit circle when the circle is expanded and contracted by ε, is the number of roots on the circle [97].

Example 2.12

Consider the characteristic polynomial

$$\Delta(z) = 2z^4 + 7z^3 + 10z^2 + 4z + 1$$

Employing stability constraints (2.75a), we get

(i) $\Delta(1) = 2 + 7 + 10 + 4 + 1 = 24 > 0$; satisfied
(ii) $\Delta(-1) = 2 - 7 + 10 - 4 + 1 = 2 > 0$; satisfied
(iii) $|1| < |2|$; satisfied

Next, we construct the Jury table:

Row	z^0	z^1	z^2	z^3	z^4
1	1	4	10	7	2
2	2	7	10	4	1
3	–3	–10	–10	–1	
4	–1	–10	–10	–3	
5	8	20	20		

Employing stability constraints (2.75b), we get

(i) $|-3| > |-1|$; satisfied
(ii) $|8| > |20|$; not satisfied

The system is, therefore, unstable.

Usefulness of the Jury stability test for the design of a digital control system from the stability point of view, is demonstrated in the next chapter.

The Jury criterion is of marginal use in designing a feedback system; it falls far short of the root locus method discussed in Chapter 4. The root locus technique of factoring a polynomial is intrinsically geometric, as opposed to the algebraic approach of the Jury criterion. The root locus method enables us to rapidly sketch the locus of all solutions to the characteristic polynomial of the closed-loop transfer function. The sketch is usually only qualitative, but even so, it offers great insight by showing how the locations of the poles of closed-loop transfer function change as the gain is varied. As we will see in Chapter 4, the root locus approach has been reduced to a set of 'rules'. Applied in an orderly fashion, these rules quickly identify all closed-loop pole locations.

2.10 SAMPLE-AND-HOLD SYSTEMS

The *sampling operation*—conversion of a continuous-time function to a sequence—has earlier been studied in Section 2.6. We developed an impulse modulation model for the sampling operation (refer to Fig. 2.19).

It is important to emphasize here that the impulse modulation model is a mathematical representation of sampling; not a representation of any physical system designed to implement the sampling operation. We have introduced this representation of the sampling operation because it leads to a simple derivation of a key result on sampling (given in the next section) and because this approach allows us to obtain a transfer function model of the *hold operation*.

2.10.1 The Hold Operation

It is the inverse of the sampling operation—conversion of a sequence to a continuous-time function. In computer-controlled systems, it is necessary to convert the control actions calculated by the computer as a sequence of numbers, to a continuous-time signal that can be applied to the process.

The problem of hold operation may be posed as follows:

Given a sequence $\{y(0), y(1), ..., y(k), ...\}$, we have to construct $y_a(t)$, $t \geq 0$.

A commonly used solution to the problem of hold operation, is polynomial extrapolation. Using Taylor's series expansion about $t = kT$, we can express $y_a(t)$ as

$$y_a(t) = y_a(kT) + \dot{y}_a(kT)\,(t - kT) + \frac{\ddot{y}_a(kT)}{2!}\,(t - kT)^2 + \cdots;\; kT \leq t < (k+1)T \tag{2.76}$$

where

$$\dot{y}_a(kT) \triangleq \left.\frac{dy_a(t)}{dt}\right|_{t = kT} \cong \frac{1}{T}[y_a(kT) - y_a((k-1)T)]$$

$$\ddot{y}_a(kT) \triangleq \left.\frac{d^2 y_a(t)}{dt^2}\right|_{t = kT} \cong \frac{1}{T}[\,\dot{y}_a(kT) - \dot{y}_a((k-1)T)]$$

$$= \frac{1}{T^2}[y_a(kT) - 2y_a((k-1)T) + y_a((k-2)T)]$$

If only the first term in expansion (2.76) is used, the data hold is called a *zero-order hold* (ZOH). Here we assume that the function $y_a(t)$ is approximately constant within the sampling interval, at a value equal to that of the function at the preceding sampling instant. Therefore, for a given input sequence $\{y(k)\}$, the output of ZOH is given by

$$y_a(t) = y(k);\; kT \leq t < (k+1)T \tag{2.77}$$

The first two terms in Eqn. (2.76) are used to realize the *first-order hold*. For a given input sequence $\{y(k)\}$, the output of the first-order hold is given by

$$y_a(t) = y(k) + \frac{t - kT}{T}[y(k) - y(k-1)] \tag{2.78}$$

It is obvious from Eqn. (2.76) that the higher the order of the derivative to be approximated, the larger will be the number of delay pulses required. The time-delay adversely affects the stability of feedback control systems. Furthermore, a high-order extrapolation requires complex circuitry and results in high costs of construction. The ZOH is the simplest, and most commonly used, data hold device. The standard D/A converters are often designed in such a way that the old value is held constant until a new conversion is ordered.

2.10.2 A Model of Sample-and-Hold Operation

In the digital control structure of Fig. 2.2, discrete-time processing of continuous-time signals is accomplished by the system depicted in Fig. 2.27. The system is a cascade of an A/D converter followed by a discrete-time system (computer program), followed by a D/A converter. Note that the overall system is equivalent to a continuous-time system, since it transforms the continuous-time input signal $x_a(t)$ into the continuous-time signal $y_a(t)$. However, the properties of the system are dependent on the choice of the discrete-time system and the sampling rate.

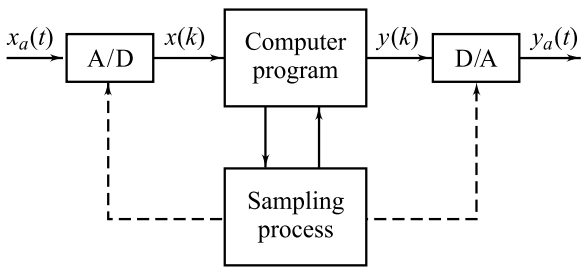


Fig. 2.27 Discrete-time processing of continuous-time signals

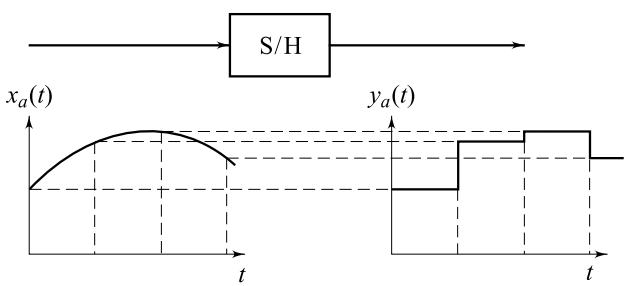


Fig. 2.28 Input-output behavior of a sample-and-hold system

In the special case of discrete-time signal processing with a unit-gain algorithm, and negligible time delay (i.e., $y(k) = x(k)$), the combined action of the A/D converter, the computer, and the D/A converter can be described as a system that samples the analog signal and produces another analog signal that is constant over the sampling periods. Such a system is called a *sample-and-hold* (S/H) system. Input-output behavior of an S/H system is described diagrammatically in Fig. 2.28. In the following, we develop an idealized model for S/H systems.

S/H operations require modeling of the following two processes:

(i) extracting the samples, and
(ii) holding the result fixed for one period.

The impulse modulator effectively extracts the samples in the form of $x(k)\,\delta(t - kT)$. The remaining problem is to construct a linear time-invariant system which will convert this impulse into a pulse of height $x(k)$ and width T. The S/H may, therefore, be modeled by Fig. 2.29a, wherein the ZOH is a system whose response to a unit impulse $\delta(t)$ is a unit pulse $g_{h0}(t)$ of width T. The Laplace transform of the impulse response $g_{h0}(t)$ is the transfer function of the hold operation, namely,

$$G_{h0}(s) = \mathscr{L}^{-1}[g_{h0}(t)] = \int_0^{\infty} g_{h0}(t)\, e^{-st} dt = \int_0^{T} e^{-st}\, dt = \frac{1 - e^{-sT}}{s} \tag{2.79}$$

Figure 2.29b is a block diagram representation of the transfer function model of the S/H operation.

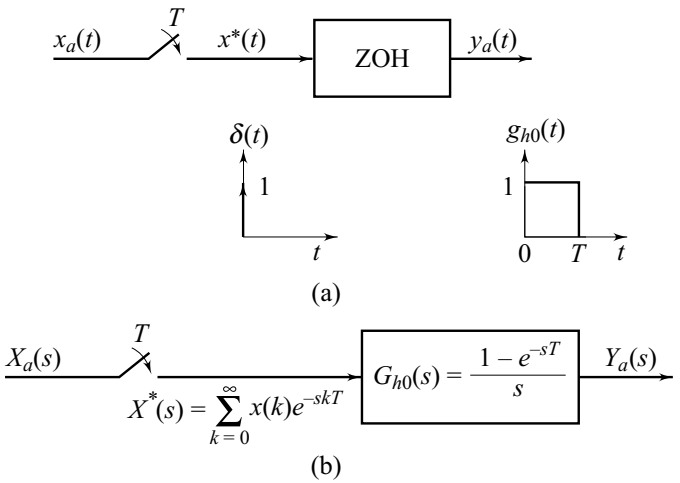


Fig. 2.29 A model of a sample-and-hold operation

Practical Sample-and-Hold Circuit

In a majority of practical digital operations, S/H functions are performed by a single S/H device. It consists of a capacitor, an electronic switch, and operational amplifiers (Fig. 2.30). Op amps are needed for isolation; the capacitor and switch cannot be connected directly to analog circuitry because of the capacitor's effect on the driving waveform.

Since the voltage between the inverting and non-inverting inputs of an op amp is measured in microvolts, we can approximate this voltage to zero. This implies that the voltage from the inverting input (−input) to ground in Fig. 2.30 is approximately V_{IN}; therefore, the output of first op amp is approximately V_{IN}.

When the switch is closed, the capacitor rapidly charges to V_{IN}, and V_{OUT} is equal to V_{IN} approximately. When the switch opens, the capacitor retains its charge; the output holds at a value of V_{IN}.

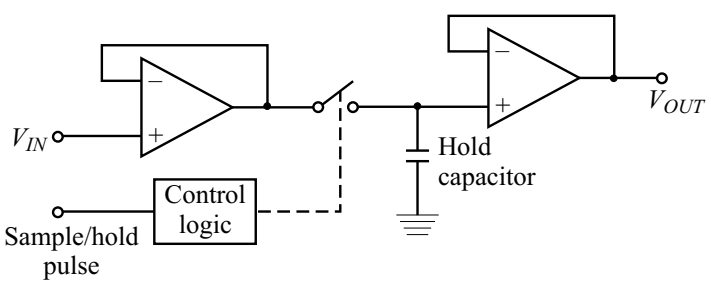


Fig. 2.30 A sample-and-hold device

If the input voltage changes rapidly while the switch is closed, the capacitor can follow this voltage because the charging time-constant is very short. If the switch is suddenly opened, the capacitor voltage represents a sample of the input voltage at the instant the switch was opened. The capacitor then holds this sample until the switch is again closed and a new sample is taken.

As an illustration of the application of a sampler/ZOH circuit, consider the A/D conversion system of Fig. 2.31. The two subsystems in this figure correspond to systems that are available as physical devices. The A/D converter converts a voltage (or current) amplitude at its input into a binary code representing a quantized amplitude value closest to the amplitude of the input. However, the conversion is not instantaneous. Input signal variation during the conversion time of the A/D converter (typical conversion times of commercial A/D units range from 100 *n*sec to 200 μsec), can lead to erroneous results. For this reason, a high performance A/D conversion system includes an S/H device, as shown in Fig. 2.31.

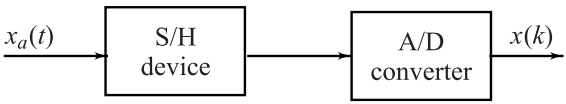


Fig. 2.31 Physical configuration for A/D conversion

Although an S/H is available commercially as one unit, it is advantageous to treat the sampling and holding operations separately for analytical purposes, as has been done in the S/H model of Fig. 2.29b. This model gives the defining equation of the sampling process and the transfer function of the ZOH. It may be emphasized here that $X^*(s)$ is not present in the physical system but appears in the mathematical model; the sampler in Fig. 2.29 does not model a physical sampler and the block does not model a physical data hold. However, the combination does accurately model a sampler/ZOH device.

2.11 SAMPLED SPECTRA AND ALIASING

We can get further insight into the process of sampling by relating the spectrum of the continuous-time signal to that of the discrete-time sequence, which is obtained by sampling.

Let us define the continuous-time signal by $x_a(t)$. Its spectrum is then given by $X_a(j\omega)$, where ω is the frequency in radians per second. The sequence $x(k)$ with value $x(k) = x_a(kT)$ is derived from $x_a(t)$ by periodic sampling. Spectrum of $x(k)$ is given by $X(e^{j\Omega})$ where the frequency Ω has units of radians per sample interval.

The Laplace transform expresses an analog signal $x_a(t)$ as a continuous sum of exponentials e^{st}; $s = \sigma + j\omega$. The Fourier transform expresses $x_a(t)$ as a continuous sum of exponentials $e^{j\omega t}$. Similarly z-transform expresses a sequence $x(k)$ as a discrete sum of phasors z^{-k}; $z = re^{j\Omega}$. Fourier transform expresses $x(k)$ as a discrete sum of exponentials $e^{j\Omega k}$ [31].

The Fourier transforms of $x_a(t)$ and $x(k)$ are, respectively,

$$X_a(j\omega) = \int_{-\infty}^{\infty} x_a(t)\, e^{-j\omega t}\, dt \tag{2.80}$$

$$X(e^{j\Omega}) = \sum_{k=-\infty}^{\infty} x(k)\, e^{-j\Omega k} \tag{2.81}$$

We use the intermediate function $x^*(t)$—the impulse modulated $x_a(t)$—to establish a relation between $X_a(j\omega)$ and $X(e^{j\Omega})$.

The Fourier transform of $x^*(t)$, denoted by $X^*(j\omega)$, is (refer to Eqn. (2.30)) given by

$$X^*(j\omega) = \int_{-\infty}^{\infty} x^*(t) e^{-j\omega t}\, dt = \int_{-\infty}^{\infty} \left[\sum_{k=0}^{\infty} x(k)\, \delta(t - kT) \right] e^{-j\omega t}\, dt$$

$$= \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t - kT)\, x(k)\, e^{-j\omega t}\, dt = \sum_{k=-\infty}^{\infty} x(k) e^{-jk\omega T} \tag{2.82}$$

(The summation over the interval $-\infty$ to ∞ is allowed, since $x(k) = 0$ for $k < 0$). We have arrived at our first intermediate result. By comparing Eqn. (2.82) with Eqn. (2.81), we observe that

$$X(e^{j\Omega}) = X^*(j\omega)\big|_{\omega = \frac{\Omega}{T}} \tag{2.83a}$$

$X(e^{j\Omega})$ is thus a *frequency-scaled version* of $X^*(j\omega)$ with the frequency scaling specified by

$$\Omega = \omega T \tag{2.83b}$$

We now determine $X^*(j\omega)$ in terms of the continuous-time spectrum $X_a(j\omega)$. From Eqn. (2.30), we have

$$x^*(t) = \sum_{k=0}^{\infty} x(k)\, \delta(t - kT) = \sum_{k=0}^{\infty} x_a(t)\, \delta(t - kT) = x_a(t) \sum_{k=0}^{\infty} \delta(t - kT)$$

The summation over the interval $-\infty$ to ∞ is allowed since $x_a(t) = 0$ for $t < 0$.

Therefore,
$$x^*(t) = x_a(t) \sum_{k=-\infty}^{\infty} \delta(t - kT)$$

A nonsinusoidal periodic signal $y(t)$ of period T_0 can be expanded through the Fourier series [31] as

$$y(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t};\ \omega_0 = 2\pi/T_0$$

$$c_n = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} y(t) e^{-jn\omega_0 t}\, dt$$

Since $\sum_{k=-\infty}^{\infty} \delta(t - kT)$ is a periodic function of period T, it can be expressed in terms of the following Fourier series expansion.

$$\sum_{k=-\infty}^{\infty} \delta(t - kT) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n t}{T}}$$

where

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} \left[\sum_{k=-\infty}^{\infty} \delta(t - kT) \right] e^{-j\frac{2\pi n t}{T}}\, dt$$

$$= \frac{1}{T} \int_{-T/2}^{T/2} \delta(t) e^{-j\frac{2\pi n t}{T}}\, dt = \frac{1}{T} e^{-j0} = \frac{1}{T} \text{ for all } n$$

Substituting this Fourier series expansion into the impulse modulation process, we get

$$x^*(t) = x_a(t) \sum_{k=-\infty}^{\infty} \delta(t - kT)$$
$$= x_a(t) \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j\frac{2\pi nt}{T}} = \frac{1}{T} \sum_{n=-\infty}^{\infty} x_a(t)\, e^{j\frac{2\pi nt}{T}}$$

The continuous-time spectrum of $x^*(t)$ is then equal to

$$X^*(j\omega) = \int_{-\infty}^{\infty} x^*(t) e^{-j\omega t}\, dt = \frac{1}{T} \int_{-\infty}^{\infty} \left[\sum_{n=-\infty}^{\infty} x_a(t)\, e^{j\frac{2\pi nt}{T}} \right] e^{-j\omega t} dt$$

Interchanging the order of summation and integration, we obtain

$$X^*(j\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \left[\int_{-\infty}^{\infty} x_a(t) e^{-j\left(\omega - \frac{2\pi n}{T}\right)t} dt \right]$$
$$= \frac{1}{T} \sum_{n=-\infty}^{\infty} X_a\left(j\omega - j\frac{2\pi n}{T} \right) \tag{2.84a}$$

where $X_a(j\omega)$ is the Fourier transform of $x_a(t)$.

We see from this equation that $X^*(j\omega)$ consists of periodically repeated copies of $X_a(j\omega)$, scaled by $1/T$. The scaled copies of $X_a(j\omega)$ are shifted by integer multiples of the sampling frequency

$$\omega_s = \frac{2\pi}{T} \tag{2.84b}$$

and then superimposed to produce $X^*(j\omega)$.

Equation (2.84a) is our second intermediate result. Combining this result with that given by Eqn. (2.83a), we obtain the following relations:

$$X^*(j\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a\left(j\omega - j\frac{2\pi k}{T} \right) \tag{2.85a}$$

$$X(e^{j\Omega}) = X^*\left(j\frac{\Omega}{T} \right) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a\left(j\frac{\Omega}{T} - j\frac{2\pi k}{T} \right) \tag{2.85b}$$

2.11.1 Aliasing

While sampling a continuous-time signal $x_a(t)$ to produce the sequence $x(k)$ with values $x(k) = x_a(kT)$, we want to ensure that all the information in the original signal is retained in the samples. There will be no information loss if we can exactly recover the continuous-time signal from the samples. To determine the condition under which there is no information loss, let us consider $x_a(t)$ to be a *band-limited* signal with maximum frequency ω_m, i.e.,

$$X_a(j\omega) = 0 \text{ for } |\omega| > \omega_m \tag{2.86}$$

as shown in Fig. 2.32a. Figure 2.32b shows a plot of $X^*(j\omega)$ under the condition

$$\frac{\omega_s}{2} = \frac{\pi}{T} > \omega_m \tag{2.87a}$$

Figure 2.32c shows the plot of $X(e^{j\Omega})$, which is derived from Fig. 2.32b by simply scaling the frequency axis.

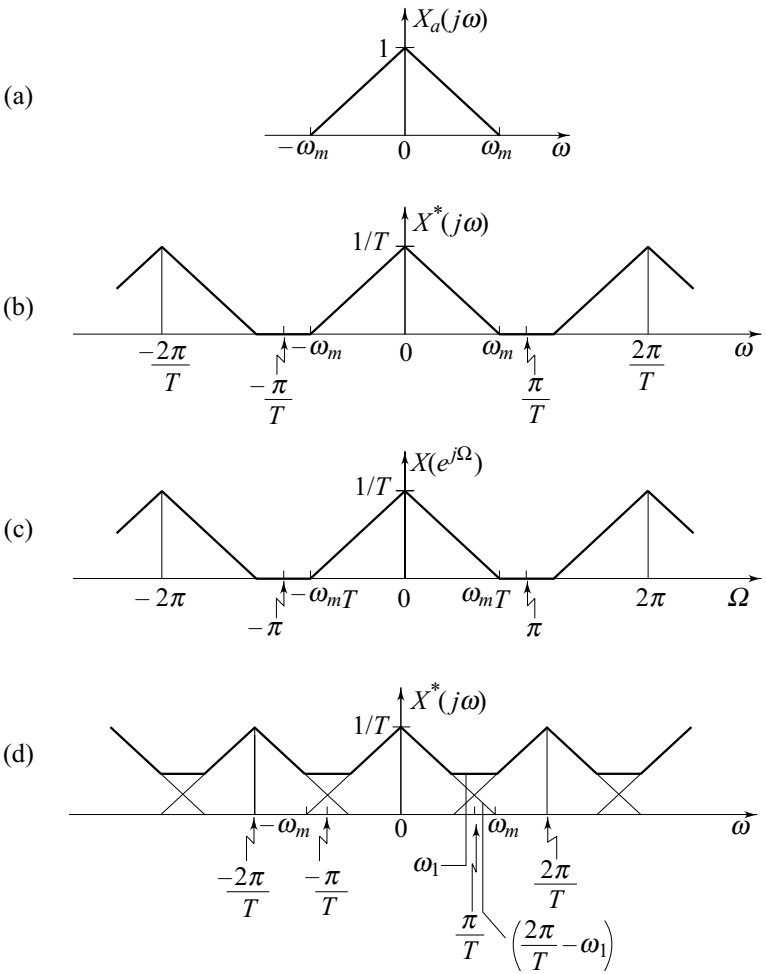


Fig. 2.32 Frequency-domain considerations in sampling and reconstruction

$X^*(j\omega)$ is seen to be a periodic function with period $2\pi/T$ ($X(e^{j\Omega})$ is a periodic function with period 2π). The spectrum $X^*(j\omega)$ for $|\omega| \le \pi/T$ is identical to the continuous-time spectrum $X_a(j\omega)$ except for linear scaling in amplitude (the spectrum $X(e^{j\Omega})$ for $|\Omega| \le \pi$ is identical to the continuous-time spectrum $X_a(j\omega)$, except for linear scaling in amplitude and frequency). The continuous-time signal $x_a(t)$ can be recovered from its samples $x(k)$ without any distortion by employing an ideal low-pass filter (Fig. 2.33).

Figure 2.32d shows a plot of $X^*(j\omega)$ under the condition

$$\frac{\omega_s}{2} = \frac{\pi}{T} < \omega_m \tag{2.87b}$$

The plot of $X(e^{j\Omega})$ can easily be derived from Fig. 2.32d by scaling the frequency axis.

It is obvious from Fig. 2.32d that the shifted versions of $X_a(j\omega)$ overlap; $X^*(j\omega)$ in the range $|\omega| \le \pi/T$ can be viewed as being found by superimposing onto this frequency range, the behavior of the shifted versions of $X_a(j\omega)$.

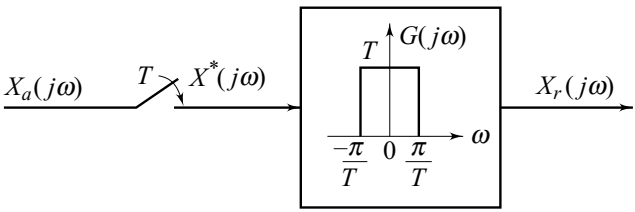


Fig. 2.33 Ideal low-pass filter

Consider an arbitrary frequency point ω_1 in Fig. 2.32d which falls in the region of the overlap of shifted versions of $X_a(j\omega)$. The frequency spectrum at $\omega = \omega_1$ is the sum of two components. One of these, the larger one in the figure, has a value equal to $X_a(j\omega_1)$. The other component comes from the spectrum centered at $2\pi/T$, and has a value equal to $X_a\left(j\left(\frac{2\pi}{T}-\omega_1\right)\right)$. Note that the high frequency $\left(\frac{2\pi}{T}-\omega_1\right)$ is 'folded in' about the *folding frequency* π/T; and appears as low frequency at ω_1. The frequency $\left(\frac{2\pi}{T}-\omega_1\right)$ which shows up at ω_1 after sampling, is called in the trade as the 'alias' of ω_1. The superimposition of the high-frequency behavior onto the low frequency is known as *frequency folding* or *aliasing*. Under the condition given by (2.87b), the form of $X^*(j\omega)$ in the frequency range $|\omega| \le \pi/T$ is no longer similar to $X_a(j\omega)$; therefore, the true spectral shape $X_a(j\omega)$ is no longer recoverable by low-pass filtering (refer to Fig. 2.33). In this case, the reconstructed signal $x_r(t)$ is related to the original signal $x_a(t)$ through a distortion introduced by aliasing and therefore, there is loss of information due to sampling.

Example 2.13

We consider a simple example to illustrate the effects of aliasing.

Figure 2.34a shows a recording of the temperature in a thermal process. From this recording we observe that there is an oscillation in temperature with a period of two minutes.

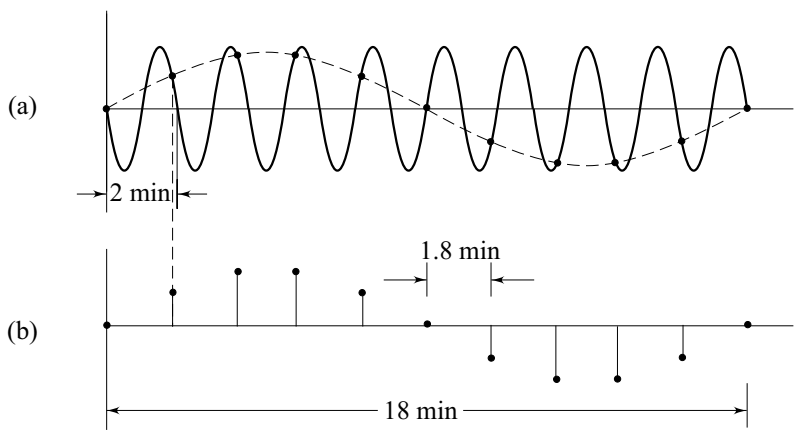


Fig. 2.34 Continuous and sampled recording of temperature

The sampled recording of the temperature obtained by measurement of temperature after every 1.8 minutes is shown in Fig. 2.34b. From the sampled recording, one might believe that there is an oscillation with a period of 18 minutes. There seems to be loss of information because of the process of sampling.

The sampling frequency is $\omega_s = 2\pi/1.8 = \pi/0.9$ rad/min, and the frequency of temperature oscillation is $\omega_0 = 2\pi/2 = \pi$ rad/min. Since ω_0 is greater than $\omega_s/2$, it does not lie in the passband of a low-pass filter with a cut-off frequency $\omega_s/2$. However, the frequency ω_0 is 'folded in' at $\omega_s - \omega_0 = \pi/9$ rad/min which lies in the passband of the low-pass filter. The reconstructed signal has, therefore, a period of 18 minutes, which is the period of the sampled recording.

2.11.2 Sampling Theorem

A corollary to the aliasing problem is the *sampling theorem* stated below.

Let $x_a(t)$ be a band-limited signal with $X_a(j\omega) = 0$ for $|\omega| > \omega_m$. Then $x_a(t)$ is uniquely determined from its samples $x(k) = x_a(kT)$ if the sampling frequency $\omega_s \left(= \frac{2\pi}{T}\right) > 2\omega_m$, i.e., the *sampling frequency must be at least twice the highest frequency present in the signal*.

We will discuss the practical aspects of the choice of sampling frequency in Section 2.13.

2.12 RECONSTRUCTION OF ANALOG SIGNALS

Digital control systems usually require the transformation of discrete-time sequences into analog signals. In such cases, we are faced with the converse problem from that of sampling $x_a(t)$ to obtain $x(k)$. The relevant question now becomes—how can $x_a(t)$ be recovered from its samples.

We begin by considering the *unaliased* spectrum of $X^*(j\omega)$ shown in Fig. 2.32b. $X_a(j\omega)$ has the same form as $X^*(j\omega)$ over $\frac{-\pi}{T} \le \omega \le \frac{\pi}{T}$. $X_a(j\omega)$ can be recovered from $X^*(j\omega)$ by a low-pass filter.

Ideal Low-Pass Filter

Consider the ideal low-pass filter shown in Fig. 2.33. It is characterized by $G(j\omega)$ defined below.

$$G(j\omega) = \begin{cases} T & \text{for } \frac{-\pi}{T} \le \omega \le \frac{\pi}{T} \\ 0 & \text{otherwise} \end{cases} \tag{2.88}$$

Note that the ideal filter given by Eqn. (2.88) has a *zero phase* characteristic. This phase characteristic stems from our requirement that any signal whose frequency components are totally within the passband of the filter, be passed undistorted.

We will need the following basic mathematical background in this section [31].

The Fourier transform pair:

$$\mathcal{F}[y(t)] = Y(j\omega) \triangleq \int_{-\infty}^{\infty} y(t)e^{-j\omega t}\, dt$$

$$\mathscr{F}^{-1}[Y(j\omega)] = y(t) \triangleq \frac{1}{2\pi}\int_{-\infty}^{\infty} Y(j\omega)e^{j\omega t}\, d\omega$$

Shifting theorem:

$$\mathscr{F}\left[y\left(t-\frac{T}{2}\right)\right] = e^{-j\omega T/2}\, Y(j\omega)$$

The impulse response of the ideal low-pass filter is given by *inverse Fourier transformation*.

$$g(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} G(j\omega)\, e^{j\omega t}\, d\omega = \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} Te^{j\omega t} d\omega$$

$$= \frac{T}{j2\pi t}(e^{j\pi t/T} - e^{-j\pi t/T}) = \frac{\sin \pi t/T}{\pi t/T} \tag{2.89}$$

Figure 2.35 shows a plot of $g(t)$ *versus* t. Notice that the response extends from $t = -\infty$ to $t = \infty$. This implies that there is a response for $t < 0$ to a unit impulse applied at $t = 0$ (i.e., the time response that begins before an input is applied). This cannot be true in the physical world. Hence, such an ideal filter is physically unrealizable.

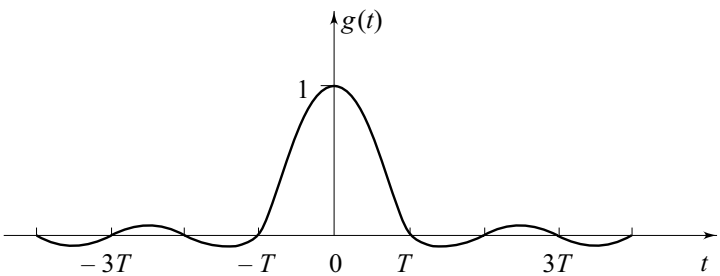


Fig. 2.35 Impulse response of an ideal low-pass filter

Zero-Order Hold

We consider polynomial holds as an approximation to the ideal low-pass filter. The ZOH was considered in Section 2.10, and its transfer function was derived to be (Eqn. (2.79)),

$$G_{h0}(s) = \frac{1-e^{-sT}}{s}$$

Its frequency response is consequently given by

$$G_{h0}(j\omega) = \frac{1-e^{-j\omega T}}{j\omega} = \frac{e^{-j\omega T/2}(e^{j\omega T/2} - e^{-j\omega T/2})}{j\omega}$$

$$= T\frac{\sin(\omega T/2)}{\omega T/2}e^{-j\omega T/2} \tag{2.90}$$

Plot of $\dfrac{\sin(\omega T/2)}{\omega T/2}$ *versus* ω will be of the form shown in Fig. 2.35 with sign reversals at $\omega = \dfrac{2\pi}{T}, \dfrac{4\pi}{T}, \ldots$

(i.e., $\omega = \omega_s, 2\omega_s, \ldots$). The sign reversals amount to a phase shift of –180º (it can be taken as +180º as well) at $\omega = k\omega_s$; $k = 1, 2, \ldots$

Equation (2.90) can, therefore, be expressed as

$$G_{h0}(j\omega) = |G_{h0}(j\omega)| \angle G_{h0}(j\omega)$$

where

$$|G_{h0}(j\omega)| = T \left| \frac{\sin(\omega T/2)}{\omega T/2} \right| \tag{2.91a}$$

and

$$\angle G_{h0}(j\omega) = \left(\frac{-\omega T}{2} - 180° \right) \text{ at } \omega = \frac{2\pi k}{T}; \; k = 1, 2, \ldots \tag{2.91b}$$

A plot of magnitude and phase characteristics of ZOH are shown in Fig. 2.36. The ideal low-pass filter is shown by dashed lines in Fig. 2.36a. The phase of the ideal filter, at all frequencies, is zero.

It is obvious that the hold device does not have the ideal filter characteristics.

(i) The ZOH begins to attenuate at frequencies considerably below $\omega_s/2$.

(ii) The ZOH allows high frequencies to pass through, although they are attenuated.

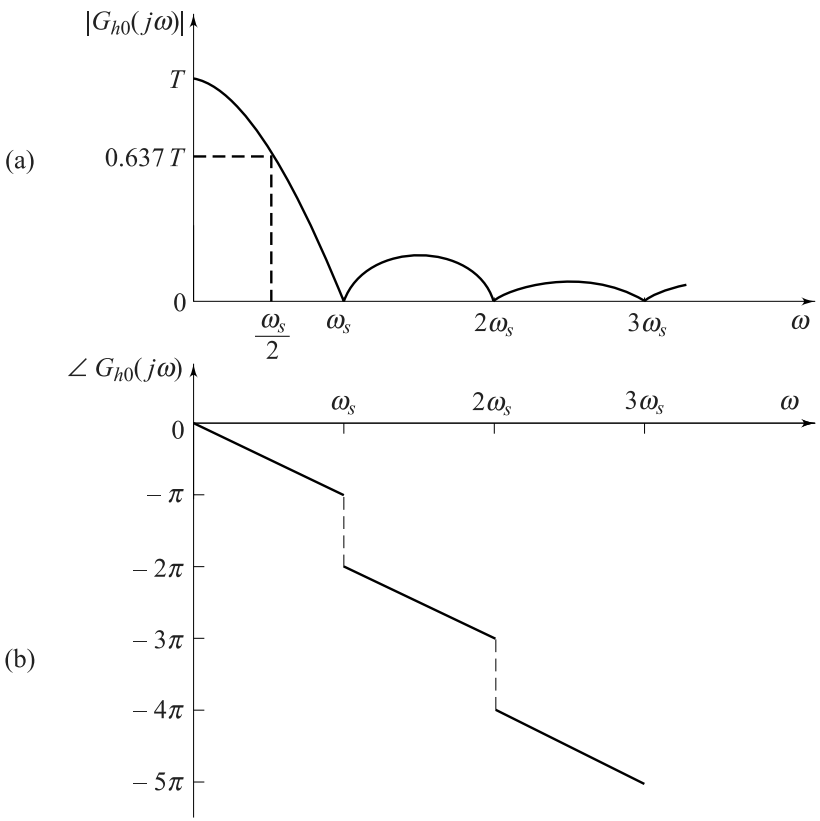


Fig. 2.36 ZOH device: (a) Gain and (b) Phase characteristics.

(iii) The factor $e^{-j\omega T/2}$ in Eqn. (2.90) corresponds to a delay of $T/2$ in the time domain. This follows from the shifting theorem of Fourier transforms. Therefore, the linear phase characteristic introduces a time delay of $T/2$. When ZOH is used in a feedback system, the lag characteristic of the device degrades the degree of system stability.

The *higher-order holds*, which are more sophisticated and which better approximate the ideal filter, are more complex and have more time delay than the ZOH. As the additional time delay in feedback control systems decreases the stability margin or even causes instability, the higher-order holds are rarely justified in terms of improved performance, and therefore, the *zero-order hold* is widely used in practice.

Anti-Aliasing Filter

In practice, signals in control systems have frequency spectra consisting of low-frequency components as well as high-frequency *noise* components. Recall that all signals with frequency higher than $\omega_s/2$ appear as signals of frequencies between 0 and $\omega_s/2$ due to the *aliasing effect*. Therefore, high-frequency noise will be folded in and will corrupt the low-frequency signal containing the desired information.

To avoid aliasing, we must either choose the sampling frequency high enough ($\omega_s > 2\omega_m$, where ω_m is the highest-frequency component present in the signal) or use an analog filter ahead of sampler (refer to Fig. 2.2) to reshape the frequency spectrum of the signal (so that the frequency spectrum for $\omega > (1/2)\omega_s$ is negligible), before the signal is sampled. Sampling at very high frequencies introduces numerical errors. Anti-aliasing filters are, therefore, useful for digital control applications.

The synthesis of analog filters is now a very mature subject area. Extensive sets of tables exist, which give, not only the *frequency* and *phase* response of many analog prototypes, but also the element values necessary to realize those prototypes. Many of the design procedures for digital filters, have been developed in ways that allow this wide body of analog filter knowledge, to be utilized effectively.

2.13 PRACTICAL ASPECTS OF THE CHOICE OF SAMPLING RATE

Every time a digital control algorithm is designed, a suitable *sampling interval* must be chosen. Choosing a long sampling interval reduces both the computational load and the need for rapid A/D conversion, and hence the hardware cost of the project.

However, as the sampling interval is increased, a number of potentially *degrading effects* start to become significant. For a particular application, one or more of these degrading effects set the upper limit for the sampling interval. The process dynamics, the type of algorithm, the control requirement and the characteristics of input and noise signals, all interact to set the maximum usable value for T.

There is also a lower limit for the sampling interval. Digital hardware dictates the minimum usable value for T.

We will discuss some of the factors which limit the choice of sampling interval. Some empirical rules for the selection of sampling interval are also reported.

2.13.1 Limiting Factors for the Choice of Sampling Rate

Information Loss due to Sampling

The *sampling theorem* states that a continuous-time signal whose frequency spectrum is bounded by upper limit ω_m, can be completely reconstructed from its samples when the sampling frequency is $\omega_s > 2\omega_m$. There are two problems associated with the use of the sampling theorem in practical control systems.

(i) The frequency spectra of real signals do not possess strictly defined ω_m. There are almost always frequency components outside the system *bandwidth*. Therefore, the selection of the sampling frequency ω_s using the sampling theorem on the basis of system bandwidth ($\omega_b = \omega_m$) is risky, as frequency components outside ω_b will appear as low-frequency signals of frequencies between 0 and $\omega_s/2$ due to the aliasing effect, and lead to loss of information.

(ii) The ideal low-pass filter needed for perfect reconstruction of a continuous-time signal from its samples is not physically realizable. Practical filters, such as the ZOH, introduce *reconstruction errors* because of the limitations of their operation.

Figure 2.28 clearly indicates that the accuracy of the *zero-order hold* as an extrapolating device depends greatly on the sampling frequency ω_s. The accuracy improves with increase in sampling frequency.

Information Loss due to Disturbances

In practice, signals in control systems include low-frequency components carrying useful information, as well as high-frequency noise components. The high-frequency components appear as low-frequency signals (of frequencies between 0 and $\omega_s/2$) due to the aliasing effect, causing a loss of information.

To avoid aliasing, we use the *analog filter* ahead of sampler (refer to Fig. 2.2) to reshape the frequency spectrum of the signal, so that the frequency spectrum for $\omega > (1/2)\omega_s$ is negligible. The cut-off frequency $\omega_s/2$ of the anti-aliasing filter must be much higher than the system bandwidth, otherwise the anti-aliasing filter becomes as significant as the system itself, in determining the sampled response.

Destabilizing Effects

Due to the conversion times and the computation times, a digital algorithm contains a *dead-time* that is absent from its analog counterpart. Dead-time has a marked destabilizing effect on a closed-loop system due to the *phase shift* caused.

A practical approach of selecting the sampling interval is to determine the stability limit of the closed-loop control system, as sampling interval T is increased. For control system applications, this approach is more useful than the use of the sampling theorem for the selection of sampling interval. In the later chapters of this book, we will use stability tests, root-locus techniques, and frequency-response plots to study the effect of the sampling interval on closed-loop stability.

Algorithm-Accuracy Effects

A number of digital control algorithms are derived from analog algorithms by a process of discretization. As we shall see in the next section, in the transformation of an algorithm, from continuous-time to

discrete-time form, errors arise and the character of the *digital algorithm* differs from that of its analog counterpart. In general, these errors occurring during the discretization process, become larger as the sampling interval increases.

This effect should rarely be allowed to dictate a shorter sampling interval, than would otherwise have been needed. We will see in Chapter 4 that the *direct digital design approach* allows a longer sampling interval without the introduction of unacceptable errors.

Word-Length Effects

As the sampling interval T becomes very short, a digital system does not tend to the continuous-time case, because of the finite word-length. To visualize this effect, we can imagine that as a signal is sampled more frequently, adjacent samples have more similar magnitudes. In order to realize the beneficial effects of shorter sampling, longer word-lengths are needed to resolve the differences between adjacent samples.

Excessively fast sampling ($T \to 0$) may also result in numerical ill-conditioning in implementation of recursive control algorithms (such as the PID control algorithm—discussed in the next section).

2.13.2 Empirical Rules for the Selection of Sampling Rate

Practical experience and simulation results have produced a number of useful approximate rules for the specification of minimum sampling rates.

(i) The recommendations given in the adjacent table for the most common process variables follow from the experience of process industries.

Type of variable	*Sampling time (seconds)*
Flow	1–3
Level	5–10
Pressure	1–5
Temperature	10–20

(ii) Fast-acting electromechanical systems require much shorter sampling intervals, perhaps down to a few milliseconds.

(iii) A rule of thumb says that, a sampling period needs to be selected that is much shorter than any of the time constants, in the continuous-time plant, to be controlled digitally. The sampling interval, equal to one tenth of the smallest time-constant, or the inverse of the largest real pole (or real part of complex pole), has been recommended.

(iv) For complex poles with the imaginary part ω_d, the frequency of transient oscillations, corresponding to the poles, is ω_d. A convenient rule suggests sampling at the rate of 6 to 10 times per cycle. Thus, if the largest imaginary part in the poles of the continuous-time plant is 1 rad/sec, which corresponds to transient oscillations with a frequency of 1/6.28 cycles per second, $T = 1$ sec may be satisfactory.

(v) Rules of thumb based on the open-loop plant model, are risky under conditions where the high closed-loop performance is forced from a plant with a low open-loop performance. The rational choice of the sampling rate, should be based on an understanding of its influence on the closed-loop performance of the control system. It seems reasonable that the highest frequency of interest, should be closely related to the 3dB-bandwidth of the closed-loop system. The selection

of sampling rates can then be based on the bandwidth of the closed-loop system. Reasonable sampling rates are 10 to 30 times the bandwidth.

(vi) Another rule of thumb, based on the closed-loop performance, is to select sampling interval T equal to, or less than, one tenth of the desired settling time.

2.14 PRINCIPLES OF DISCRETIZATION

Most of the industrial processes that we are called upon to control are continuous-time processes. Mathematical models of continuous-time processes are usually based around differential equations or, equivalently, around transfer functions in the operator s. A very extensive range of well-tried methods for control system analysis and design are in the continuous-time form.

To move from the continuous-time form to the discrete-time form requires some mechanism for time discretization (we shall refer to this mechanism simply as *discretization*). In this section, principles and various methods of discretization will be presented. An understanding of various possible approaches helps the formation of a good theoretical foundation for the analysis and design of digital control systems.

The main point is to be aware of the significant features of discretization and to have a rough quantitative understanding of the errors that are likely to be introduced by various methods. We will shortly see that none of the discretization methods preserves the characteristics of the continuous-time system exactly.

The specific problem of this section is: given a transfer function $G(s)$, what discrete-time transfer function will have approximately the same characteristics?

We present four methods for solution of this problem.

(i) *Impulse-invariant* discretization
(ii) *Step-invariant* discretization
(iii) Discretization based on *finite-difference approximation of derivatives*
(iv) Discretization based on *bilinear transformation*

2.14.1 Impulse Invariance

If we are given a continuous-time impulse response $g_a(t)$, we can consider transforming it to a discrete-time system with impulse response $g(k)$ consisting of equally spaced samples of $g_a(t)$ so that

$$g(k) = g_a(t)|_{t=kT} = g_a(kT)$$

where T is a (positive) number to be chosen as part of the discretization procedure.

The transformation of $g_a(t)$ to $g(k)$ can be viewed as impulse modulation (refer of Fig. 2.19) giving impulse-train representation $g^*(t)$ to the samples $g(k)$:

$$g^*(t) = \sum_{k=0}^{\infty} g(k)\,\delta(t - kT) \tag{2.92}$$

From the discussion in Section 2.11, and specifically Eqns (2.85), it follows that

$$G^*(j\omega) = \frac{1}{T}\sum_{k=-\infty}^{\infty} G_a\left(j\omega - j\frac{2\pi k}{T}\right) \tag{2.93a}$$

$$G(e^{j\Omega}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} G_a\left(j\frac{\Omega}{T} - j\frac{2\pi k}{T}\right) \tag{2.93b}$$

ω, in radians/second, is the *physical frequency* of the continuous-time function and $\Omega = \omega T$, in radians, is the *observed frequency* in its samples.

Thus, for a discrete-time system obtained from a continuous-time system through *impulse invariance*, the discrete-time frequency response $G(e^{j\Omega})$ is related to the continuous-time frequency response $G_a(j\omega)$ through *replication* of the continuous-time frequency response and *linear scaling* in amplitude and frequency. If $G_a(j\omega)$ is band-limited and T is chosen so that aliasing is avoided, the discrete-time frequency response is then identical to continuous-time frequency response, except for linear scaling in amplitude and frequency.

Let us explore further the properties of impulse invariance. Applying the Laplace transform to Eqn. (2.92), we obtain (refer to Eqn. (2.32b))

$$G^*(s) = \sum_{k=0}^{\infty} g(k)\, e^{-skT} \tag{2.94a}$$

On the other hand, the z-transform of $g(k)$ is, by definition,

$$G(z) = \sum_{k=0}^{\infty} g(k)\, z^{-k} \tag{2.94b}$$

Comparing Eqns (2.94a) and (2.94b), it follows that

$$G(z)\Big|_{z=e^{sT}} = G^*(s) \tag{2.95a}$$

Rewriting Eqn. (2.93a) in terms of the general transform variable s, gives a relationship between $G^*(s)$ and $G_a(s)$:

$$G^*(s) = \frac{1}{T}\sum_{k=-\infty}^{\infty} G_a\left(s - \frac{2\pi k}{T}\right) \tag{2.95b}$$

Therefore,

$$G(z)\Big|_{z=e^{sT}} = \frac{1}{T}\sum_{k=-\infty}^{\infty} G_a\left(s - \frac{2\pi k}{T}\right) \tag{2.95c}$$

We note that impulse invariance corresponds to a transformation between $G^*(s)$ and $G(z)$ represented by the mapping

$$z = e^{sT} = e^{(\sigma \pm j\omega)T} = e^{\sigma T} \angle \pm \omega T \tag{2.96}$$

between the s-plane and the z-plane.

In the following, we investigate in more detail the mapping $z = e^{sT}$. We begin by letting $g_a(t) = e^{-at}\mu(t)$; $a > 0$. The Laplace transform of this function is

$$G_a(s) = \frac{1}{s + a} \tag{2.97a}$$

The starred transformation is (refer to Eqn. (2.32b))

$$G^*(s) = \sum_{k=0}^{\infty} e^{-akT} e^{-kTs} = \frac{e^{sT}}{e^{sT} - e^{-aT}} \tag{2.97b}$$

Under the mapping $z = e^{sT}$, $G^*(s)$ is mapped into the z-transform

$$G(z) = \frac{z}{z - e^{-aT}} \tag{2.97c}$$

The function $G_a(s)$ has a pole at $s = -a$. The poles of $G^*(s)$ are at $s = -a \pm j2m\pi/T$; $m = 0, 1, 2, \ldots$. Thus $G^*(s)$ has a countably *infinite* number of poles, one of which is the pole of $G_a(s)$ at $s = -a$, as shown in Fig. 2.37a. We see that the poles of $G^*(s)$ consist of a pole of $G_a(s)$ and copies of this pole, repeated at intervals of $2\pi/T$. The same would be true for any other function whose Laplace transform exists. The poles of the Laplace transform will lie in a strip of width $2\pi/T$ centered on the real axis of the s-plane. This strip is called the *primary strip*. These poles are then repeated in *complementary strips*, above and below the primary strip (refer to Fig. 2.37a).

Suppose we map the primary strip of the s-plane into the z-plane. We begin by mapping the points of a vertical line $s = \sigma + j\omega$, where $\sigma < 0$ is fixed. Under the mapping $z = e^{sT}$, a point on this line maps to

$$z = e^{(\sigma + j\omega)T} = e^{\sigma T} e^{j\omega T}$$

The term $e^{\sigma T}$ is a *real* number that can be thought of as a scaling factor for the *unit phasor* $e^{j\omega T}$. If $\frac{-\pi}{T} \le \omega \le \frac{\pi}{T}$, and σ is fixed with $\sigma < 0$, then the mapping is a circle with radius less than one. If $\sigma = 0$, the line segment, maps onto the unit circle. For clarity, the circles in Fig. 2.37b have been divided

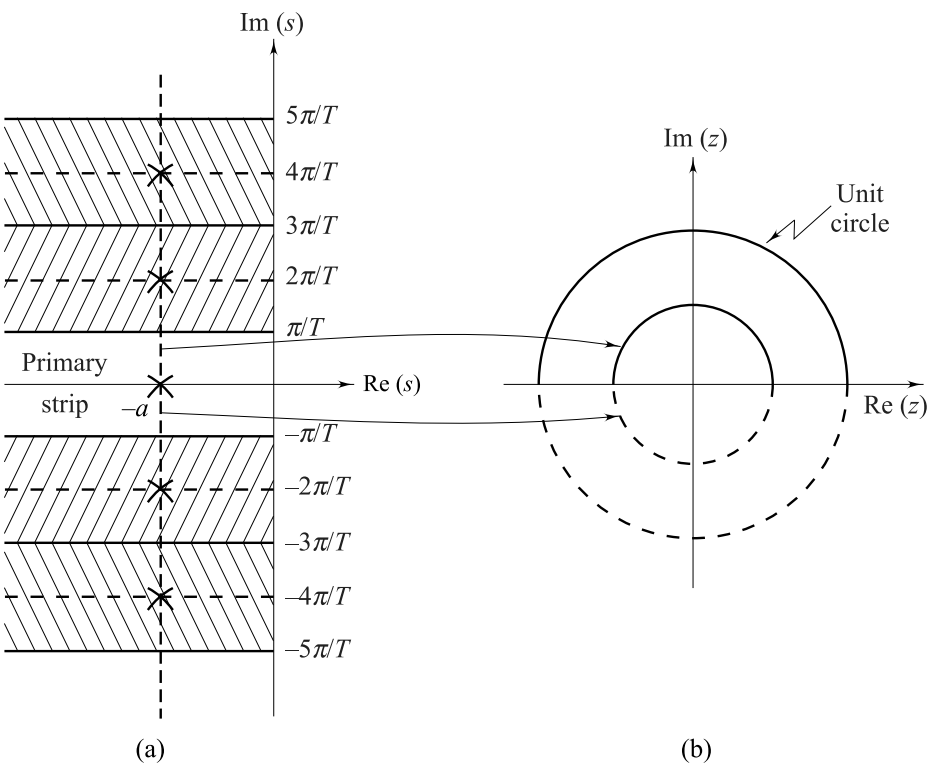


Fig. 2.37 Mapping primary strip to the z-plane

into *dashed* and *solid* portions; the solid portions correspond to mapping for $0 \le \omega \le \pi/T$ and the dashed portions correspond to mapping for $-\frac{\pi}{T} \le \omega \le 0$.

The following points are worth noting at this juncture.

(i) The left half of the primary strip in the *s*-plane maps onto the interior of the unit circle, in the *z*-plane.

(ii) The imaginary axis between $-j\pi/T$ and $j\pi/T$ associated with primary strip in the *s*-plane, maps onto the unit circle in the *z*-plane.

(iii) The right half of the primary strip in the *s*-plane, maps onto the region, exterior to the unit circle, in the *z*-plane.

(iv) The same pattern holds for each of the complementary strips.

The fourth point needs further discussion. We consider

$$g_a(t) = \cos(\omega t)$$

The corresponding Laplace transform is

$$G_a(s) = \frac{s}{s^2 + \omega^2} = \frac{s}{(s - j\omega)(s + j\omega)}$$

and the *z*-transform is

$$G(z) = \frac{z(z - \cos \omega T)}{(z - e^{j\omega T})(z - e^{-j\omega T})}$$

The *s*-plane poles: $s = j\omega$ and $s = -j\omega$, in the primary strip are mapped to the *z*-plane poles: $z = e^{j\omega T}$ and $z = e^{-j\omega T}$, respectively. However, these *z*-plane poles are also maps of *s*-plane poles: $s = j\omega + 2\pi/T$ and $s = -j\omega + 2\pi/T$; $s = j\omega - 2\pi/T$ and $s = -j\omega - 2\pi/T$; $s = j\omega + 4\pi/T$ and $s = -j\omega + 4\pi/T$; $s = j\omega - 4\pi/T$ and $s = -j\omega - 4\pi/T$;..., in the complementary strips. The *z*-plane poles: $z = e^{j\omega T}$ and $z = e^{-j\omega T}$, cannot distinguish the poles in the *primary strip* from the poles in the *complementary strips*. Thus the largest frequency we can distinguish is $\omega = \pi/T$, which is half of the sampling frequency $2\pi/T$.

While sampling a continuous-time signal to produce the discrete-time sequence, we want to ensure that all the information in the original signal is retained in the samples. There will be no information loss if we can exactly recover the continuous-time signal from the samples. To determine the condition under which there is no information loss, let us consider the continuous-time signal to be band-limited signal with maximum frequency ω_m. From Fig. 2.37, we observe that there is no information loss if

$$\frac{\pi}{T} < \omega_m$$

This, in fact, is the *sampling theorem*.

In summary, the use of impulse invariance corresponds to converting the continuous-time impulse response to a discrete-time impulse response through sampling. To avoid aliasing, the procedure is restricted to transforming band-limited frequency responses. Except for aliasing, the discrete-time frequency response is a replication of the continuous-time frequency response, linearly scaled in amplitude and frequency.

Although useful for discretizing band-limited analog systems, the impulse-invariance method is unsuccessful for discretizing transfer functions $G_a(s)$ for which $|G_a(j\omega)|$ does not approach zero for large ω. In these cases, an appropriate sampling rate cannot be found to prevent aliasing.

To overcome the problem of aliasing, we need a method in which the entire $j\omega$-axis in the s-plane, maps uniquely onto the unit circle in the z-plane. This is accomplished by the *bilinear transformation* method, described later in this section.

For a given analog system $G_a(s)$, the impulse-invariant discrete-time system is obtained following the procedure given below:

(i) Obtain the impulse response,

$$g_a(t) = \mathscr{L}^{-1}[G_a(s)]$$

(ii) Select a suitable sampling interval and derive samples $g(k)$ from $g_a(t)$,

$$g(k) = g_a(t)|_{t=kT}$$

(iii) Obtain z-transform of the sequence $g(k)$,

$$G(z) = \mathscr{Z}[g(k)]$$

The three steps given above can be represented by the following relationship:

$$G(z) = \mathscr{Z}[\mathscr{L}^{-1}[G_a(s)]|_{t=kT}] \tag{2.98a}$$

This z-transform operation is commonly indicated as

$$G(z) = \mathscr{Z}[G_a(s)] \tag{2.98b}$$

Single factor building blocks of the Laplace and z-transform pairs are given in Table 2.1. Expanding any $G_a(s)$ into partial fractions, $G(z)$ can be found by use of this table.

Example 2.14

With the background on analog design methods, the reader will appreciate the value of being able to correlate particular patterns in the s-plane with particular features of system behavior. Some of the useful s-plane patterns, which have been used in analog design, are the loci of points in the s-plane with (i) constant damping ratio ζ, and (ii) constant undamped natural frequency ω_n. In this example, we translate these patterns in the primary strip of the s-plane onto the z-plane using the basic relation $z = e^{sT}$, where T is some chosen sampling period.

Consider a second-order system with transfer function

$$G_a(s) = \frac{K}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where ζ = damping ratio, and ω_n = undamped natural frequency.

The characteristic root locations in the s-plane are

$$s_1, s_2 = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -\zeta\omega_n \pm j\omega_d$$

Figure 2.38a shows a locus of the characteristic roots, with ζ held constant and ω_n varying. Figure 2.38b shows a locus with ω_n held constant and ζ varying. The loci in Figs 2.38a and 2.38b, correspond to an underdamped second-order system.

Define the sampling frequency by

$$\omega_s = 2\pi/T$$

Corresponding to each point $s = \sigma + j\omega$ in the primary strip of the s-plane, there is a point

$$z = \exp[(\sigma + j\omega)2\pi/\omega_s]$$

in the z-plane.

(i) *Mapping of constant damping ratio loci*

A point on the constant-ζ line in the second quadrant (Fig. 2.38a), can be expressed as

$$s = \sigma + j\omega_d = -\zeta\omega_n + j\omega_n\sqrt{1-\zeta^2}$$

Since

$$\cot\theta = \frac{\zeta\omega_n}{\omega_d} = \frac{\zeta\omega_n}{\omega_n\sqrt{1-\zeta^2}} = \frac{\zeta}{\sqrt{1-\zeta^2}},$$

the s-plane point may be described by the relation

$$s = -\omega_d \cot\theta + j\omega_d$$

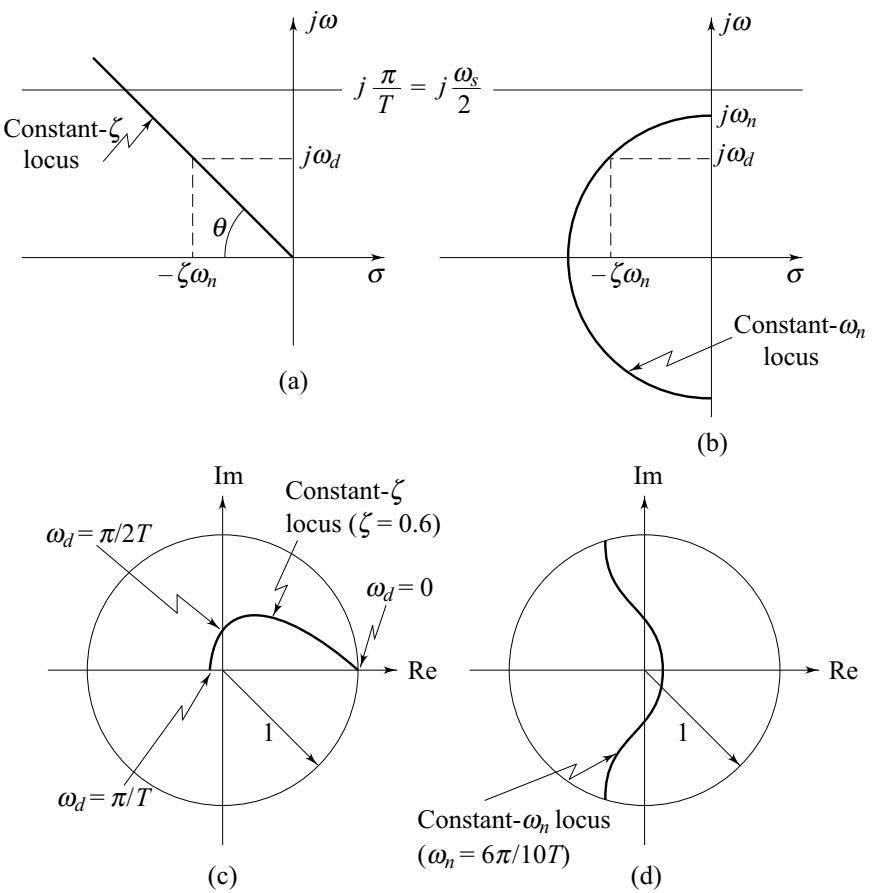


Fig. 2.38 Mapping of constant-ζ, and constant-ω_n loci from the s-plane to the z-plane

The z-plane relation becomes

$$z = \exp[(-\omega_d \cot\theta + j\omega_d)\, 2\pi/\omega_s] = \exp[((-2\pi\cot\theta)/\omega_s)\ \omega_d] \exp[j(2\pi/\omega_s)\ \omega_d]$$

where ω_d varies from 0 to $\omega_s/2$.

As per this equation, for a constant value of ζ (and hence $\cot\theta$), z is a function of ω_d only. The constant damping ratio line in the s-plane maps into the z-plane as a *logarithmic spiral* (except for $\theta = 0°$ and $\theta = 90°$). The portion of ζ-line between $\omega_d = 0$ and $\omega_d = \omega_s/2$ corresponds to one half revolution of the logarithmic spiral in the z-plane. Mapping of one representative constant-ζ line is shown in Fig. 2.38c.

(ii) *Mapping of constant undamped natural frequency loci*

A point on the constant-ω_n locus in the second quadrant (Fig. 2.38b), can be expressed as

$$s = \sigma + j\omega_d = -\zeta\omega_n + j\omega_d$$

It lies on the circle given by

$$\sigma^2 + \omega_d^2 = \omega_n^2$$

For points in the second quadrant,

$$\sigma = -\sqrt{\omega_n^2 - \omega_d^2}$$

The locus of constant-ω_n in the z-plane is given by the relation

$$z = e^{\left(-\sqrt{\omega_n^2 - \omega_d^2} + j\omega_d\right)T}$$

where ω_d varies from 0 to $\omega_s/2$.

Mapping of one representative constant-ω_n locus is shown in Fig. 2.38d.

2.14.2 Step Invariance

The basis for impulse invariance is to choose an impulse response for the discrete-time system that is similar to the impulse response of the analog system. The use of this procedure is often motivated not so much by a desire to preserve the impulse-response shape, as by the knowledge that if the analog system is band-limited, then the discrete-time frequency response will closely approximate the continuous-time frequency response.

In some design problems, a primary objective may be to control some aspect of the time response, such as the step response. In such cases, a natural approach might be to discretize the continuous-time system by *waveform-invariance criteria*. In this subsection, we consider the *step-invariant discretization*.

The step-invariant discrete-time system is obtained by placing a unit step on the input to the analog system $G_a(s)$, and a sampled unit step on the input to the discrete-time system. The transfer function $G(z)$ of the discrete-time system is adjusted, until the output of the discrete-time system represents samples of the output of the analog system. The input to the analog system $G_a(s)$ is $\mu(t)$—a unit-step function. Since $\mathscr{L}[\mu(t)] = 1/s$, the output $y(t)$ of the analog system is given by

$$y(t) = \mathscr{L}^{-1}\left\{\frac{G_a(s)}{s}\right\}$$

Output samples of the discrete-time system are defined to be

$$y(kT) = \mathscr{L}^{-1}\left\{\frac{G_a(s)}{s}\right\}\Bigg|_{t=kT}$$

The z-transform of this quantity yields the z-domain output of the discrete-time system. This gives

$$Y(z) = \mathscr{Z}\left[\mathscr{L}^{-1}\left\{\frac{G_a(s)}{s}\right\}\Bigg|_{t=kT}\right] \tag{2.99a}$$

Since $\mathscr{Z}[\mu(k)] = \dfrac{z}{z-1}$, where $\mu(k)$ is the unit-step sequence, the output $y(k)$ of the discrete-time system $G(z)$ is given by

$$Y(z) = G(z)\left\{\frac{z}{z-1}\right\} \tag{2.99b}$$

Comparing Eqn. (2.99b) with Eqn. (2.99a), we obtain

$$G(z) = (1 - z^{-1})\left[\mathscr{Z}\left\{\mathscr{L}^{-1}\left(\frac{G_a(s)}{s}\right)\Bigg|_{t=kT}\right\}\right] \tag{2.100a}$$

or

$$G(z) = (1 - z^{-1})\left[\mathscr{Z}\left(\frac{G_a(s)}{s}\right)\right] \tag{2.100b}$$

Notice that Eqn. (2.100b) can be rewritten as follows:

$$G(z) = \mathscr{Z}\left[\frac{1-e^{-sT}}{s}G_a(s)\right] \tag{2.100c}$$

This can easily be established.

Let $$\mathscr{L}^{-1}\left[\frac{G_a(s)}{s}\right] = g_1(t), \text{ and } \mathscr{Z}[g_1(kT)] = G_1(z)$$

Then $$\mathscr{L}^{-1}\left[e^{-sT}\frac{G_a(s)}{s}\right] = g_1(t - T), \text{ and } \mathscr{Z}[g_1(kT - T)] = z^{-1}G_1(z)$$

Therefore,

$$\mathscr{Z}\left[\frac{G_a(s)}{s} - e^{-sT}\frac{G_a(s)}{s}\right] = (1 - z^{-1})\,\mathscr{Z}\left[\frac{G_a(s)}{s}\right]$$

This establishes the equivalence of Eqns (2.100b) and (2.100c).

The right-hand side of Eqn (2.100c) can be viewed as the z-transform of the analog system $G_a(s)$, preceded by zero-order hold (ZOH). Introducing a *fictitious* sampler and ZOH for analytical purposes, we can use the model of Fig. 2.39 to derive a step-invariant equivalent of analog systems. For obvious reasons, step-invariant equivalence is also referred to as ZOH *equivalence*. In the next chapter, we will use the ZOH equivalence to obtain discrete-time equivalents of the plants of feedback control systems.

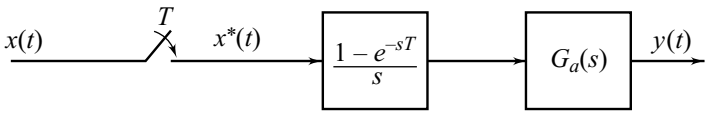


Fig. 2.39 $G_a(s)$ preceded by a fictitious sample-and-hold device

Equivalent discrete-time systems obtained by the step-invariance method, may exhibit the *frequency folding* phenomena and may, therefore, present the same kind of aliasing errors as found in impulse-invariance method. Notice, however, that the presence of $1/s$ term in $G_a(s)/s$ causes high-frequency attenuation. Consequently, the equivalent discrete-time system obtained by the step-invariance method, will exhibit smaller aliasing errors than that obtained by the impulse-invariance method.

As for stability, the equivalent discrete-time system obtained by the step-invariance method is stable if the original continuous-time system is a stable one (refer to Review Example 6.2).

Example 2.15

Figure 2.40 shows the model of a plant driven by a D/A converter. In the following, we derive the transfer function model relating $y(kT)$ to $r(kT)$.

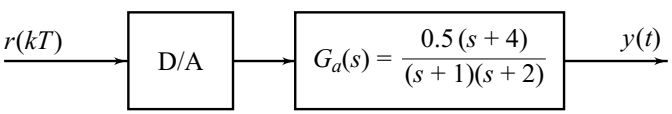


Fig. 2.40 A plant driven by a D/A converter

The standard D/A converters are designed in such a way, that the old value of the input sample is held constant until a new sample arrives. The system of Fig. 2.40 can, therefore, be viewed as an analog system $G_a(s)$, preceded by zero-order hold, and we can use ZOH equivalence to obtain the transfer function model relating $y(kT)$ to $r(kT)$.

Zero-order hold equivalent (step-invariant equivalent) of $G_a(s)$ can be determined as follows:

Since
$$\frac{1}{s}G_a(s) = \frac{0.5\,(s+4)}{s(s+1)\,(s+2)} = \frac{1}{s} - \frac{1.5}{s+1} + \frac{0.5}{s+2}$$

we have (refer to Table 2.1)

$$\mathcal{Z}\left[\frac{1}{s}G_a(s)\right] = \frac{z}{z-1} - \frac{1.5z}{z-e^{-T}} + \frac{0.5z}{z-e^{-2T}}$$

From Eqn. (2.100b),

$$G(z) = \frac{z-1}{z}\left[\frac{z}{z-1} - \frac{1.5z}{z-e^{-T}} + \frac{0.5z}{z-e^{-2T}}\right] = 1 - \frac{1.5(z-1)}{z-e^{-T}} + \frac{0.5(z-1)}{z-e^{-2T}}$$

Let the sampling frequency be 20 rad/sec, so that

$$T = \frac{2\pi}{20} = 0.31416 \text{ sec};\ e^{-T} = 0.7304;\ e^{-2T} = 0.5335$$

With these values, we get the following step-invariant equivalent of the given analog system:

$$G(z) = \frac{0.17115z - 0.04535}{z^2 - 1.2639z + 0.3897}$$

2.14.3 Finite-Difference Approximation of Derivatives

Another approach to transforming a continuous-time system into a discrete-time one is to approximate derivatives in a differential equation representation of the continuous-time system by *finite differences*. This is a common procedure in digital simulations of analog systems, and is motivated by the intuitive notion that the derivative of a continuous-time function, can be approximated by the difference between consecutive samples of the signal to be differentiated. To illustrate the procedure, consider the first-order differential equation

$$\frac{dy(t)}{dt} + ay(t) = r(t) \tag{2.101}$$

The *backward-difference method* consists of replacing $r(t)$ by $r(k)$, $y(t)$ by $y(k)$; and the first derivative $dy(t)/dt$ by the first backward-difference

$$\left.\frac{dy(t)}{dt}\right|_{t=kT} = \frac{y(k) - y(k-1)}{T} \tag{2.102}$$

This yields the difference equation

$$\frac{y(k) - y(k-1)}{T} + ay(k) = r(k) \tag{2.103}$$

If T is sufficiently small, we would expect the solution $y(k)$ to yield a good approximation to the samples of $y(t)$.

To interpret the procedure in terms of a mapping of continuous-time function $G_a(s)$ to a discrete-time function $G(z)$, we apply the Laplace transform to Eqn. (2.101) and z-transform to Eqn. (2.103), to obtain

$$sY(s) + aY(s) = R(s)\ ;\ \text{so that } G_a(s) = \frac{Y(s)}{R(s)} = \frac{1}{s+a}$$

$$\left(\frac{1-z^{-1}}{T}\right) Y(z) + a\,Y(z) = R(z)\ ;\ \text{so that } G(z) = \frac{Y(z)}{R(z)} = \frac{1}{\left(\frac{1-z^{-1}}{T}\right) + a}$$

Comparing $G_a(s)$ with $G(z)$, we see that

$$G(z) = G_a(s)\Big|_{s=(1-z^{-1})/T}$$

Therefore,

$$s = \frac{1-z^{-1}}{T}\ ;\ z = \frac{1}{1-sT} \tag{2.104}$$

is a mapping from the s-plane to the z-plane when the backward-difference method is used to discretize Eqn. (2.101).

The stability region in the s-plane can be mapped by Eqn. (2.104) into the z-plane as follows. Noting that the stable region in the s-plane is given by $\text{Re}(s) < 0$, the stability region in the z-plane under the mapping (2.104), becomes

$$\text{Re}\left(\frac{1-z^{-1}}{T}\right) = \text{Re}\left(\frac{z-1}{Tz}\right) < 0$$

Writing the complex variable z as $\alpha + j\beta$, we may write the last inequality as

$$\text{Re}\left(\frac{\alpha + j\beta - 1}{\alpha + j\beta}\right) < 0$$

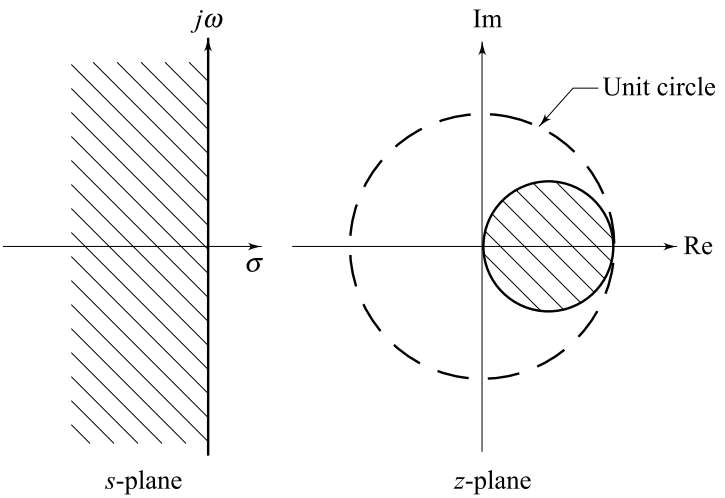
or

$$\text{Re}\left[\frac{(\alpha + j\beta - 1)(\alpha - j\beta)}{\alpha^2 + \beta^2}\right] = \text{Re}\left[\frac{\alpha^2 - \alpha + \beta^2 + j\beta}{\alpha^2 + \beta^2}\right] = \frac{\alpha^2 - \alpha + \beta^2}{\alpha^2 + \beta^2} < 0$$

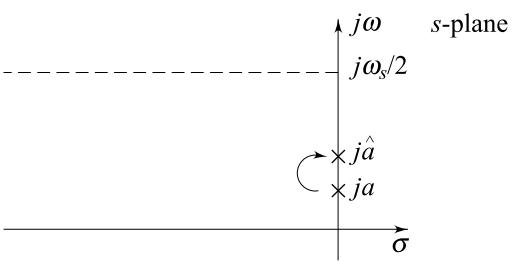
which can be written as

$$(\alpha - 1/2)^2 + \beta^2 < (1/2)^2$$

The stable region in the s-plane can thus be mapped into a circle with center at $\alpha = 1/2$, $\beta = 0$ and radius equal to 1/2, as shown in Fig. 2.41a.



(a) Mapping of the s-plane to the z-plane using backward-difference approximation of the derivative



(b) Warping effect

Fig. 2.41

The *backward-difference* method is simple and will produce a stable discrete-time system for a stable continuous-time system. Also, the entire *s*-plane imaginary axis is mapped once and only once onto the small *z*-plane circle; the folding or aliasing problems do not occur. The penalty is a 'warping' of the equivalent *s*-plane poles, as shown in Fig. 2.41b. This situation is reflected in the relationship between the exact *z-transformation*, and the *backward-difference approximation*.

Consider a pole in *z*-plane at $z = e^{jaT}$. Inverse mapping of this pole to the *s*-plane, using the transformation $s = \ln z/T$, gives $s = ja$ (shown in Fig. 2.41b). Inverse mapping of the pole in the *z*-plane at $z = e^{jaT}$ to the *s*-plane, using the backward-difference approximation $s = (1 - z^{-1})/T$, gives $s = j\hat{a} = (1 - e^{-jaT})/T$ (also shown in Fig. 2.41b).

Thus a nonlinear relationship or 'warping':

$$j\hat{a} = (1 - e^{-jaT})/T \tag{2.105}$$

exists between the two poles ja and $j\hat{a}$ in the *s*-plane. Note that for small aT, using the first two terms of the expansion of the exponential in Eqn. (2.105), yields

$$\begin{aligned} j\hat{a} &\cong \frac{1}{T}\left[1-(1-jaT)\right] \\ &\cong ja \end{aligned}$$

The 'warping' effect is thus negligible for relatively small aT (about 17° or less).

Let us now investigate the behavior of the equivalent discrete-time system when the derivative $dy(t)/dt$ in Eqn. (2.101), is replaced by *forward difference*:

$$\left.\frac{dy(t)}{dt}\right|_{t=kT} = \frac{y(k+1)-y(k)}{T}$$

This yields the following difference equation approximation for Eqn. (2.101):

$$\frac{y(k+1)-y(k)}{T} + ay(k) = r(k) \tag{2.106}$$

Applying Laplace transform to Eqn. (2.101) and *z*-transform to Eqn. (2.106), we obtain

$$\frac{Y(s)}{R(s)} = G_a(s) = \frac{1}{s+a} \tag{2.107a}$$

and

$$\frac{Y(z)}{R(z)} = G(z) = \frac{1}{\dfrac{z-1}{T}+a} \tag{2.107b}$$

The right-hand sides of Eqns (2.107a) and (2.107b) become identical if we let

$$s = \frac{z-1}{T} \tag{2.108}$$

We may consider Eqn. (2.108) to be the mapping from the *s*-plane to the *z*-plane, when the forward-difference method is used to discretize Eqn. (2.101).

One serious problem with the forward-difference approximation method is regarding stability. The left-hand side of the *s*-plane is mapped into the region $\text{Re}\left(\dfrac{z-1}{T}\right) < 0$ or $\text{Re}\,(z) < 1$. This mapping shows

that the poles of the left half of the *s*-plane, may be mapped outside the unit circle in *z*-plane. Hence the discrete-time system obtained by this method may become unstable.

Rectangular Rules for Integration

Consider the continuous-time system (2.101):

$$\dot{y}(t) = -ay(t) + r(t) \tag{2.109a}$$

or

$$y(t) = y(0) - a\int_0^t y(\tau)\,d\tau + \int_0^t r(\tau)\,d\tau \tag{2.109b}$$

In numerical analysis, the procedure known as the *rectangular rule for integration* proceeds by approximating the continuous-time function by continuous rectangles, as illustrated in Fig. 2.42, and then adding their areas to compute the total integral. We thus approximate the area as given below.

(i) *Forward rectangular rule for integration*

$$\int_{(k-1)T}^{kT} y(t)\,dt \cong [y(k-1)]T \tag{2.110a}$$

(ii) *Backward rectangular rule for integration*

$$\int_{(k-1)T}^{kT} y(t)\,dt \cong [y(k)]T \tag{2.110b}$$

With the forward rule for integration, the continuous-time system (2.109) is converted to the following recursive algorithm:

$$y(k) = y(k-1) - aTy(k-1) + Tr(k-1)$$

The *z*-transformation of this equation gives

$$Y(z) = z^{-1}\,Y(z) - aT\,z^{-1}\,Y(z) + T\,z^{-1}\,R(z)$$

or

$$\frac{Y(z)}{R(z)} = \frac{1}{\dfrac{z-1}{T} + a}$$

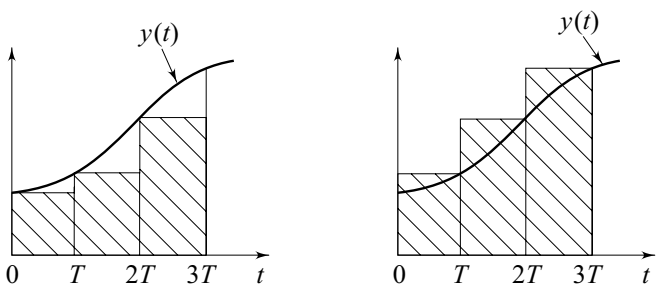


Fig. 2.42 (a) Forward rectangular rule, and (b) backward rectangular rule for integral approximation

Laplace transformation of Eqn. (2.109a) gives the transfer function of the continuous-time system:

$$\frac{Y(s)}{R(s)} = \frac{1}{s+a}$$

The forward rectangular rule for integration thus results in the s-plane to z-plane mapping:

$$s = \frac{z-1}{T}$$

which is same as the one obtained by forward-difference approximation of derivatives (Eqn. (2.108)).

Similarly, it can easily be established that the backward rectangular rule for integration results in s-plane to z-plane mapping, which is same as the one obtained by backward-difference approximation of derivatives (Eqn. (2.104)).

Example 2.16

The simplest formula for the PID or three-mode controller is the addition of the proportional, integral, and derivative modes:

$$u(t) = K_c \left[e(t) + \frac{1}{T_I} \int_0^t e(t)dt + T_D \frac{de(t)}{dt} \right] \tag{2.111}$$

where

u = controller output signal;
T_I = integral or reset time;
e = error (controller input) signal;
T_D = derivative or rate time; and
K = controller gain.

For the digital realization of the PID controller, it is necessary to approximate each mode in Eqn. (2.111) using the sampled values of $e(t)$.

The *proportional mode* requires no approximation since it is a purely static part:

$$u_P(k) = K_c\, e(k)$$

The *integral mode* may be approximated by the backward rectangular rule for integration. If $S(k-1)$ approximates the area under the $e(t)$ curve up to $t = (k-1)T$, then the approximation to the area under the $e(t)$ curve up to $t = kT$ is given by (refer to Eqn. (2.110b))

$$S(k) = S(k-1) + Te(k)$$

A digital realization of the integral mode of control is as follows:

$$u_I(k) = \frac{K_c}{T_I}\, S(k)$$

where $S(k)$ = sum of the areas under the error curve = $S(k-1) + Te(k)$

The *derivative mode* may be approximated by the backward-difference approximation:

$$\left.\frac{de(t)}{dt}\right|_{t=kT} = \frac{e(k)-e(k-1)}{T}$$

Therefore,
$$u_D(k) = \frac{K_c T_D}{T}\,[e(k) - e(k-1)]$$

Bringing all the three modes together, results in the following PID algorithm:

$$\begin{aligned} u(k) &= u_P(k) + u_I(k) + u_D(k) \\ &= K_c\left[e(k) + \frac{1}{T_I}S(k) + \frac{T_D}{T}[e(k)-e(k-1)]\right] \end{aligned} \tag{2.112a}$$

where
$$S(k) = S(k-1) + Te(k) \tag{2.112b}$$

We can directly use the s-plane to z-plane mapping given by Eqn. (2.104) to obtain the discrete equivalent (2.112) of the PID controller (2.111).

The PID controller (2.111), expressed in terms of operator s, is given by the input-output relation

$$U(s) = K_c\left[1 + \frac{1}{T_I s} + T_D s\right] E(s) \tag{2.113a}$$

The mapping (2.104):

$$s = \frac{1-z^{-1}}{T}$$

corresponds to backward-difference approximation of derivatives. This mapping transforms Eqn. (2.113a) to the following system:

$$U(z) = K_c\left[1 + \frac{T}{T_I}\left(\frac{1}{1-z^{-1}}\right) + \frac{T_D}{T}(1-z^{-1})\right] E(z) \tag{2.113b}$$

This is the input-output relation of the PID controller in terms of operator z. By the inverse transform operation, we can express individual control modes by difference equations:

(i) $$u_P(k) = K_c\, e(k)$$

(ii) $$u_I(k) - u_I(k-1) = \frac{K_c T}{T_I}\, e(k)$$

or
$$u_I(k) = u_I(k-1) + \frac{K_c T}{T_I}\, e(k) = \frac{K_c}{T_I}\, S(k)$$

where
$$S(k) = S(k-1) + T\, e(k)$$

(iii) $$u_D(k) = \frac{K_c T_D}{T}\,[e(k) - e(k-1)]$$

Bringing all the three modes together results in the PID algorithm given by Eqns (2.112).

2.14.4 Bilinear Transformation

The technique, based on finite-difference approximation to differential equations, for deriving a discrete-time system from an analog system, has the advantage that z-transform of the discrete-time system, is trivially derived from the Laplace transform of the analog system by an algebraic substitution. The disadvantages of these mappings are that $j\omega$-axis in the s-plane, generally does not map into the unit circle in the z-plane, and (for the case of forward-difference method) stable analog systems may not always map into stable discrete-time systems.

A nonlinear one-to-one mapping from the s-plane to the z-plane which eliminates the disadvantages mentioned above and which preserves the desired algebraic form is the *bilinear*[8] *transformation* defined by

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \tag{2.114}$$

This transformation is invertible with the inverse mapping given by

$$z = \frac{1+(T/2)s}{1-(T/2)s} \tag{2.115}$$

The bilinear transformation also arises from a particular approximation method—the *trapezoidal rule* for numerically integrating differential equations.

Let us consider a continuous-time system for which the describing equation is (Eqn. (2.101))

$$\dot{y}(t) = -ay(t) + r(t) \tag{2.116a}$$

or

$$y(t) = y(0) - a\int_0^t y(\tau)\,d\tau + \int_0^t r(\tau)\,d\tau \tag{2.116b}$$

Laplace transformation of Eqn. (2.116a) gives the transfer function of the continuous-time system.

$$\frac{Y(s)}{R(s)} = G_a(s) = \frac{1}{s+a}$$

Applying bilinear transformation (Eqn. (2.114)) to this transfer function, we obtain

$$G(z) = \frac{1}{\frac{2}{T}\left(\frac{z-1}{z+1}\right)+a}$$

In numerical analysis, the procedure known as the *trapezoidal rule for integration* proceeds by approximating the continuous-time function by continuous trapezoids, as illustrated in Fig. 2.43, and then adding their areas to compute the total integral. We thus approximate the area

$$\int_{(k-1)T}^{kT} y(t)dt \text{ by } \frac{1}{2}[y(k)+y(k-1)]T$$

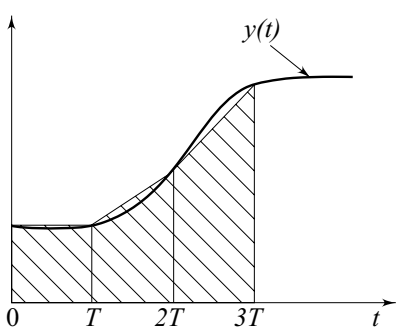


Fig. 2.43 Trapezoidal rule for integral approximation

[8] The transformation is called *bilinear* from consideration of its mathematical form.

With this approximation, Eqn. (2.116b) can be converted to the following recursive algorithm:

$$y(k) = y(k-1) - \frac{aT}{2}\,[y(k) + y(k-1)] + \frac{T}{2}\,[r(k) + r(k-1)]$$

The z-transformation of this equation gives

$$Y(z) = z^{-1}\,Y(z) - \frac{aT}{2}\,[Y(z) + z^{-1}\,Y(z)] + \frac{T}{2}\,[R(z) + z^{-1}\,R(z)]$$

or

$$\frac{Y(z)}{R(z)} = \frac{\frac{T}{2}(1+z^{-1})}{(1-z^{-1}) + \frac{aT}{2}(1+z^{-1})} = \frac{1}{\frac{2}{T}\left(\frac{z-1}{z+1}\right) + a}$$

This result is identical to the one obtained from the transfer function of the continuous-time system by bilinear transformation.

The nature of bilinear transformation is best understood from Fig. 2.44, which shows how the s-plane is mapped onto the z-plane. As seen in the figure, the entire $j\omega$-axis in the s-plane, is mapped onto the unit circle in the z-plane. The left half of the s-plane is mapped inside the unit circle in the z-plane, and the right half of the s-plane is mapped outside the z-plane unit circle. These properties can easily be established. Consider, for example, the left half of the s-plane defined by $\mathrm{Re}(s) < 0$. By means of Eqn. (2.114), this region of the s-plane is mapped onto the z-plane region defined by

$$\mathrm{Re}\left(\frac{2}{T}\frac{z-1}{z+1}\right) < 0 \quad \text{or} \quad \mathrm{Re}\left(\frac{z-1}{z+1}\right) < 0$$

By taking the complex variable $z = \alpha + j\beta$, this inequality becomes

$$\mathrm{Re}\left(\frac{z-1}{z+1}\right) = \mathrm{Re}\left(\frac{\alpha + j\beta - 1}{\alpha + j\beta + 1}\right) = \mathrm{Re}\left[\frac{(\alpha - 1 + j\beta)\,(\alpha + 1 - j\beta)}{(\alpha + 1 + j\beta)\,(\alpha + 1 - j\beta)}\right]$$

$$= \mathrm{Re}\left[\frac{\alpha^2 - 1 + \beta^2 + j2\beta}{(\alpha+1)^2 + \beta^2}\right] < 0$$

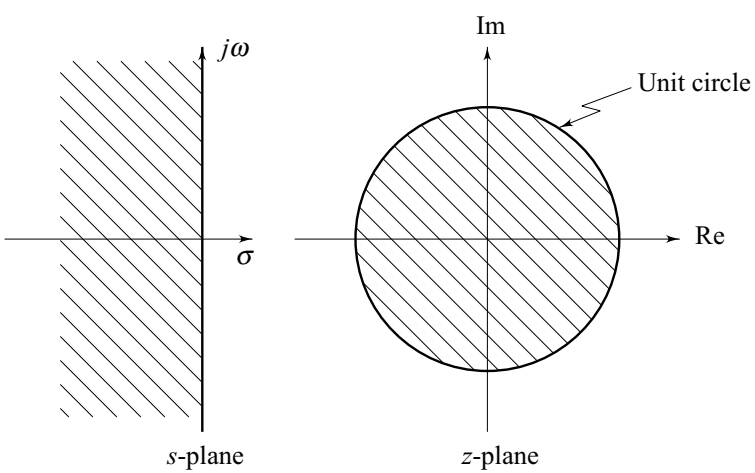


Fig. 2.44 Mapping of the s-plane to the z-plane using bilinear transformation

which is equivalent to

$$\alpha^2 - 1 + \beta^2 < 0 \quad \text{or} \quad \alpha^2 + \beta^2 < 1^2$$

which corresponds to the inside of the unit circle in z-plane. The bilinear transformation thus produces a stable discrete-time system for a stable continuous-time system.

Since the entire $j\omega$-axis of the s-plane is mapped once and only once onto the unit circle in the z-plane, the aliasing errors inherent with impulse-invariant transformations are eliminated. However, there is again a warping penalty.

Consider a pole in z-plane at $z = e^{jaT}$, shown in Fig. 2.45c. Its inverse mapping in the s-plane using the transformation $s = \ln z/T$, gives $s = ja$ (shown in Fig. 2.45a). Inverse mapping of the z-plane pole (Fig. 2.45b) at $z = e^{jaT}$, to the s-plane obtained using the bilinear transformation $s = \frac{1}{T}\frac{z-1}{z+1}$, is also shown in Fig. 2.45a.

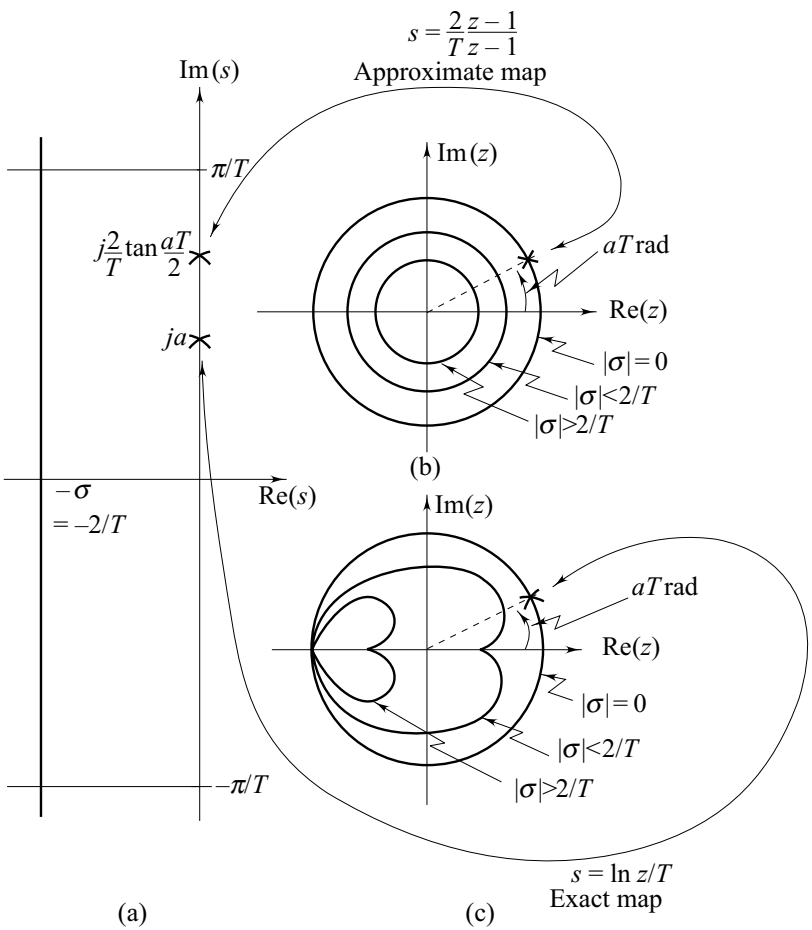


Fig. 2.45 Mapping of vertical line in the s-plane to the z-plane

$$s = j\hat{a} = \frac{2e^{jaT} - 1}{Te^{jaT} + 1} = \frac{2e^{jaT/2} - e^{-jaT/2}}{Te^{jaT/2} + e^{-jaT/2}}$$
$$= j\frac{2}{T}\tan\frac{aT}{2} \tag{2.117}$$

Equation (2.117) is a measure of the frequency distortion or warping caused by the bilinear transformation. Whereas the exact mapping gives the s-plane pole at $s = ja$, approximate mapping gives the pole at $s = j\frac{2}{T}\tan\frac{aT}{2}$. This mapping will be correct only if $\frac{aT}{2} << \pi$, i.e., if $\omega_s = \frac{2\pi}{T} >> a$. When $\frac{aT}{2} < 17°$, or about 0.3 rad, then $j\frac{2}{T}\tan\frac{aT}{2} \cong a$. This means that in frequency domain, the bilinear transformation is good only for small values $\omega T/2$.

Some additional information about the nature of the bilinear mapping can be obtained, by considering the mapping of s-plane poles, $s = -\sigma + j\omega$. Figure 2.45a shows the regions of poles with $|\sigma| > \frac{2}{T}$, and $|\sigma| < \frac{2}{T}$; the vertical line through $-\sigma = -\frac{2}{T}$ in the left half of the s-plane, maps to a closed curve inside the unit circle in z-plane (Fig. 2.45b). The approximations introduced by bilinear mapping are clearly visible when we compare this mapping with exact mapping $z = e^{sT}$ shown in Fig. 2.45c.

Example 2.17

A method that has been frequently used by practicing engineers to approximate a sampled-data system by a continuous-time system, relies on the approximation of the sample-and-hold operation by means of a pure time delay. Consider the sampled-data system of Fig. 2.46a. The sinusoidal steady-state transfer function of the zero-order hold is (refer to Eqn. (2.90))

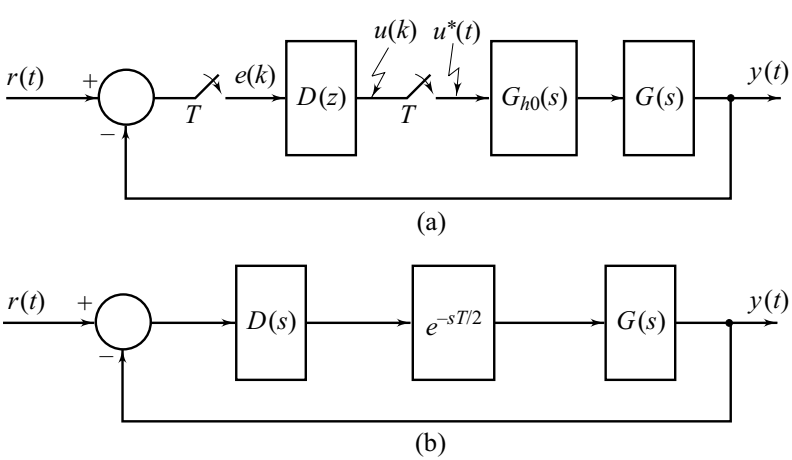


Fig. 2.46 Discretization of analog design

$$G_{h0}(j\omega) = T\,\frac{\sin(\omega T/2)}{\omega T/2}\,e^{-j\omega T/2}$$

The sinusoidal steady-state transfer function of the impulse-modulation model of the open-loop system is given by (refer to Eqn. (2.84a))

$$G_{h0}G^*(j\omega) = \frac{1}{T}\sum_{n=-\infty}^{\infty} G_{h0}\left(j\omega - j\frac{2\pi n}{T}\right)G\left(j\omega - j\frac{2\pi n}{T}\right)$$

Since in most control systems, $G(j\omega)$ has low-pass filter characteristics, we can approximate the right-hand side of the equation given above just by the $n = 0$ term. At low frequencies, the magnitude of $\sin(\omega T/2)/(\omega T/2)$ is approximately unity; therefore (refer to Eqn. (2.90))

$$G_{h0}G^*(j\omega) = G(j\omega)\,e^{-j\omega T/2}$$

This means that the sample-and-hold operation can be approximated by a pure time delay of one half the sampling period T. Figure 2.46b shows the continuous-time approximation of the sampled-data system.

The approximating continuous-time system can be used for the design of the discrete-time system as illustrated by the following example.

Consider that the transfer function of the controlled process of the system shown in Fig. 2.46a is

$$G(s) = \frac{10}{(1+0.5s)\,(1+0.1s)\,(1+0.05s)}$$

We wish to design a digital controller for the process so that the closed-loop system acquires a damping ratio of 0.4 without loss of steady-state accuracy. We select sampling time $T = 0.04$ sec for the proposed digital controller. Our approach will be to first design an analog controller $D(s)$ for the approximating continuous time system in Fig. 2.46b (with $G(s) = 10/[(1+0.5s)(1+0.1s)(1+0.05s)]$), that meets the given design specifications and then discretizing $D(s)$ to obtain the digital controller $D(z)$.

The plant model for the design of $D(s)$ becomes

$$G_P(s) = \frac{10e^{-0.02s}}{(1+0.5s)\,(1+0.1s)\,(1+0.05s)}$$

Since the design is handled more conveniently in frequency domain in the presence of time delay, we can translate damping ratio specification into equivalent phase margin specification and then proceed. Using the standard frequency-domain design procedure[9], we obtain the following analog compensator:

$$D(s) = \frac{0.67s+1}{2s+1}$$

The bandwidth of the compensated system is $\omega_b = 10$ rad/sec. Note that the sampling rate $\omega_s = 2\pi/T = 157$ rad/sec is factor of 16 faster than ω_b; therefore, our selection of $T = 0.04$ sec is quite 'safe' (refer to Section 2.13).

Using the bilinear transformation given by Eqn. (2.114), we obtain

$$D(z) = \frac{0.67\left[\dfrac{50(z-1)}{z+1}\right]+1}{2\left[\dfrac{50(z-1)}{z+1}\right]+1} = \frac{34.5z-32.5}{101z-99} = \frac{U(z)}{E(z)}$$

[9] Chapters 8–10 of reference [155].

which leads to

$$u(k) = 0.9802\, u(k-1) + 0.3416\, e(k) - 0.3218\, e(k-1), \text{ where } e(k) = r(k) - y(k).$$

This is the proposed algorithm for the digital control system of Fig. 2.46a. To make sure that with the proposed design, the system will behave as expected, we must analyze the system response. Methods for analysis of digital control systems are covered in the next chapter.

In this section, we have presented several methods for obtaining discrete-time equivalents for continuous-time systems. The response between sampling points is different for each discretization method used. Furthermore, none of the equivalent discrete-time systems can have complete fidelity. The actual (continuous-time) response between any two consecutive sampling points, is always different from the response between the same two consecutive sampling points that is taking place in each equivalent discrete-time system, no matter what method of discretization is used.

It is not possible to say which equivalent discrete-time system is best for any given analog system, since the degree of distortions in transient response and frequency response characteristics, depends on the sampling frequency, the highest frequency component involved in the system, transportation lag present in the system, etc. It may be advisable for the designer to try a few alternate forms of the equivalent discrete-time systems, for the given analog system.

REVIEW EXAMPLES

Review Example 2.1

Consider a first-order discrete-time system described by the difference equation

$$y(k+1) + a_1\, y(k) = b_0\, r(k+1) + b_1\, r(k) \tag{2.118}$$

The input is switched to the system at $k = 0$ ($r(k) = 0$ for $k < 0$); the initial state $y(-1)$ of the system is specified. Obtain a simulation diagram for the system.

Solution State variable models of discrete-time systems can easily be translated into digital computer simulation diagrams. Methods of conversion of difference equation models to state variable models, is presented in Chapter 6.

It can easily be verified that the following state variable model represents the given difference equation (2.118):

$$\begin{aligned} x(k+1) &= -a_1 x(k) + r(k) \\ y(k) &= (b_1 - a_1 b_0)\, x(k) + b_0 r(k) \end{aligned} \tag{2.119}$$

In terms of the specified initial condition $y(-1)$ of the difference equation model (2.118), the initial state $x(0)$ of the state variable model (2.119) is given by

$$x(0) = \frac{-a_1}{b_1 - a_1 b_0}\, y(-1)$$

Note that if the first-order discrete-time system is relaxed before switching on the input $r(k)$ at $k = 0$, the initial condition $y(-1) = 0$ for the model (2.118), and equivalently the initial condition $x(0) = 0$ for the model (2.119).

Figure 2.47 shows a simulation diagram for the given discrete-time system.

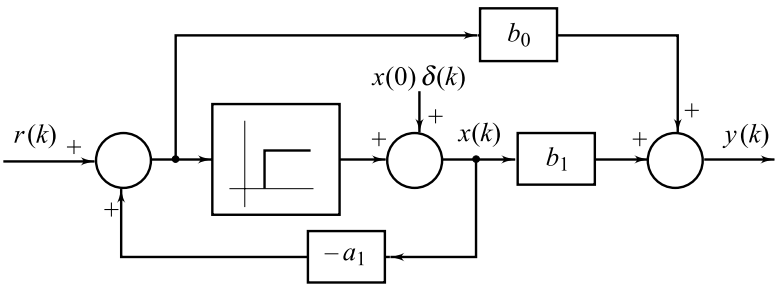


Fig. 2.47 Simulation diagram for the state model

Review Example 2.2

Consider a discrete-time system

$$y(k+2) + \frac{1}{4}\,y(k+1) - \frac{1}{8}\,y(k) = 3r(k+1) - r(k) \tag{2.120}$$

with input

$$r(k) = (-1)^k\,\mu(k)$$

and initial conditions

$$y(-1) = 5,\ y(-2) = -6$$

Find the output $y(k)$; $k \geq 0$.

Solution The difference equation (2.120) is first converted to the equivalent form

$$y(k) + \tfrac{1}{4}\,y(k-1) - \tfrac{1}{8}\,y(k-2) = 3r(k-1) - r(k-2) \tag{2.121}$$

z-transformation of the linear difference equation (2.121) requires the following results:

$$\mathcal{Z}[y(k-1)] = \sum_{k=0}^{\infty} y(k-1)z^{-k}$$

$$= y(-1) + z^{-1}\sum_{k=0}^{\infty} y(k)z^{-k} = y(-1) + z^{-1}\,Y(z) \tag{2.122a}$$

$$\mathcal{Z}[y(k-2)] = y(-2) + z^{-1}[y(-1) + z^{-1}\,Y(z)]$$
$$= y(-2) + z^{-1}\,y(-1) + z^{-2}\,Y(z) \tag{2.122b}$$

z-transformation of each term in Eqn. (2.121) yields

$$Y(z) + \tfrac{1}{4}\,[z^{-1}\,Y(z) + y(-1)] - \tfrac{1}{8}\,[z^{-2}\,Y(z) + z^{-1}\,y(-1) + y(-2)]$$
$$= 3\,[z^{-1}\,R(z) + r(-1)] - [z^{-2}\,R(z) + z^{-1}\,r(-1) + r(-2)]$$

Since $r(-1) = r(-2) = 0$, we have

$$\left(1 + \tfrac{1}{4}z^{-1} - \tfrac{1}{8}z^{-2}\right)\,Y(z) = (3z^{-1} - z^{-2})R(z) + \tfrac{5}{8}\,z^{-1} - 2$$

or

$$\left(z^2 + \tfrac{1}{4}z - \tfrac{1}{8}\right)\,Y(z) = (3z - 1)\,R(z) + \tfrac{5}{8}\,z - 2z^2$$

Therefore,

$$Y(z) = \frac{3z-1}{z^2 + \frac{1}{4}z - \frac{1}{8}}R(z) + \frac{-2z^2 + \frac{5}{8}z}{z^2 + \frac{1}{4}z - \frac{1}{8}}$$

For (refer to Example 2.10)

$$R(z) = \mathcal{Z}\,[(-1)^k] = \frac{z}{z+1},$$

$$Y(z) = \frac{z(3z-1)}{\left(z^2 + \frac{1}{4}z - \frac{1}{8}\right)(z+1)} + \frac{-2z^2 + \frac{5}{8}z}{z^2 + \frac{1}{4}z - \frac{1}{8}}$$

$$= \frac{-2z^3 + \frac{13}{8}z^2 - \frac{3}{8}z}{\left(z + \frac{1}{2}\right)\left(z - \frac{1}{4}\right)(z+1)}$$

Expanding $Y(z)/z$ into partial fractions,

$$\frac{Y(z)}{z} = \frac{-2z^2 + \frac{13}{8}z - \frac{3}{8}}{\left(z + \frac{1}{2}\right)\left(z - \frac{1}{4}\right)(z+1)} = \frac{\frac{9}{2}}{z + \frac{1}{2}} + \frac{-\frac{1}{10}}{z - \frac{1}{4}} + \frac{\frac{-32}{5}}{z+1}$$

Then (refer to Table 2.1)

$$y(k) = \left[\tfrac{9}{2}\left(-\tfrac{1}{2}\right)^k - \tfrac{1}{10}\left(\tfrac{1}{4}\right)^k - \tfrac{32}{5}(-1)^k\right]\mu(k)$$

Review Example 2.3

Consider a second-order discrete-time system described by the difference equation

$$y(k+2) - \tfrac{3}{2}\,y(k+1) + \tfrac{1}{2}\,y(k) = r(k+1) + \tfrac{1}{2}\,r(k)$$

The system is initially relaxed ($y(k) = 0$ for $k < 0$) and is excited by the input

$$r(k) = \begin{cases} 0\,; & k = 0 \\ 1\,; & k > 0 \end{cases}$$

Shifting the difference equation by two sampling intervals, we obtain

$$y(k) - \tfrac{3}{2}\,y(k-1) + \tfrac{1}{2}\,y(k-2) = r(k-1) + \tfrac{1}{2}\,r(k-2)$$

z-transformation of this equation gives

$$Y(z) = \frac{z^{-1} + \frac{1}{2}z^{-2}}{1 - \frac{3}{2}z^{-1} + \frac{1}{2}z^{-2}} R(z) = \frac{z + \frac{1}{2}}{\left(z - \frac{1}{2}\right)(z-1)} R(z)$$

The system modes are $\left(\frac{1}{2}\right)^k$ and $(1)^k$. The mode $\left(\frac{1}{2}\right)^k$ decays as $k \to \infty$, and the mode $(1)^k$ is constant (i.e., it remains within finite bounds for all k).

The input $r(k) = \mu(k-1)$

Therefore, $R(z) = z^{-1}\left[\frac{z}{z-1}\right] = \frac{1}{z-1}$

For this input, $Y(z) = \frac{z + \frac{1}{2}}{\left(z - \frac{1}{2}\right)(z-1)^2}$

It is observed that excitation pole matches one of the system poles. Though the system modes, as well as the input, do not grow with increasing k, the effect of the pole-matching is to give rise to a time function, in forced response of the system, that grows indefinitely as $k \to \infty$. This is evident from the inverse transform of $Y(z)$ (refer to Table 2.1)

$$Y(z) = \frac{A_1}{z - \frac{1}{2}} + \frac{A_2}{(z-1)^2} + \frac{A_3}{z-1}$$

$$A_1 = \left(z - \frac{1}{2}\right) Y(z)\Big|_{z=\frac{1}{2}} = 4$$

$$A_2 = (z-1)^2 Y(z)\Big|_{z=1} = 3$$

$$A_3 = \frac{d}{dz}[(z-1)^2 Y(z)]\Big|_{z=1} = -4$$

Therefore,

$$Y(z) = \frac{4}{z - \frac{1}{2}} + \frac{3}{(z-1)^2} + \frac{-4}{z-1}$$

$$y(k) = 4\left(\frac{1}{2}\right)^{k-1} + 3(k-1)(1)^{k-1} - 4(1)^{k-1}$$

$$= 4\left(\frac{1}{2}\right)^{k-1} + 3(k-1) - 4;\ k \geq 1$$

$$= 4\left(\frac{1}{2}\right)^{k-1} + 3k - 7;\ k \geq 1$$

Review Example 2.4

Solve for $y(k)$ the equation:

$$y(k) = r(k) - r(k-1) - y(k-1),\ k \geq 0$$

where $$r(k) = \begin{cases} 1; & k \text{ even} \\ 0; & k \text{ odd} \end{cases};\ y(-1) = r(-1) = 0$$

Solution The z-transformation of the given equation yields

$$Y(z) = \frac{1 - z^{-1}}{1 + z^{-1}} R(z) = \frac{z-1}{z+1} R(z)$$

For the given input,

$$R(z) = 1 + z^{-2} + z^{-4} + \cdots = \left.\frac{1}{1-x}\right|_{x=z^{-2}} = \frac{1}{1-z^{-2}} = \frac{z^2}{z^2-1}$$

Thus $$Y(z) = \left(\frac{z-1}{z+1}\right)\frac{z^2}{z^2-1} = \frac{z^2}{z^2+2z+1}$$

We can expand $Y(z)$ into a power series by dividing the numerator of $Y(z)$ by its denominator:

$$\begin{array}{r} 1 - 2z^{-1} + 3z^{-2} - 4z^{-3} + \cdots \\ z^2 + 2z + 1 \overline{\big)\ z^2} \\ \underline{z^2 + 2z + 1} \\ -2z - 1 \\ \underline{-2z - 4 - 2z^{-1}} \\ 3 + 2z^{-1} \\ \underline{3 + 6z^{-1} + 3z^{-2}} \\ -4z^{-1} - 3z^{-2} \\ \vdots \end{array}$$

Therefore, $$Y(z) = 1 - 2z^{-1} + 3z^{-2} - 4z^{-3} + \cdots$$

and the values of $y(k)$ are $\{1, -2, 3, -4, \ldots\}$

Review Example 2.5

Through this simple example, we explain the phenomenon of aliasing.

Suppose the sampling rate is 10 Hz; $\omega_s = 20\ \pi$ rad/sec, $T = 0.1$ sec. The primary strip in the s-plane corresponding to this sampling rate is shown in Fig. 2.48a.

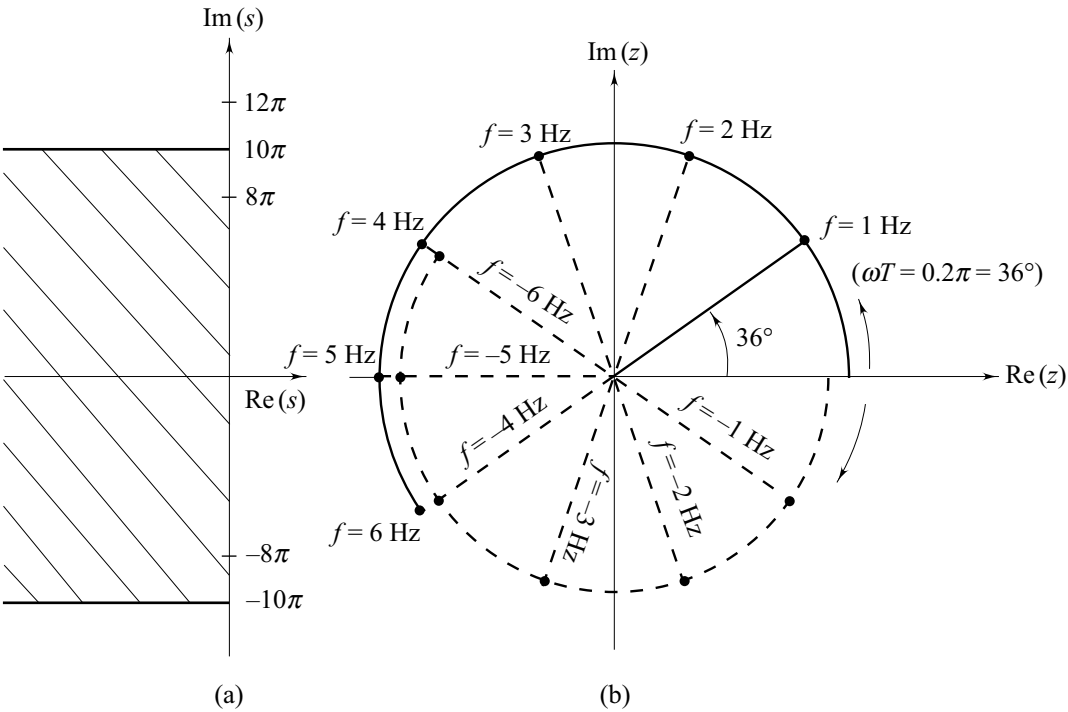


Fig. 2.48

We try to sample 6 Hz sine wave ($\omega_0 = 12\pi$). Note that the signal lies outside the primary strip. Consider mapping of the imaginary axis of the *s*-plane to the *z*-plane, as frequency increases from 0 to 6 Hz. The paths followed as the frequency increases are shown in Fig. 2.48b.

Note that at a frequency of 5 Hz, the two paths meet at $z = -1$. The 6 Hz ($\omega_0 = 12\ \pi$) sine wave will appear to be (10 Hz – 6 Hz) = 4 Hz sine wave. The high frequency $\omega_0 = 12$ rad/sec is 'folded in' about the folding frequency $\pi/T = 10\pi$; and appears as low frequency at $(\omega_s - \omega_0) = \left(\frac{2\pi}{T} - \omega_0\right) = 8\pi$.

The high frequency ω_0, which shows up at $(\omega_s - \omega_0)$ after sampling, is called the 'alias' of the primary-strip frequency $(\omega_s - \omega_0)$. The superimposition of the high-frequency behavior onto the low frequency is known as *frequency folding or aliasing.*

Take a sine wave of 6 Hz frequency and extract the samples with $T = 0.1$ sec. Examine the sampled recording carefully; it has a frequency of 4 Hz.

The phenomenon of aliasing has a clear meaning in time. Two continuous sinusoids of different frequencies (6 Hz and 4 Hz in the example under consideration) appear at the same frequency when sampled. We cannot, therefore, distinguish between them, based on their samples alone.

To avoid aliasing, the requirement is that the *sampling frequency ω_s, must be at least twice the highest frequency ω_m present in the signal*, i.e., $\omega_s > 2\omega_m$. This requirement is formally known as the *sampling theorem*.

Review Example 2.6

Find the response of the system shown in Fig. 2.49 to a unit-impulse input.

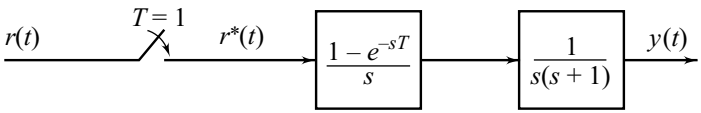


Fig. 2.49

Solution The discrete-time transfer function of the given system is obtained as follows (refer to Eqns (2.100)):

$$G_{h0}(s) = \frac{1-e^{-sT}}{s},\ G_a(s) = \frac{1}{s(s+1)}$$

$$\frac{Y(z)}{R(z)} = \mathcal{Z}[G_{h0}(s)G_a(s)] = (1-z^{-1})\,\mathcal{Z}\left(\frac{G_a(s)}{s}\right)$$

$$= (1-z^{-1})\,\mathcal{Z}\left[\frac{1}{s^2(s+1)}\right] = (1-z^{-1})\,\mathcal{Z}\left[\frac{1}{s^2} - \frac{1}{s} + \frac{1}{s+1}\right]$$

Using Table 2.1, we obtain

$$\frac{Y(z)}{R(z)} = (1-z^{-1})\left[\frac{Tz}{(z-1)^2} - \frac{z}{z-1} + \frac{z}{z-e^{-T}}\right]$$

$$= \left[\frac{(ze^{-T} - z + Tz) + (1 - e^{-T} - Te^{-T})}{(z-1)(z-e^{-T})}\right]$$

For $T = 1$, we have

$$\frac{Y(z)}{R(z)} = \frac{ze^{-1} + 1 - 2e^{-1}}{(z-1)(z-e^{-1})}$$

$$= \frac{0.3678z + 0.2642}{(z-1)(z-0.3679)} = \frac{0.3678z + 0.2642}{z^2 - 1.3678z + 0.3679}$$

For unit-impulse input, $R(z) = 1$.

Therefore, $$Y(z) = \frac{0.3678z + 0.2642}{z^2 - 1.3678z + 0.3679}$$

We can expand $Y(z)$ into a power series by dividing the numerator of $Y(z)$ by its denominator:

$$z^2 - 1.3678z + 0.3678 \overline{)\ 0.3678z + 0.2644}\ \Big)\ 0.3678z^{-1} + 0.7675z^{-2} + 0.9145z^{-3} + \cdots$$

$$0.3678z - 0.5031 + 0.1353z^{-1}$$

$$+ 0.7675 - 0.1353z^{-1}$$

$$+ 0.7675 - 1.0497z^{-1} + 0.2823z^{-2}$$

$$0.9145z^{-1} - 0.2823z^{-2}$$

$$\vdots$$

This calculation yields the response at the sampling instants, and can be carried on as far as needed. In this case we have obtained $y(kT)$ as follows:

$$y(0) = 0,\ y(T) = 0.3678,\ y(2T) = 0.7675, \text{ and } y(3T) = 0.9145.$$

Review Example 2.7

A PID controller is described by the following relation between input $e(t)$ and output $u(t)$:

$$u(t) = K_c \left[e(t) + \frac{1}{T_I} \int_0^t e(t)\, dt + T_D \frac{de(t)}{dt} \right] \tag{2.123}$$

Using the trapezoidal rule for integration and backward-difference approximation for the derivatives, obtain the difference-equation model of the PID algorithm. Also obtain the transfer function $U(z)/E(z)$.

Solution By the trapezoidal rule for integration, we obtain

$$\int_0^{kT} e(t)\, dt = T\left[\frac{e(0)+e(T)}{2} + \frac{e(T)+e(2T)}{2} + \cdots + \frac{e((k-1)T)+e(kT)}{2} \right]$$

$$= T\left[\sum_{i=1}^{k} \frac{e((i-1)T)+e(iT)}{2} \right]$$

By backward-difference approximation for the derivatives (refer to Eqn. (2.102)), we get

$$\left.\frac{de(t)}{dt}\right|_{t=kT} = \frac{e(kT)-e((k-1)T)}{T}$$

A difference-equation model of the PID controller is, therefore, given by

$$u(k) = K_c \left\{ e(k) + \frac{T}{T_I} \sum_{i=1}^{k} \frac{e(i-1)+e(i)}{2} + \frac{T_D}{T}[e(k)-e(k-1)] \right\} \tag{2.124}$$

Let us now obtain the transfer function model of the PID control algorithm given by Eqn. (2.124).

Define (refer to Fig. 2.50)

$$\frac{e(i-1)+e(i)}{2} = f(i);\ f(0) = 0$$

Then

$$\sum_{i=1}^{k} \frac{e(i-1)+e(i)}{2} = \sum_{i=1}^{k} f(i)$$

Taking the z-transform of this equation (refer to Eqn. (2.51)), we obtain

$$\mathcal{Z}\left[\sum_{i=1}^{k} \frac{e(i-1)+e(i)}{2} \right] = \mathcal{Z}\left[\sum_{i=1}^{k} f(i) \right]$$

$$= \frac{z}{z-1} F(z)$$

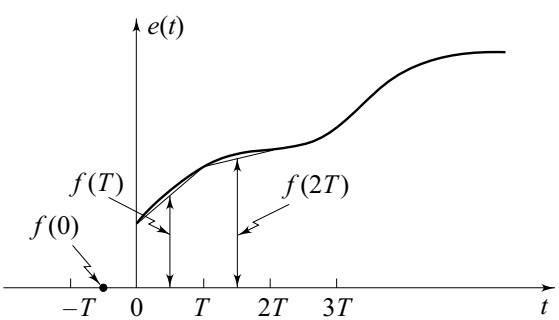


Fig. 2.50 Diagram depicting function $f(kT)$

Notice that $$F(z) = \mathcal{Z}\left[\frac{e(i-1)+e(i)}{2}\right] = \frac{1+z^{-1}}{2}E(z)$$

Hence $$\mathcal{Z}\left[\sum_{i=1}^{k}\frac{e(i-1)+e(i)}{2}\right] = \frac{1+z^{-1}}{2(1-z^{-1})}E(z) = \frac{z+1}{2(z-1)}E(z)$$

The z-transform of Eqn. (2.124) becomes

$$\begin{aligned} U(z) &= K_c\left[1+\frac{T}{2T_I}\frac{1+z^{-1}}{1-z^{-1}}+\frac{T_D}{T}(1-z^{-1})\right]E(z) \\ &= K_c\left[1+\frac{T}{2T_I}\left(\frac{z+1}{z-1}\right)+\frac{T_D}{T}\left(\frac{z-1}{z}\right)\right]E(z) \end{aligned} \tag{2.125}$$

This equation gives the transfer function model of the PID control algorithm. Note that we can obtain the discrete-time transfer function model (2.125) by expressing the PID controller (2.123) in terms of operator s and then using the mapping (2.104) for the derivative term, and the mapping (2.114) for the integral term of the controller.

Review Example 2.8

Derive the difference equation model for the numerical solution of the differential equation

$$\frac{d^2y(t)}{dt^2}+a_1\frac{dy(t)}{dt}+a_2y(t) = r(t); y(0) = y_1^0,\ \frac{dy}{dt}(0) = y_2^0,\ 0 \le t \le t_f \tag{2.126}$$

Use backward-difference approximation for the derivatives.

Solution We divide the interval $0 \le t \le t_f$ into N equal intervals of width equal to *step-length* T:

$$\frac{t_f}{N} = T;\ t = kT,\ k = 0, 1, 2, ..., N$$

By backward-difference approximation,

$$\left.\frac{dy(t)}{dt}\right|_{t=kT} \triangleq \dot{y}(k) = \frac{y(k)-y(k-1)}{T} \tag{2.127}$$

$$\left.\frac{d^2y(t)}{dt^2}\right|_{t=kT} \triangleq \ddot{y}(k) = \frac{\dot{y}(k)-\dot{y}(k-1)}{T} = \frac{1}{T^2}\,[y(k)-2y(k-1)+y(k-2)] \tag{2.128}$$

From Eqn. (2.127), we have

$$\dot{y}(0) = y_2^0 = \frac{y_1^0 - y(-1)}{T}$$

Substituting Eqns (2.127) and (2.128) into (2.126) at $t = kT$, we obtain

$$\frac{1}{T^2}\,[y(k)-2y(k-1)+y(k-2)] + \frac{a_1}{T}\,[y(k)-y(k-1)] + a_2\,y(k) = r(k)$$

or

$$\left(a_2+\frac{a_1}{T}+\frac{1}{T^2}\right)y(k) - \left(\frac{a_1}{T}+\frac{2}{T^2}\right)y(k-1) + \frac{1}{T^2}\,y(k-2) = r(k);$$

$$y(0) = y_1^0, y(-1) = y_1^0 - Ty_2^0 \tag{2.129}$$

Incrementing k to take on values $k = 1, 2, ..., N$, we can easily obtain $y(1), ..., y(N)$ from Eqn. (2.129) by the iterative procedure.

PROBLEMS

2.1 Consider the signal processing algorithm shown in Fig. P2.1.

(a) Assign the state variables and obtain a state variable model for the system.

(b) Represent the algorithm of Fig. P2.1 by a signal flow graph and from there obtain the transfer function model of the system using Mason's gain formula.

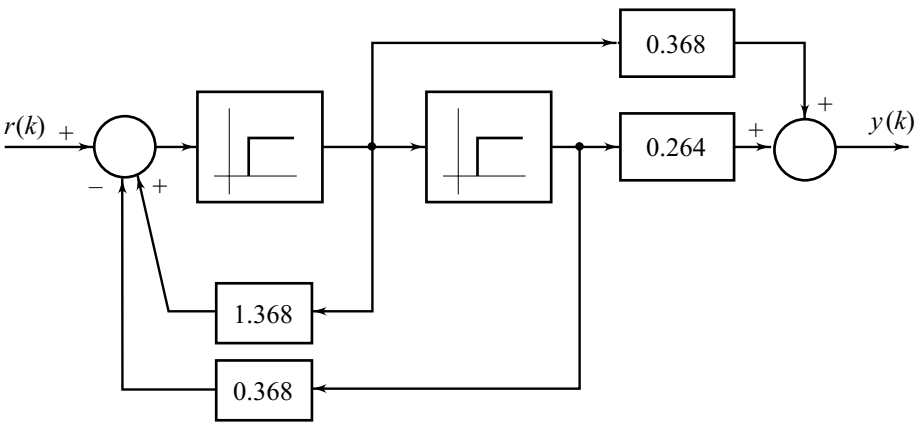


Fig. P 2.1

2.2 Consider the signal processing algorithm shown in Fig. P2.2. Represent the algorithm by (a) difference equation model, (b) a state variable model, and (c) a transfer function model.

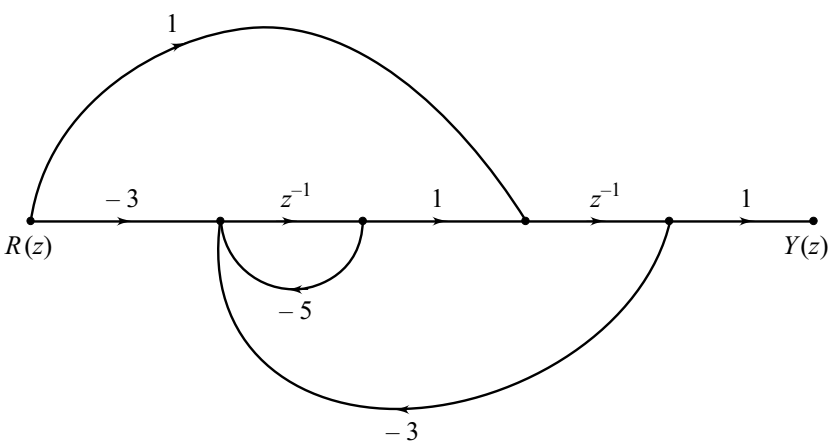


Fig. P 2.2

2.3 Consider the discrete-time system shown in Fig. P2.3.

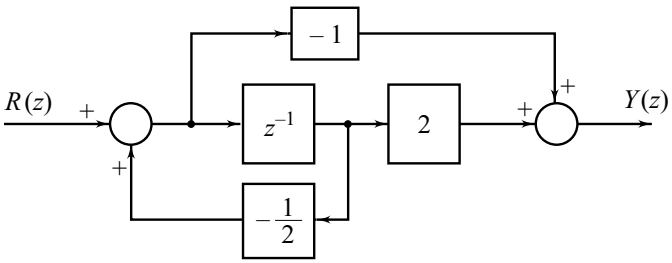


Fig. P 2.3

(a) Obtain the difference equation model and therefrom the transfer function model of the system.
(b) Find the impulse response of the system.
(c) Find the response of the system to unit-step input $\mu(k)$.

2.4 A filter often used as part of computer-controlled algorithms is shown in Fig. P2.4 (β and α are real constants).
(a) Find the impulse response to an impulse of strength A.
(b) Find the step response to a step of strength A.
(c) Find the ramp response to a ramp function with a slope A.
(d) Find the response to sinusoidal input $A\cos(\Omega k)$.

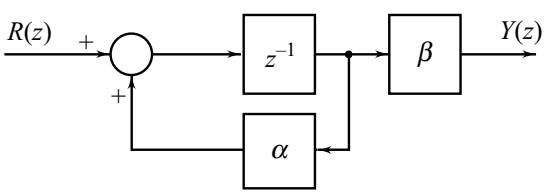


Fig. P2.4

2.5 Solve the following difference equations:

(a) $y(k+2) + 3y(k+1) + 2y(k) = 0;\ y(-1) = -\frac{1}{2},\ y(-2) = \frac{3}{4}$

(b) $2y(k) - 2y(k-1) + y(k-2) = r(k)$
$y(k) = 0$ for $k < 0$ and

$$r(k) = \begin{cases} 1; & k = 0, 1, 2, \ldots \\ 0; & k < 0 \end{cases}$$

2.6 Consider the difference equation:

$$y(k+2) - 1.3679\, y(k+1) + 0.3679\, y(k) = 0.3679\, r(k+1) + 0.2642\, r(k)$$
$$y(k) = 0 \text{ for } k \le 0, \text{ and } r(k) = 0 \text{ for } k < 0,$$
$$r(0) = 1,\ r(1) = 0.2142,\ r(2) = -0.2142;\ r(k) = 0,\ k = 3, 4, 5, \ldots$$

Determine the output $y(k)$.

2.7 Solve the following difference equation using z-transforms:

$$y(k) - 3y(k-1) + 2y(k-2) = r(k)$$

where $$r(k) = \begin{cases} 1 \text{ for } k = 0, 1 \\ 0 \text{ for } k \geq 2 \end{cases} ; y(-2) = y(-1) = 0$$

Will the final value theorem give the correct value of $y(k)$ as $k \to \infty$? Why?

2.8 For the transfer function models and inputs given below, find the response $y(k)$ as a function of k:

(a) $$G(z) = \frac{Y(z)}{R(z)} = \frac{2z-3}{(z-0.5)(z+0.3)}$$

$$r(k) = \begin{cases} 1; & k = 1 \\ 0; & k = 0, 2, 3, 4, \ldots \end{cases}$$

(b) $$G(z) = \frac{Y(z)}{R(z)} = \frac{-6z+1}{\left(z-\frac{1}{2}+j\frac{1}{4}\right)\left(z-\frac{1}{2}-j\frac{1}{4}\right)}$$

$$r(k) = \begin{cases} 0; & k < 1 \\ 1; & k = 0, 1, 2, 3, \ldots \end{cases}$$

2.9 For the transfer function models and inputs given below, find the response $y(k)$ as a function of k:

(a) $$G(z) = \frac{Y(z)}{R(z)} = \frac{1}{(z-0.5)(z+0.3)}$$

$$r(k) = \begin{cases} 1; & k \text{ even} \\ 0; & k \text{ odd} \end{cases}$$

(b) $$G(z) = \frac{Y(z)}{R(z)} = \frac{1}{(z-0.5)^2 (z-0.1)}$$

$$r(k) = \begin{cases} 0; & k < 0 \\ 1; & k = 0, 1, 2, 3, \ldots \end{cases}$$

2.10 Determine $y(\infty)$ for the following $Y(z)$ function (a is a real constant):

$$Y(z) = \frac{K[z^3 - 2az^2 + (a^3 - a^2 + a)z]}{(z-1)(z-a)(z-a^2)}$$

Assuming stable response, determine what the value of K must be for $y(\infty) = 1$.

2.11 Given: $$y(k) = \frac{k!(\alpha)^{k-m+1}}{(k-m+1)!\,(m-1)!}\mu(k-m+1)$$

Prove that $y(k)$ decays to zero as $k \to \infty$ if $|a| < 1$.

2.12 A system has the transfer function

$$G(z) = \frac{Y(z)}{R(z)} = \frac{1}{z^2+1}.$$

Show that when the input $r(k)$ is a unit-step function, the output $y(k)$ is bounded; and when the input

$$r(k) = \{1, 0, -1, 0, 1, 0, -1, \ldots\},$$

the output $y(k)$ is unbounded.

Explain why a bounded input produces a bounded output in the first case but an unbounded output in the second case.

2.13 Using Jury stability criterion, check if all the roots of the following characteristic equations lie within the unit circle:

(a) $z^3 - 1.3\,z^2 - 0.08z + 0.24 = 0$

(b) $z^4 - 1.368z^3 + 0.4z^2 + 0.08z + 0.002 = 0$

2.14 Using Jury stability criterion, find if all the poles of the following transfer function lie inside the unit circle on the z-plane.

$$G(z) = \frac{3z^4 + 2z^3 - z^2 + 4z + 5}{z^4 + 0.5z^3 - 0.2z^2 + z + 0.4}$$

2.15 Figure P2.15 shows the input-output description of a D/A converter. The converter is designed in such a way that the old value of the input sample is held constant, until a new sample arrives. Treating each sample of the sequence $r(kT)$ as an impulse function of strength equal to the value of the sample, the system of Fig. P2.15 becomes a continuous-time system. Determine the transfer function model of the system.

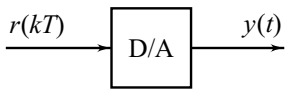


Fig. P 2.15

2.16 (a) State and prove the sampling theorem.

(b) Given: $E(s) = \dfrac{10(s+2)}{s^2(s^2+2s+2)}$

Based upon the sampling theorem, determine the maximum value of the sampling interval T that can be used to enable us to reconstruct $e(t)$ from its samples.

(c) Consider a system with sampling frequency 50 rad/sec. A noise signal cos $50t$ enters into the system. Show that it can cause a dc component in the system output.

2.17 Draw the magnitude and phase curves of the zero-order hold, and compare these curves with those of the ideal low-pass filter.

2.18 Consider a signal $f(t)$, which has discrete values $f(kT)$ at the sampling rate $1/T$. If the signal $f(t)$ is imagined to be impulse sampled at the same rate, it becomes

$$f^*(t) = \sum_{k=0}^{\infty} f(kT)\delta(t - kT)$$

(a) Prove that $F(z)\Big|_{z=e^{sT}} = F^*(s)$

(b) Determine $F(z)\Big|_{z=e^{sT}}$ in terms of $F(s)$. Using this result, explain the relationship between the z-plane and the s-plane.

2.19 Figure P2.19 shows two root paths in the s-plane:

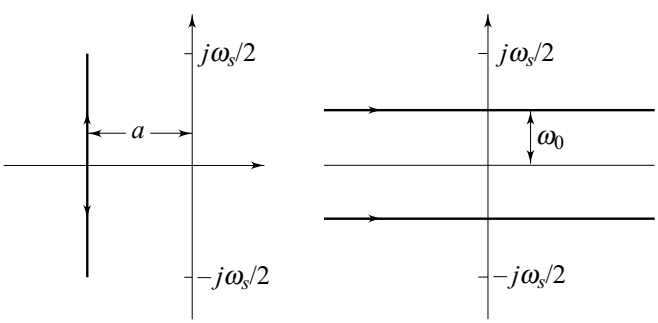


Fig. P 2.19

(i) roots with the same time-constant $\tau = 1/a$, and

(ii) roots with the same oscillation frequency ω_0.

Derive and sketch the corresponding root paths in the z-plane under the impulse-invariant transformation. Sampling frequency is ω_s.

2.20 Figure P2.20 shows a discrete-time system. Determine the transfer function $G(z)$ of this system assuming that the samplers operate synchronously at intervals of T sec. Also find the unit-step response of the system.

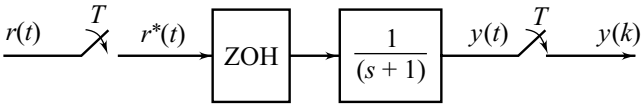


Fig. P 2.20

2.21 Figure P2.21 shows the model of a plant driven by a D/A converter. Derive the transfer function model relating $r(kT)$ and $y(kT)$; $T = 0.4$ sec.

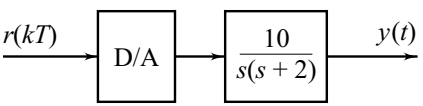


Fig. P 2.21

2.22 Show that if y is the integral of a function r, then

(i) by the backward rectangular rule for integration,

$$Y(z) = \frac{Tz}{z-1} R(z);$$

(ii) by the trapezoidal rule for integration

$$Y(z) = \frac{T}{2}\frac{z+1}{z-1} R(z)$$

2.23 Consider the discretization method based on the backward-difference approximation of derivatives, as a mapping from the s-plane to the z-plane. Show that the mapping transforms the left half of the s-plane into a circle in the z-plane. Is the size of the circle dependent on the choice of the sampling interval?

2.24 Prove that the bilinear transformation maps the left half of the s-plane into the unit circle in the z-plane.

The transformation $z = e^{sT}$ also maps the left half of the s-plane into the unit circle in the z-plane. What is the difference between the two maps?

2.25 A PID controller is described by the following relation between input $e(t)$ and output $u(t)$:

$$u(t) = K_c \left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D \frac{de(t)}{dt} \right]$$

Obtain the PID control algorithm by the discretization of the equation:

$$\dot{u}(t) = K_c \left[\dot{e}(t) + \frac{1}{T_I} e(t) + T_D \ddot{e}(t) \right]$$

using the backward-difference approximation of the derivatives. Also find the transfer function $U(z)/E(z)$.

2.26 For a plant $1.57/[s(s + 1)]$, we are required to design a digital controller so that the closed-loop system acquires a damping ratio of 0.45 without loss of steady-state accuracy. The sampling period $T = 1.57$ sec. The following design procedure may be followed:

(i) First we design the analog controller $D(s)$ defined in Fig. P2.26. The transfer function $G_h(s)$ has been inserted in the analog control loop, to take into account the effect of the hold that must be included in the equivalent digital control system. Verify that $D(s) = \dfrac{25s+1}{62.5s+1}$ meets the design requirements.

(ii) Discretize $D(s)$ using bilinear transformation.

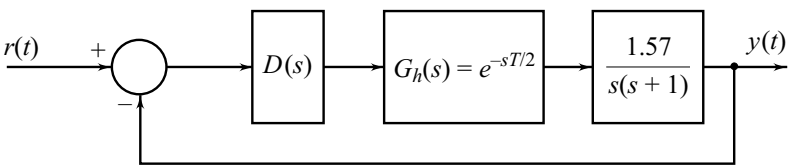


Fig. P 2.26

2.27 A PID controller is described by the following relation between input $e(t)$ and output $u(t)$:

$$U(s) = K_c\left[1 + \frac{1}{T_I s} + T_D s\right]E(s)$$

(a) Derive the PID algorithm using the s-plane to z-plane maps—bilinear transformation for integration and backward-difference approximation for the derivatives.

(b) Convert the transfer function model of the PID controller obtained in step (a) into a difference equation model.

2.28 Derive difference equation models for the numerical solution of the following differential equation using (a) the backward rectangular rule for integration, and (b) the forward rectangular rule for integration:

$$\dot{y}(t) + ay(t) = r(t);\ y(0) = y^0$$

2.29 Consider the second-order system

$$\ddot{y} + a\dot{y} + by = 0;\ y(0) = \alpha,\ \dot{y}(0) = \beta$$

Approximate this equation with a second-order difference equation for computer solution. Use backward-difference approximation for the derivatives.

Chapter 3

Models of Digital Control Devices and Systems

3.1 INTRODUCTION

Now that we have developed the prerequisite signal processing techniques in Chapter 2, we can use them to study closed-loop digital control systems. A typical topology of the type of systems to be considered in this chapter, is shown in Fig. 3.1.

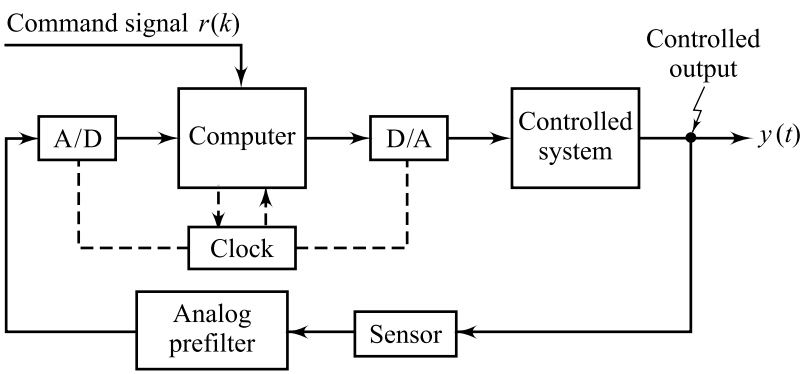


Fig. 3.1 Configuration of the basic digital control scheme

Digital control systems with analog sensors include an analog prefilter between the sensor and the sampler (A/D converter) as an *anti-aliasing* device. The prefilters are low-pass, and the simplest transfer function is

$$H_{pf}(s) = \frac{a}{s+a}$$

so that the noise above the prefilter breakpoint a, is attenuated. The design goal is to provide enough attenuation at half the sample rate ($\omega_s/2$) so that the noise above $\omega_s/2$, when aliased into lower frequencies by the sampler, will not be detrimental to the control-system performance.

Since the phase lag from the prefilter can significantly affect system stability, it is required that the control design be carried out with the analog prefilter included in the loop transfer function. An alternative design

procedure is to select the breakpoint and ω_s sufficiently higher than the system bandwidth, so that the phase lag from the prefilter does not significantly alter the system stability, and thus the prefilter design problem can be divorced from the control-law design problem. Our treatment of the subject is based on this alternative design procedure. We, therefore, ignore the prefilter design and focus on the basic control-system design. The basic configuration for this design problem is shown in Fig. 3.2, where

$G(s)$ = transfer function of the controlled plant (continuous-time system);
$H(s)$ = transfer function of the analog sensor; and
$D(z)$ = transfer function of the digital control algorithm.

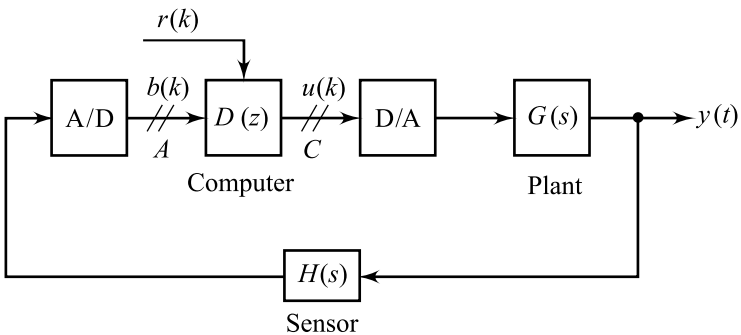


Fig. 3.2 Block diagram for basic control system design

The analog and digital parts of the system are connected through D/A and A/D converters. The computer, with its internal clock, drives the D/A and A/D converters. It compares the command signal $r(k)$ with the feedback signal $b(k)$ and generates the control signal $u(k)$, to be sent to the final control elements of the controlled plant. These signals are computed from the digital control algorithm $D(z)$, stored in the memory of the computer.

There are two different approaches for the design of digital algorithms.

(i) Discretization of Analog Design The controller design is done in the s-domain using analog design methods.[1] The resulting analog control law is then converted to discrete-time form, using one of the approximation techniques given in Section 2.14.

(ii) Direct Digital Design In this approach, we first develop the discrete-time model of the analog part of the loop—from C to A in Fig. 3.2—that includes the controlled plant. The controller design is then performed using discrete-time analysis.

An actual design process is often a combination of the two methods. First iteration to a digital design can be obtained using discretization of an analog design. Then the result is tuned up using direct digital analysis and design.

The intent of this chapter is to provide basic tools for the analysis and design of a control system that is to be implemented using a computer. Mathematical models of commonly used digital control devices and systems are developed. Different ways to implement digital controllers (obtained by the discretization of analog design (Section 2.14) or by direct digital design (Chapter 4), are also given in this chapter.

[1] Chapters 7–10 of reference [155].

3.2 *z*-DOMAIN DESCRIPTION OF SAMPLED CONTINUOUS-TIME PLANTS

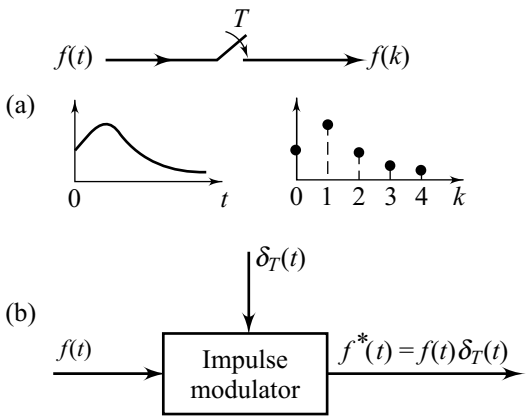


Fig. 3.3 A model of an A/D converter

Whenever a digital computer is used to control a continuous-time plant, there must be some type of *interface system* that takes care of the communication between the discrete-time and the continuous-time systems. In the system of Fig. 3.2, the interface function is performed by A/D and D/A converters.

Simple models of the interface actions of A/D and D/A converters have been developed in Chapter 2. A brief review is in order here.

Model of an A/D Converter

A simple model of an A/D converter is shown in Fig. 3.3a. A continuous-time function $f(t)$, $t \geq 0$, is the input and the sequence of real numbers $f(k)$, $k = 0, 1, 2, \ldots$, is the output. The following relation holds between input and output:

$$f(k) = f(t = kT);\ T \text{ is the time interval between samples.} \tag{3.1}$$

The sequence $f(k)$ can be treated as a train of impulses represented by continuous-time function $f^*(t)$:

$$\begin{aligned} f^*(t) &= f(0)\ \delta(t) + f(1)\ \delta(t - T) + f(2)\ \delta(t - 2T) + \cdots \\ &= \sum_{k=0}^{\infty} f(k)\ \delta(t - kT) \end{aligned} \tag{3.2}$$

The sampler of Fig. 3.3a can thus be viewed as an 'impulse modulator' with the carrier signal

$$\delta_T(t) = \sum_{k=0}^{\infty} \delta(t - kT) \tag{3.3}$$

and modulating signal $f(t)$. A schematic representation of the modulation process is shown in Fig. 3.3b.

Model of a D/A Converter

A simple model of a D/A converter is shown in Fig. 3.4a. A sequence of numbers $f(k)$, $k = 0, 1, 2, \ldots$, is the input, and the continuous-time function $f^+(t)$, $t \geq 0$ is the output. The following relation holds between input and output:

$$f^+(t) = f(k);\ kT \leq t < (k + 1)T \tag{3.4}$$

Each sample of the sequence $f(k)$ may be treated as an impulse function of the form $f(k)\,\delta(t - kT)$. The *Zero-Order Hold* (ZOH) of Fig. 3.4a can thus be viewed as a linear time-invariant system that converts the impulse $f(k)\,\delta(t - kT)$ into a pulse of height $f(k)$ and width T. The D/A converter may, therefore, be modeled by Fig. 3.4b, where the ZOH is a system whose response to a unit impulse $\delta(t)$, is a unit pulse

$g_{h0}(t)$ of width T. The Laplace transform of $g_{h0}(t)$ is the transfer function of hold operation, namely (refer to Eqn. (2.79))

$$G_{h0}(s) = \mathscr{Z}[g_{h0}(t)] = \frac{1-e^{-sT}}{s} \qquad (3.5)$$

Figure 3.5 illustrates a typical example of an interconnection of discrete-time and continuous-time systems. In order to analyze such a system, it is often convenient to represent the continuous-time system together with the ZOH by an *equivalent discrete-time system*.

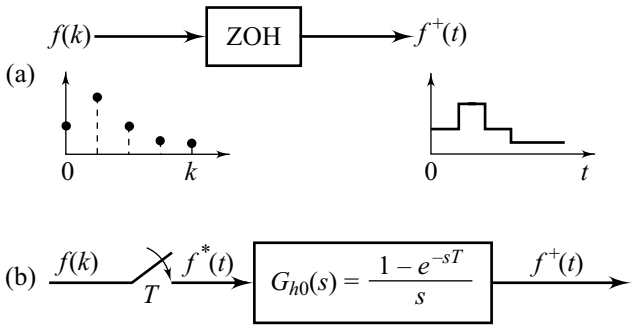


Fig. 3.4 A model of a D/A converter

We assume that the continuous-time system of Fig. 3.5 is a linear system with the transfer function $G(s)$. A block diagram model of the equivalent discrete-time system is shown in Fig. 3.6a. As seen from this figure, the impulse modulated signal $u^*(t)$ is applied to two s-domain transfer functions in tandem. Since the two blocks with transfer functions $G_{h0}(s)$ and $G(s)$ are not separated by an impulse modulator, we can consider them as a single block with transfer function $[G_{h0}(s)G(s)]$, as shown in Fig. 3.6b. The continuous-time system with transfer function $[G_{h0}(s)G(s)]$ has input $u^*(t)$ and output $y(t)$. The output signal $y(t)$ is read off at discrete synchronous sampling instants kT; $k = 0, 1, ...$, by means of a mathematical sampler $T(M)$.

We assume that $\hat{g}(t)$ is the impulse response of the continuous-time system $G_{h0}(s)G(s)$:

$$\hat{g}(t) = \mathscr{L}^{-1}[G_{h0}(s)G(s)] \qquad (3.6)$$

The input signal to the system is given by (refer to Eqn. (3.2)),

$$u^*(t) = \sum_{k=0}^{\infty} u(kT)\,\delta(t-kT) \qquad (3.7)$$

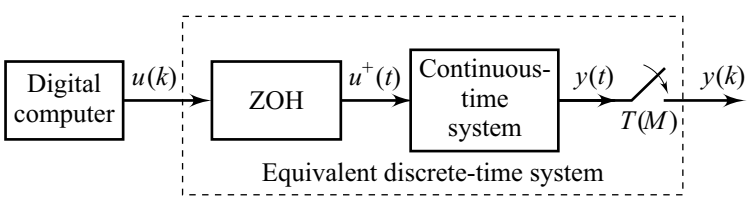


Fig. 3.5 Interconnection of discrete-time and continuous-time systems

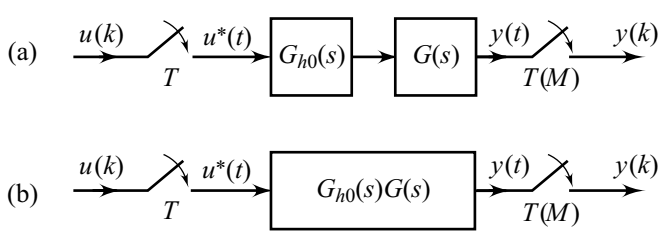


Fig. 3.6 Block diagrams for an equivalent discrete-time system

This is a sequence of impulses with intensities given by $u(kT)$. Since $\hat{g}(t)$ is the impulse response of the system (response to the input $\delta(t)$), by superposition from Eqn. (3.7),

$$y(t) = \sum_{j=0}^{\infty} u(jT)\ \hat{g}(t - jT)$$

At the sampling times $t = kT$, $y(t)$ is given by

$$y(kT) = \sum_{j=0}^{\infty} u(jT)\ \hat{g}(kT - jT) \tag{3.8}$$

We can recognize it at once as discrete-time convolution (refer to Eqn. (2.25)). Taking the z-transform of both sides of Eqn. (3.8), we obtain (refer to Eqn. (2.29)),

$$Y(z) = U(z)\,\hat{G}(z) \tag{3.9a}$$

where

$$\begin{aligned}\hat{G}(z) &= \mathcal{Z}[\hat{g}(kT)] \\ &= \mathcal{Z}[\mathcal{L}^{-1}\{G_{h0}(s)G(s)\}|_{t=kT}]\end{aligned} \tag{3.9b}$$

The z-transforming operation of Eqn. (3.9b) is commonly indicated as

$$\hat{G}(z) = \mathcal{Z}[G_{h0}(s)G(s)] = G_{h0}G(z) \tag{3.10}$$

It may be carefully noted that since the two blocks $G_{h0}(s)$ and $G(s)$ are not separated by an impulse modulator,

$$\hat{G}(z) \neq \mathcal{Z}[G_{h0}(s)]\ \mathcal{Z}[G(s)] \tag{3.11a}$$

$$\neq G_{h0}(z)\ G(z) \tag{3.11b}$$

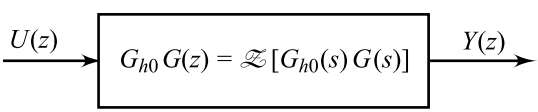


Fig. 3.7 The z-domain equivalent of Fig. 3.6

It follows from Eqns (3.9), that the block diagram of Fig. 3.6b becomes the z-domain block diagram of Fig. 3.7. We could, of course, directly draw the z-domain block diagram of Fig. 3.7 from Fig. 3.6b, which implies that

$$Y(s) = G_{h0}(s)G(s)U^*(s) \leftrightarrow Y(z) = G_{h0}G(z)U(z) \tag{3.12a}$$

or

$$\begin{aligned}\mathcal{Z}[G_{h0}(s)G(s)U^*(s)] &= \mathcal{Z}[G_{h0}(s)G(s)]\ \mathcal{Z}[U^*(s)] \\ &= G_{h0}G(z)U(z)\end{aligned} \tag{3.12b}$$

We can use the following relation (refer to Eqns (2.100 b) and (2.100 c)) for evaluation of $G_{h0}G(z)$:

$$\begin{aligned}G_{h0}G(z) &= \mathcal{Z}[G_{h0}(s)G(s)] \\ &= \mathcal{Z}\left[(1 - e^{-sT})\frac{G(s)}{s}\right] \\ &= (1 - z^{-1})\ \mathcal{Z}\left[\frac{G(s)}{s}\right]\end{aligned} \tag{3.13}$$

Single factor building blocks of the Laplace and z-transform pairs are given in Table 2.1. Expanding $G(s)/s$ into partial fractions, $\mathcal{Z}[G(s)/s]$ can be found by using this table.

Consider now the basic sampled-data feedback system, whose block diagram is depicted in Fig. 3.8a. In terms of impulse modulation, this block diagram can be redrawn as shown in Fig. 3.8b.

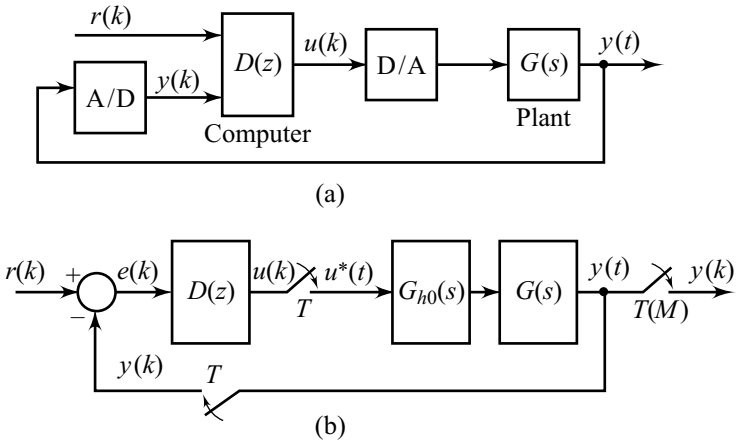


Fig. 3.8 A sampled-data feedback system

Using the relations previously established in this section, we have

$$U(z) = D(z)\, E(z) \tag{3.14a}$$

$$Y(z) = G_{h0}G(z)U(z) \tag{3.14b}$$

Since $$e(kT) = r(kT) - y(kT)$$

we have $$E(z) = R(z) - Y(z) \tag{3.14c}$$

Combining Eqns (3.14a), (3.14b), and (3.14c) gives

$$\frac{Y(z)}{R(z)} = \frac{D(z)G_{h0}G(z)}{1 + D(z)G_{h0}G(z)} \tag{3.15}$$

Figure 3.9 gives the z-domain equivalent of Fig. 3.8.

Having become familiar with the technique, from now onwards we may directly write z-domain relationships, without introducing impulse modulators in block diagrams of sampled-data systems.

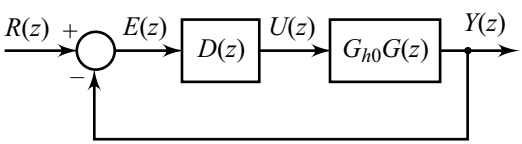


Fig. 3.9 The z-domain equivalent of Fig. 3.8

Consider the sampled-data feedback system of Fig. 3.10 where the sensor dynamics is represented by transfer function $H(s)$. The following equations easily follow:

$$E(z) = R(z) - B(z) \tag{3.16a}$$

$$U(z) = D(z)\, E(z) \tag{3.16b}$$

$$Y(z) = G_{h0}G(z)\, U(z) = \mathcal{Z}[G_{h0}(s)G(s)]\, U(z) \tag{3.16c}$$

$$B(z) = G_{h0}GH(z)U(z) = \mathcal{Z}[G_{h0}(s)G(s)H(s)]U(z) \tag{3.16d}$$

Equations (3.16a), (3.16b) and (3.16d) give

$$\frac{E(z)}{R(z)} = \frac{1}{1 + D(z)\, G_{h0}GH(z)} \tag{3.17}$$

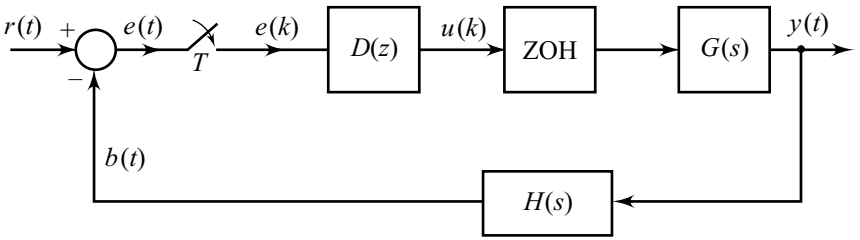


Fig. 3.10 A sampled-data feedback system

Combining Eqns (3.16b), (3.16c) and (3.17), we get

$$\frac{Y(z)}{R(z)} = \frac{D(z)G_{h0}G(z)}{1+D(z)G_{h0}GH(z)} \tag{3.18}$$

Figure 3.11 illustrates a phenomenon that we have not yet encountered. When an input signal is acted upon by a dynamic element before being sampled, it is impossible to obtain a transfer function for the system. The system in Fig. 3.11 differs from that in Fig. 3.10, in that the analog error $e(t)$ is first amplified before being converted to digital form for the control computer. The amplifier's dynamics are given by $G_1(s)$.

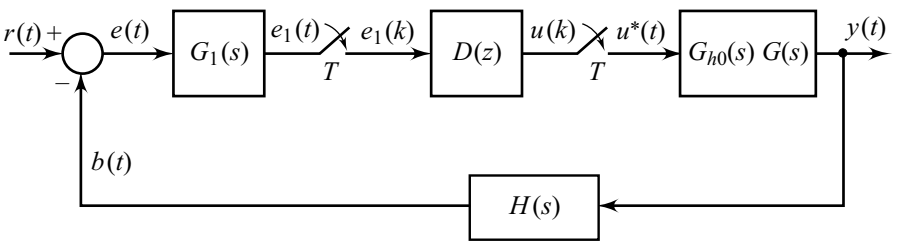


Fig. 3.11 A sampled-data system

Consider first the subsystem shown in Fig. 3.12a. We can equivalently represent it as a block $[G_1(s)E(s)]$ with input $\delta(t)$, as in Fig. 3.12b. Now the input, and therefore, the output, does not change by imagining a fictitious impulse modulator through which $\delta(t)$ is applied to $[G_1(s)E(s)]$ as in Fig. 3.12c.

On application of Eqn. (3.12), we can write

$$E_1(z) = \mathcal{Z}[G_1(s)E(s)]\ \mathcal{Z}[\delta(k)] = \mathcal{Z}[G_1(s)E(s)] \tag{3.19}$$

Now, for the system of Fig. 3.11,

$$\begin{aligned} E(s) &= R(s) - B(s) = R(s) - H(s)Y(s) \\ &= R(s) - H(s)\ G_{h0}(s)\ G(s)\ U^*(s) \end{aligned} \tag{3.20}$$

Therefore, from Eqns (3.19) and (3.20), we obtain

$$\begin{aligned} E_1(z) &= \mathcal{Z}[G_1(s)R(s)] - \mathcal{Z}[G_1(s)H(s)G_{h0}(s)G(s)]U(z) \\ &= G_1R(z) - G_1G_{h0}GH(z)\ U(z) \end{aligned} \tag{3.21}$$

Since $U(z) = D(z)\ E_1(z)$,

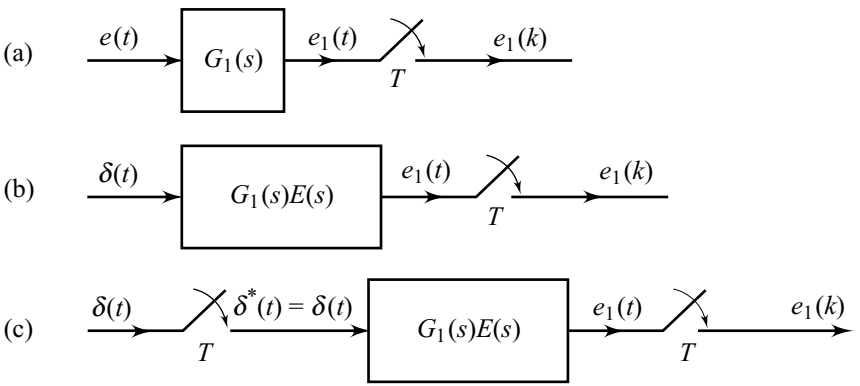


Fig. 3.12 Equivalent *s*-transfer function with impulse modulated input

we can write from Eqn. (3.21),

$$E_1(z) = \frac{G_1R(z)}{1+D(z)G_1G_{h0}H(z)}$$

The output $Y(z) = G_{h0}G(z)\ U(z) = D(z)\ G_{h0}G(z)\ E_1(z)$

$$= \frac{D(z)G_{h0}G(z)G_1R(z)}{1+D(z)G_1G_{h0}GH(z)} \tag{3.22}$$

Since it is impossible to form the ratio $Y(z)/R(z)$ from Eqn. (3.22), we can not obtain a transfer function for the system of Fig. 3.11, and we cannot analyze it further without specifying a functional form for the input $r(t)$.

It is not permissible to create a transfer function for the system in Fig. 3.11 by inserting a fictitious sampler at the input, because this would change the physics of the situation represented by the diagram (the analog input would be replaced by the train of impulses). Fictitious samplers are permissible at only the output, because they are simply a means of selecting the values of the output at the times of interest to us, namely, the sample times.

Example 3.1

Consider the sampled-data system shown in Fig. 3.13a. From the block diagram, we obtain (refer to Eqn. (3.15))

$$\frac{Y(z)}{R(z)} = \frac{G_{h0}G(z)}{1+G_{h0}G(z)} \tag{3.23}$$

Figure 3.13b gives the *z*-domain equivalent of Fig. 3.13a. The forward path transfer function:

$$G_{h0}G(z) = \mathcal{Z}[G_{h0}(s)G(s)]$$

$$= (1-z^{-1})\ \mathcal{Z}\left[\frac{G(s)}{s}\right] = (1-z^{-1})\ \mathcal{Z}\left[\frac{1}{s^2(s+1)}\right]$$

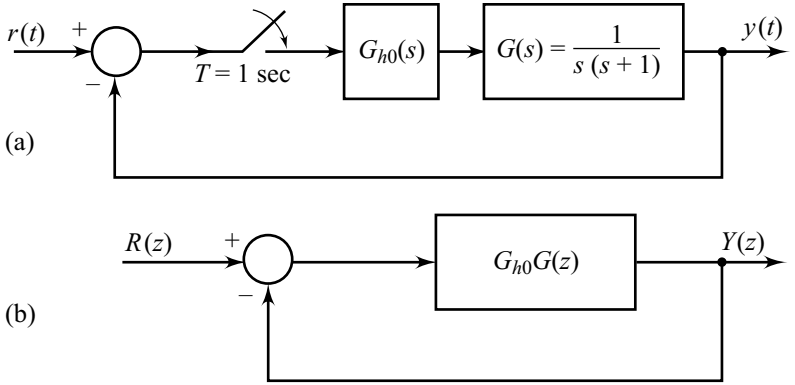


Fig. 3.13 A closed-loop sampled-data system

$$= (1 - z^{-1})\,\mathcal{Z}\left[\frac{1}{s^2} - \frac{1}{s} + \frac{1}{s+1}\right] = (1 - z^{-1})\left[\frac{Tz}{(z-1)^2} - \frac{z}{z-1} + \frac{z}{z - e^{-T}}\right]$$

$$= \frac{z(T - 1 + e^{-T}) + (1 - e^{-T} - Te^{-T})}{(z-1)\,(z - e^{-T})}$$

When $T = 1$, we have

$$G_{h0}G(z) = \frac{ze^{-1} + 1 - 2e^{-1}}{(z-1)\,(z - e^{-1})} = \frac{0.3679z + 0.2642}{z^2 - 1.3679z + 0.3679}$$

Substituting in Eqn. (3.23), we obtain

$$\frac{Y(z)}{R(z)} = \frac{0.3679z + 0.2642}{z^2 - z + 0.6321}$$

For a unit-step input,

$$R(z) = \frac{z}{z-1}$$

and therefore,

$$Y(z) = \frac{z(0.3679z + 0.2642)}{(z-1)\,(z^2 - z + 0.6321)} = \frac{0.3679z^2 + 0.2642z}{z^3 - 2z^2 + 1.6321z - 0.6321}$$

By long-division process, we get

$$Y(z) = 0.3679\,z^{-1} + z^{-2} + 1.3996\,z^{-3} + 1.3996\,z^{-4} + 1.1469\,z^{-5} + 0.8944\,z^{-6} + 0.8015\,z^{-7} + \cdots$$

Therefore, the sequence ($k = 1, 2, ...$)

$$y(kT) = \{0.3679,\ 1,\ 1.3996,\ 1.3996,\ 1.1469,\ 0.8944,\ 0.8015,\ ...\}$$

Note that the final value of $y(kT)$ is (refer to Eqn. (2.52),

$$\lim_{k \to \infty} y(kT) = \lim_{z \to 1} (z - 1)Y(z) = \frac{0.3679 + 0.2642}{0.6321} = 1$$

The unit-step response is shown in Fig. 3.14. Also shown in this figure is the unit-step response of the continuous-time system (i.e., when $T = 0$). The overshoot of the sampled system is 45%, in contrast to 17% for the continuous-time system.

The performance of the digital system is, thus, dependent on the sampling period T. Larger sampling periods usually give rise to higher overshoots in the step response, and may eventually cause instability if the sampling period is too large.

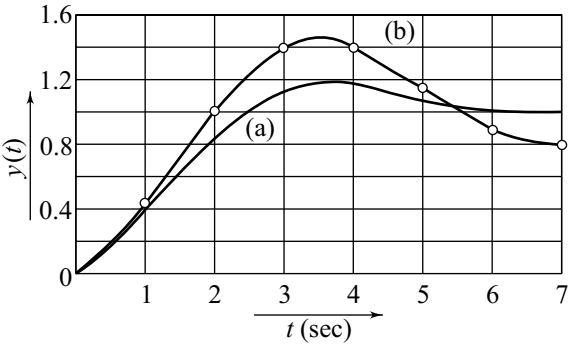


Fig. 3.14 The response of a second-order system: (a) Analog; (b) Sampled

Example 3.2

Let us compare the stability properties of the system shown in Fig. 3.15, with and without a sample-and-hold on the error signal.

Without sample-and-hold, the system in Fig. 3.15 has the transfer function

$$\frac{Y(s)}{R(s)} = \frac{K}{s^2 + 2s + K}$$

This system is stable for all values of $K > 0$.

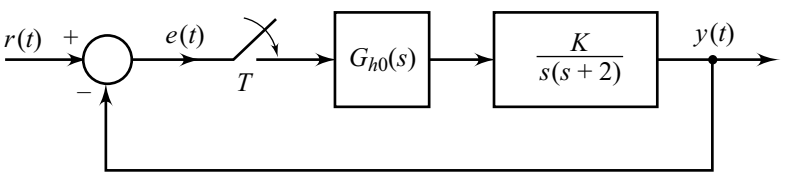


Fig. 3.15 A closed-loop sampled-data system

For the system with sample-and-hold, the forward-path transfer function is given by

$$G_{h0}G(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{K}{s^2(s+2)}\right]$$

$$= (1 - z^{-1})\,\mathcal{Z}\left[\frac{K}{2}\left(\frac{1}{s^2} - \frac{1/2}{s} + \frac{1/2}{s+2}\right)\right]$$

$$= \frac{K}{2}(1 - z^{-1})\left[\frac{Tz}{(z-1)^2} - \frac{(1/2)z}{z-1} + \frac{(1/2)z}{z - e^{-2T}}\right]$$

$$= \frac{K}{2}\left[\frac{2T(z - e^{-2T}) - (z-1)(1 - e^{-2T})}{2(z-1)(z - e^{-2T})}\right]$$

The characteristic equation of the sampled-data system is

$$1 + G_{h0}G(z) = 0$$

or $$4(z-1)(z-e^{-2T}) + 2KT(z-e^{-2T}) - K(z-1)(1-e^{-2T}) = 0$$

or $$z^2 + z\left[\frac{1}{2}K\left(T - \frac{1}{2} + \frac{1}{2}e^{-2T}\right) - 1 - e^{-2T}\right] + e^{-2T} + \frac{1}{2}K\left[\frac{1}{2} - \frac{1}{2}e^{-2T} - Te^{-2T}\right] = 0$$

Case 1: $T = 0.4$ *sec*

For this value of sampling period, the characteristic polynomial becomes

$$\Delta(z) = z^2 + (0.062K - 1.449)z + 0.449 + 0.048K$$

Applying the Jury stability test (refer to Eqns (2.73)–(2.75)), we find that the system is stable if the following conditions are satisfied:

$$\Delta(1) = 1 + 0.062K - 1.449 + 0.449 + 0.048K > 0$$

$$\Delta(-1) = 1 - 0.062K + 1.449 + 0.449 + 0.048K > 0$$

$$|\,0.449 + 0.048K\,| < 1$$

These conditions are satisfied for $0 < K < 11.479$.

Case II: $T = 3$ *sec*

For this value of sampling period, the characteristic polynomial becomes

$$\Delta(z) = z^2 + (1.2506K - 1.0025)\,z + 0.0025 + 0.2457K$$

The system is found to be stable for $0 < K < 1.995$.

Thus, the system which is stable for all $K > 0$ when $T = 0$ (continuous-time system), becomes unstable for $K > 11.479$ when $T = 0.4$ sec. When T is further increased to 3 sec, it becomes unstable for $K > 1.995$. It means that increasing the sampling period (or decreasing the sampling rate), reduces the margin of stability.

3.3 *z*-DOMAIN DESCRIPTION OF SYSTEMS WITH DEAD-TIME

Figure 3.16 is the block diagram of a computer-controlled continuous-time system with dead-time. We assume that the continuous-time system is described by transfer function of the form

$$G_p(s) = G(s)\,e^{-\tau_D s} \tag{3.24}$$

where τ_D is the dead-time, and $G(s)$ contains no dead-time.

The equivalent discrete-time system, shown by dotted lines in Fig. 3.16, is described by the model

$$\frac{Y(z)}{U(z)} = \mathcal{Z}\,[G_{h0}(s)G_p(s)] = G_{h0}G_p(z) \tag{3.25a}$$

$$= (1 - z^{-1})\,\mathcal{Z}\left[\frac{1}{s}e^{-\tau_D s}G(s)\right] \tag{3.25b}$$

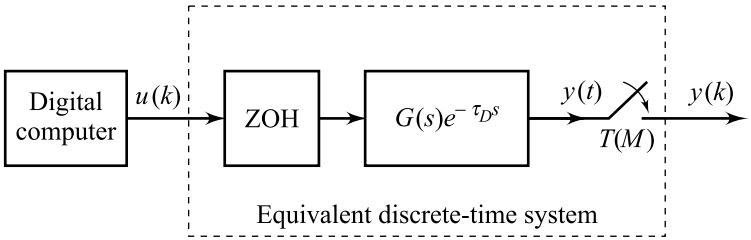


Fig. 3.16 A sampled continuous-time system with dead-time

If N is the largest integer number of sampling periods in τ_D, we can write

$$\tau_D = NT + \Delta T;\ 0 \le \Delta < 1 \tag{3.26}$$

(Given the capabilities of modern microprocessors, we can adjust the sampling frequency slightly so that $\tau_D = NT$)

Therefore, $$G_{h0}G_p(z) = (1 - z^{-1})z^{-N}\,\mathcal{Z}\left[\frac{1}{s}e^{-\Delta Ts}G(s)\right]$$

Let us take an example where

$$G(s) = \frac{1}{s+a} \tag{3.27}$$

For this example, $$\frac{Y(z)}{U(z)} = G_{h0}G_p(z) = (1 - z^{-1})z^{-N}\,\mathcal{Z}\left[\frac{e^{-\Delta Ts}}{s(s+a)}\right]$$

$$= \frac{1}{a}(1 - z^{-1})\,z^{-N}\,\mathcal{Z}\left[\frac{e^{-\Delta Ts}}{s} - \frac{e^{-\Delta Ts}}{s+a}\right] \tag{3.28}$$

Now $$\mathcal{L}^{-1}\left[\frac{e^{-\Delta Ts}}{s}\right] = g_1(t) = \mu(t - \Delta T);\ \mathcal{L}^{-1}\left[\frac{e^{-\Delta Ts}}{s+a}\right] = g_2(t) = e^{-a(t-\Delta T)}\,\mu(t - \Delta T)$$

where $\mu(t)$ is a unit-step function.

Therefore, $$g_1(kT) = \mu(kT - \Delta T);\ g_2(kT) = e^{-a(kT-\Delta T)}\,\mu(kT - \Delta T)$$

$$\mathcal{Z}[g_1(kT)] = \sum_{k=0}^{\infty} g_1(kT)z^{-k} = z^{-1} + z^{-2} + z^{-3} + \cdots$$

$$= z^{-1}(1 + z^{-1} + z^{-2} + \cdots) = z^{-1}\left(\frac{1}{1 - z^{-1}}\right) = \frac{1}{z-1} \tag{3.29}$$

$$\mathcal{Z}[g_2(kT)] = \sum_{k=0}^{\infty} g_2(kT)z^{-k} = e^{-a(T-\Delta T)}z^{-1} + e^{-a(2T-\Delta T)}z^{-2} + e^{-a(3T-\Delta T)}z^{-3} + \cdots$$

We introduce a parameter m, such that

$$m = 1 - \Delta$$

Then $$\mathcal{Z}[g_2(kT)] = e^{-amT}z^{-1} + e^{-amT}e^{-aT}z^{-2} + e^{-amT}e^{-2aT}z^{-3} + \cdots$$

$$= e^{-amT}z^{-1}\left[1 + e^{-aT}z^{-1} + e^{-2aT}z^{-2} + \cdots\right]$$

$$= e^{-amT} z^{-1} \left[\frac{1}{1 - e^{-aT} z^{-1}} \right]$$

$$= \frac{e^{-amT}}{z - e^{-aT}} \tag{3.30}$$

Substituting the z-transform results given by Eqns (3.29) and (3.30) in Eqn. (3.28), we get

$$\frac{Y(z)}{U(z)} = G_{h0} G_p(z) = \frac{1}{a} (1 - z^{-1}) z^{-N} \left[\frac{1}{z-1} - \frac{e^{-amT}}{z - e^{-aT}} \right]$$

$$= \frac{\frac{1}{a} \left[\left(1 - e^{-amT} \right) z + e^{-amT} - e^{-aT} \right]}{z^{N+1} (z - e^{-aT})} \tag{3.31}$$

Table 3.1 has been generated by applying the procedure outlined above, to commonly occurring functions.

Table 3.1 Table of transform pairs for systems with dead-time

Laplace transform $F(s)e^{-\Delta Ts}$; $0 \le \Delta < 1$	*z-transform* $\mathcal{Z}[F(s)e^{-\Delta Ts}]$; $m = 1 - \Delta$
$\frac{e^{-\Delta Ts}}{s}$	$\frac{1}{z-1}$
$\frac{e^{-\Delta Ts}}{s^2}$	$\frac{mT}{z-1} + \frac{T}{(z-1)^2}$
$\frac{2e^{-\Delta Ts}}{s^3}$	$T^2 \left[\frac{m^2 z^2 + (2m - 2m^2 + 1) z + (m-1)^2}{(z-1)^3} \right]$
$\frac{e^{-\Delta Ts}}{s+a}$	$\frac{e^{-amT}}{z - e^{-aT}}$
$\frac{e^{-\Delta Ts}}{(s+a)(s+b)}$	$\frac{1}{(b-a)} \left[\frac{e^{-amT}}{z - e^{-aT}} - \frac{e^{-bmT}}{z - e^{-bT}} \right]$
$\frac{ae^{-\Delta Ts}}{s(s+a)}$	$\frac{(1 - e^{-amT}) z + (e^{-amT} - e^{-aT})}{(z-1)(z - e^{-aT})}$
$\frac{ae^{-\Delta Ts}}{s^2 (s+a)}$	$\frac{T}{(z-1)^2} + \frac{amT - 1}{a(z-1)} + \frac{e^{-amT}}{a(z - e^{-aT})}$

Example 3.3

The scheme of Fig. 3.17 produces a steady-stream flow of fluid with controlled temperature θ. A stream of hot fluid is continuously mixed with a stream of cold fluid, in a mixing valve. The valve characteristic is such that, the total flow rate Q (m^3/sec) through it is maintained constant, but the inflow q_i (m^3/sec) of hot fluid may be linearly varied by controlling valve stem position x. The valve stem position x, thus controls the temperature θ_i (ºC) of the outflow from the mixing valve. Due to the distance between the valve and the point of discharge into the tank, there is a time delay between the change in θ_i and the discharge of the flow with the changed temperature, into the tank.

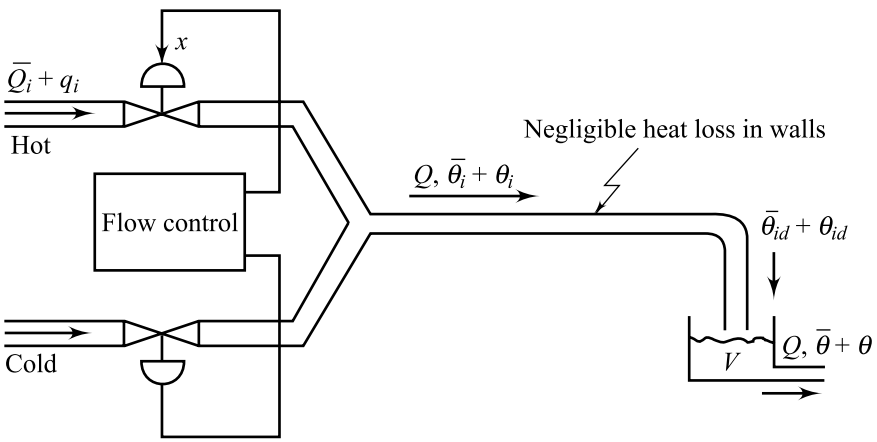


Fig. 3.17 Tank fluid temperature control

The differential equation governing the tank temperature is (assuming an initial equilibrium and taking all variables as perturbations)

$$V\rho c \frac{d\theta}{dt} = Q\rho c\,(\theta_{id} - \theta) \tag{3.32}$$

where

θ = tank fluid temperature, ºC
= temperature of the outflowing fluid from the tank;
c = specific heat of the fluid, Joules/(kg)(ºC);
V = volume of the fluid in the tank, m^3;
ρ = fluid density, kg/m^3;
Q = fluid flow rate, m^3/sec; and
θ_{id} = temperature of the fluid entering the tank, ºC.

The temperature θ_{id} at the input to the tank at time t, however, is the mixing valve output temperature τ_D seconds in the past, which may be expressed as

$$\theta_{id}(t) = \theta_i(t - \tau_D) \tag{3.33}$$

From Eqns (3.32)–(3.33), we obtain

$$\dot{\theta}(t) + a\theta(t) = a\theta_i(t - \tau_D)$$

where $a = Q/V$

Therefore,
$$G_p(s) = \frac{\theta(s)}{\theta_i(s)} = \frac{ae^{-\tau_D s}}{s+a} \tag{3.34}$$

To form the discrete-time transfer function of $G_p(s)$ preceded by a zero-order hold, we must compute

$$G_{h0}G_p(z) = \mathscr{Z}\left[\left(\frac{1-e^{-sT}}{s}\right)\frac{ae^{-\tau_D s}}{s+a}\right]$$
$$= (1-z^{-1})\,\mathscr{Z}\left[\frac{a}{s(s+a)}e^{-\tau_D s}\right] \tag{3.35}$$

For the specific values of $\tau_D = 1.5$, $T = 1$, $a = 1$, Eqn. (3.35) reduces to

$$G_{h0}G_p(z) = (1-z^{-1})\,\mathscr{Z}\left[\frac{1}{s(s+1)}e^{-s}e^{-0.5s}\right]$$
$$= (1-z^{-1})\,z^{-1}\,\mathscr{Z}\left[\frac{1}{s(s+1)}e^{-0.5s}\right]$$

Using transform pairs of Table 3.1, we obtain

$$G_{h0}G_p(z) = (1-z^{-1})\,z^{-1}\left[\frac{(1-e^{-0.5})z + (e^{-0.5}-e^{-1})}{(z-1)(z-e^{-1})}\right]$$
$$= \frac{0.3935(z+0.6066)}{z^2(z-0.3679)} = \frac{\theta(z)}{\theta_i(z)} \tag{3.36}$$

The relationship between x and θ_i is linear, as is seen below.

$$(\bar{Q}_i + q_i)\,\rho c\,\theta_H + [Q - (\bar{Q}_i + q_i)]\,\rho c\,\theta_C = Q\rho c\,(\bar{\theta}_i + \theta_i)$$

where θ_H and θ_C are constant temperatures of hot and cold streams, respectively.

$$q_i = K_v\,x$$

where K_v is the valve gain.

The perturbation equation is obtained as (neglecting second-order terms in perturbation variables),

$$K_v(\theta_H - \theta_C)\,x(t) = Q\theta_i(t)$$

or
$$x(t) = K\theta_i(t);\ K = Q/[K_v(\theta_H - \theta_C)]$$

Therefore,

$$\frac{\theta(z)}{X(z)} = \frac{(0.3935/K)(z+0.6066)}{z^2(z-0.3679)}$$

3.4 IMPLEMENTATION OF DIGITAL CONTROLLERS

The application of conventional 8- and 16-bit microprocessors to control systems, is now well established. Such processors have general-purpose architectures which make them applicable for a wide range of tasks, though none are remarkably efficient. In control applications, such devices may pose problems such as inadequate speed, difficulties with numerical manipulation, and relatively high cost for the complete system; the latter being due to both the programming effort and the cost of the peripheral hardware (memories, I/O ports, timers/counters, A/D converters, D/A converters, PWM circuit, etc.).

In applications requiring small amounts of program ROM, data RAM, and I/O ports, single-chip microcontrollers are ideally suited. In these chips, the capabilities in terms of speed of computation, on-chip resources, and software facilities are optimized for control applications. Should the on-chip features be insufficient to meet control requirements, the microcontroller chips allow for easy expansion.

The Intel microcontroller family (MCS-48 group, MCS-51 group, MCS-96 group) includes 8- and 16-bit processors with the following on-chip resources—ROM, RAM, I/O lines, timer/counter, A/D converter, and PWM output. The Motorola microcontroller family (HC 05 group, HC 11 group, HC 16 group) also provides microcontroller chips with similar features.

In many application areas, processing requirements for digital control systems, such as execution time and algorithm complexity, have increased dramatically. For example, in motor control, short sampling time constraints can place exacting requirements on algorithm execution time. New airframe designs and extended aircraft performance envelopes, increase the complexity of flight control laws. Controller complexity also increases with number of interacting loops (e.g., in robotics), or the number of sensors (e.g., in vision systems). For a growing number of real-time control applications, conventional single-processor systems are unable to satisfy the new demands for increased speed and greater complexity and flexibility.

The dramatic advances in VLSI technology leading to high transistor packing densities have enabled computer architects to develop parallel-processing architectures consisting of multiple processors; thus realizing high-performance computing engines at relatively low cost. The control engineer can exploit a range of architectures for a variety of functions.

Parallel-processing speeds up the execution time for a task. This is achieved by dividing the problem into several subtasks, and allocating multiple processors to execute multiple subtasks simultaneously. Parallel architectures differ from one another in respect of nature of interconnectivity between the processing elements and the processing power of each individual processing element.

The transputer is a family of single-chip computers, which incorporates features to support parallel processing. It is possible to use a network of transputers to reduce the execution time of a real-time control law.

Digital signal processors (DSPs) offer an alternative strategy for implementation of digital controllers. They use architectures and dedicated arithmetic circuits, that provide high resolution and high speed arithmetic, making them ideally suited for use as controllers.

Many DSP chips, available commercially, can be applied to a wide range of control problems. The Texas Instruments TMS 320 family provides several beneficial features through its architecture, speed, and instruction set.

TMS 320 is designed to support both numeric-intensive operations, such as required in signal processing, and also general-purpose computation, as would be required in high speed control. It uses a modified architecture, which gives it speed and flexibility—the program and data memory are allotted separate sections on the chip, permitting a full overlap of the instruction fetch and execution cycle. The processor also uses hardware to implement functions which had previously been achieved using software. As a result, a multiplication takes only 200 nsec, i.e., one instruction cycle, to execute. Extra hardware has also been included to implement shifting and some other functions. This gives the design engineer the type of power previously unavailable on a single chip.

Implementation of a control algorithm on a computer consists of the following two steps:

(i) Block diagram realization of the transfer function (obtained by the discretization of analog controller (Section 2.14), or by the direct digital design (Chapter 4) that represents the control algorithm.

(ii) Software design based on the block diagram realization.

In the following, we present several different structures of block diagram realizations of digital controllers using delay elements, adders, and multipliers. Different realizations are equivalent from the input-output point of view if we assume that the calculations are done with infinite precision. With finite precision in the calculations, the choice of the realization is very important. A bad choice of the realization may give a controller that is very sensitive to *errors* in the computations.

Assume that we want to realize the controller

$$D(z) = \frac{U(z)}{E(z)} = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_{n-1} z + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_{n-1} z + \alpha_n} \tag{3.37a}$$

where the α_i's and β_i's are real coefficients (some of them may be zero).

Transfer functions of all digital controllers can be rearranged in this form. For example, the transfer function of PID controller, given by Eqn. (2.113b), can be rearranged as follows:

$$\begin{aligned} D(z) = \frac{U(z)}{E(z)} &= K_c\left[1 + \frac{T}{T_I}\left(\frac{1}{1-z^{-1}}\right) + \frac{T_D}{T}(1-z^{-1})\right] \\ &= K_c + \frac{K_c T}{T_I}\left(\frac{1}{1-z^{-1}}\right) + \frac{K_c T_D}{T}(1-z^{-1}) \\ &= \frac{\beta_0 z^2 + \beta_1 z + \beta_2}{z^2 + \alpha_1 z + \alpha_2} \end{aligned}$$

where

$$\alpha_1 = -1;\ \alpha_2 = 0$$

$$\beta_0 = K_c\left(1 + \frac{T}{T_I} + \frac{T_D}{T}\right);\ \beta_1 = -K_c\left(1 + \frac{2T_D}{T}\right);\ \beta_2 = \frac{K_c T_D}{T}$$

We shall now discuss different ways of realizing the transfer function (3.37a), or equivalently the transfer function:

$$D(z) = \frac{U(z)}{E(z)} = \frac{\beta_0 + \beta_1 z^{-1} + \beta_2 z^{-2} + \cdots + \beta_{n-1} z^{-(n-1)} + \beta_n z^{-n}}{1 + \alpha_1 z^{-1} + \alpha_2 z^{-2} + \cdots + \alpha_{n-1} z^{-(n-1)} + \alpha_n z^{-n}} \tag{3.37b}$$

The methods for realizing digital systems of the form (3.37) can be divided into two classes—*recursive* and *nonrecursive*. The functional relation between the input sequence $e(k)$ and the output sequence $u(k)$ for a *recursive realization* has the form

$$u(k) = f(u(k-1), u(k-2),..., e(k), e(k-1), ...) \tag{3.38}$$

For the linear time-invariant system of Eqn. (3.37b), the recursive realization has the form

$$u(k) = -\alpha_1 u(k-1) - \alpha_2 u(k-2) - \cdots - \alpha_n u(k-n) + \beta_0 e(k) + \beta_1 e(k-1) + \cdots + \beta_n e(k-n) \tag{3.39}$$

The current output sample $u(k)$ is a function of past outputs and present and past input samples. Due to the recursive nature, the errors in previous outputs may accumulate.

The impulse response of the digital system defined by Eqn. (3.39), where we assume not all α_i's are zero, has an infinite number of nonzero samples, although their magnitudes may become negligibly small as k increases. This type of digital system is called an *Infinite Impulse Response* (IIR) system.

The input-output relation for a *nonrecursive realization* is of the form

$$u(k) = f(e(k), e(k-1), ...) \tag{3.40a}$$

For a linear time-invariant system, this relation takes the form

$$u(k) = b_0 e(k) + b_1 e(k-1) + b_2 e(k-2) + \cdots + b_N e(k-N) \tag{3.40b}$$

The current output sample $u(k)$ is a function only of the present and past values of the input.

The impulse response of the digital system defined by Eqn. (3.40b), is limited to a finite number of samples defined over a finite range of time intervals, i.e., the impulse response sequence is finite. This type of digital system is called a *finite impulse response* (FIR) system.

The digital controller given by Eqn. (3.37b) is obviously an FIR digital system when the coefficients α_i are all zero. When not all α_i's are zero, we can obtain FIR approximation of the digital system by dividing its numerator by the denominator and truncating the series at z^{-N}; $N \geq n$:

$$\frac{U(z)}{E(z)} = D(z) \cong a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N};\ N \geq n \tag{3.41}$$

Notice that we may require a large value of N to obtain a good level of accuracy.

In the following sections, we discuss the most common types of recursive and nonrecursive realizations of digital controllers of the form (3.37).

3.4.1 Recursive Realizations

The transfer function (3.37) represents an nth-order system. Recursive realization of this transfer function will require at least *n unit delayers*. Each unit delayer will represent a first-order dynamic system. Each of the three recursive realization structures given below, uses the minimum number (n) of delay elements in realizing the transfer function (3.37).

Direct Realization

Let us multiply the numerator and denominator of the right-hand side of Eqn. (3.37b) by a variable $X(z)$. This operation gives

$$\frac{U(z)}{E(z)} = \frac{(\beta_0 + \beta_1 z^{-1} + \beta_2 z^{-2} + \cdots + \beta_{n-1} z^{-(n-1)} + \beta_n z^{-n})\, X(z)}{(1 + \alpha_1 z^{-1} + \alpha_2 z^{-2} + \cdots + \alpha_{n-1} z^{-(n-1)} + \alpha_n z^{-n})\, X(z)} \tag{3.42}$$

Equating the numerators on both sides of this equation gives

$$U(z) = (\beta_0 + \beta_1 z^{-1} + \cdots + \beta_n z^{-n})\, X(z) \tag{3.43a}$$

The same operation on the denominator brings

$$E(z) = (1 + \alpha_1 z^{-1} + \cdots + \alpha_n z^{-n})\, X(z) \tag{3.43b}$$

In order to construct a block diagram for realization, Eqn. (3.43b) must first be written in a *cause-and-effect* relation. Solving for $X(z)$ in Eqn. (3.43b) gives

$$X(z) = E(z) - \alpha_1 z^{-1} X(z) - \cdots - \alpha_n z^{-n} X(z) \tag{3.43c}$$

A block diagram portraying Eqns (3.43a) and (3.43c) is now drawn in Fig. 3.18 for $n = 3$. Notice that we use only three delay elements. The coefficients α_i and β_i (which are real quantities) appear as multipliers. The block diagram schemes where the coefficients α_i and β_i appear directly as multipliers are called *direct structures*.

Basically, there are three sources of error that affect the accuracy of a realization (Section 2.1):

(i) the error due to the quantization of the input signal into a finite number of discrete levels;

(ii) the error due to accumulation of round-off errors in the arithmetic operations in the digital system; and

(iii) the error due to quantization of the coefficients α_i and β_i of the transfer function. This error may become large for higher-order transfer functions. That is, in a higher-order digital controller in direct structure, small errors in the coefficients α_i and β_i cause large errors in the locations of the poles and zeros of the controller (refer to Review Example 3.3).

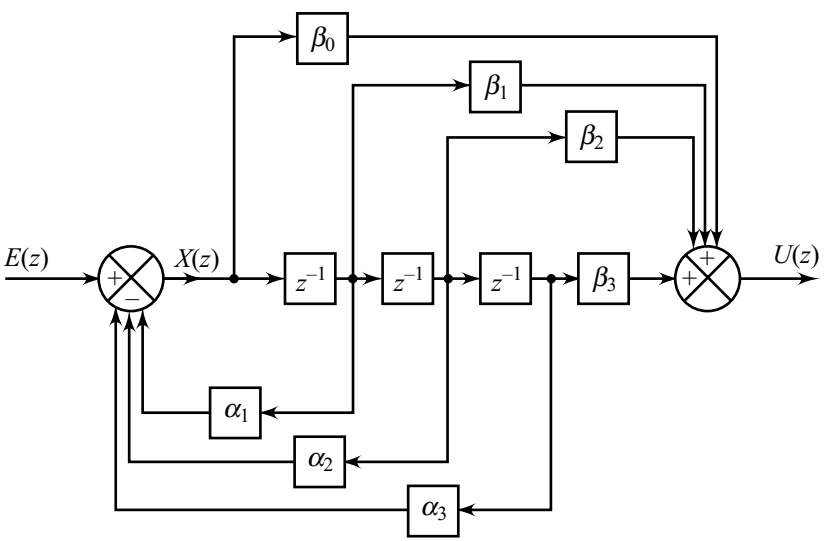


Fig. 3.18 Block diagram realization of the transfer function of Eqn. (3.37) with $n = 3$

These three errors arise because of the practical limitations of the number of bits that represent various signal samples and coefficients. The third type of error listed above may be reduced by mathematically decomposing a higher-order transfer function into a combination of lower-order transfer functions. In this way, the system may be made less sensitive to coefficient inaccuracies.

For decomposing higher-order transfer functions in order to reduce the coefficient sensitivity problem, the following two approaches are commonly used. It is desirable to analyze each of these structures for a given transfer function, to see which one is better with respect to the number of arithmetic operations required, the range of coefficients, and so forth.

Cascade Realization

The sensitivity problem may be reduced by implementing the transfer function $D(z)$ as a cascade connection of first-order and/or second-order transfer functions. If $D(z)$ can be written as a product of transfer functions $D_1(z)$,... , $D_m(z)$, i.e.,

$$D(z) = D_1(z)\, D_2(z) \cdots D_m(z),$$

then a digital realization for $D(z)$ may be obtained by a cascade connection of m component realizations for $D_1(z)$, $D_2(z)$, ... , and $D_m(z)$, as shown in Fig. 3.19.

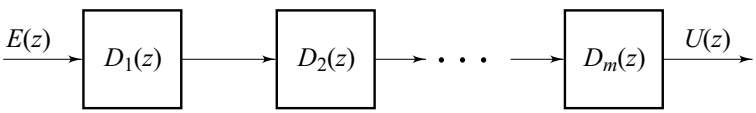


Fig. 3.19 Cascade decomposition of $D(z)$

In most cases, the $D_i(z)$; $i = 1, 2, ..., m$, are chosen to be either first-order or second-order functions. If the poles and zeros of $D(z)$ are known, then $D_i(z)$ can be obtained by grouping real poles and real zeros to produce first-order functions, or by grouping a pair of complex-conjugate poles and a pair of complex-conjugate zeros to produce a second-order function. It is, of course, possible to group two real poles with a pair of complex-conjugate zeros and vice versa. The grouping is, in a sense, arbitrary. It is desirable to group several different ways, to see which one is best with respect to the number of arithmetic operations required, the range of coefficients, and so forth.

In general, $D(z)$ may be decomposed as follows:

$$D(z) = \prod_{i=1}^{p} \frac{1+b_i z^{-1}}{1+a_i z^{-1}} \prod_{j=p+1}^{m} \frac{1+e_j z^{-1}+f_j z^{-2}}{1+c_j z^{-1}+d_j z^{-2}}$$

The block diagram for

$$D_i(z) = \frac{1+b_i z^{-1}}{1+a_i z^{-1}} = \frac{U_i(z)}{E_i(z)}$$

and that for

$$D_j(z) = \frac{1+e_j z^{-1}+f_j z^{-2}}{1+c_j z^{-1}+d_j z^{-2}} = \frac{U_j(z)}{E_j(z)}$$

are shown in Figs 3.20a and 3.20b, respectively. The realization for the digital controller $D(z)$ is a cascade connection of p first-order systems of the type shown in Fig. 3.20a, and $(m - p)$ second-order systems of the type shown in Fig. 3.20b.

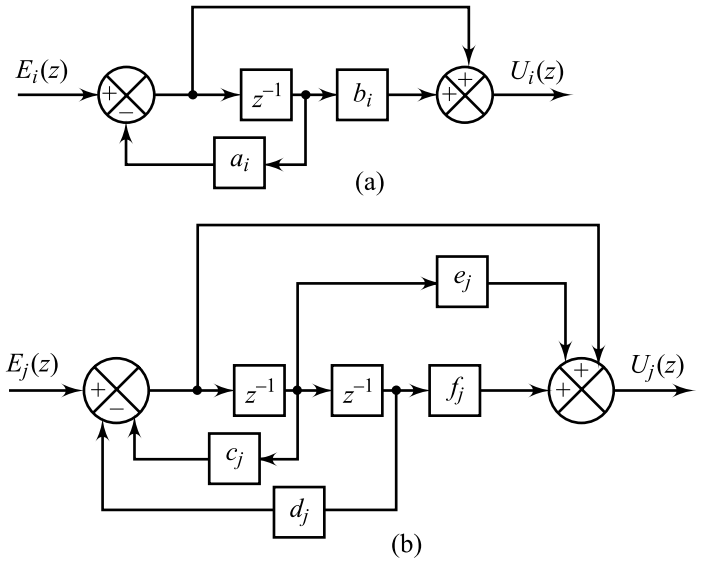


Fig. 3.20 Realizations of (a) first-order and (b) second-order functions

Parallel Realization

Another approach to reduce the coefficient sensitivity problem is to expand the transfer function $D(z)$ into partial fractions. If $D(z)$ is expanded so that

$$D(z) = A + D_1(z) + D_2(z) + \cdots + D_r(z),$$

where A is simply a constant, then a digital realization for $D(z)$ may be obtained by a parallel connection of $(r + 1)$ component realizations for A, $D_1(z)$, ... , $D_r(z)$, as shown in Fig. 3.21. Due to the presence of the constant term A, the first-order and second-order functions can be chosen in simpler forms:

$$D(z) = A + \sum_{i=1}^{q} \frac{b_i}{1 + a_i z^{-1}} + \sum_{j=q+1}^{r} \frac{e_j + f_j z^{-1}}{1 + c_j z^{-1} + d_j z^{-2}}$$

The block diagram for

$$D_j(z) = \frac{b_i}{1 + a_i z^{-1}} = \frac{U_i(z)}{E(z)}$$

and that for

$$D_j(z) = \frac{e_j + f_j z^{-1}}{1 + c_j z^{-1} + d_j z^{-2}} = \frac{U_j(z)}{E(z)}$$

are shown in Figs 3.22a and 3.22b, respectively.

3.4.2 Nonrecursive Realizations

Nonrecursive structures for $D(z)$ are similar to the recursive structures presented earlier in this section. In the nonrecursive form, the direct and cascade structures are commonly used; the parallel structure is not used since it requires more elements.

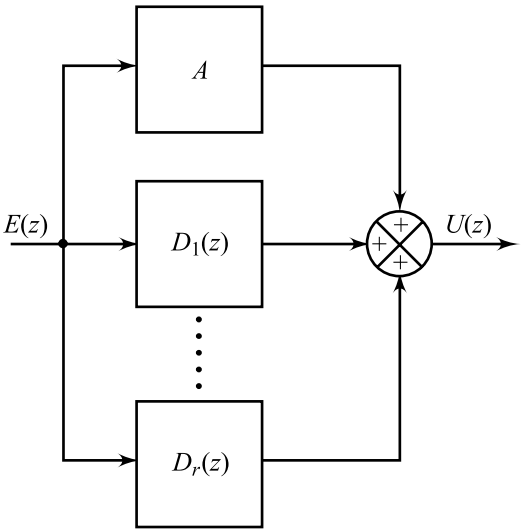


Fig. 3.21 Parallel decomposition of $D(z)$

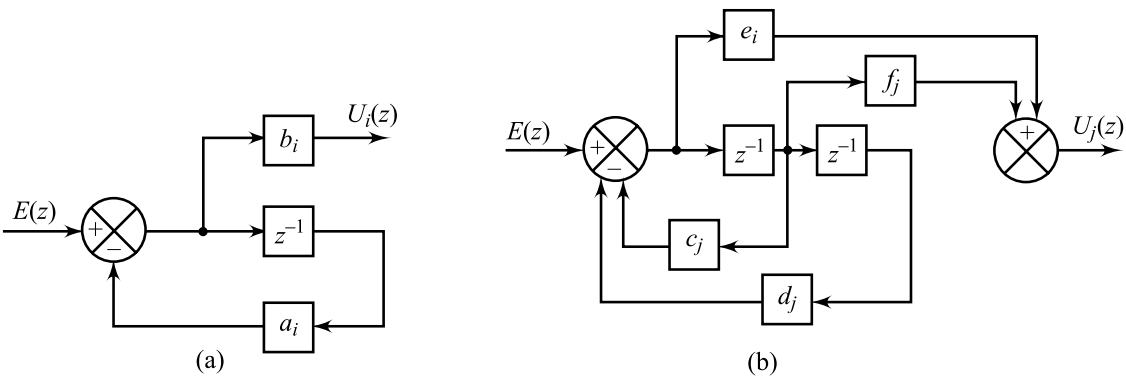


Fig. 3.22 Realization of (a) first-order and (b) second-order functions

Example 3.4

Consider the digital controller with transfer function model

$$D(z) = \frac{U(z)}{E(z)} = \frac{2 - 0.6z^{-1}}{1 + 0.5z^{-1}}$$

Recursive realization of $D(z)$ yields the block diagram shown in Fig. 3.23a. By dividing the numerator of $D(z)$ by the denominator, we obtain

$$D(z) = 2 - 1.6z^{-1} + 0.8z^{-2} - 0.4z^{-3} + 0.2z^{-4} - 0.1z^{-5} + 0.05z^{-6} - 0.025z^{-7} + \cdots$$

Truncating this series at z^{-5}, we obtain the following FIR digital system:

$$\frac{U(z)}{E(z)} = 2 - 1.6\,z^{-1} + 0.8\,z^{-2} - 0.4\,z^{-3} + 0.2\,z^{-4} - 0.1\,z^{-5}$$

Figure 3.23b gives a realization for this FIR system. Notice that we need a large number of delay elements to obtain a good level of accuracy. An advantage of this realization is that, because of the lack of feedback, the accumulation of errors in past outputs is avoided in the processing of the signal.

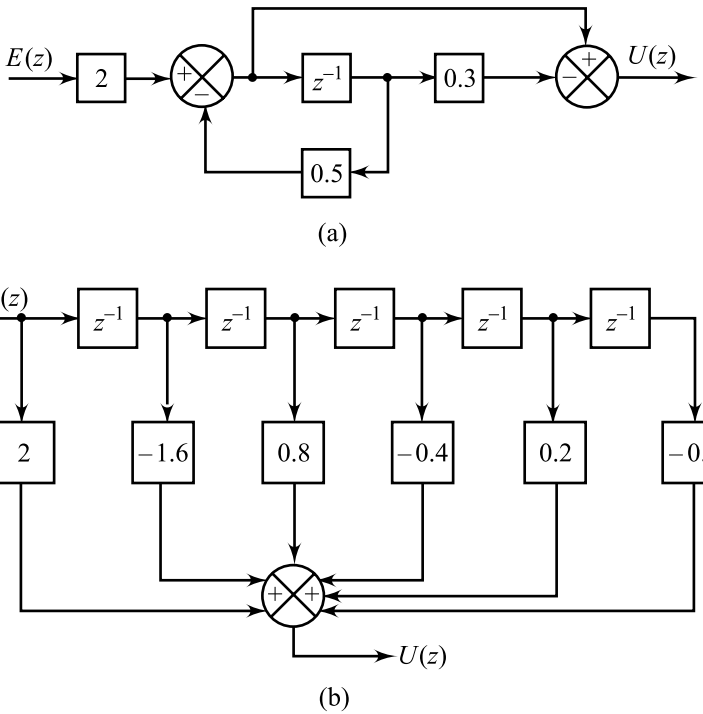


Fig. 3.23 (a) Recursive realization (b) Nonrecursive realization of a first-order function

3.5 TUNABLE PID CONTROLLERS

The ultimate goal of control systems engineering is to build real physical systems to perform some specified tasks. To accomplish this goal, design and physical implementation of a control strategy are required. The standard approach to design this is as follows. A mathematical model is built making necessary assumptions about various uncertain quantities on the dynamics of the system. If the objective is well defined in precise mathematical terms, then control strategies can be derived mathematically (e.g., by optimizing some criterion of performance). This is the basis of all *model-based control* strategies. This approach is feasible when it is possible to specify the objective and the model, mathematically. Many sophisticated methods based on model-based control approach will appear in later chapters of the book.

For motion control applications (position, and speed control systems), identification of mathematical models of systems close enough to reality is usually possible. However, for process control applications (pressure, flow, liquid-level, temperature, and composition control systems), identification of process dynamics precisely can be expensive even if meaningful identification is possible. This is because industrial processes are relatively slow and complex. In process-control field, therefore, it is not uncommon to follow an ad hoc approach for controller development, when high demands on control-system performance are not made. In the *ad hoc approach*, we select a certain type of controller based on past experience with the process to be controlled, and then set controller parameters by experiment once the controller is installed. The 'experimental design' of controller settings has come to be known as *controller tuning*.

Many years of experience have shown that a PID controller is versatile enough to control a wide variety of industrial processes. The common practice is to interface a PID controller (with adjustment features) to the process and adjust the parameters of the controller online, by trial-and-error, to obtain acceptable performance. A number of *tuning methods* have been introduced to obtain fast convergence to control solution. These methods consist of the following two steps:

(i) experimental determination of the dynamic characteristics of the control loop; and
(ii) estimation of the controller tuning parameters that produce a desired response for the dynamic characteristics determined in the first step.

It may be noted that for tuning purposes, simple experiments are performed to estimate important dynamic attributes of the process. The approximate models have proven to be quite useful for process control applications (For processes whose dynamics are precisely known, the use of trial-and-error tuning is not justified since many model-based methods to the design of PID controllers are available which predict the controller parameters fairly well at the design stage itself). The predicted parameter values based on approximate models simply provide initial trial values for the online trial-and-error approach. These trial values may turn out to be a poor guess. *Fine tuning* the controller parameters online is usually necessary to obtain acceptable control performance.

Some of the tuning methods which have been successfully used in process industry, will be described here.

3.5.1 Analog PID Controllers

Approximately 75% of feedback controllers in the process industry are PI controllers; most of the balance are PID controllers. Some applications require only P, or PD controllers, but these are few.

Proportional Controller

The equation that describes the proportional controller is

$$u(t) = K_c e(t) \tag{3.44a}$$

or

$$U(s) = K_c E(s) \tag{3.44b}$$

where K_c is the *controller gain*, e is the error, and u is the perturbation in controller output signal from the base value corresponding to the normal operating conditions; the base value on the controller is adjusted to produce zero error under the conditions of no disturbance and/or set-point change.

Some instrument manufacturers calibrate the controller gain as *proportional band* (*PB*). A 10% *PB* means that a 10% change in the controller input causes a full-scale (100%) change in controller output. The conversion relation is thus

$$K_c = \frac{100}{PB} \tag{3.45}$$

A proportional controller has only one adjustable or tuning parameter: K_c or *PB*.

A proportionally controlled process with no integration property will always exhibit error at steady state in the presence of disturbances and changes in set-point. The error, of course, can be made negligibly small by increasing the gain of the proportional controller. However, as the gain is increased, the performance of the closed-loop system becomes more oscillatory and takes longer to settle down after being disturbed. Further, most process plants have a considerable amount of dead-time, which severely restricts the value of the gain that can be used. In processes where the control within a band from the set-point is acceptable, proportional control is sufficient. However, in processes which require perfect control at the set-point, proportional controllers will not provide satisfactory performance [155].

Proportional-Integral Controller

To remove the steady-state offset in the controlled variable of a process, an extra amount of intelligence must be added to the proportional controller. This extra intelligence is the integral or reset action, and consequently, the controller becomes a PI controller. The equation describing a PI controller is as follows:

$$u(t) = K_c \left[e(t) + \frac{1}{T_I} \int_0^t e(t)\, dt \right] \tag{3.46a}$$

or

$$U(s) = K_c \left[1 + \frac{1}{T_I s} \right] E(s) \tag{3.46b}$$

where T_I is the *integral* or *reset time*.

A PI controller has thus two adjustable or tuning parameters: K_c (or *PB*) and T_I. The integral or reset action in this controller removes the steady-state offset in the controlled variable. However, the integral mode of control has a considerable destabilizing effect which, in most of the situations, can be compensated by adjusting the gain K_c [155].

Some instrument manufacturers calibrate the integral mode parameter as the *reset rate*, which is simply the reciprocal of the reset time.

Proportional-Integral-Derivative Controller

Sometimes a mode faster than the proportional mode is added to the PI controller. This new mode of control is the derivative action, also called the rate action, which responds to the rate of change of error with time. This speeds up the controller action. The equation describing the PID controller is as follows:

$$u(t) = K_c \left[e(t) + \frac{1}{T_I} \int_0^t e(t) dt + T_D \frac{de(t)}{dt} \right] \tag{3.47a}$$

or
$$U(s) = K_c\left[1 + \frac{1}{T_I s} + T_D s\right]E(s) \tag{3.47b}$$

where T_D is the *derivative* or *rate time*.

A PID controller has thus three adjustable or tuning parameters: K_c (or PB), T_I, and T_D. The derivative action anticipates the error, initiates an early corrective action, and tends to increase the stability of the system. It does not affect the steady-state error directly. A derivative control mode, in isolation, produces no corrective effort for any constant error, no matter how large, and would, therefore, allow uncontrolled steady-state errors. Thus, we cannot consider derivative modes in isolation; they will always be considered as augmenting some other mode [155].

The block diagram implementation of Eqn. (3.47b) is sketched in Fig. 3.24a. The alternative form, Fig. 3.24b, is more commonly used, because it avoids taking the rate of change of the set-point input

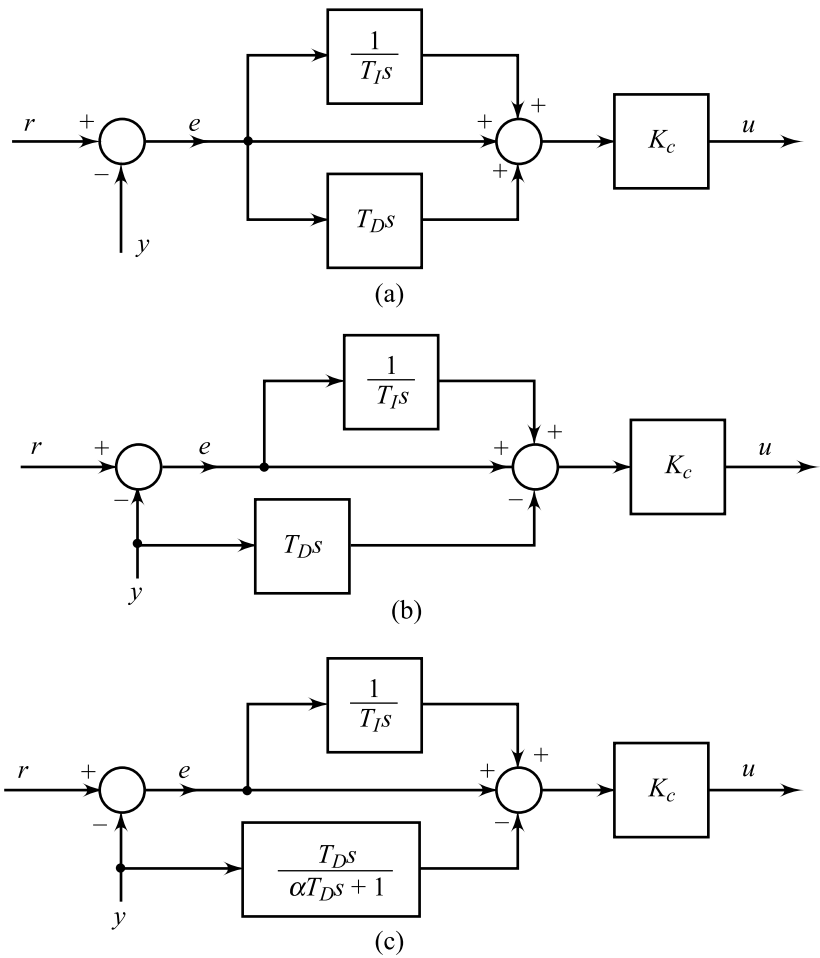


Fig. 3.24 Block diagram of PID controller

to the controller, thus preventing the undesirable derivative 'kick' on set-point changes by the process operator.

Due to the noise-accentuating characteristics of derivative operation, the low-pass-filtered derivative $T_D s/(\alpha T_D s + 1)$ is actually preferred in practice (Fig. 3.24c). The value of the filter parameter α is not adjustable but is built into the design of the controller. It is usually of the order of 0.05 to 0.3.

The controller of Fig. 3.24 is considered to be *non-interacting* in that its derivative and integral modes operate independently of each other (although proportional gain affects all the three modes). Non-interaction is provided by the parallel functioning of integral and derivative modes. By contrast, many controllers have derivative and integral action applied serially to the controlled variable, resulting in interaction between them. Many of the analog industrial controllers realize the following *interacting* PID control action.

$$U(s) = K_c' \left[\frac{T_D' s + 1}{\alpha T_D' s + 1} \right] \left[1 + \frac{1}{T_I' s} \right] E(s) \tag{3.48}$$

The first term in brackets is a derivative unit attached to the standard PI controller serially, to create the PID controller (Fig. 3.25a). The derivative unit installed on the controlled-variable input to the controller avoids the derivative kick (Fig. 3.25b).

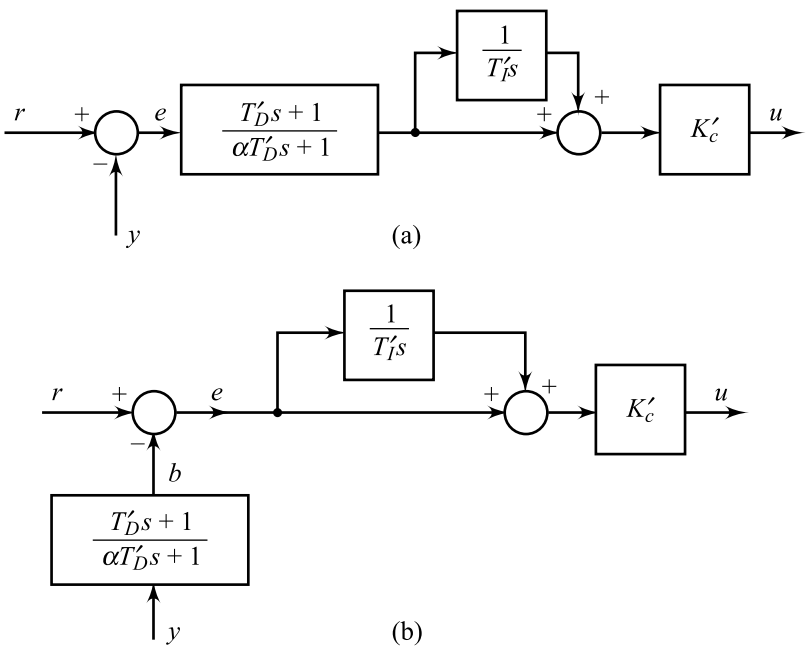


Fig. 3.25 Alternative realization schemes for PID control action

Most commercially available tunable controllers use the non-interacting version of PID control. The discussion that follows applies to non-interacting PID control.

3.5.2 Adjustment Features in Industrial Controllers

A great number of manufacturers are now making available in the market, process controllers (electronic, and computer-based) with features that permit adjusting the set-point, transferring between manual and automatic control modes, adjusting the output signal from the control-action unit (tuning the parameters K_c, T_I, and T_D), and displaying the controlled variable, set-point, and control signal. Figure 3.26 shows the basic structure of an industrial controller. The controller has been broken down into the following three main units:

(i) the set-point control unit;
(ii) the PID control unit; and
(iii) the manual/automatic control unit.

The set-point control unit receives the measurement y of controlled variable of the process, together with the set-point r of the control. A switch gives an option of choosing between local and remote (external) set-point operation. If the set-point to the controller is to be set by the operating personnel, then the local option r_L is chosen. If the set-point to the controller is to be set by another control module, then remote (external) option r_e is chosen. This is the case, for example, in cascade control where the drive of the controller in the major loop constitutes the set-point of the minor-loop controller.

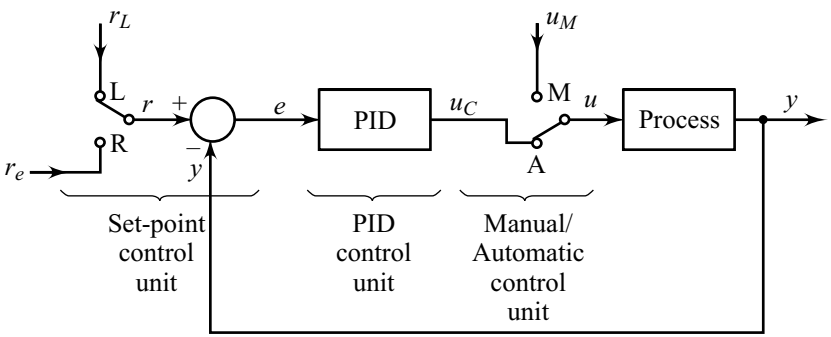


Fig. 3.26 Basic structure of an industrial controller

The PID control unit receives the error signal e developed by the set-point control unit, and generates an appropriate control signal u_C. Adjustment features provided in the control unit, for generating appropriate control signals, include tuning of the three parameters K_c, T_I, and T_D.

The manual/automatic control unit has a switch which determines the mode of control action. When the switch is in the auto (A) position, the control signal u_C calculated by PID control unit is sent to the process (in such a case, the process is controlled in closed loop). When the switch is in the manual (M) position, the PID control unit 'freezes' its output. The control signal u_M can then be changed manually by the operating personnel (the process is then controlled in open loop).

The basic structure of a process controller shown in Fig. 3.26 is common for electronic, and computer-based controllers. These controllers are different in terms of realization of adjustment features.

3.5.3 Ziegler–Nichols Tuning Method Based on Ultimate Gain and Period

This pioneer method, also known as the closed-loop or on-line tuning method, was proposed by J G Ziegler and N B Nichols around 1940. In this method, the parameters by which the dynamic characteristics of the process are represented are the ultimate gain and period. These parameters are used in tuning the controller for a specified response: the quarter-decay ratio (QDR) response.

Determination of Ultimate Gain and Period

When the process is under closed-loop proportional (P) control, the gain of the P controller at which the loop oscillates with constant amplitude, has been defined as the *ultimate gain* K_{cu}. *Ultimate period* T_u is the period of these sustained oscillations. The ultimate gain is, thus, a measure of difficulty in controlling a process; the higher the ultimate gain, the easier it is to control the process loop. The ultimate period is, in turn, a measure of speed of response of the loop; the larger the period, the slower the loop.

By its definition, it can be deduced that the ultimate gain is the gain at which the loop is at the threshold of instability. At gains just below the ultimate, the loop signals will oscillate with decreasing amplitude, and at gains above the ultimate, the amplitude of the oscillations will grow with time.

For experimental determination of K_{cu} and T_u, the controller is set in 'auto' mode and the following procedure is followed (refer to Fig. 3.26).

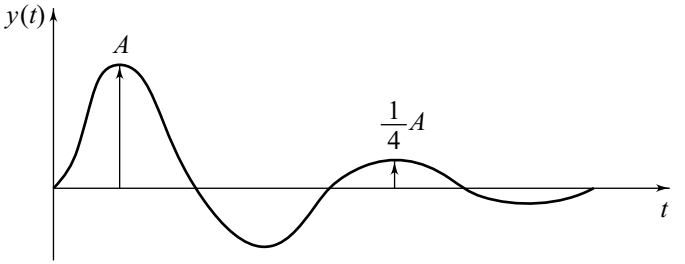
(i) Remove the integral mode by setting the integral time to its highest value. Alternatively, if the PID controller allows for switching off the integral mode, switch it off.
(ii) Switch off the derivative mode, or set the derivative time to its lowest value, usually zero.
(iii) Increase the proportional gain in steps. After each increase, disturb the loop by introducing a small step change in set-point and observe the response of the controlled variable, preferably on a trend recorder. The controlled variable should start oscillating as the gain is increased. When the amplitude of the oscillations remains approximately constant, the ultimate controller gain has been reached. Record it as K_{cu}.
(iv) Measure the period of the oscillations from the trend recording. This parameter is T_u.

The procedure just outlined is simple and requires a minimum upset to the process, just enough to be able to observe the oscillations. Nevertheless, the prospect of taking a process control loop to the verge of instability is not an attractive one from a process operation standpoint.

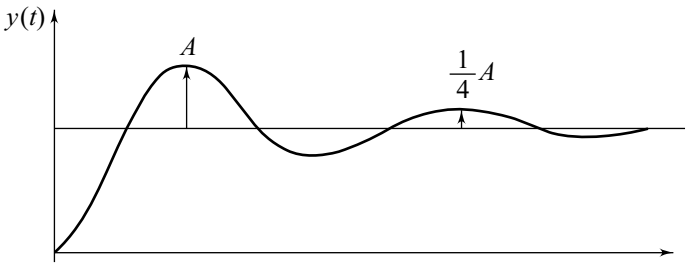
Tuning for Quarter-Decay Ratio Response

Ziegler and Nichols proposed that the parameters K_{cu} and T_u, characterizing a process, be used in tuning the controller for QDR response. The QDR response is illustrated in Fig. 3.27 for a step change in disturbance, and for a step change in set-point. Its characteristic is that each oscillation has an amplitude that is one fourth of the previous oscillation.

Empirical relations [12] for calculating the QDR tuning parameters of P, PI and PID controllers, from the ultimate gain K_{cu} and period T_u, are given in Table 3.2.



(a) Disturbance input



(b) Set-point change

Fig. 3.27 QDR response

PI and PID tuning parameters that produce quarter-decay response, are not unique. For each setting of the integral and derivative times, there will usually be a setting of the controller gain that produces quarter-decay response. The settings given in Table 3.2 are the figures based on experience; these settings have produced fast response for most industrial loops.

Table 3.2 QDR tuning formulas based on ultimate gain and period

Controller	*Gain*	*Integral time*	*Derivative time*
P	$K_c = 0.5\ K_{cu}$	—	—
PI	$K_c = 0.45\ K_{cu}$	$T_I = T_u/1.2$	—
PID	$K_c = 0.75\ K_{cu}$	$T_I = T_u/1.6$	$T_D = T_u/10$

3.5.4 Ziegler–Nichols Tuning Method Based on Process Reaction Curve

Although the tuning method based on ultimate gain and period is simple and fast, other methods of characterizing the dynamic response of feedback control loops have been developed over the years. The need for these alternative methods is based on the fact that, it is not always possible to determine the ultimate gain and period of a loop; some loops would not exhibit sustained oscillations with a proportional

controller. Also, the ultimate gain and period do not give insight into which process or control system characteristics could be modified to improve the feedback controller performance. A more fundamental method of characterizing process dynamics is needed to guide such modifications. In the following, we present an open-loop method for characterizing the dynamic response of the process in the loop.

Process Reaction Curve

Process control is characterized by systems which are relatively slow and complex and which, in many cases, include an element of pure time delay (dead-time). Even where a dead-time element is not present, the complexity of the system which will typically contain several first-order subsystems, will often result in a *process reaction curve* (dynamic response to a step change in input), which has the appearance of pure time delay.

Process reaction curve may be obtained by carrying out the following step-test procedure.

With the controller on 'manual', i.e., the loop opened (refer to Fig. 3.26), a step change of magnitude Δu in the control signal $u(t)$ is applied to the process. The magnitude Δu should be large enough for the consequent change $\Delta y(t)$ in the process output variable to be measurable, but not so large that the response will be distorted by process nonlinearities. The process output is recorded for a period from the introduction of the step change in the input, until the process reaches a new steady state.

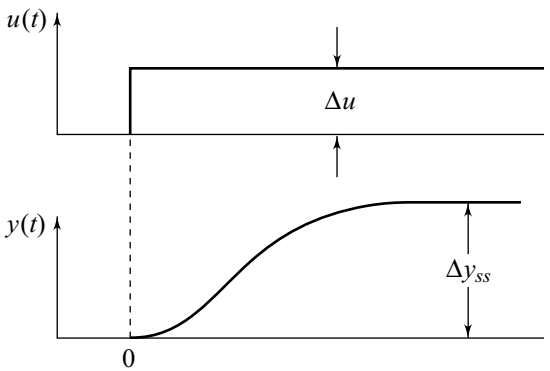


Fig. 3.28 A process reaction curve

A typical process reaction curve is sketched in Fig. 3.28. The most common model used to characterize the process reaction curve is the following:

$$\frac{Y(s)}{U(s)} = G(s) = \frac{Ke^{-\tau_D s}}{\tau s + 1} \tag{3.49}$$

where K = the process steady-state gain;
τ_D = the effective process dead-time; and
τ = the effective process time-constant.

This is a *first-order plus dead-time model*. The model response for a step change in the input signal of magnitude Δu, is given by

$$Y(s) = \frac{Ke^{-\tau_D s}}{\tau s + 1} \frac{\Delta u}{s} = K\Delta u\, e^{-\tau_D s}\left[\frac{1}{s} - \frac{\tau}{\tau s + 1}\right] \tag{3.50}$$

Inverting with the help of a transform table (Table 2.1), and applying the real translation theorem of Laplace transforms $\mathscr{L}[y(t-t_0)\mu(t-t_0)] = e^{-st_0}\,Y(s)$; $t_0 > 0$ [155], we get

$$\begin{aligned} \Delta y(t) &= K\Delta u\,[1 - e^{-(t-\tau_D)/\tau}]\,; \; t > \tau_D \\ &= 0 \qquad\qquad\qquad\quad ; \; t \le \tau_D \end{aligned} \tag{3.51}$$

The term Δy is the perturbation or change in the output from its initial value:

$$\Delta y(t) = y(t) - y(0)$$

Figure 3.29 shows the model response to a step change of magnitude Δu in the input signal. Δy_{ss} is the steady-state change in the process output (refer to Eqn. (3.51)):

$$\Delta y_{ss} = \lim_{t \to \infty} \Delta y(t) = K\Delta u$$

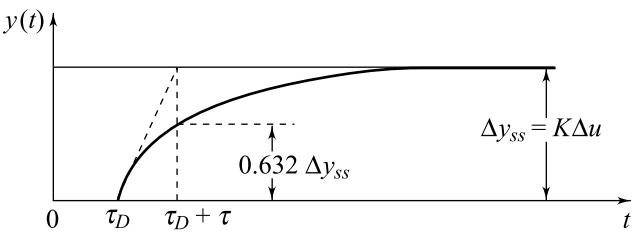


Fig. 3.29 Step-response of first-order plus dead-time model

At the point $t = \tau_D$ on the time axis, the process output variable leaves the initial steady state with a maximum rate of change (refer to Eqn. (3.51)):

$$\left.\frac{d}{dt}\Delta y(t)\right|_{t=\tau_D} = K\Delta u\left(\frac{1}{\tau}\right) = \frac{\Delta y_{ss}}{\tau}$$

The time-constant τ is the distance on the time axis between the point $t = \tau_D$, and the point at which the tangent to the model response curve, drawn at $t = \tau_D$, crosses the new steady state.

Note that the model response at $t = \tau_D + \tau$ is given by

$$\Delta y\,(\tau_D + \tau) = K\Delta u(1 - e^{-1}) = 0.632\,\Delta y_{ss}$$

The process reaction curve of Fig. 3.28 can be matched to the model response of Fig. 3.29 by the following estimation procedure.

The model parameter K is given by

$$K = \frac{\text{Change in process output at steady state}}{\text{Step change in process input}} = \frac{\Delta y_{ss}}{\Delta u} \tag{3.52}$$

The estimation of the model parameters τ_D and τ can be done by, at least, three methods; each of which results in different values.

Tangent Method

This method makes use of the line that is tangent to the process reaction curve at the point of maximum rate of change. The time-constant is then defined as the distance on the time axis, between the point where the

tangent crosses the *initial* steady state of the output variable, and the point where it crosses the *new* steady-state value. The *dead-time* is the distance on the time axis, between the occurrence of the input step change, and the point where the tangent line crosses the initial steady state. These estimates are indicated in Fig. 3.30a.

Tangent-and-Point Method

In this method, τ_D is determined in the same manner as in the earlier method, but the value of τ is the one that forces the model response to coincide with the actual response at $t = \tau_D + \tau$. Construction for this method is shown in Fig. 3.30b. The value of τ obtained by this method is usually less than that obtained by the earlier method, and the process reaction curve is usually closer to the response of the model obtained by this method compared to the one obtained by the earlier method.

Two-Points Method

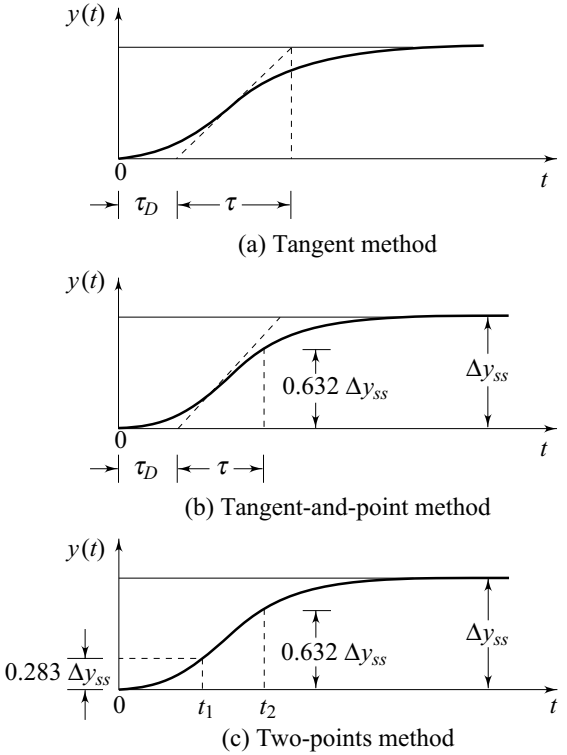


Fig. 3.30 Estimation of model parameters

The least precise step in the determination of τ_D and τ by the previous two methods, is the drawing of the line tangent to the process reaction curve at the point of maximum rate of change. To eliminate this dependence on the tangent line, it is proposed that the values of τ_D and τ be selected such that the model and the actual response coincide at two points in the region of high rate of change. The two points recommended are $(\tau_D + \frac{1}{3}\tau)$ and $(\tau_D + \tau)$. To locate these points, we make use of Eqn. (3.51):

$$\Delta y(\tau_D + \tfrac{1}{3}\tau) = K\Delta u[1 - e^{-1/3}] = 0.283\ \Delta y_{ss}$$

$$\Delta y(\tau_D + \tau) = K\Delta u\ [1 - e^{-1}] = 0.632\ \Delta y_{ss}$$

These two points are labeled t_1 and t_2 in Fig. 3.30c. Knowing t_1 and t_2, we can obtain the values of τ_D and τ.

$$\tau_D + \tau = t_2;\ \tau_D + \tfrac{1}{3}\tau = t_1$$

which reduces to

$$\tau = \tfrac{3}{2}\,(t_2 - t_1);\ \tau_D = t_2 - \tau \tag{3.53}$$

where t_1 = time at which $\Delta y(t) = 0.283\ \Delta y_{ss}$; and
t_2 = time at which $\Delta y(t) = 0.632\ \Delta y_{ss}$.

Tuning for QDR Response

Besides the formulas for QDR response tuning based on the ultimate gain and period of the loop (refer to Table 3.2), Ziegler and Nichols also developed tuning formulas based on the parameters of a first-order model fit to the process reaction curve. These formulas are given in Table 3.3 [12].

Table 3.3 QDR tuning formulas based on process reaction curve: Process model $G(s) = \dfrac{Ke^{-\tau_D s}}{\tau s + 1}$

Controller	*Gain*	*Integral time*	*Derivative time*
P	$K_c = \tau/K\tau_D$	—	—
PI	$K_c = 0.9\,\tau/K\tau_D$	$T_I = 3.33\ \tau_D$	—
PID	$K_c = 1.5\,\tau/K\tau_D$	$T_I = 2.5\,\tau_D$	$T_D = 0.4\,\tau_D$

Three major conclusions can be drawn from this table.

(i) The controller gain is inversely proportional to the process gain K which represents the product of gain of all the elements in the loop other than the controller. It means, that if the gain of any of the elements were to change because of recalibration, resizing, or nonlinearity, the response of the feedback loop will change, unless the controller gain is readjusted.

(ii) The controller gain must be reduced when the ratio of the process dead-time to its time-constant, increases. This means, that the difficulty in controlling the loop increases when the ratio of the process dead-time to its time-constant, increases. This ratio, which can be used as a measure of difficulty in controlling a process, will be called the *normalized dead-time* τ_{ND}.

$$\frac{\text{Apparent dead-time } \tau_D}{\text{Apparent time-constant } \tau} = \text{Normalized dead-time } \tau_{ND} \tag{3.54}$$

τ_{ND} can be estimated from the process reaction curve. Processes with small τ_{ND} are easy to control and processes with large τ_{ND} are difficult to control. The parameter τ_{ND} has been called the *controllability ratio* in the literature. To avoid confusion with the standard terminology of modern control theory (Chapter 5), the word normalized dead-time is used here.

Notice that having a long dead-time parameter means that the loop is difficult to control only if the time-constant is short. In other words, a loop with a dead-time of several minutes, would be just as difficult to control as one with a dead-time of a few seconds—if the normalized dead-time for both the loops is the same.

(iii) The speed of response of the controller, which is determined by integral and derivative times, must match the speed of response of the process. The formulas in Table 3.3 match these response speeds by relating the integral and derivative times of the controller to the process dead-time.

In using the formulas in Table 3.3, we must keep in mind that they were developed empirically for the most common range of the normalized dead-time parameter, which is between 0.1 and 0.3, based on the fact that most processes do not exhibit significant transportation lag (rather, the dead-time is the result of several first-order lags in series).

As was pointed out in the earlier discussion on QDR tuning based on ultimate gain and period, the difficulty of the QDR performance specification for PI and PID controllers is that there is an infinite set of values of the controller parameters that can produce it; i.e., for each setting of the integral time on a PI controller, and for each reset-derivative time combination on a PID controller, there is a setting of the gain that results in QDR response. The settings given in Table 3.3 are the figures based on experience; these settings have produced fast response for most industrial loops.

3.5.5 Digital PID Controllers

Most process industries today, use computers to carry out the basic feedback control calculations. The formulas that are programmed to calculate the controller output are mostly the discrete versions of the analog controllers presented earlier in this section. This practice allows the use of established experience with analog controllers and in principle, their well-known tuning rules which could be applied.

As there is no extra cost in programming all the three modes of control, most computer-based algorithms contain all the three, and then use flags and logic to allow the process engineer to specify any single mode or, a combination of two or three modes. Most tunable commercially available controllers use the non-interacting version of PID control (refer to Eqn. (3.47b)). The discussion that follows applies to non-interacting PID control.

Position PID Algorithm

The equation describing an idealized non-interacting PID controller is as follows (refer to Eqn. (3.47a)):

$$u(t) = K_c \left[e(t) + \frac{1}{T_I} \int_0^t e(t)dt + T_D \frac{de(t)}{dt} \right] \tag{3.55}$$

with parameters

K_c = controller gain; T_I = integral time; and T_D = derivative time.

For small sample times T, this equation can be turned into a difference equation by discretization. Various methods of discretization were presented in Section 2.14.

Approximating the derivative mode by the backward-difference approximation and the integral mode by backward integration rule, we obtain (refer to Eqns (2.112))

$$u(k) = K_c \left[e(k) + \frac{1}{T_I} S(k) + \frac{T_D}{T} (e(k) - e(k-1)) \right] \tag{3.56}$$

$$S(k) = S(k-1) + Te(k)$$

where $u(k)$ = the controller output at sample k;

$S(k)$ = the sum of the errors; and

T = the sampling interval.

This is a nonrecursive algorithm. For the formation of the sum, all past errors $e(\cdot)$ have to be stored.

Equation (3.56) is known as the 'absolute form' or 'position form' of the PID algorithm. It suffers from one particular disadvantage, which is manifest when the process it is controlling, is switched from manual

to automatic control. The initial value of the control variable u will simply be ($e(k-1) = S(k-1) = 0$ in Eqn. (3.56)):

$$u(0) = K_c \left[1 + \frac{T}{T_I} + \frac{T_D}{T}\right] e(0)$$

Since the controller has no knowledge of the previous sample values, it is not likely that this output value will coincide with that previously available under manual control. As a result, the transfer of control will cause a 'bump', which may seriously disturb the plant's operation. This can only be overcome by laboriously aligning the manual and computer outputs or, by adding complexity to the controller so that it will automatically 'track' the manual controller.

Practical implementation of the PID algorithm includes the following additional features:

(i) It is seldom desirable for the derivative mode of the controller to respond to set-point changes. This is because the set-point changes cause large changes in the error that last for only one sample; when the derivative mode acts on this error, undesirable pulses or 'derivative kicks' occur on the controller output—right after the set-point is changed. These pulses, which last for one sampling interval, can be avoided by having the derivative mode act on the controlled variable, rather than on the error.

(ii) A pure derivative term should not be implemented, because it will give a very large amplification of the measurement noise. The gain of the derivative must thus be limited. This can be done by approximating the transfer function $T_D s$ as follows:

$$T_D s \cong \frac{T_D s}{\alpha T_D s + 1}$$

where α is the filter parameter, whose value is not adjustable but is built into the design of the controller. It is usually of the order of 0.05 to 0.3.

The PID controller, therefore, takes the form (refer to Fig. 3.24c):

$$U(s) = K_c \left[E(s) + \frac{1}{T_I s} E(s) - \frac{T_D s}{\alpha T_D s + 1} Y(s)\right] \tag{3.57}$$

Discretization of this equation results in the following PID algorithm:

$$u(k) = K_c \left[e(k) + \frac{1}{T_I} S(k) + D(k)\right] \tag{3.58}$$

$$S(k) = S(k-1) + Te(k)$$

$$D(k) = \frac{\alpha T_D}{\alpha T_D + T} D(k-1) - \frac{T_D}{\alpha T_D + T} [y(k) - y(k-1)]$$

Velocity PID Algorithm

This is a recursive algorithm characterized by the calculation of the current control variable $u(k)$ based on the previous control variable $u(k-1)$ and correction terms. To derive the recursive algorithm, we subtract from Eqn. (3.56)

$$u(k-1) = K_c\left[e(k-1) + \frac{1}{T_I}S(k-1) + \frac{T_D}{T}(e(k-1) - e(k-2))\right] \tag{3.59}$$

This gives

$$u(k) - u(k-1) = K_c\left[e(k) - e(k-1) + \frac{T}{T_I}e(k) + \frac{T_D}{T}[e(k) - 2e(k-1) + e(k-2)]\right] \tag{3.60}$$

Now, only the current change in the control variable

$$\Delta u(k) = u(k) - u(k-1) \tag{3.61}$$

is calculated. This algorithm is known as the 'incremental form' or 'velocity form' of the PID algorithm. The distinction between the position and velocity algorithms is significant only for controllers with integral effect.

The velocity algorithm provides a simple solution to the requirement of bumpless transfer. The problem of bumps arises mainly from the need for an 'initial condition' on the integral; and the solution adopted is to externalize the integration, as shown in Fig. 3.31. The external integration may take the form of an electronic integrator but frequently the type of actuating element is changed, so that recursive algorithm is used with actuators which, by their very nature, contain integral action. Stepper motor (refer to Section 3.8) is one such actuating element.

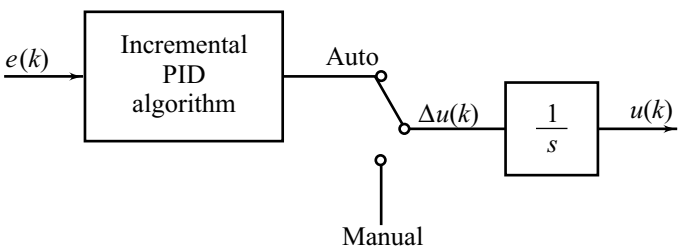


Fig. 3.31 Control scheme of bumpless transfer

Practical implementation of this algorithm includes the features of avoiding derivative kicks and filtering measurement noise. Using Eqn. (3.58) we obtain

$$\Delta u(k) = K_c\left[e(k) - e(k-1) + \frac{T}{T_I}e(k) - \frac{T}{\alpha T_D + T}D(k-1) - \frac{T_D}{\alpha T_D + T}(y(k) - y(k-1))\right] \tag{3.62}$$

$$D(k) = \frac{\alpha T_D}{\alpha T_D + T}D(k-1) - \frac{T_D}{\alpha T_D + T}[y(k) - y(k-1)] \tag{3.63}$$

where $y(k)$ = controlled variable; $\Delta u(k)$ = incremental control variable = $u(k) - u(k-1)$;
$e(k)$ = error variable; K_c = controller gain;
T_I = integral time; T_D = derivative time; and T = sampling interval.

Tuning Rules for Digital Controllers

Though tuning formulas that are specifically applicable to digital control algorithms have been developed [12], the most popular and widely used tuning approach for digital PID controllers is to apply rules in Tables 3.2–3.3 with a simple correction to account for the effect of sampling. When a continuous-time signal is sampled at regular intervals of time, and is then reconstructed by holding the sampled values constant for each sampling interval, the reconstructed signal is effectively delayed by approximately one half of the sampling interval, as shown in Fig. 3.32a (also refer to Example 2.17). In the digital control configuration of Fig. 3.32b, the D/A converter holds the output of the digital controller constant between updates, thus adding one half the sampling time to the dead-time of the process components. The correction for sampling is then, simply, to add one half the sampling time to the dead-time obtained from the process reaction curve.

$$\tau_{CD} = \tau_D + \tfrac{1}{2}T \tag{3.64}$$

where τ_{CD} is the corrected dead-time, τ_D is the dead-time of the process, and T is the sampling interval.

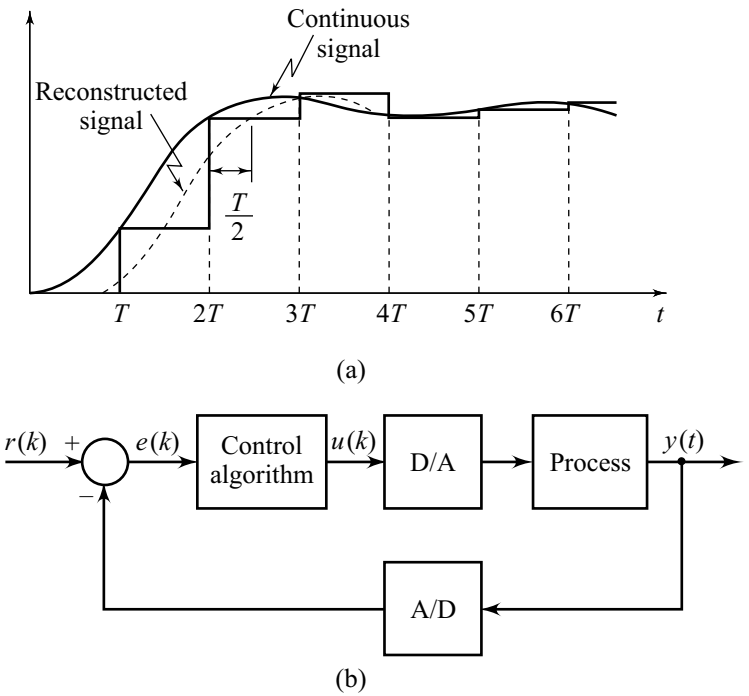


Fig. 3.32 Time delay introduced by sampling and reconstruction

The tuning formulas given in Table 3.3 can directly be used for digital PID controllers with τ_D replaced by τ_{CD}.

Notice that the online tuning method, based on ultimate gain and period, inherently incorporates the effect of sampling when the ultimate gain and period are determined with the digital controller included in the loop. Tuning rules in Table 3.2 can, therefore, be applied to digital control algorithms without any correction.

3.6 DIGITAL TEMPERATURE CONTROL SYSTEM

This section describes the hardware features of the design of a microprocessor-based controller for temperature control in an air-flow system.

Figure 3.33 shows the air-flow system, provided with temperature measurement and having a heater grid with controlled power input. Air, drawn through a variable orifice by a centrifugal blower, is driven past the heater grid and through a length of tubing, to the atmosphere again. The temperature sensing element consists of a bead thermistor fitted to the end of a probe, inserted into the air stream 30 cms from the heater. The task is to implement a controller, in the position shown by dotted box, to provide temperature control of the air stream. It is a practical process-control problem in miniature, simulating the conditions found in furnaces, boilers, air-conditioning systems, etc.

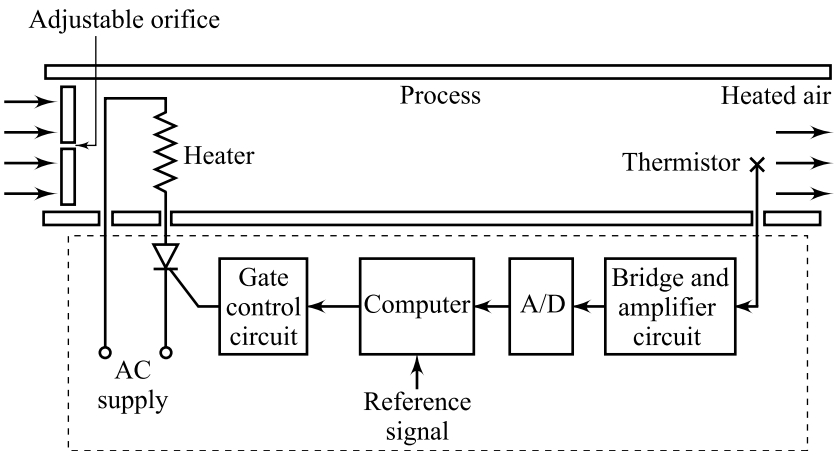


Fig. 3.33 Temperature control in an air-flow system

The functions within the control loop can be broken down as follows:

(a) sampling of the temperature measurement signal at an appropriate rate;
(b) transfer of the measurement signal into the computer;
(c) comparison of the measured temperature with a stored desired temperature, to form an error signal;
(d) operation on the error signal by an appropriate algorithm, to form an output signal; and
(e) transfer of the output signal, through the interface, to the power control unit.

3.6.1 Digital Measurement/Control Schemes

Figure 3.34 gives hardware description of the temperature control system. Let us examine briefly the function of each block. The block labeled *keyboard matrix*, interfaced to the microcomputer through a programmable keyboard/display interface chip, enables the user to feed reference input to the temperature control system. The LED *display* unit provides display of the actual temperature of the heating chamber.

The temperature range for the system under consideration is 20 to 60ºC. When a thermistor is used as temperature sensor, it is necessary to convert the change in its resistance to an equivalent analog voltage. This is accomplished with *Wheatstone bridge*; the thermistor exposed to the process air, forms one arm of the bridge. The millivolt range of the bridge error voltage is *amplified* to the range required by A/D converter. The output of the A/D converter is the digital measurement of the actual temperature of the process air. This data is fed to the microcomputer through an input port. The microcomputer compares the actual temperature with the desired temperature at each sampling instant, and generates an error signal. The error signal is then processed as per the control algorithm (to be given later), resulting in a control signal in digital form. The control signal is, in fact, the amount of power required to be applied to the plant, in order to reduce the error between the desired temperature and the actual temperature. The power input to the plant may be controlled with the help of *triacs and firing circuit* interface.

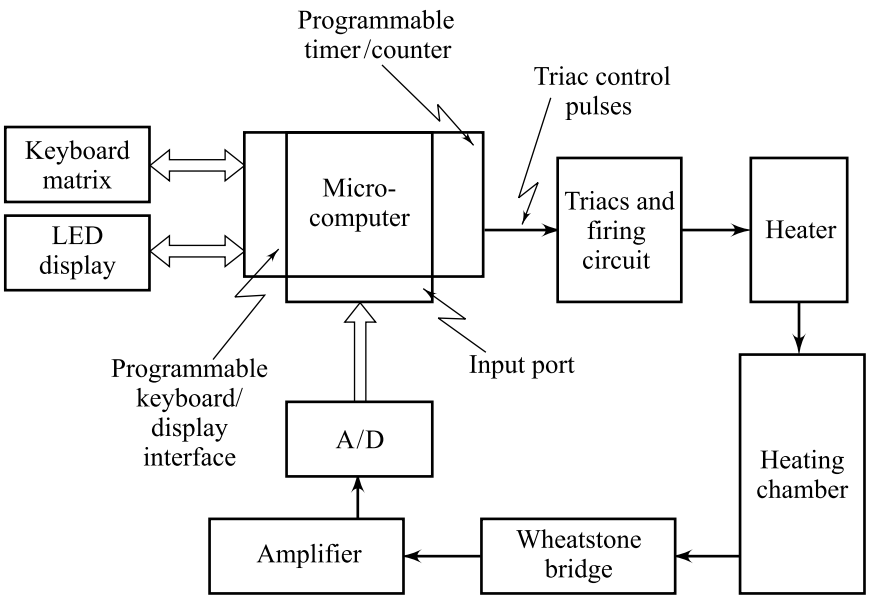


Fig. 3.34 Block diagram of the temperature control system

A basic circuit using a triac (bidirectional thyristor) which controls the flow of alternating current through the heater is shown in Fig. 3.35a. If the triac closes the circuit for t_p seconds out of T seconds, the average power applied to the plant over the sampling period T is

$$u = \frac{1}{T}\int_0^{t_p} \frac{V^2}{R}\,dt = \frac{V^2}{R}\frac{t_p}{T}$$

V = rms value of the voltage applied to the heater; and

R = resistance of the heater.

This gives

$$t_p = \frac{u}{V^2/R}T \tag{3.65}$$

Depending on the control signal u (power required to be applied to the plant), t_p is calculated in the microcomputer. A number is latched in a down counter (in the programmable timer/counter chip interfaced with the microcomputer) which is determined by the value of t_p and the counter's clock frequency. A pulse of required width t_p is thus available at each sampling instant from the programmable timer/counter chip. This, in fact, is a *pulse width modulated* (PWM) wave whose time period is constant and width is varied in accordance with the power required to be fed to the plant (Fig. 3.35b).

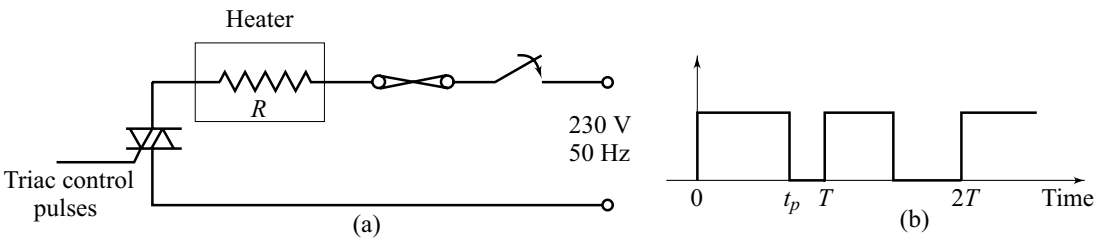


Fig. 3.35 Control scheme for the thermal process

The function of the triacs and firing circuit interface is thus, to process the PWM output of the microcomputer, such that the heater is ON when the PWM output is logic 1, and OFF when it is logic 0. Since the heater is operated off 230 V ac at 50 Hz, the firing circuit should also provide adequate isolation between the high voltage ac signals and the low voltage digital signals.

3.6.2 Control Algorithm

A model for the temperature control system under study is given by the block diagram of Fig. 3.36. A gain of unity in the feedback path corresponds to the design of feedback circuit (temperature transducer + amplifier + A/D converter) which enables us to interpret the magnitude of the digital output of A/D converter directly as temperature in ºC. The temperature command is given in terms of the digital number with magnitude equal to the desired temperature in ºC. The error e (ºC) is processed by the control algorithm with transfer function $D(z)$. The computer generates a PWM wave whose time period is equal to the sampling interval, and width is varied in accordance with the control signal u (watts). The PWM wave controls the power input to the plant through the triacs and the firing circuit interface. Since the width of PWM remains constant over a sampling interval, we can use S/H to model the input-output relation of the triacs and the firing circuit interface.

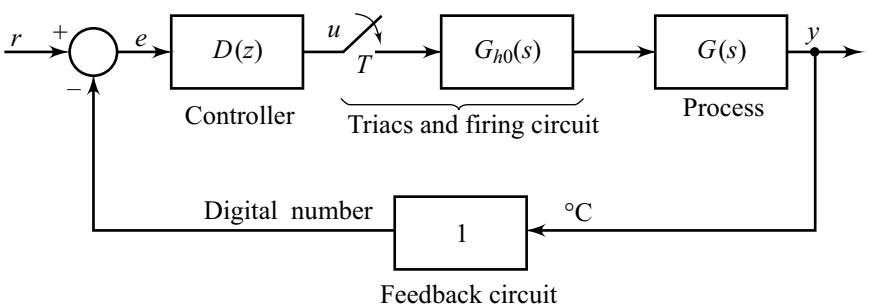


Fig. 3.36 A model for the temperature control system

To develop the digital controller $D(z)$ for the process, we will follow the approach of controller tuning (refer to Section 3.5). A simple tuning procedure consists of the following steps:

(i) Obtain experimentally the dynamic characteristics of the process, either by open-loop or closed-loop tests.

(ii) Based on dynamic characteristics of a process, tuning rules have been developed by Ziegler and Nichols (refer to Tables 3.2–3.3). Use these rules to obtain initial settings of the controller parameters K_c, T_I, and T_D of the PID controller

$$D(s) = K_c \left[1 + \frac{1}{T_I s} + T_D s\right] \tag{3.66}$$

(iii) Discretize the PID controller to obtain digital control algorithm for the temperature control process. Thumb rules given in Section 2.13 may be followed for initial selection of sampling interval T.

In digital mode, the PID controller takes the form (refer to Eqn. (2.125))

$$D(z) = K_c \left[1 + \frac{T}{2T_I}\left(\frac{z+1}{z-1}\right) + \frac{T_D}{T}\left(\frac{z-1}{z}\right)\right] = \frac{U(z)}{E(z)} \tag{3.67}$$

(iv) Implement the digital PID controller. Figure 3.37 shows a realization scheme for the controller; the proportional, integral, and derivative terms are implemented separately and summed up at the output.

(v) Fine tune K_c, T_I, T_D and T to obtain acceptable performance.

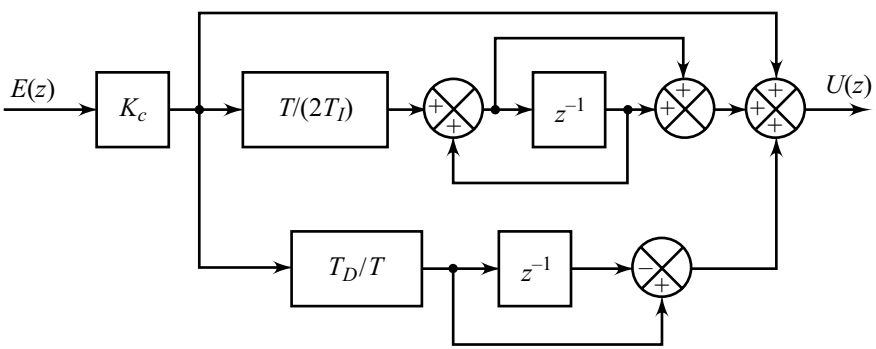


Fig. 3.37 A realization scheme for the PID controller

An open-loop test was performed on the air-flow system (Fig. 3.33) to obtain its dynamic characteristics.

Input : heater power
Output : air temperature

The test was carried out with a dc input signal. A wattmeter, on the input side, measured the heater power, and a voltmeter, on the output side, measured the output (in volts) of the bridge circuit, which is proportional to the air temperature in ºC.

Figure 3.38 shows the response for a step input of 20 watts. This process reaction curve was obtained for a specific orifice setting.

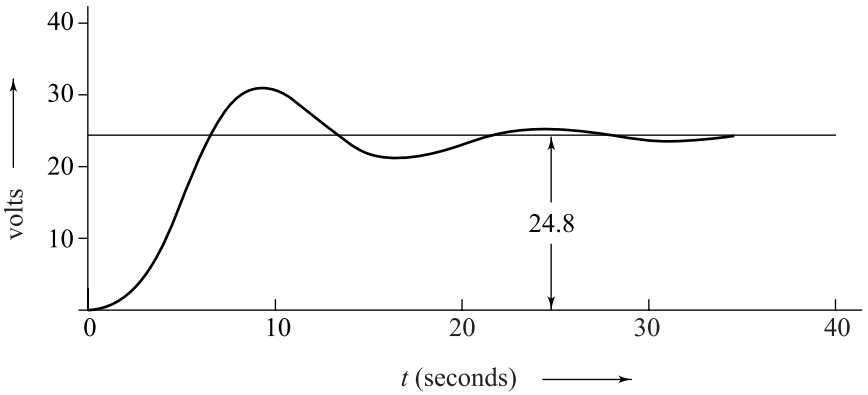


Fig. 3.38 Process reaction curve of the air-flow system

Approximation of the process reaction curve by a first-order plus dead-time model is obtained as follows (refer to Fig. 3.29):

The change in the process output at steady state is found to be $\Delta y_{ss} = 24.8$ volts. Therefore, the process gain

$$K = \frac{24.8}{20} = 1.24 \text{ volts/watt}$$

The line that is tangent to the process reaction curve at the point of maximum rate of change gives $\tau_D = 0.3$ sec. The time at which the response is $0.632\,\Delta y_{ss}$ is found to be 0.83 sec. Therefore, $\tau + \tau_D = 0.83$; which gives $\tau = 0.53$ sec. (It may be noted that the response is oscillatory in nature; therefore, a second-order model will give a better fit. However, for coarse tuning, we have approximated the response by a first-order plus dead-time model).

The process reaction curve of the air-flow system is thus represented by the model:

$$G(s) = \frac{Ke^{-\tau_D s}}{\tau s + 1} = \frac{1.24e^{-0.3s}}{0.53s + 1} \tag{3.68}$$

Taking a sampling interval $T = 0.1$ sec, we have (refer to Eqn. (3.64)):

$$\tau_{CD} = \tau_D + \frac{1}{2}T = 0.35 \text{ sec}$$

Using tuning formulas of Table 3.3, we obtain the following parameters for the PID controller:

$$\begin{aligned} K_c &= 1.5\,\tau/(K\tau_{CD}) = 1.832 \\ T_I &= 2.5\,\tau_{CD} = 0.875 \\ T_D &= 0.4\,\tau_{CD} = 0.14 \end{aligned} \tag{3.69}$$

3.7 DIGITAL POSITION CONTROL SYSTEM

This section describes hardware features of the design of a microprocessor-based controller for a position control system. The plant of our digital control system is an inertial load, driven by an armature-

controlled dc servo motor. The plant also includes a motor-drive circuit. The output of the drive circuit is fed to the armature of the motor which controls the position of the motor shaft. In addition, it also controls the direction of rotation of the motor shaft.

Figure 3.39 gives hardware description of the position control system. Let us examine briefly the function of each block.

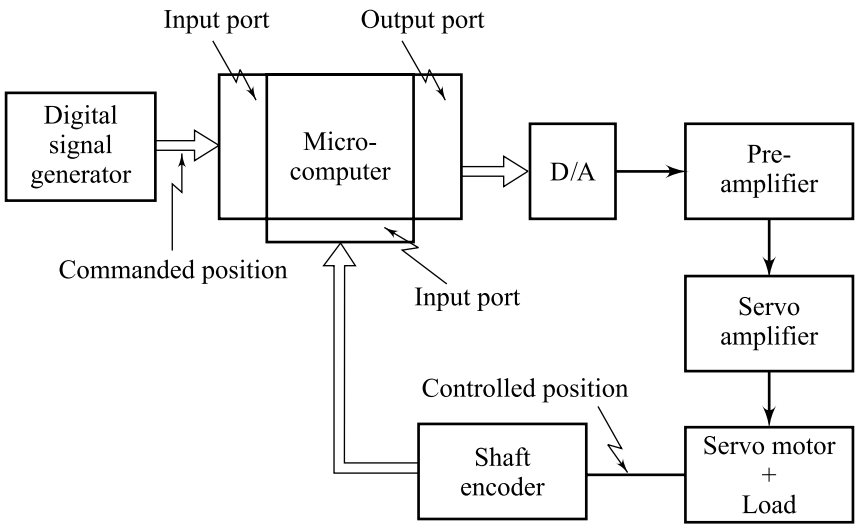


Fig. 3.39 Block diagram of the digital positioning system

The block labeled *digital signal generator*, interfaced with the microcomputer through an input port, enables the user to feed the desired position (set-point) of the motor shaft. A keyboard matrix can be used for entering numerical commands into the digital system.

The microcomputer compares the actual position of the motor shaft with the desired position at each sampling instant, and generates an error signal. The error signal is then processed as per the control algorithm (to be given later), resulting in a control signal in digital form. The digital control signal is converted to a bipolar (can be + ve or –ve) analog voltage in the D/A converter interfaced to the microcomputer. This bipolar signal is processed in a preamplifier and servo amplifier (power amplifier), enabling the motor to be driven in one direction for positive voltage at preamplifier input, and in opposite direction for a negative voltage.

With these units, the block diagram of Fig. 3.39 also shows a shaft encoder for digital measurement of shaft position/speed. We now examine in detail the principle of operation of this digital device.

3.7.1 Digital Measurement of Shaft Position/Speed

The digital measurement of shaft position requires conversion from the analog quantity 'shaft angle' to a binary number. One way of doing this would be to change shaft angle to a voltage using a potentiometer, and then to convert it to a binary number through an electronic A/D converter. This is perfectly feasible, but is not sensible because of the following reasons:

(i) high quality potentiometers of good accuracy are expensive and subject to wear; and
(ii) the double conversion is certain to introduce more errors than a single conversion would.

We can go straight from angle to number, using an optical angular *absolute-position encoder*. It consists of a rotary disk made of a transparent material. The disk is divided into a number of equal angular sectors—depending on the resolution required. Several tracks, which are transparent in certain sectors but opaque in others, are laid out. Each track represents one digit of a binary number. Detectors on these tracks sense whether the digit is a '1' or a '0'. Figure 3.40 gives an example. Here, the disk is divided into eight 45º sectors. To represent eight angles in binary code requires three digits ($2^3 = 8$), hence, there are three tracks. Each track has a light source sending a beam on the disk and, on the opposite side, a photoelectric sensor receiving this beam. Depending upon the angular sector momentarily facing the sensors, they transmit a bit pattern representing the angular disk position. For example, if the bit pattern is 010, then Sector IV is facing the sensors.

Figure 3.40 is an example of an 'absolute encoder'. It is so called because for a given angle, the digital output must always be the same. Note that a cyclic (Gray) binary code is normally used on absolute encoders (in cyclic codes, only one bit changes between adjacent numbers). If a natural binary-code pattern were used, a transition from, say, 001 to 010, would produce a race between the two right-hand bits. Depending on which photosensor responded faster, the output would go briefly through 011 or 000. In either case, a momentary *false* bit pattern would be sent. Cyclic codes avoid such races. A cyclic code can be converted into a natural binary code by using either hardware or computer software.

Encoders similar to Fig. 3.40 have been widely used. However, they have certain disadvantages.

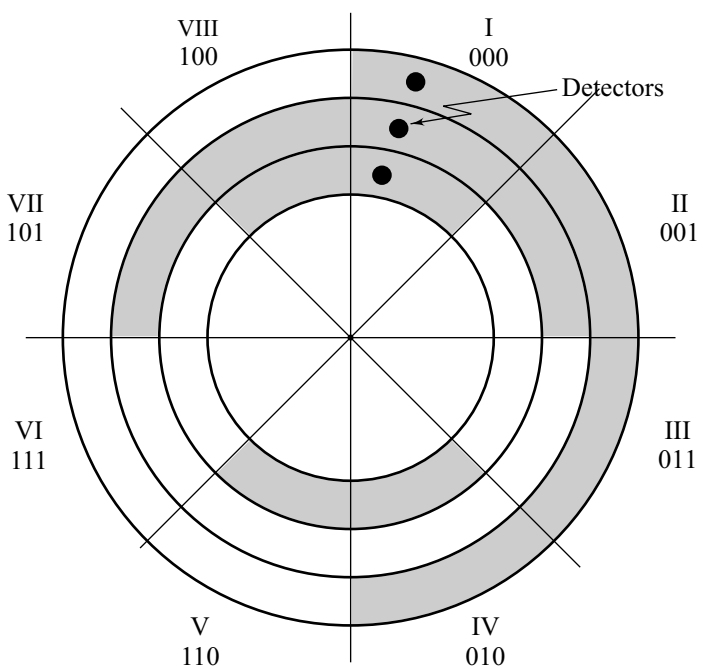


Fig. 3.40 Absolute encoder

(i) The resolution obtainable with these encoders is limited by the number of tracks on the encoder disk. The alignment of up to ten detectors and the laying out of ten tracks is still quite difficult and thus expensive.
(ii) The resulting digital measurement is in a cyclic code and must usually be converted to natural binary before use.
(iii) The large number of tracks and detectors, inevitably increases the chance of mechanical and/or electrical failure.

For these reasons, another form of encoder is commonly used today and is known as the *incremental encoder*. The basis of an incremental encoder is a single track served by a single detector, and laid out in equal segments of '0' and '1', as in Fig. 3.41. As the track moves relative to the detector, a pulse train is generated, and can be fed to a counter to record how much motion has occurred. With regard to this scheme of measurement, the following questions may be raised:

(i) How do we know which direction the motion was?
(ii) If we can record only the distance moved, how do we know where we were?

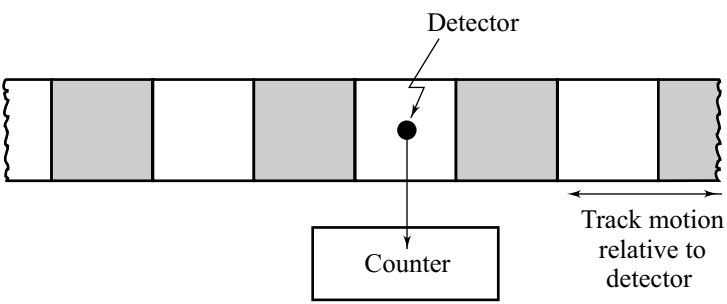


Fig. 3.41 Incremental encoder

The answer to the first question involves the addition of a second detector. Figure 3.42a shows two detectors, spaced one half of a segment apart. As the track moves relative to the detectors (we assume at a constant rate), the detector outputs vary with time, as shown in the waveforms of Fig. 3.42b. We can see that the relative 'phasing' of the *A* and *B* signals depends upon the direction of motion, and so gives us a means of detecting the direction.

For example, if signal *B* goes from '0' to '1' while signal *A* is at '1', the motion is positive. For the same direction, we see that *B* goes from '1' to '0' whilst *A* is at '0'. For negative motion, a similar but different pair of statements can be made. By application of some fairly simple logic, it is possible to control a reversible counter as is indicated in Fig. 3.43

This method of direction-sensing is referred to as *quadrature encoding*. The detectors are one half of a segment apart, but reference to the waveforms of Fig. 3.42 shows that there are two segments to one cycle; so the detectors are one quarter of a cycle apart, and hence the name.

The solution to the second problem also requires an additional detector working on a datum track, as shown in Fig. 3.44. The datum resets the counter every time it goes by.

We have thus three detectors in an incremental encoder. But this is still a lot less than on an absolute encoder.

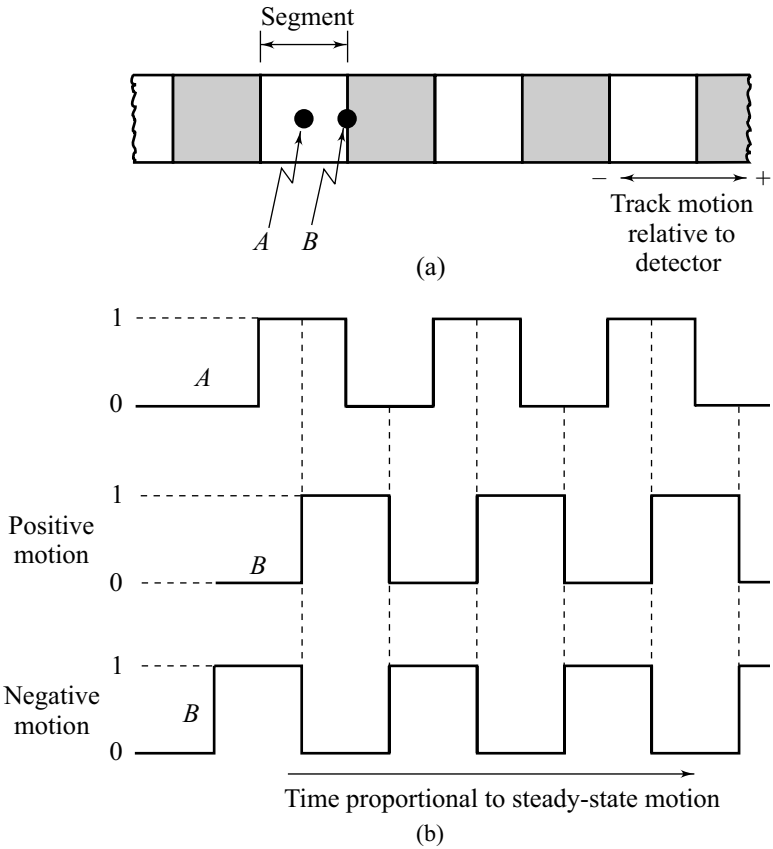


Fig. 3.42 Quadrature encoding

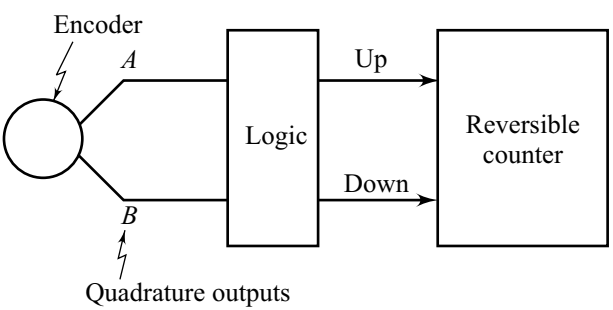


Fig. 3.43 Direction-detecting circuit for an incremental encoder

In an analog system, speed is usually measured by a tachogenerator attached to the motor shaft. This is because the time differentiation of analog position signal presents practical problems.

In a digital system, however, it is relatively easy to carry out step-by-step calculation of the 'slope' of the position/time curve. We have the position data in digital form from the shaft encoder, so the rest is fairly straightforward.

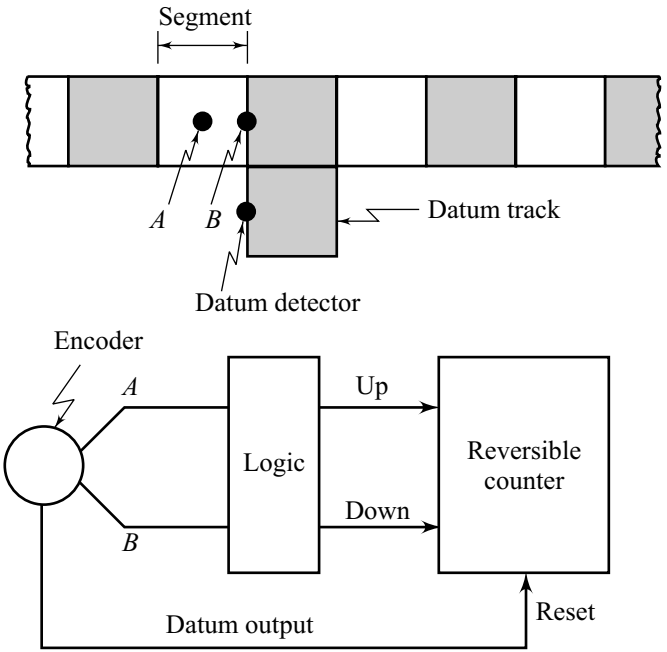


Fig. 3.44 Datum detector in an incremental encoder

Encoder Interface Implementation

In a position control system, the load is connected to the motor through a gear train. The encoder may be connected to the load shaft/motor shaft directly or through a pulley system.

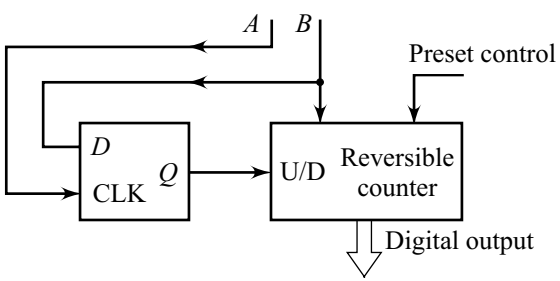


Fig. 3.45 A simple encoder interface circuit

To know the shaft position, the number of pulses obtained at the output of detector *A* or detector *B* have to be counted. To know the direction of rotation, the relative phasing of the outputs of detectors *A* and *B* has to be sensed. To implement the direction sensing, a negative edge-triggered *D*-flipflop may be used (Fig. 3.45). This flipflop has the following two inputs:

(i) clock input, derived from the output of detector *A* of the encoder; and
(ii) '*D*' input, derived from the output of detector *B* of the encoder.

Every time the flipflop is triggered on the $1 \rightarrow 0$ transition of waveform *A*, the output of the *D* flipflop is either 1 or 0, depending on the direction of rotation of the shaft. The output of the *D* flipflop thus serves as the control for the up/down input of the counter, reversing the direction of its count whenever the shaft reverses its direction.

3.7.2 Control Algorithm

The mathematical model of the position control under study is given by the block diagram of Fig. 3.46. The magnitude of the digital output of the shaft encoder can be interpreted directly as the position of the motor shaft in degrees, by proper design of the encoder interface. Similarly, the magnitude of the digital reference input can be interpreted directly as reference input in degrees, by proper design of the keyboard matrix interface. The error e (degrees) in position is processed by the control algorithm with transfer function $D(z)$. The control signal u (in volts) is applied to the preamplifier through the D/A converter. The plant (preamplifier + servo amplifier + dc motor + load) is described by the transfer function

$$\frac{\theta(s)}{V(s)} = G(s) = \frac{94}{s(0.3s+1)} \tag{3.70}$$

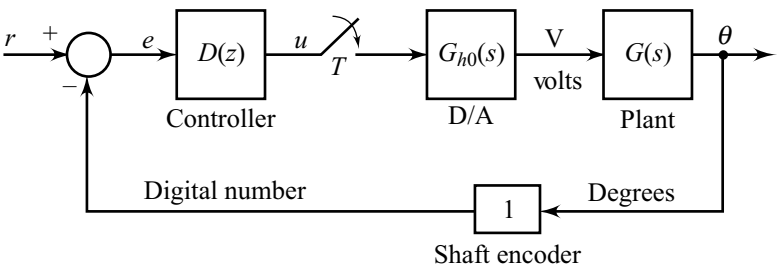


Fig. 3.46 Mathematical model for the position control system

To design the digital controller $D(z)$ for this plant, we will follow the approach of discretization of analog design (refer to Section 2.14). The design requirements may be fixed as $\zeta = 0.7$ and $\omega_n \cong 10$. The first step is to find a proper analog controller $D(s)$ that meets the specifications. The transfer function

$$D(s) = K_c \frac{(s+3.33)}{s+\alpha}$$

cancels the plant pole at $s = -3.33$. The characteristic roots of

$$1 + D(s)G(s) = 0$$

give $\zeta = 0.7$ and $\omega_n = 10$ if we choose $K_c = 0.32$ and $\alpha = 14$.

The controller

$$D(s) = \frac{0.32(s+3.33)}{s+14} \tag{3.71}$$

gives the following steady-state behavior:

$$K_v = \lim_{s \to 0} sG(s)D(s) = 7.15$$

This may be considered satisfactory.

The discretized version of the controller $D(s)$ is the proposed digital controller $D(z)$ for the control loop of Fig. 3.46. The $D(z)$ will perform as per the specifications if the lagging effect of zero-order hold is negligible. We take a small value for sampling interval T to satisfy this requirement. For a system with $\omega_n = 10$ rad/sec, a very 'safe' sample rate would be—a factor of 20 faster than ω_n, yielding

$$\omega_s = 10 \times 20 = 200 \text{ rad/sec}$$

and $$T = \frac{2\pi}{\omega_s} \cong 0.03 \text{ sec}$$

The dominant time constant of the plant is 0.3 sec. The sampling interval T is one tenth of this value.

We use the bilinear transformation given by

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$$

to digitize $D(s)$. This results in

$$D(z) = \frac{22.4z - 20.27}{80.67z - 52.67} = \frac{0.278 - 0.25z^{-1}}{1 - 0.653z^{-1}} = \frac{U(z)}{E(z)} \tag{3.72a}$$

The control algorithm is, therefore, given by

$$u(k) = 0.653\, u(k-1) + 0.278\, e(k) - 0.25\, e(k-1) \tag{3.72b}$$

This completes the digital algorithm design.

3.8 STEPPING MOTORS AND THEIR CONTROL

The explosive growth of the computer industry in recent years has also meant an enormous growth for stepping motors, because these motors provide the driving force in many computer peripheral devices. Stepping motors can be found, for example, driving the paper-feed mechanism in printers. These motors are also used exclusively in floppy disk drives, where they provide precise positioning of magnetic head on the disks. The X and Y coordinate pens in plotters, are driven by stepping motors.

The stepping motor can be found performing countless tasks outside the computer industry as well. The most common application is probably in analog quartz watches where tiny stepping motors drive the hands. These motors are also popular in numerical-control applications (positioning of the workpiece and/or the tool in a machine according to previously specified numerical data).

A stepping motor is especially suited for applications mentioned above because, essentially, it is a device that serves to convert input information in *digital form* to an output that is mechanical. It thereby provides a natural interface with the digital computer. A stepping motor, plus its associated drive electronics, accepts a pulse-train input and produces an increment of rotary displacement for each pulse. We can control average speed by manipulating pulse rate, and motor position by controlling the total pulse count.

Two types of stepping motors are in common use—the permanent magnet (PM), and the variable reluctance (VR). We will discuss the PM motor first.

3.8.1 Permanent Magnet Motor

Constructional Features

A PM stepping motor in its simplest form is shown in Fig. 3.47. The motor has a permanent magnet rotor that, in this example, has two poles, though often many more poles are used. The stator is made of soft

iron with a number of pole pieces and associated windings. Only four windings (grouped into two sets of two windings each) are used in this example. These windings must be excited sequentially in a certain order. Although this is commonly done by solid-state switching circuits, mechanical switches are shown in the figure since their operation is easier to visualize.

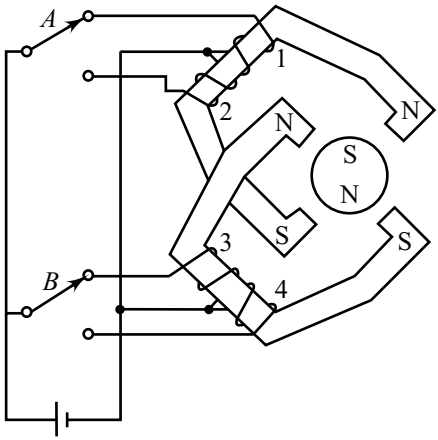


Fig. 3.47 A PM stepping motor (four phase)

Assume the switches to be in the positions shown. Windings 1 and 3 are energized and, as a result, the pole pieces have the polarities shown. The rotor is thus found in the position shown with its *S* pole centered between the two upper *N* pole pieces, and its *N* pole between the two lower *S* pole pieces. With the field maintained, if we try to twist the shaft away from its standstill (equilibrium) position, we feel a 'magnetic spring' restoring torque. However, a sufficiently large external torque can overcome the magnetic spring.

With the rotor energized and in equilibrium position, the torque required from an external source to break away the motor from this position is called the *holding torque*. The holding torque is a basic characteristic of the stepping motors, and provides positional integrity under standstill conditions.

If we now imagine the position of switch *A* changed, then winding 2 is energized instead of winding 1. As a result, the right upper pole piece becomes *S* instead of *N*, and the left lower one *N*, so that the rotor is forced to rotate 90º counterclockwise. Changing switch *B* produces the next 90º step, etc. The rotor is thus forced to realign itself continuously according to the prevalent magnetic fields. If it is desired to reverse the direction of rotation, the order of changing the switch positions need only be reversed.

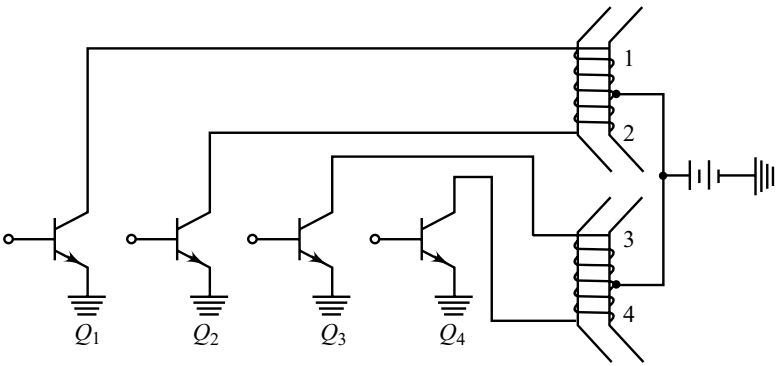
One characteristic feature of PM stepping motors is that they have a so-called *residual* or *detent torque* when power to the stator windings is cut off. It is the result of the permanent-magnet flux of the PM motor acting on residual flux on stator poles. The detent torque is naturally much lower than the holding torque, produced when the stator is energized, but it does help in keeping the shaft from moving due to outside forces.

Many motors have more than four stator pole pieces—and possibly also more rotor poles—resulting in smaller step angles. Typical step angles for PM motors range from 90º to as low as 1.8º. Stator pole windings are connected in so-called *phases*, with all windings belonging to the same phase energized at the same time. Typically, the phases can range from as low as two to as high as eight. The more phases the motor has, the smoother is its output torque.

Sequence Logic and Drive Amplifier

Figure 3.48 shows a simple power drive scheme; each time the power transistors are switched as per the sequence given in the chart, the motor moves through a fixed angle, referred to as the *step angle*. The chart is circular in the sense that, the next entry after Step 4 is Step 1. To rotate the motor in a clockwise direction, the chart is traversed from top to bottom, and to rotate the motor in counterclockwise direction,

the chart is traversed from bottom to top. Number of step movements/sec gives the *stepping rate*—a parameter that gives a measure of the speed of operation of the stepping motor. The stepping rate is controlled by changing the switching frequency of the transistors.



CW rotation ↓

Step	Q_1	Q_2	Q_3	Q_4
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF

↑ CCW rotation

Fig. 3.48 A simple stepping motor drive scheme

From the foregoing description of the method of operation of a stepping motor, we observe that the stepping action of the motor is dependent on a specific switching sequence that serves to energize and de-energize the stator windings. In addition to the sequence requirement, the windings must be provided with sufficient current. These requirements are met by the stepping motor driver, whose block diagram is shown in Fig. 3.49. The sequence-logic section of the motor driver accepts the pulse-train input, and also receives a binary direction signal indicating the direction in which the motor is to step. It then produces an appropriate switching sequence, so that each phase of the motor is energized at the proper time. The drive-amplifier section consists of power transistors supplying sufficient current to drive the motor.

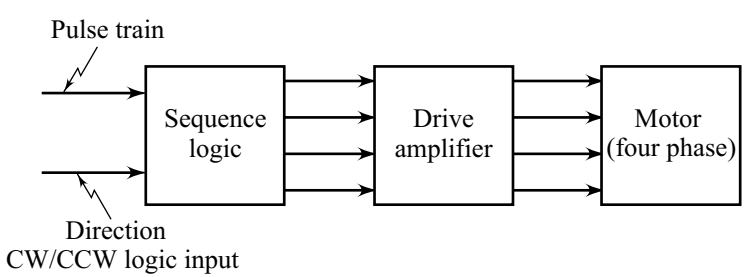


Fig. 3.49 Stepping motor driver (four phase)

3.8.2 Variable Reluctance Motor

Figure 3.50 illustrates a typical Variable Reluctance (VR) motor. The rotor is made of magnetic material, but it is not a permanent magnet, and it has a series of teeth (eight in this case) machined into it. As with the PM stepping motor, the stator consists of a number of pole pieces with windings connected in phases; all windings belonging to the same phase are energized at the same time. The stator in Fig. 3.50 is designed for 12 pole pieces with 12 associated windings arranged in three phases (labeled 1, 2, and 3, respectively). The figure shows a set of four windings for Phase 1; the windings for the other two phases have been omitted for clarity.

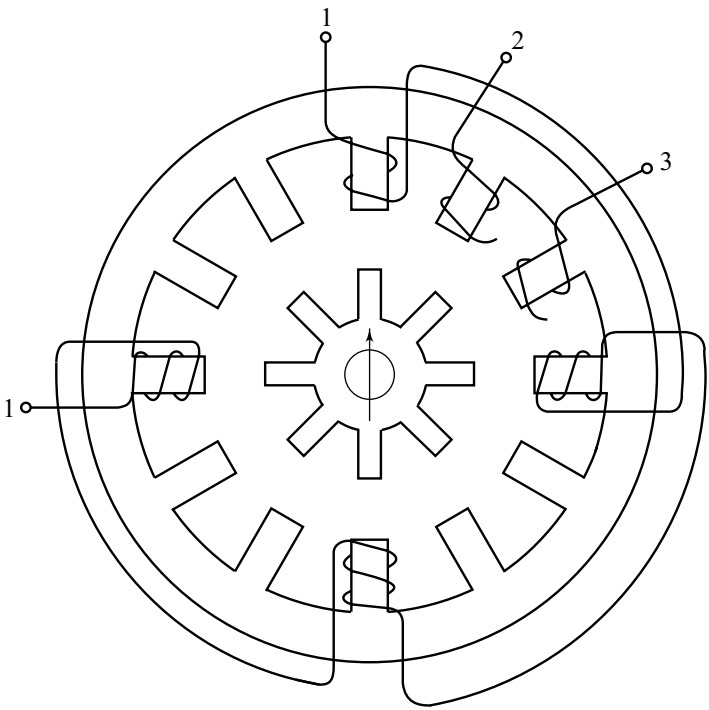


Fig. 3.50 A VR stepping motor (three phase)

The operating principle of the VR motor is straightforward. Let any phase of the windings be energized with a dc signal. The magnetomotive force set up will position the rotor such that the teeth of the rotor section in the neighborhood of the excited phase of the stator, are aligned opposite to the pole pieces associated with the excited phase. This is the position of minimum reluctance, and the motor is in stable equilibrium. Figure 3.50 illustrates the rotor in the position it would assume when Phase 1 is energized. If we now de-energize Phase 1 and energize Phase 2, the rotor rotates counterclockwise so that the four rotor teeth nearest to the four pole pieces belonging to Phase 2, align themselves with these. The step angle of the motor equals the difference in angular pitch between adjacent rotor teeth and adjacent pole pieces; in this case 45 – 30 = 15º. Due to this difference relationship, VR motors can be designed to operate with considerably smaller step angles than PM motors. Other advantages of VR motors include faster dynamic response and the ability to accept higher pulse rates.

Among the drawbacks—their output torque is lower than that of a PM motor of a similar size, and they do not provide any detent torque when not energized.

3.8.3 Torque-Speed Curves of a Stepping Motor

Torque *versus* speed curves of a stepping motor give the dynamic torque, produced by the stepping motor at a given stepping rate, on excitation under rated conditions. The dynamic torque of a motor is the most important data and it plays a major role in the selection of a motor for a specified application. In a load-positioning application, for instance, the rotor would typically start from rest and accelerate the load to the desired position. To provide this type of motion, the motor must develop sufficient torque to overcome friction, and to accelerate the total inertia. In accelerating the inertia, the motor may be required to develop a large amount of torque, particularly if the acceleration must be completed in a short time—so as to position the load quickly. Inability of the motor to develop sufficient torque during motion may cause the motor to stall, resulting in a loss of synchronization between the motor steps and phase excitation, and consequently, resulting in incorrect positioning of the load.

A typical torque *versus* stepping rate characteristic graph is shown in Fig. 3.51, in which curve *a* gives pull-in torque *versus* rotor steps/sec and curve *b* gives pull-out torque *versus* rotor steps/sec.

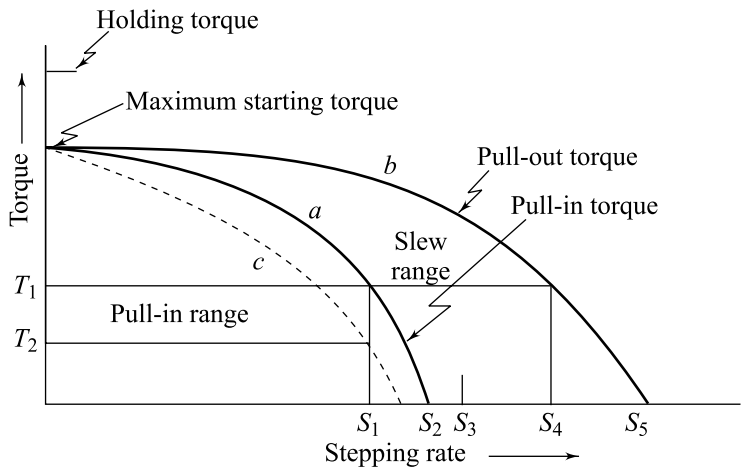


Fig. 3.51 Torque versus stepping rate characteristics

The *pull-in range* (the area between axes and curve *a*) of the motor is the range of switching speeds at which the motor can start and stop, without losing steps. For a frictional load requiring torque T_1 to overcome friction, the maximum *pull-in rate* is S_1 steps per sec. S_2 is the *maximum pull-in rate* at which the unloaded motor can start and stop, without losing steps.

When the motor is running, the stepping rate can be increased above the maximum pull-in rate, and when this occurs, the motor is operating in the *slew-range* region (the area between horizontal axis, and curves *a* and *b*). The slew range gives the range of switching speeds within which the motor can run unidirectionally, but cannot be started or reversed (at shaft torque T_1, the motor cannot be started or reversed at step rate S_3). When the motor is running in the slew range, it can follow changes in the stepping rate without losing steps, but only with a certain acceleration limit.

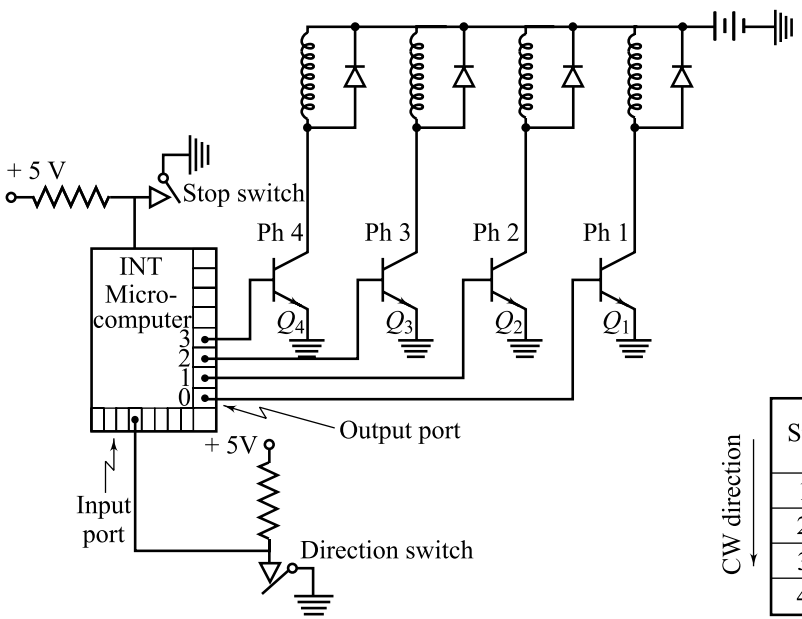
For a frictional load requiring torque T_1 to overcome friction, the maximum *slewing rate* at which the motor can run is S_4. S_5 is the maximum slewing rate at which the unloaded motor can run without losing steps.

Curve *c* in Fig. 3.51 gives the pull-in torque with external inertia. It is obvious that if the external load results in a pull-in torque curve *c*, the torque developed by the motor at step rate S_1 is $T_2 < T_1$. Stepping motors are more sensitive to the inertia of the load than they are to its friction.

3.8.4 Interfacing of Stepping Motors to Microprocessors

In motion control technology, the rise of stepping motors, in fact, began with the availability of easy-to-use integrated circuit chips to drive these stepping motors. These chips require, as inputs a pulse train at the stepping frequency, a logic signal to specify CW and CCW rotation, and a logic signal for STOP/START operation. An adjustable frequency pulse train is readily obtained from another integrated circuit chip—a voltage-controlled oscillator.

The application of stepping motors has shot up with the availability of low-cost microprocessors. A simplified form of microprocessor-based stepping motor drive is shown in Fig. 3.52. The system requires an input port and an output port (this requirement is reduced to one port if a programmable I/O port is used). Output port handles the binary pattern applied to the stepping motor (which is assumed to be a four-phase motor). The excitation sequence is usually stored in a table of numbers. A pattern for four-phase motor is shown in the chart of Fig. 3.52. The chart is circular in the sense that the next entry after Step 4 is Step 1. To rotate the motor in a clockwise direction, the chart is traversed from top to



Step	Port line			
	3	2	1	0
1	0	1	0	1
2	1	0	0	1
3	1	0	1	0
4	0	1	1	0

CW direction ↓

↓ CCW direction

Fig. 3.52 Microprocessor-based stepping motor drive

bottom, and to rotate the motor in counterclockwise direction, the chart is traversed from bottom to top. By controlling the number of bit-pattern changes, and the speed at which they change, it is possible to control the angle through which the motor rotates and the speed of rotation. These controls can easily be realized through software.

The system operator has control over the direction of rotation of the motor by means of a DIRECTION switch, which is interfaced to the CPU through the input port. The operator is also provided with a STOP switch which is connected to an interrupt line of the CPU. The interrupt routine must stop the motor by sending out logic '0's on the data bus lines connected to the stepping motor windings through the output port.

Figure 3.52 also shows a simple drive circuit for the stepping motor. Power transistors Q_1–Q_4 act as switching elements.

When a power transistor is turned off, a high voltage builds up due to di/dt, which may damage the transistor. This surge in voltage can be suppressed by connecting a diode in parallel with each winding in the polarity shown in Fig. 3.52. Now, there will be a flow of circulating current after the transistor is turned off, and the current will decay with time.

3.8.5 Comments

Stepping motors present a number of pronounced advantages, as compared to conventional electric motors:

(i) Since the stepping-motor shaft angle bears an exact relation to the number of input pulses, the motor provides an accurate open-loop positioning system without the need for closing the loop with a position encoder, comparator, and servo amplifier, as is done in conventional closed-loop systems.

(ii) If the stepping motor receives a continuous train of pulses at constant frequency, it rotates at a constant speed, provided neither the load torque nor the pulse frequency are excessive for the given motor. The stepping motor can thus take the place of a velocity servo, again, without the need for a closed-loop system. By changing pulse frequency, the motor speed can be controlled. Even low velocities can be maintained accurately, which is difficult to do with conventional dc motors.

(iii) By driving several motors from the same frequency source, synchronized motions at different points in a machine are easily obtained. Using standard frequency-divider chips, we can drive a motor at a precise fraction of another motor's speed, giving an electronic gear train.

(iv) If the motor stator is kept energized during standstill, the motor produces an appreciable holding torque. Thus, the load position can be locked without the need for clutch-brake arrangements. The motor can be stalled in this manner indefinitely, without adverse effects.

There are, of course, also certain drawbacks.

(i) If the input pulse rate is too fast, or if the load is excessive, the motor will 'miss' steps, making the speed and position inaccurate.

(ii) If the motor is at rest, an external disturbing torque greater than the motor's holding torque, can twist the motor shaft away from its commanded position by any number of steps.

(iii) With high load inertias, overshooting and oscillations can occur unless proper damping is applied, and under certain conditions, the stepping motor may become unstable.

(iv) Stepping motors are only available in low or medium hp ratings, up to a couple of hp (in theory, larger stepping motors could be built, but the real problem lies with the controller—how to get large currents into and out of motor windings at a sufficiently high rate, in spite of winding inductance).

(v) Stepping motors are inherently low-speed devices, more suited for low-speed applications because gearing is avoided. If high speeds are required, this of course becomes a drawback.

Since the cost and simplicity advantages of stepping-motor control systems erode when motion sensors and feedback loops are added, much effort has gone into improving the performance of open-loop systems:

(i) As explained earlier in connection with Fig. 3.51, the permissible pulse rate for starting an inertia load (i.e., the pull-in rate), is much lower than the permissible pulse rate once the motor has reached maximum speed (pull-out rate). A good controller brings the motor up to its maximum speed gradually, a process called *ramping*[2], in such a manner that no pulses are lost. Similarly, a good controller controls deceleration when the motor is to be stopped.

(ii) Various schemes for improving damping to prevent overshooting and oscillations when the motor is to be stopped, are available. Mechanical damping devices provide a simple solution, but these devices reduce the available motor torque and also, mostly, require a motor with a double-ended shaft. Therefore, electronic damping methods are usually preferred. A technique called *back-phase damping* consists of switching the motor into the reverse direction using the last few pulses of a move.

(iii) The more sophisticated controllers are able to provide so-called *microstepping*. This technique permits the motor shaft to be positioned at places other than the natural stable points of the motor. It is accomplished by proportioning the current in two adjacent motor windings. Instead of operating the winding in the on-off mode, the current in one winding is decreased slightly, but increased in the adjacent winding.

(iv) Complex drive circuits that offer good current build-up without loss at high stepping rates, are used.

Although the advantages of stepping-motor drives in open-loop systems are most obvious, closed-loop applications also exist. A closed-loop stepping motor drive can be analyzed using classical techniques employed for continuous-motion systems. For a detailed account of stepping motors, refer to [52].

3.9 PROGRAMMABLE LOGIC CONTROLLERS

A great deal of what has been said in this book so far about control systems seems exotic: algorithms for radar tracking, drives for rolling mills, filters for extracting information from noisy data, methods for numerical control of machine tools, fluid-temperature control in process plants, etc. Underlying most of these are much more mundane tasks: turning equipment (pumps, conveyor belts, etc.) on and off;

[2] Refer to [96] for detailed description of hardware and software.

opening and closing of valves (pneumatic, hydraulic); checking sensors to be certain they are working; sending alarms when monitored signals go out of range; etc. Process control plants and manufacturing floors share this need for simple, but important, tasks.

These so-called *logic control* functions can be implemented using one of the most ingenious devices ever devised to advance the field of *industrial automation*. So versatile are these devices, that they are employed in the automation of almost every type of industry. The device, of course, is the *programmable controller*, and thousands of these devices go unrecognized in process plants and factory environments—quietly monitoring security, manipulating valves, and controlling machines and automatic production lines.

Industrial applications of logic control are mainly of two types; those in which the control system is entirely based on logic principles, and those that are mainly of a continuous feedback nature and use a 'relatively small' amount of logic in auxiliary functions, such as start-up/shut-down, safety interlocks and overrides, and mode switching. Programmable controllers, originally intended for '100%' logic systems, have, in recent years, added the capability of conventional feedback control; making them very popular—since one controller can now handle, in an integrated way, *all* aspects of operation of a practical system, that includes both types of control problems. General-purpose digital computers could also handle such situations, but they are not as popular as the programmable controllers, for the reasons mentioned below.

In theory, general-purpose computers can be programmed to perform most of the functions of programmable controllers. However, these machines are not built to operate reliably under industrial conditions, where they can be exposed to heat, humidity, corrosive atmosphere, mechanical shock and vibration, electromagnetic noise, unreliable ac power with dropping voltages, voltage spikes, etc. A programmable controller is a special-purpose computer, especially designed for industrial environments. A general-purpose computer is a complex machine, capable of executing several programs or tasks simultaneously, and in any order. By contrast, a programmable controller typically executes its tiny program continuously hundreds of millions of times before being interrupted to introduce a new program. General-purpose computers can be interfaced with external equipment with special circuit cards. In programmable controllers by comparison, the hardware interfaces for connecting the field devices are actually a part of the controller and are easily connected. The software of the controllers is designed for easy use by plant technicians. A programmable controller is thus a special-purpose device for industrial automation applications—requiring logic control functions and simple PID control functions; it cannot compete with conventional computers when it comes to complex control algorithms and/or fast feedback loops, requiring high program execution speeds.

Early devices were called 'programmable logic controllers (PLCs)', and were designed to accept on-off (binary logic) voltage inputs from sensors, switches, relay contacts, etc., and produce on-off voltage outputs to actuate motors, solenoids, control relays, lights, alarms, fans, heaters, and other electrical equipment. As many of today's 'programmable controllers' also accept analog data, perform simple arithmetic operations, and even act as PID (proportional-integral-derivative) process controllers, the word 'logic' and the letter 'L' were dropped from the name long ago. This frequently causes confusion, since the letters 'PC' mean different things to different people; the most common usage of these letters being for 'Personal Computer'. To avoid this confusion, there has been a tendency lately to restore the letter 'L' and revive the designation 'PLC'. We have followed this practice in the book.

Before the era of PLCs, hardwired relay control panels were, in fact, the major type of logic systems, and this historical development explains why the most modern, microprocessor-based PLCs still are usually programmed according to relay ladder diagrams. This feature has been responsible for much of the widespread and rapid acceptance of PLCs; the computer was forced to *learn* the already familiar human language rather than making the humans learn a new computer language. Originally cost-effective for only large-scale systems, small versions of PLCs are now available.

A sequenced but brief presentation of building blocks of a PLC, ladder diagrams, and examples of industrial automation, follows [23–25]. It is not appropriate to discuss here the internal details, performance specifications and programming details for any particular manufacturer's PLC. These aspects are described in every manufacturers' literature.

3.9.1 Logic Controls for Industrial Automation

A definition of *logic controls,* that adequately describes most applications, is that they are controls that work with one-bit binary signals. That is, the system needs only to know that a signal is absent or present; its *exact* size is not important. This definition excludes the field of digital computer control discussed so far in the book. Conventional computer control also uses binary signals (though usually with many bits); the type of application and the analysis methods are quite different for logic controls and conventional computer controls, which is why we make the distinction.

Logic control systems can involve both *combinational* and *sequential* aspects. Combinational aspects are implemented by a proper interconnection of basic logic elements such as AND, OR, NOT, so as to provide a desired output or outputs, when a certain combination of inputs exists. Sequential effects use logic elements together with memory elements (counters, timers, etc.), to ensure that a chain of events occurs in some desired sequence. The present status of outputs depends, both, on the past and present status of inputs.

It is important to be able to distinguish between the nature of variables in a logic control system, and those in a conventional feedback control system. To define the difference, we consider an example that employs both the control schemes.

Figure 3.53 shows a tank with a valve that controls flow of liquid into the tank, and another valve that controls flow out of the tank. A transducer is available to measure the level of the liquid in the tank. Also shown is the block diagram of a feedback control system, whose *objective* is to maintain the level of the liquid in the tank at some preset, or set-point value. We assume that the controller operates according to PID mode of control, to regulate the level against variations induced from external influences. This is a *continuous variable* control system because, both the level and the control valve setting can vary over a range to achieve the desired regulation.

The liquid-level control system is a part of the continuous bottle filling process. Periodically, a bottle comes into position under the outlet valve. The level must be maintained at the set-point *while the outlet valve is opened and the bottle is filled*. This requirement is necessary to assure a constant pressure head during bottle-filling. Figure 3.53 shows a pictorial representation of process hardware for continuous bottle-filling control. The objective is to fill bottles moving on a conveyor, from the constant-head tank. This is a typical logic control problem. We are to implement a control program that will detect the position of a bottle under the tank outlet via a mechanically actuated *limit switch*, stop the feed motor

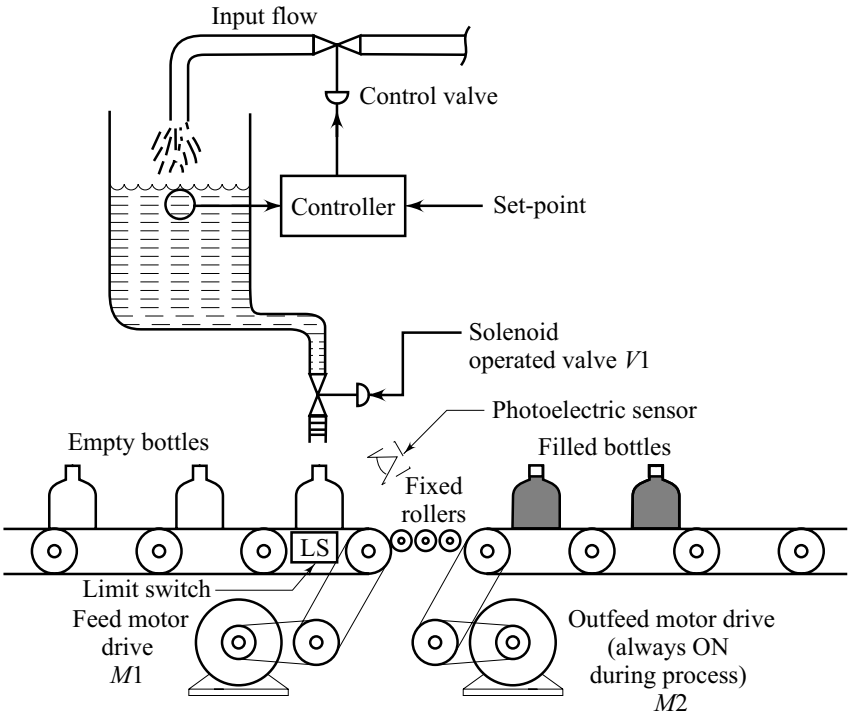


Fig. 3.53 Composite conventional and logic control

$M1$ to stop the feed conveyor, open the solenoid-operated outlet valve $V1$, and then fill the bottle until the photosensor detects the filled position. After the bottle is filled, it will close the valve $V1$, and restart the conveyor to continue to the next bottle. The start and stop pushbuttons (PB) will be included for the outfeed motor, and for the start of the bottle-filling process. Once the start PB is pushed, the outfeed motor $M2$ will be ON until the stop PB is pushed. The feed motor $M1$ is energized once the system starts ($M2$ ON), and is stopped when the limit switch detects the correct position of the bottle.

The sensors used for the logic control problem have characteristics different from those used for the regulator problem. For the regulator problem, the level sensor is an analog device producing analog signal as its output. For the logic control problem, sensors used are binary sensors producing on-off (binary logic) signals. For example, a limit-switch consists of mechanically actuated electrical contacts. The contacts open or close when some object reaches a certain position (i.e., limit), and actuates the switch. Hence, limit-switches are binary sensors. Photoelectric sensors consist, basically, of a source emitting a light beam and a light-sensing detector receiving the beam. The object to be sensed interrupts the beam, thereby making its presence known without physical contact between sensor and object. The filled-bottle state of the product can thus be sensed by a binary photoelectric sensor.

The system of Fig. 3.53 involves solenoid and electric motors as motion actuators. Thus, when the logic controller specifies that 'output valve be opened', it may mean moving a solenoid. This is not done by a simple toggle switch. Instead, one would logically assume that a small switch may be used to energize a relay with contact ratings that can handle the heavy load. Similarly, an on-off voltage signal from the logic controller may actuate a thyristor circuit to run a motor.

3.9.2 Building Blocks of a PLC

The programmable logic controllers are basically computer-based; and therefore, their architecture is very similar to computer architecture. The memory contains the operating system stored in fixed memory (ROM), and the application programs stored in alterable memory (RAM). The Central Processing Unit (CPU) is a microprocessor that coordinates the activities of the PLC system. Figure 3.54 shows basic building blocks of a PLC.

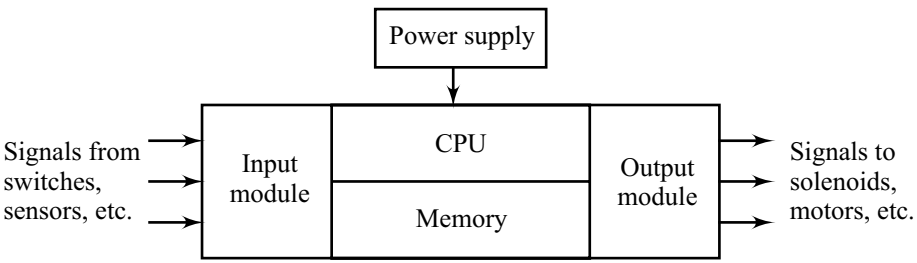


Fig. 3.54 Basic building blocks of a PLC

Input devices such as pushbuttons, sensors, and limit switches are connected to the input interface circuit, called the *input module*. This section gathers information from the outside environment, and sends it to the CPU. Output devices such as solenoids, motor controls, indicator lights and alarms are connected to the output interface circuit, called the *output module*. This section is where the calculation results from the CPU are output to the outside environment. With the control application program (stored within the PLC memory) in execution, the PLC constantly monitors the state of the system through the field input devices; and based on the program logic, it determines the course of action to be carried out at the field output devices. This process of sequentially reading the inputs, executing the program in memory, and updating the outputs is known as *scanning*.

Input Module

Intelligence of an automated system is greatly dependent on the ability of a PLC to read in the signals from various types of input field devices. The most common class of input devices in an automated system is the binary type. These devices provide input signals that are ON/OFF or OPEN/CLOSED. To the input interface circuit, all binary input devices are essentially a switch that is either open or closed, signaling a 1(ON) or 0(OFF). Some of the binary input field devices along with their symbolic representation are listed in Fig. 3.55.

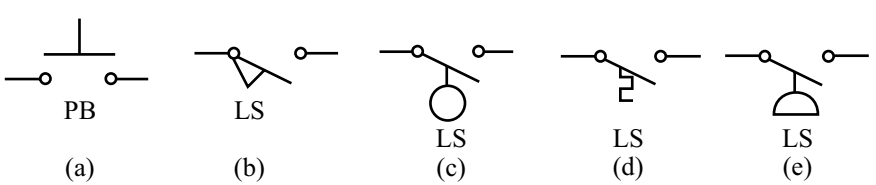


Fig. 3.55 (a) Push-button (b) Position limit switch (c) Level limit switch (d) Temperature limit switch (e) Pressure limit switch

As mentioned earlier, a switch is a symbolic representation of the field input device, interfaced to the input module of the PLC. The device may be a manually operated pushbutton, mechanically actuated limit switch (the contacts open/close when some object reaches a certain position and actuates the switch), proximity switch (device based on inductive/capacitive/magnetic effect which, with appropriate electronics, can sense the presence of an object without a physical contact with the object), photoelectric sensor, level sensor, temperature sensor, shaft encoder, etc. The main purpose of the input module is to condition the various signals, received from the field devices, to produce an output to be sensed by the CPU. The signal conditioning involves converting power-level signals from field devices to logic-level signals acceptable to the CPU, and providing *electrical isolation* so that there is no electrical connection between the field device (power) and the controller (logic). The coupling between the power and the logic sections is normally provided by an optical coupler.

During our discussion on PLC programming, it will be helpful if we keep in mind the relationship between the interface signals (ON/OFF) and their mapping and addressing used in the program. When in operation, if an input signal is energized (ON), the input interface circuit senses the field device's supplied voltage and converts it to a logic-level signal acceptable to the CPU, to indicate the status of the device. The field status information provided to the standard input module is placed into the *input table* in memory through PLC instructions. The I/O address assignment document of the PLC manufacturer identifies each field device by an address. During scanning, the PLC reads the status of all field input devices, and places this information at the corresponding address locations.

Output Module

An automation system is incomplete without means for interface to the field output devices. The *output module* provides connections between the CPU and output field devices. The output module receives from the CPU logic-level signals (1 or 0).

The main purpose of the output interface circuit is to condition the signals received from the CPU, to produce outputs to actuate the output field devices. The signal conditioning circuit consists, primarily, of the logic and power sections, coupled by an isolation circuit. The output interface can be thought of as a simple switch through which power can be provided to control the output device.

During normal operation, the CPU sends to the *output table*, at predefined address locations, the output status according to the logic program. If the status is 1, ON signal will be passed through the isolation circuit, which, in turn, will switch the voltage to the field device through the power section of the module.

The power section of the output module may be transistor based, triac based, or simply, relay 'contact based' circuit. The relay circuit output interface allows the output devices to be switched by NO (normally open) or NC (normally closed) relay contact. When the processor sends the status (1 or 0) to the module (through output table) during the output update, the state of the contact will change. If a 1 is sent to the module from the processor, a normally open contact will close, and a normally closed contact will change to an open position. If a 0 is sent, no change occurs to the normal state of the contacts. The contact output can be used to switch either ac or dc loads; switching small currents at low voltages. High power contact outputs are also available for switching of high currents.

Some of the output field devices, along with their symbolic representation, are given in Fig. 3.56.

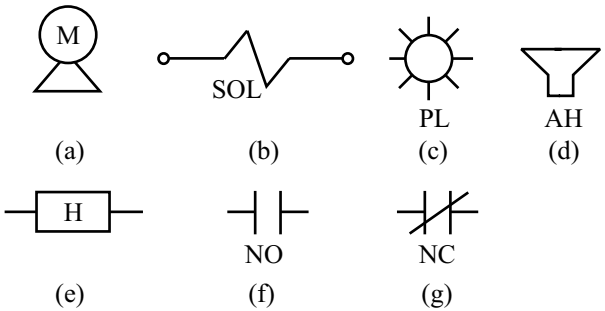


Fig. 3.56 (a) Motor (b) Solenoid (c) Pilot lamp (d) Alarm horn (e) Heater (f) Relay contact (Normally Open) (g) Relay contact (Normally Closed)

I/O Rack Enclosures

Once we have the CPU programmed, we get information in and out of the PLC through the use of input and output modules. The input module terminals receive signals from switches, and other input information devices. The output module terminals provide output voltages to energize motors and valves, operate indicating devices, and so on.

For small PLC systems, the input and output terminals may be included on the same frame as the CPU. In large systems, the input and output modules are separate units; modules are placed in groups on racks, and the racks are connected to the CPU via appropriate connector cables.

Generally speaking, there are three categories of rack enclosures—the *master rack*, the *local rack*, and the *remote rack*. A master rack refers to the enclosure containing the CPU module. This rack may, or may not, have slots available for the insertion of I/O modules. The larger the PLC system, the less likely that the master rack will have I/O housing capability or space. A local rack is an enclosure which is placed in the same location or area where the master rack is housed. If a master rack contains I/O, it can also be considered a local rack. In general, a local rack contains a local I/O processor which receives and sends data to and from the CPU.

As the name implies, remote racks are enclosures containing I/O modules located far away from the CPU. A remote rack contains an I/O processor which communicates I/O information just like the local rack.

Timers/Counters

Timers and counters play an important part in many industrial automation systems. The timers are used to initiate events at defined intervals. The counters, on the other hand, are used to count the occurrences of any defined event.

Basically the operation of both the timer and the counter is same, as a timer operates like a counter. The counter shown in Fig. 3.57a, counts down from *set value* when its execution condition (count input) goes from OFF to ON. When the value reaches zero, the counter contact point is turned ON. It is reset with a reset input. The set value is decided by the programmer, and stored in the internal register of the counter through control program instructions. The count input signal may refer to any event which may occur randomly.

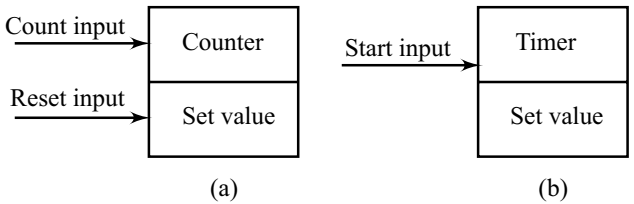


Fig. 3.57 (a) Counter (b) Timer

When a count input signal occurs at fixed frequency, i.e., after every fixed interval of time, the counter performs as a timer. Now 10 pulses, i.e., counts, will mean an elapsed time of 5 seconds, if the signal is occurring after a regular interval of 0.5 seconds. The timer, shown in Fig. 3.57b, is activated when its execution condition goes ON and starts decreasing from the set value. When the value reaches zero, the timer contact point is turned ON. It is reset to set value when the execution condition goes OFF.

Memory Map

It is unlikely that two different PLCs will have identical memory maps, but a generalization of memory organization is still valid in the light of the fact that all PLCs have similar storage requirements. In general, all PLCs must have memory allocated for the four items described below.

Executive The executive software is a permanently stored collection of programs that are considered a part of the system itself. These programs direct system activities such as execution of the control program, communication with peripheral devices, and other housekeeping activities. The executive area of memory is not accessible to the user.

Scratch Pad It is a temporary storage used by the CPU to store a relatively small amount of data for interim calculations or control.

Data Table This area stores any data associated with the control program, such as timer/counter set values, and any other stored constants or variables that are used by the control program. This section also retains the status information of the system inputs once they have been read, and the system outputs once they have been set by the control program.

User Program This area provides storage for any programmed instructions entered by the user. The control program is stored in this area.

The Data Table and the User Program areas are accessible and are required by the user for control application. The Executive and Scratch Pad areas together are normally referred to as 'system memory', and Data Table and User Program areas together are labeled as 'application memory'.

Data Table Memory Area

The data table area of the PLC's application memory is composed of several sections described below.

Input Table The input table is an array of bits that stores the status of discrete inputs which are connected to input interface circuit. The maximum number of bits in the input table is equal to the maximum number of field inputs that can be connected to the PLC. For instance, a controller with 128

inputs would require an input table of 128 bits. If the PLC system has 8 input modules, each with 16 terminal points, then the input table in PLC memory (assuming 16 bit word length) will look like that in Fig. 3.58.

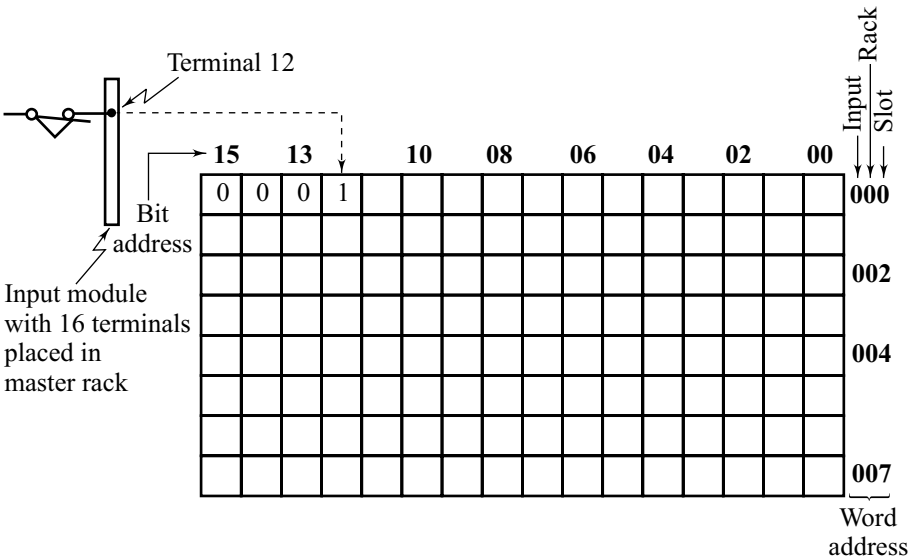


Fig. 3.58 An illustrative input table

Each terminal point on each of the input modules will have an *address* by which it is referenced. This address will be a pointer to a bit in the input table. Thus, each connected input has a bit in the input table that corresponds exactly to the terminal to which the input is connected. The address of the input device can be interpreted as *word* location in the input table corresponding to the input module, and *bit* location in the word corresponding to the terminal of the input module, to which the device is connected.

Several factors determine the address of the word location of each module. The type of module, input or output, determines the first number in the address from left to right (say, **0** for input, and **1** for output). The next two address numbers are determined by the rack number and the slot location where the module is placed. Figure 3.58 graphically illustrates a mapping of the input table, and the modules placed in rack **0** (master rack). Note that the numbers associated with address assignment depend on the PLC model used. These addresses can be represented in octal, decimal, or hexadecimal. We have used decimal numbers.

The limit switch connected to the input interface (refer to Fig. 3.58) has an address of **00012** for its corresponding bit in the input table. This address comes from the word location **000** and the bit number **12**; which are related to the rack position where the module is installed, and the module's terminal connected to the field device, respectively. If the limit switch is ON (closed), the corresponding bit **00012** will be 1; if the limit switch is OFF (open), its corresponding bit will be 0.

During PLC operation, the processor will read the status of each input in the input modules, and then place this value (1 or 0) in the corresponding address in the input table. The input table is constantly changing—to reflect the changes in the field devices connected to the input modules. These changes in the input table take place during the reading part of the processor scan.

Output Table The output table is an array of bits that controls the status of output devices, which are connected to the output interface circuit. The maximum number of bits available in the output table is equal to the maximum number of output field devices that can be interfaced to the PLC. For instance, a controller with a maximum number of 128 outputs would require an output table of 128 bits.

Each connected output has a bit in the output table that corresponds exactly to the terminal to which the output is connected. The bits in the output table are controlled (ON/OFF) by the processor, as it interprets the control program logic. If a bit in the table is tuned ON (logic 1), then the connected output is switched ON. If a bit is cleared or turned OFF (logic 0), the output is switched OFF. Remember that the turning ON or OFF, of the field devices occurs during the update of outputs after the *end* of the scan.

Storage Area This section of the data table may be subdivided in two parts consisting of a *work bit storage area* and a *word storage area*. The purpose of this data table section is to store data that can change, whether it is a bit or a word (16 bits). Work bits are *internal outputs* which are normally used to provide interlocking logic. The internal outputs do not directly control output field devices. When the processor evaluates the control program, and any of these outputs is energized (logic 1), then this internal output, in conjunction with other internal and/or real signals from field devices, forms an interlocking logic sequence that drives an output device or another internal output.

The outputs of timers and counters are used as internal outputs which are generated after a time interval has expired, or a count has reached a set value.

Assume that the timer/counter table in storage area has 512 points. Address assignment for these points depends on the PLC model used. We will use **TIM/CNT000** to **TIM/CNT512** as the addresses of these points. The word storage area will store the set values of timers/counters.

In our application examples given in the next subsection, we shall use word addresses **000** to **007** for input table, and addresses **100** to **107** for output table. The input devices will be labeled with numbers such as **00000**,..., **00015**, and output devices with numbers such as **10000**, ..., **10015**.

We shall use word addresses **010** to **017** for internal outputs. Examples of typical work bits (internal outputs) are **01000**, ..., **01015**. **TIM/CNT000** to **TIM/CNT512** are the typical addresses of timer/counter points.

3.9.3 Ladder Diagrams

Although specialized functions are useful in certain situations, most logic control systems may be implemented with the three basic logic functions AND, OR, and NOT. These functions are used either singly or in combinations, to form instructions that will determine if an output field device is to be switched ON or OFF. The most widely used language for implementing these instructions are *ladder diagrams*. Ladder diagrams are also called *contact symbology*, since the instructions, as we shall see, are relay-equivalent contact symbols shown in Figs 3.56f and 3.56g.

An AND device may have any number of inputs and one output. To turn the output ON, *all* the inputs must be ON. This function is most easily visualized in terms of switch arrangement of Fig. 3.59a, and timing chart of Fig. 3.59b. The corresponding ladder diagram is given in Fig. 3.59c. Figure 3.59d gives the *Boolean algebra* expression for the two-input AND, read as "*A* AND *B* equals *C*".

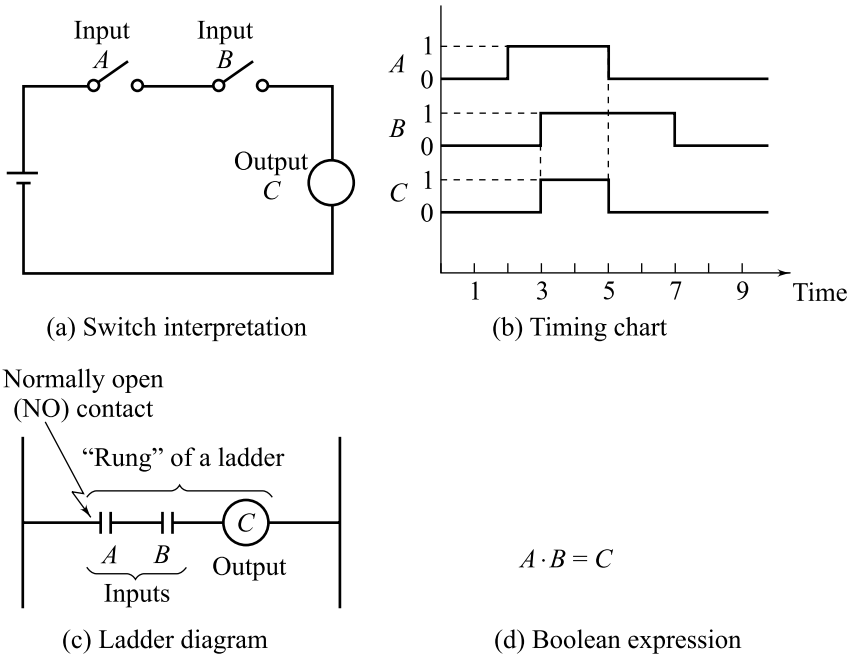


Fig. 3.59 The AND function

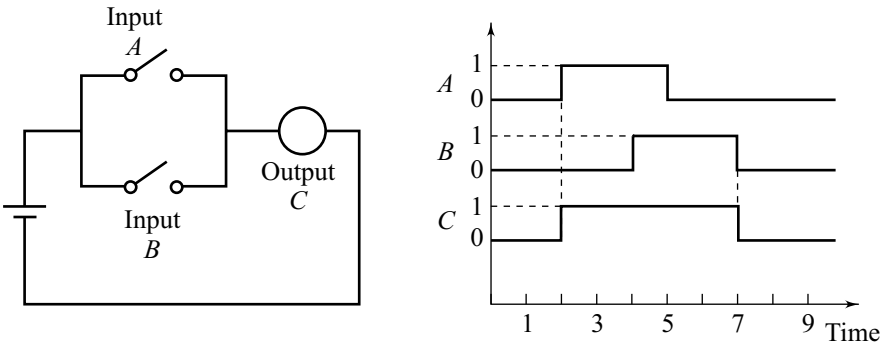
The timing chart in Fig. 3.59b is simply a series of graphs, each representing a logic variable, in which the horizontal axis is time and the vertical axis is logic state, that is, 0 or 1. The graphs are placed so that their time-axis are synchronized; in this way, a vertical line at any point on the graph describes a point in time, and all input and output variables can be evaluated at that point. The graph of the output variable is determined by the structure of the logic system and, of course, the pattern of the input.

The input contacts in Fig. 3.59c are normally open (NO) contacts (Do not confuse this symbol with the familiar electrical symbol for capacitors). If the status of the input *A* is '1', the contact *A* in ladder diagram will close, and allow current to flow through the contact. If the status of the input *A* is '0', the contact will remain open, and not allow current to flow through the contact.

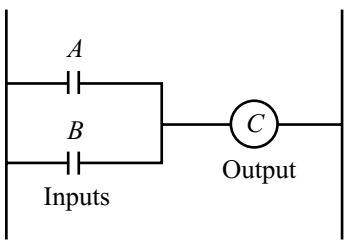
The ladder diagram of Fig. 3.59c can be thought of as a circuit having many inputs. A circuit is known as the 'rung' of the ladder. A complete PLC ladder diagram consists of several rungs; a rung controls an output field device either through an output module or an internal output. The input to a rung can be logic commands from input modules, or from output modules connected to field devices, or from internal outputs.

Figure 3.60 gives similar details for logical OR operation and should be self-explanatory. The Boolean expression is read as "*A* OR *B* equals *C*".

The contact in Fig. 3.61 in normally closed (NC) contact. If the status of the input *A* is '0', the contact will remain closed, thus allowing current to flow through the contact. If the status of the input *A* is '1', the contact will open and *not* allow current to flow through the contact. This symbol permits the use of logic NOT operator. The Boolean expression is read as "NOT *A* equals *B*"; the overbar is used in general, for applying NOT function.



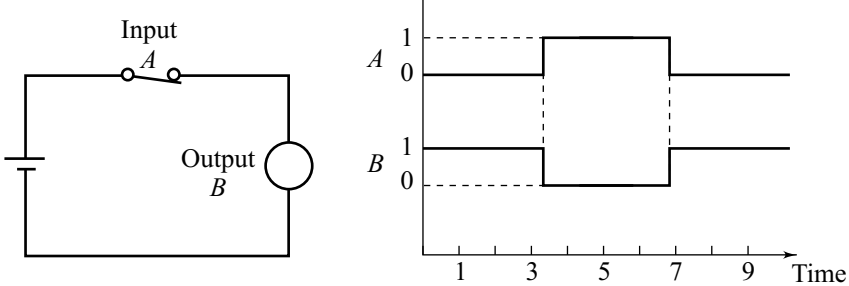
(a) Switch interpretation (b) Timing chart



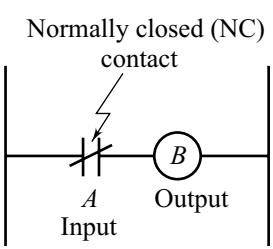
$A + B = C$

(c) Ladder diagram (d) Boolean expression

Fig. 3.60 The OR function



(a) Switch interpretation (b) Timing chart



$\overline{A} = B$

(c) Ladder diagram (d) Boolean expression

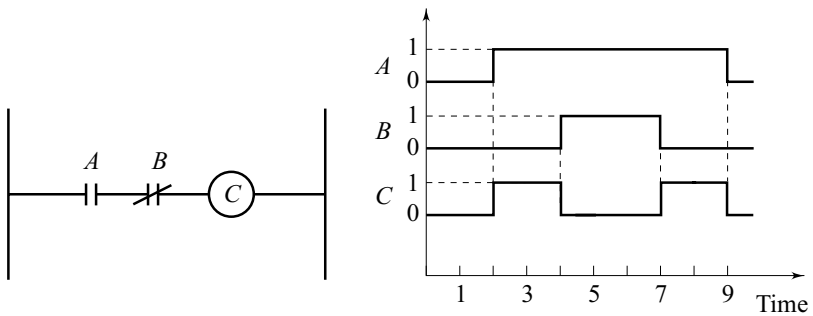
Fig. 3.61 The NOT function

Consider the logic system

$$A \cdot \overline{B} = C$$

read as "*A* AND NOT *B* equals *C*".

The ladder diagram and timing chart for this system are given in Fig. 3.62.



(a) Ladder diagram (b) Timing chart

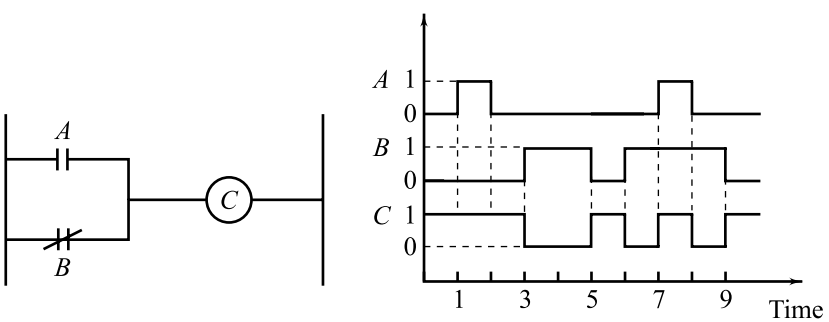
Fig. 3.62 AND NOT function

Consider now the logic system

$$A + \overline{B} = C$$

read as "*A* OR NOT *B* equals *C*".

The ladder diagram and timing chart for this system are given in Fig. 3.63.



(a) Ladder diagram (b) Timing chart

Fig. 3.63 OR NOT function

Example 3.5 *Start/Stop Pushbutton System*

Most large industrial machines are turned on and off by means of separate spring pushbuttons for start and stop. This has safety implications in that the stop pushbutton can be given priority to shut down the machine in an emergency, regardless of the status of the start pushbutton. The start/stop pushbutton system is a logic control system with three variables, each of which can take on two, and only two values, or states. These variables and their states are defined as follows:

Assume that **000** is the word address of the input module, and **100** is the word address of the output module of a PLC. Each module is assumed to have 16 terminals: **00** to **15**. The start pushbutton is connected to terminal **00**, and stop pushbutton is connected to terminal **01** of the input module **000**; and the signal from terminal **00** of the output module **100** controls the machine. The system variables may, therefore, be designated as **00000**, **00001**, and **10000**.

The bit **00000** of the input table in PLC memory is 1 when the start pushbutton is pressed, and is 0 when start pushbutton is released. The bit **00001** of the input table is 1 when the stop pushbutton is pressed, and is 0 when the stop pushbutton is released. The bit **10000** of the output table is 1 when the machine is running, and 0 when the machine is not running.

The logic system has three input variables and one output variable. There appears to be a contradiction, but the statement is true. The variable **10000**, representing the start of the machine, is both an input variable and an output variable. This makes sense because the current state of the machine may affect the future state.

Figure 3.64a illustrates a simple situation in which pushbutton **00000** turns ON machine **10000**. This of course would not be satisfactory pushbutton switch because as soon as pushbutton is released, the machine comes to OFF state. Figure 3.64b adds an OR condition that keeps the machine ON if it is already ON. This is an improvement, but now there is a new problem; once turned ON, the output will

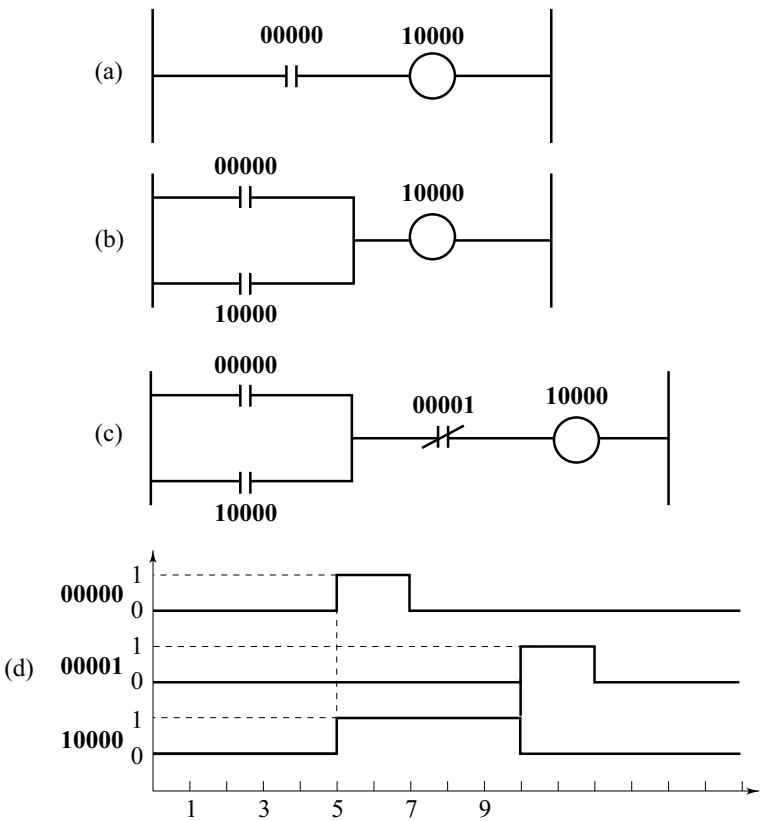


Fig. 3.64 Start/Stop pushbutton system

never be turned OFF by the logic system. We add another input switch in Fig. 3.64c. Note that **00001** contact is normally closed. Input **00000** turns ON output **10000**; input **10000** keeps output **10000** ON until input **00001** turns it OFF.

The timing chart of the logic system is shown in Fig. 3.64d.

Example 3.6 *Automatic Weigh Station*

Consider the conveyer system of Fig. 3.65, in which an automatic weigh station activates a trap door, or diverter, in the event an overweight item passes over the weigh station. The trap door opens immediately, and remains open for 4 seconds to allow sufficient time for the item to drop through to the overweight track (For the system to work properly, it is necessary for successive items on the conveyor to be separated by distances of at least 5 seconds or so).

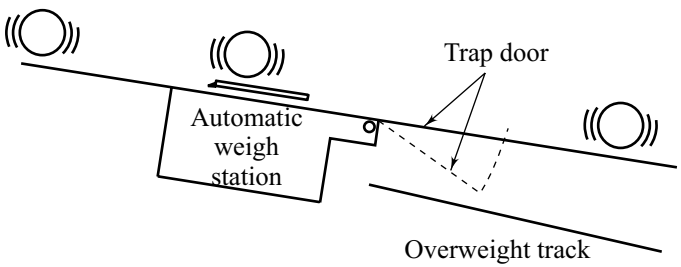
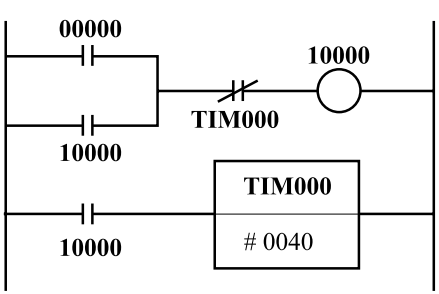


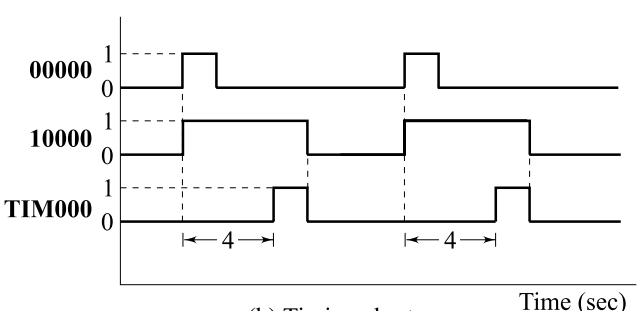
Fig. 3.65 Automatic weigh station on a conveyor system

The variables of the logic system are defined as follows (refer to Fig. 3.66a). **00000** represents the pressure switch connected to terminal **00** of input module **000**. It senses the overweight item on the automatic weigh station. The bit **00000** in the input table latches 1 for the overweight item. **10000** represents a solenoid connected to terminal **00** of the output module **100**, which pushes the trap door open. When the bit **10000** in the output table latches 1, the trap door is open and when the bit is 0, the trap door is closed.

For a 4 sec delay, the set value 0040 is stored in word storage area of memory. Countdown of this number to 0000 will give an elapsed time of 4 seconds in our PLC system, wherein we assume that the timer counts time-based intervals of 0.1 sec (40 counts will mean an elapsed time of 4 sec).



(a) Ladder diagram of logic system



(b) Timing chart

Fig. 3.66

The timer **TIM000** is activated when its execution condition goes ON and starts decreasing from the set value. When the value reaches 0000, the timer contact point is tuned ON. It is reset to set value when the execution condition goes OFF. The timer contact point works as internal work bit (A work bit/internal output is for use of program only; it does not turn ON/OFF external field devices).

It is obvious from the ladder diagram of Fig. 3.66a that once an overweight item is detected, the trap door opens; it remains open for 4 sec, and thereafter it closes. Figure 3.66b shows the timing chart for the logic system.

Example 3.7 *Packaging Line Control*

Figure 3.67 shows a pictorial representation of process hardware of a conveyor system used for automatic packaging. The objective is to fill boxes moving on Box Conveyor from the Part Conveyor System.

Input devices	:	Box proximity sensor	**00003**
		Part proximity sensor	**00002**
		Stop pushbutton	**00001**
		Start pushbutton	**00000**
Output devices	:	Part conveyor motor	**10000**
		Box conveyor motor	**10001**

Pushbutton **00000** starts the packaging line control which stops when pushbutton **00001** is pressed. Let us generate a work bit (internal output) **01000,** which depends on the state of both the pushbuttons (refer to Fig. 3.68). The work bit is 1 when packaging line control is ON, and it is 0 when packaging line control is OFF. The work bit is useful where the same combination of input signals appears repeatedly in the ladder diagram. We will shortly see that work bit **01000** is helpful in simplifying the ladder diagram.

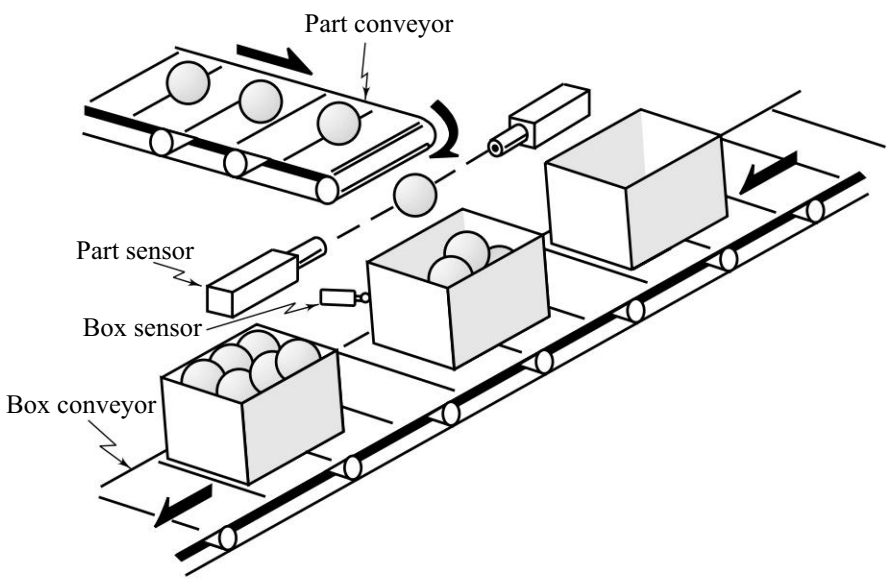


Fig. 3.67 Packaging line control

The event sequences are as follows.

(i) Box proximity sensor in '0' state; box conveyor will start, keeping part conveyor standstill.

(ii) Box proximity sensor in '1' state; box conveyor will stop, signaling the start of the part conveyor.

(iii) Part proximity sensor in '1' state and box proximity sensor in '1' state; this state signals the execution of the counter.

A counter counts down how many times input is turned ON. It counts down from set value, when its execution condition (count input) goes from OFF to ON. It decrements one count every time the input signal goes from OFF to ON. When the value reaches 0000, the counter contact point is turned ON. It is reset with a reset input. The counter contact point works as internal work bit.

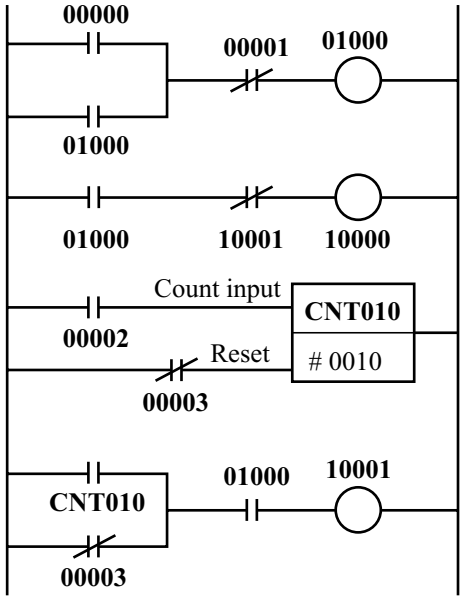


Fig. 3.68 Ladder diagram of logic system

The execution signal for our counter is generated by part proximity sensor **00002** and reset signal is generated by box proximity sensor **00003**. For counting to occur, we need both the part proximity sensor and the box proximity sensor in '1' state.

(iv) Assume that set value 0010 of count has been loaded in word storage area of memory. When the value reaches 0000, the work bit **CNT010** takes '1' state, which starts the box conveyor.

(v) Box proximity switch gets deactivated and count stops. The timing chart is shown in Fig. 3.69.

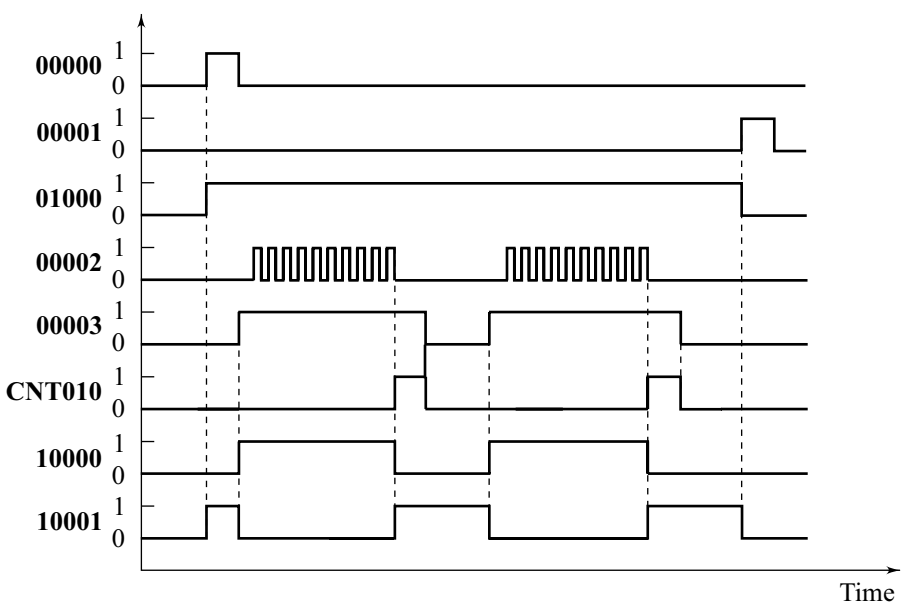


Fig. 3.69 Timing chart

Example 3.8 *Automatic Bottle Filling Control*

In this application (shown in Fig. 3.53), we are to implement a control program that will detect the position of a bottle via a limit switch, wait for 5 seconds, and then fill the bottle until the photosensor detects the filled position. After the bottle is filled, it will wait for 7 seconds to continue to the next bottle. The start and stop circuits will also be included for the outfeed motor and for the start of the process. The I/O assignment follows:

Start-process pushbutton	: **00000**
Stop-process pushbutton	: **00001**
Limit switch (position detect)	: **00002**
Photosensor (level detect)	: **00003**
Feed-motor drive	: **10000**
Outfeed-motor drive	: **10001**
Solenoid control	: **10002**
The work bits may be assigned as follows	:
Timer for 5 sec delay	: **TIM000**
Timer for 7 sec delay	: **TIM001**

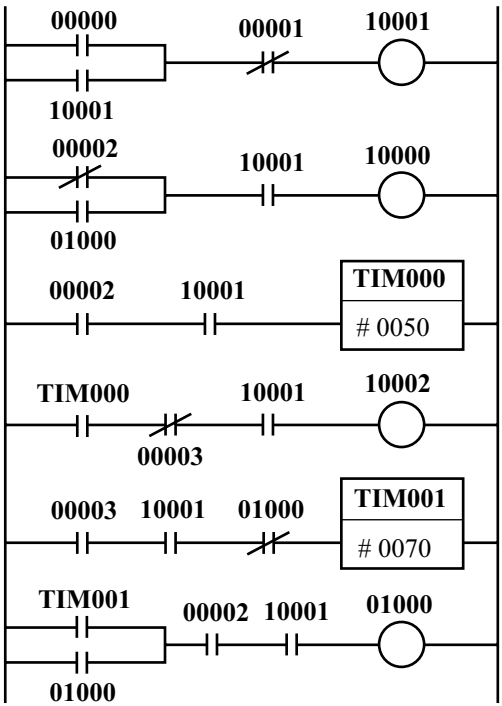


Fig. 3.70 Ladder diagram of logic system

Ladder diagram for automatic bottle filling controller is shown in Fig. 3.70. Rung 1 provides a start/stop latch for the system. The outfeed motor is always ON during process operation. Rung 2 drives the feed conveyor until a bottle is in position. Rung 3 introduces a time delay of 5 sec. The work bit **TIM000** turns ON the valve solenoid (Rung 4). Rung 5 introduces a time delay of 7 sec after detecting bottle filled position. Rung 6 is necessary to detect that the bottle is full and 7 sec waiting period is over, and to restart the conveyor to move the bottle out (**01000** is a work bit).

3.9.4 PLC Programming

PLC programming methods vary from manufacturer to manufacturer, but the basic ladder diagram approach appears to be the standard throughout the industry. A CRT connected to the CPU of the PLC through a peripheral port, is perhaps the most common device used for programming the controller. A CRT is a self-contained, video display unit with a keyboard and the necessary electronics to communicate with the CPU. The graphic display on the CRT screen appears as a ladder diagram. This ladder diagram takes form while the programmer builds it up using the keyboard. The keys themselves have symbols such as: ⊣⊢; ⊣/⊢, which are interpreted exactly as explained earlier in this section.

A limitation of CRT is that the device is not interchangeable from one manufacturer's PLC family to another. However, with the increasing number of products in the manufacturers' product lines and user standardization of products, these programming devices may be a good choice, especially if the user has standardized with one brand of PLCs.

At the other end of the spectrum of PLC programming devices is a Programming Console for programming small PLCs (up to 128 I/O). Physically, these devices resemble handheld calculators but have a larger display and somewhat different keyboard. The Programming Console uses keys with two- or three-letter abbreviations, to write programs that bear some semblance to computer coding. The display at the top of the Console exhibits the PLC instruction located in the User Program memory area. As with CRTs, Programming Consoles are designed so that they are compatible with controllers of the product family.

Common usage of a Personal Computer (PC) in our daily lives has led to a new breed of PLC programming devices. Due to the PC's general-purpose architecture and *de facto* standard operating system, PLC manufacturers provide the necessary software to implement the ladder diagram entry, editing and real-time monitoring of the PLC's control program. PCs will soon be the programming device of choice, not so much because of its PLC programming capabilities, but because these PCs may already be present at the location where the user may be performing the programming.

The programming device is connected to the CPU through a peripheral port. After the CPU has been programmed, the programming device is no longer required for CPU and process operation; it can be disconnected and removed. Therefore, we may need only one programming device for a number of operational PLCs. The programming device may be moved about in the plant as needed.

Programming details for any manufacturer's PLC are not included here. These aspects are described in every manufacturers' literature.

3.9.5 Scope of Applications

Programmable logic controllers are available in many sizes, covering a wide spectrum of capability. On the low end are 'relay replacers' with minimum I/O and memory capability. At the high end are large supervisory controllers, which play an important role in distributed control systems—by performing a variety of control and data acquisition functions. In between these two extremes are multifunctional controllers with communication capability which allow integration with various peripherals, and expansion capability which allows the product to grow, as the application requirements change.

PLCs with analog input modules and analog output modules, for driving analog valves and actuators using the PID control algorithms, are being used in process industries.

Large PLCs are used for complicated control tasks that require analog control, data acquisition, data manipulation, numerical computations and reporting. The enhanced capabilities of these controllers allow them to be used effectively in applications where LAN (local area network) may be required.

Some PLCs offer the ability to program in other languages beside the conventional ladder language. An example is the BASIC programming language. Other manufacturers use what is called 'Boolean Mnemonics', to program a controller. The Boolean language is a method used to enter and explain the control logic which follows Boolean algebra.

REVIEW EXAMPLES

Review Example 3.1

We have so far used the z-transform technique to obtain system response at the sampling instants only. In a large number of cases, this information is adequate because, if the sampling theorem is satisfied, then the output will not vary too much between any two consecutive sampling instants. In certain cases, however, we may need to find the response between consecutive sampling instants. Often, for example, hidden oscillations, that may or may not decay with time, are present. We can compute *ripple* (response between sampling instants) by introducing a fictitious delay of ΔT seconds at the output of the system, where $0 \le \Delta \le 1$ and T is the sampling period. By varying Δ between 0 and 1, the output $y(t)$ at $t = kT - \Delta T$ (where $k = 1, 2, ...$) may be obtained.

In Example 3.1, unit-step response of the sampled-data system of Fig. 3.13a was obtained. The sampling period $T = 1$ sec, and the output at the sampling instants is given by

$$y(T) = 0.3679,\ y(2T) = 1,\ y(3T) = 1.3996,\ y(4T) = 1.3996,$$
$$y(5T) = 1.1469,\ y(6T) = 0.8944,\ y(7T) = 0.8015,\ ...$$

We now introduce at the output a fictitious delay of ΔT seconds with $\Delta = 0.5$ as shown in Fig. 3.71. The output $\hat{y}(kT)$ can be determined as follows:

$$\hat{Y}(z) = \mathcal{Z}[G_{h0}(s)\, e^{-\Delta Ts}]\, E(z)$$
$$E(z) = R(z) - Y(z)$$
$$Y(z) = \mathcal{Z}[G_{h0}(s)G(s)]\, E(z)$$

Therefore,

$$E(z) = \frac{R(z)}{1 + \mathcal{Z}[G_{h0}(s)G(s)]}$$

$$\hat{Y}(z) = \frac{\mathcal{Z}[G_{h0}(s)\, G(s)e^{-\Delta Ts}]}{1 + \mathcal{Z}[G_{h0}(s)\, G(s)]}\, R(z) \tag{3.73}$$

Referring to Example 3.1, we have

$$\mathcal{Z}[G_{h0}(s)G(s)] = \frac{0.3679\, z + 0.2642}{z^2 - 1.3679\, z + 0.3679}$$

Referring to Table 3.1, we get

$$\mathcal{Z}[G_{h0}(s)G(s)e^{-\Delta Ts}] = (1 - z^{-1})\left[\frac{1}{(z-1)^2} - \frac{0.5}{(z-1)} + \frac{0.6065}{(z - 0.3679)}\right]$$
$$= \frac{0.1065\, z^2 + 0.4709\, z + 0.0547}{z^3 - 1.3679\, z^2 + 0.3679\, z}$$

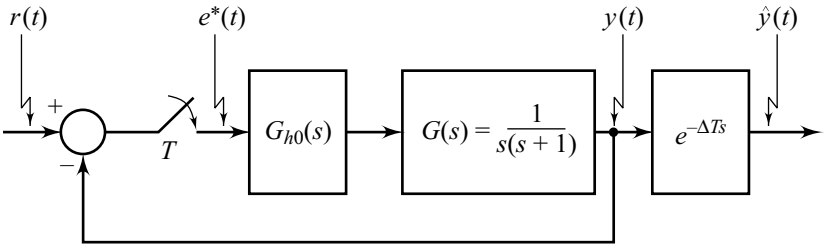


Fig. 3.71 Closed-loop discrete-time system for Review Example 3.1

Referring to Eqn. (3.73) and noting that $R(z) = z/(z-1)$, we have

$$\hat{Y}(z) = \frac{0.1065\,z^{-1} + 0.4709\,z^{-2} + 0.0547\,z^{-3}}{1 - 2\,z^{-1} + 1.6321\,z^{-2} - 0.6321\,z^{-3}}$$

This equation can be expanded into an infinite series in z^{-1}:

$$\hat{Y}(z) = 0.1065\,z^{-1} + 0.6839\,z^{-2} + 1.2487\,z^{-3} + 1.4485\,z^{-4} + 1.2913\,z^{-5} + 1.0078\,z^{-6} + 0.8236\,z^{-7} + 0.8187\,z^{-8} + \cdots$$

Therefore,

$$\hat{y}(T) = y(0.5T) = 0.1065;\ \hat{y}(2T) = y(1.5T) = 0.6839;\ \hat{y}(3T) = y(2.5T) = 1.2487;$$
$$\hat{y}(4T) = y(3.5T) = 1.4485;\ \hat{y}(5T) = y(4.5T) = 1.2913;\ \hat{y}(6T) = y(5.5T) = 1.0078;$$
$$\hat{y}(7T) = y(6.5T) = 0.8236;\ \hat{y}(8T) = y(7.5T) = 0.8187;\ \cdots$$

These values give the response at the midpoints between pairs of consecutive sampling points. Note that by varying the value of Δ between 0 and 1, it is possible to find the response at any point between two consecutive sampling points.

Review Example 3.2

Reconsider the sampled-data system of Example 3.2 (Fig. 3.15). The characteristic polynomial of the system is

$$\Delta(z) = 1 + G_{h0}G(z) = z^2 - az + b$$

where

$$a = 1 + e^{-2T} - 0.5K\,(T + 0.5e^{-2T} - 0.5)$$
$$b = e^{-2T} + 0.5K\,(0.5 - 0.5e^{-2T} - Te^{-2T}).$$

Employing stability constraints (2.75a) of the Jury stability criterion, we get

(i) $\Delta(1) = 1 - a + b > 0$;
(ii) $\Delta(-1) = 1 + a + b > 0$; and
(iii) $b < 1$.

Thus, for a stable system,

$$1 + a + b > 0 \tag{3.74a}$$
$$1 - b > 0 \tag{3.74b}$$
$$1 - a + b > 0 \tag{3.74c}$$

Substituting a and b into (3.74a) and solving for K yields

$$K < \frac{4}{T - (1 - e^{-2T})/(1 + e^{-2T})}$$

From (3.74b), we get

$$K < \frac{2}{0.5 - (Te^{-2T})/(1 - e^{-2T})}$$

and from (3.74c),

$$e^{-2T} < 1$$

Table 3.4 indicates the values of K obtained for various values of T from (3.74a) and (3.74b). A sketch of T *versus* the boundary value of K for a stable system is shown in Fig. 3.72. Note that $\lim_{T \to 0} K = \infty$.

Table 3.4 Stability requirements for Review Example 3.2

T	*Inequality (3.74a)*	*Inequality (3.74b)*	*Value of K for stability*
0.01	$K < 3986844$	$K < 401.4$	$K < 401.4$
0.1	$K < 12042.7$	$K < 41.378$	$K < 41.378$
1.0	$K < 16.778$	$K < 5.8228$	$K < 5.8228$
1.0	$K < 4$	$K < 4$	$K < 4$

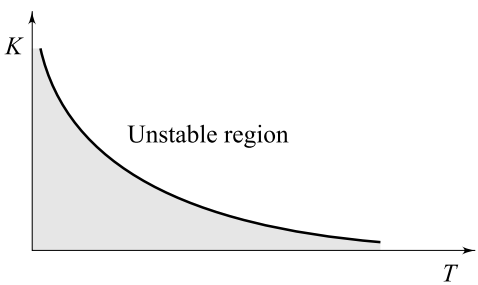


Fig. 3.72 T versus boundary value of K for stability

Review Example 3.3

Consider a digital control function

$$D(z) = \frac{N(z)}{\Delta(z)} = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n} \tag{3.75}$$

For direct realization of this control function, the computer must store the parameters α_1, α_2, ..., α_n, β_0, β_1, ..., β_n (refer to Fig. 3.18). If the machine uses fixed-point arithmetic, the parameter values will be rounded off to the accuracy of the machine. Thus, a program designed to realize Eqn. (3.75) actually realizes[3]

[3] In addition to parameter quantization error, accuracy of a realization is affected by the error due to quantization of the input signal, and the error due to accumulation of round-off errors in arithmetic operations.

$$\hat{D}(z) = \frac{(\beta_0 + \delta\beta_0)z^n + (\beta_1 + \delta\beta_1)z^{n-1} + \cdots + (\beta_n + \delta\beta_n)}{z^n + (\alpha_1 + \delta\alpha_1)z^{n-1} + \cdots + (\alpha_n + \delta\alpha_n)}$$

To study the effects of this realization on the dynamic response, we consider the characteristic equation and determine how a particular root changes when a particular parameter undergoes change.

$$\Delta(z, \alpha_1, \alpha_2, ..., \alpha_n) = z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n = 0 \tag{3.76}$$

is the characteristic equation with roots $\lambda_1, \lambda_2, \ldots, \lambda_n$:

$$\Delta(z, \alpha_1, \alpha_2, ..., \alpha_n) = (z - \lambda_1)(z - \lambda_2) \cdots (z - \lambda_n) \tag{3.77}$$

We shall consider the effect of parameter α_j on the root λ_k. By definition,

$$\Delta(\lambda_k, \alpha_j) = 0$$

If α_j is changed to $\alpha_j + \delta\alpha_j$, then λ_k also changes and the new polynomial is

$$\Delta(\lambda_k + \delta\lambda_k, \alpha_j + \delta\alpha_j) = \Delta(\lambda_k, \alpha_j) + \left.\frac{\partial\Delta}{\partial z}\right|_{z=\lambda_k} \delta\lambda_k + \left.\frac{\partial\Delta}{\partial\alpha_j}\right|_{z=\lambda_k} \delta\alpha_j + \cdots = 0$$

Neglecting the higher-order terms, we obtain

$$\delta\lambda_k = -\left(\frac{\partial\Delta/\partial\alpha_j}{\partial\Delta/\partial z}\right)\Bigg|_{z=\lambda_k} \delta\alpha_j \tag{3.78}$$

From Eqn. (3.76),

$$\left.\frac{\partial\Delta}{\partial\alpha_j}\right|_{z=\lambda_k} = \lambda_k^{n-j}$$

and from Eqn. (3.77),

$$\left.\frac{\partial\Delta}{\partial z}\right|_{z=\lambda_k} = \prod_{i=k} (\lambda_k - \lambda_i)$$

Therefore, Eqn. (3.78) gives

$$\delta\lambda_k = -\frac{\lambda_k^{n-j}}{\prod_{i \neq k} (\lambda_k - \lambda_i)} \delta\alpha_j$$

A measure of the sensitivity of the root λ_k to the parameter α_j is

$$S_{\alpha_j}^{\lambda_k} = \frac{\partial\lambda_k}{\partial\alpha_j/\alpha_j} = \frac{-\lambda_k^{n-j}\alpha_j}{\prod_{i \neq k} (\lambda_k - \lambda_i)} \tag{3.79}$$

Following observations are made from Eqn. (3.79):

(i) The numerator term in Eqn. (3.79) varies with *j*—the index number of the parameter whose variation is under consideration. For a stable system, $\lambda_k < 1$ and, therefore, the numerator term

in Eqn. (3.79) is largest for $j = n$. Therefore, the most sensitive parameter in the characteristic equation (3.76), is α_n.

(ii) The denominator in Eqn. (3.79) is the product of vectors from the characteristic roots to λ_k. Thus, if all the roots are in a cluster, the sensitivity is high.

In the cascade and parallel realizations, the coefficients, mechanized in the algorithm, are poles themselves; these realizations are generally less sensitive than the direct realization.

PROBLEMS

3.1 Find $Y(z)/R(z)$ for the sampled-data closed-loop system of Fig. P3.1.

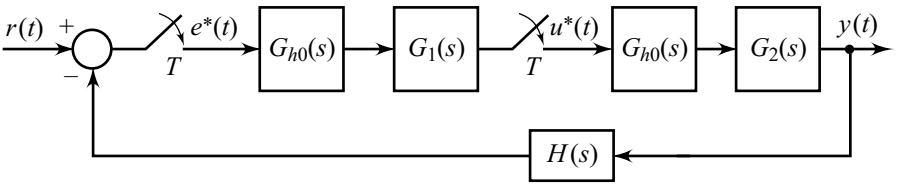


Fig. P3.1

3.2 For the sampled-data feedback system with digital network in the feedback path as shown in Fig. P3.2, find $Y(z)/R(z)$.

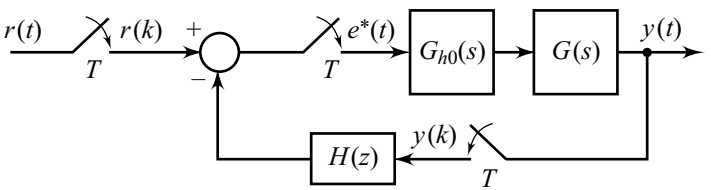


Fig. P3.2

3.3 Find $Y(z)$ for the sampled-data closed-loop system of Fig. P3.3.

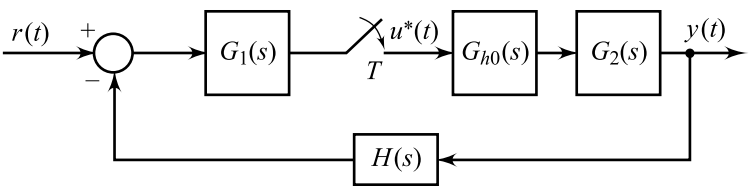


Fig. P3.3

3.4 Obtain the z-transform of the system output for the block diagram of Fig. P3.4.

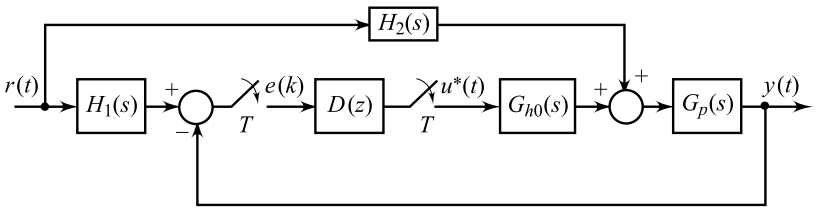


Fig. P3.4

3.5 Obtain the transfer function $Y(z)/R(z)$ of the closed-loop control system shown in Fig. P3.5. Also obtain the transfer function between $X(z)$ and $R(z)$.

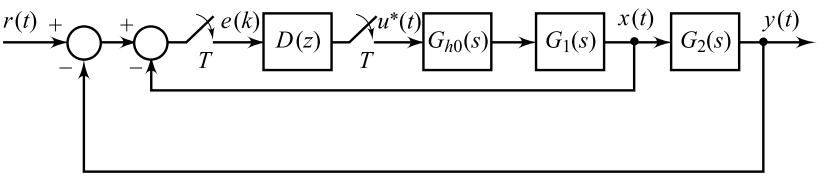


Fig. P3.5

3.6 Consider the block diagram of a digital control system shown in Fig. P3.6; $r(t)$ stands for reference input and $w(t)$ for disturbance. Obtain the z-transform of the system output when $r(t) = 0$.

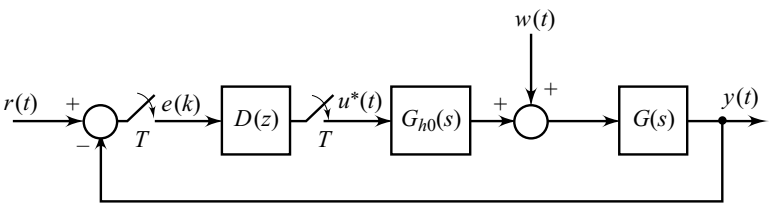


Fig. P3.6

3.7 Shown in Fig. P3.7 is the block diagram of the servo control system for one of the joints of a robot. With $D(z) = 1$, find the transfer function model of the closed-loop system. Sampling period $T = 0.25$ sec.

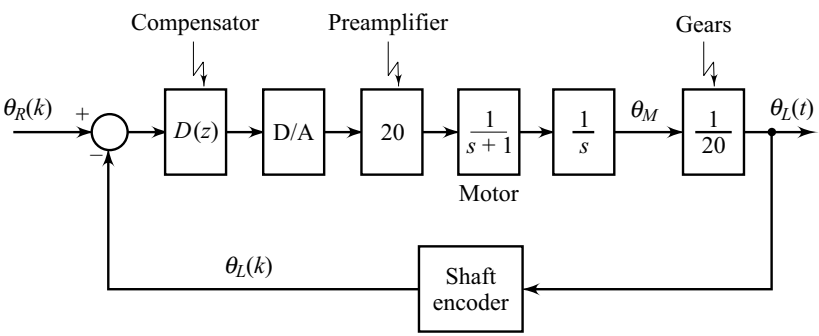


Fig. P3.7

3.8 The plant of the speed control system shown in Fig. P3.8 consists of load, armature-controlled dc motor and a power amplifier. Its transfer function is given by

$$\frac{\omega(s)}{V(s)} = \frac{185}{0.025s + 1}$$

Find the discrete-time transfer function $\omega(z)/\omega_r(z)$ for the closed-loop system. Sampling period $T = 0.05$ sec.

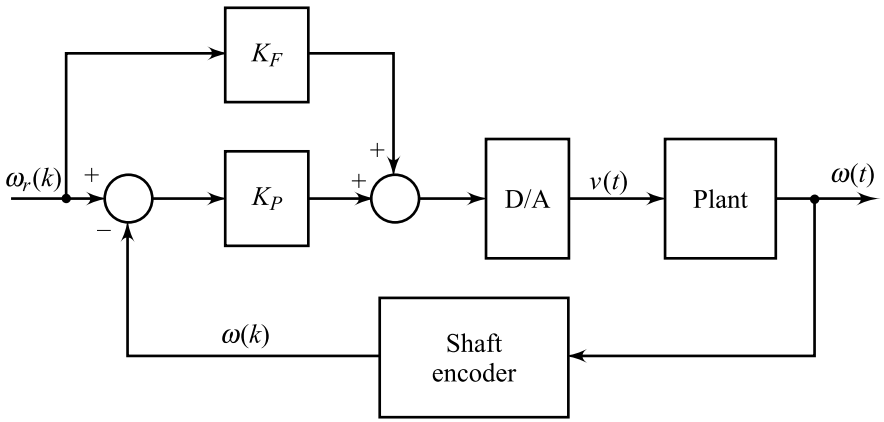


Fig. P3.8

3.9 For the system shown in Fig. P3.9, the computer solves the difference equation $u(k) = u(k-1) + 0.5\,e(k)$, where $e(k)$ is the filter input and $u(k)$ is the filter output. If the sampling rate $f_s = 5$ Hz, find $Y(z)/R(z)$.

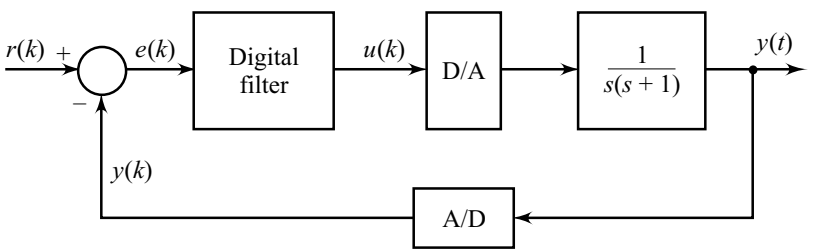


Fig. P3.9

3.10 Consider the sampled-data system shown in Fig. P3.10. Find $Y(z)/R(z)$ when (i) $\tau_D = 0.4$ sec, (ii) $\tau_D = 1.4$ sec.

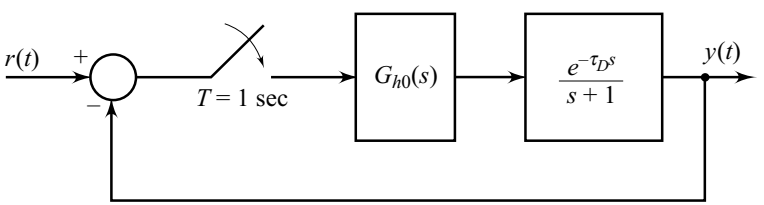


Fig. P3.10

3.11 Figure P3.11 shows an electrical oven provided with temperature measurement by a thermocouple and having a remotely controlled, continuously variable power input. The task is to design a microprocessor-based control system to provide temperature control of the oven.

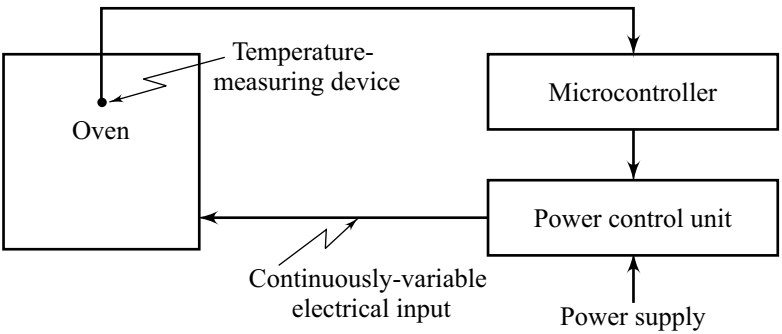


Fig. P3.11

The functions within the control loop can be enumerated as follows:

(i) sampling of the output of thermocouple;
(ii) transfer of temperature signal into the computer;
(iii) comparison of the measured temperature with the stored desired temperature, to form an error signal;
(iv) operation on the error signal by an appropriate algorithm, to form a control signal; and
(v) processing of the control signal and its transfer through the interface to the power control unit.

Suggest suitable hardware to implement these control-loop functions. Make a sketch of the system showing how the hardware is connected.

3.12 (a) A unity-feedback system has the open-loop transfer function

$$G(s) = \frac{5}{s(s+1)(s+2)}$$

Using the Routh stability criterion, show that the closed-loop system is stable.

(b) A sampler and ZOH are now introduced in the forward path (Fig. P3.12). Show that the stable linear continuous-time system becomes unstable upon the introduction of sampler and ZOH.

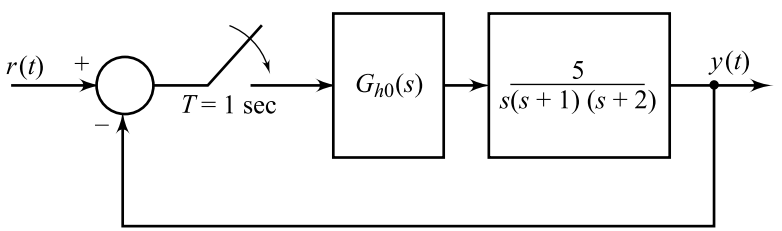


Fig. P3.12

3.13 The characteristic equation of a linear digital system is

$$z^3 - 0.1\,z^2 + 0.2Kz - 0.1K = 0$$

Using Jury stability criterion, determine the values of $K > 0$ for which the system is stable.

3.14 Compare the stability properties of the system shown in Fig. P3.14 with (i) $T = 0.5$, and (ii) $T = 1$. Assume $K > 0$.

3.15 The block diagram of a digital control system is shown in Fig. P3.15. Apply the Jury stability criterion to determine the range of values that $K > 0$ can have for a stable response. Also show graphically how these values are affected by the sampling period T.

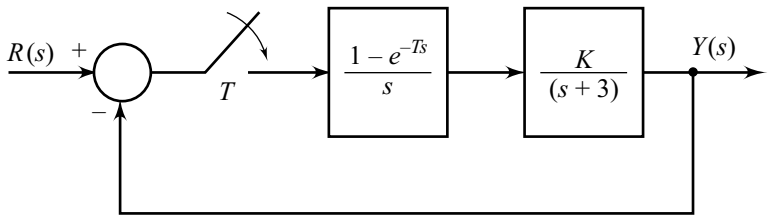


Fig. P3.14

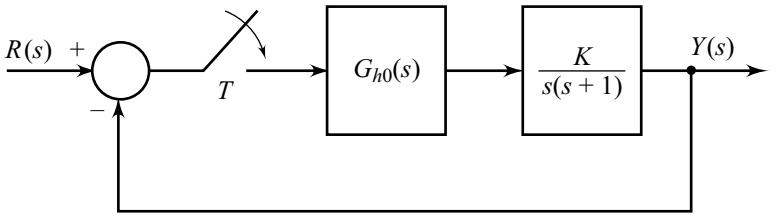


Fig. P3.15

3.16 Consider the system shown in Fig. P3.16. Using Jury stability criterion, find the range of $K > 0$ for which the system is stable.

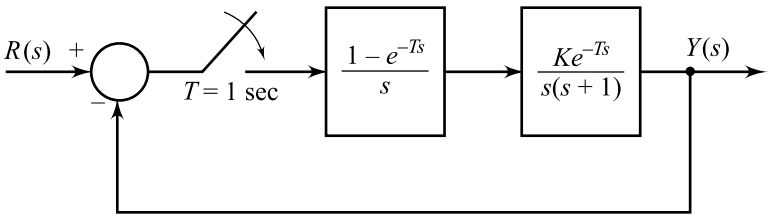


Fig. P3.16

3.17 (a) A unity-feedback system has the open-loop transfer function

$$G(s) = \frac{Y(s)}{R(s)} = \frac{4500K}{s(s+361.2)};\ K = 14.5$$

Find the response $y(t)$ of the system to a unit-step input.

(b) A sampler and ZOH are now introduced in the forward path (Fig. P3.17). For a unit-step input, determine the output $y(k)$ for first five sampling instants when (i) $T = 0.01$ sec, and (ii) $T = 0.001$ sec. Compare the result with that obtained earlier in part (a) above.

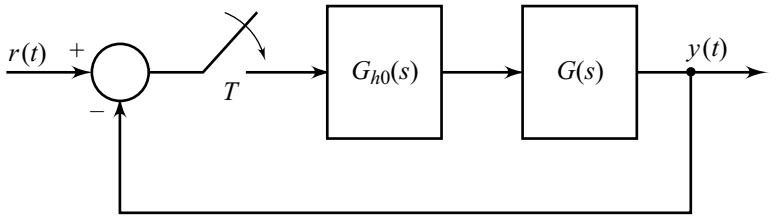


Fig. P3.17

3.18 For the sampled-data system shown in Fig. P3.18, find the output $y(k)$ for $r(t)$ = unit step.

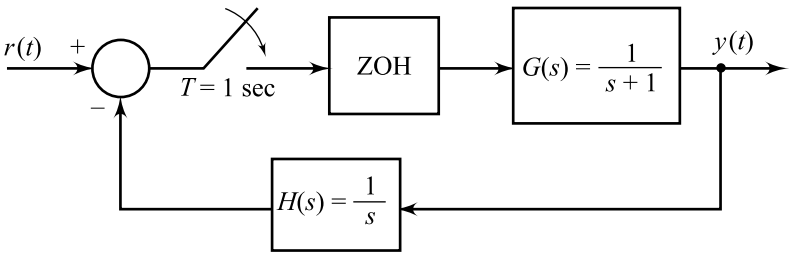


Fig. P3.18

3.19 For the sampled-data system of Fig. P3.19, find the response $y(kT)$; $k = 0, 1, 2, ...$, to a unit-step input $r(t)$. Also, obtain the output at the midpoints between pairs of consecutive sampling points.

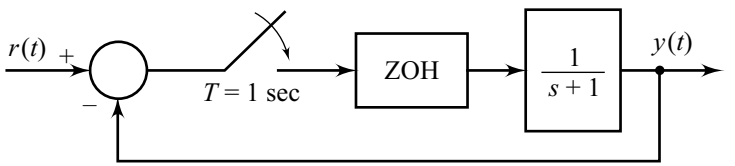


Fig. P3.19

3.20 Consider the digital controller defined by

$$D(z) = \frac{U(z)}{E(z)} = \frac{4(z-1)(z^2+1.2z+1)}{(z+0.1)(z^2-0.3z+0.8)}$$

Realize this digital controller in the cascade scheme and in parallel scheme. Use one first-order section and one second-order section.

3.21 Consider the digital controller defined by

$$D(z) = \frac{U(z)}{E(z)} = \frac{10(z^2+z+1)}{z^2(z-0.5)(z-0.8)}$$

Draw a parallel realization diagram.

3.22 Consider the temperature control system shown in Fig. P3.22a. A typical experimental curve, obtained by opening the steam valve at $t = 0$ from fully closed position to a position that allows a flow Q_m of 1 kg/min with initial sensor temperature θ of 0 ºC is shown in Fig. P3.22b.

(a) Approximate the process reaction curve by a first-order plus dead-time model using two-points method of approximation.

(b) Calculate the QDR tuning parameters for a PID controller. The PID control is to be carried out with a sampling period of 1 min on a computer control installation.

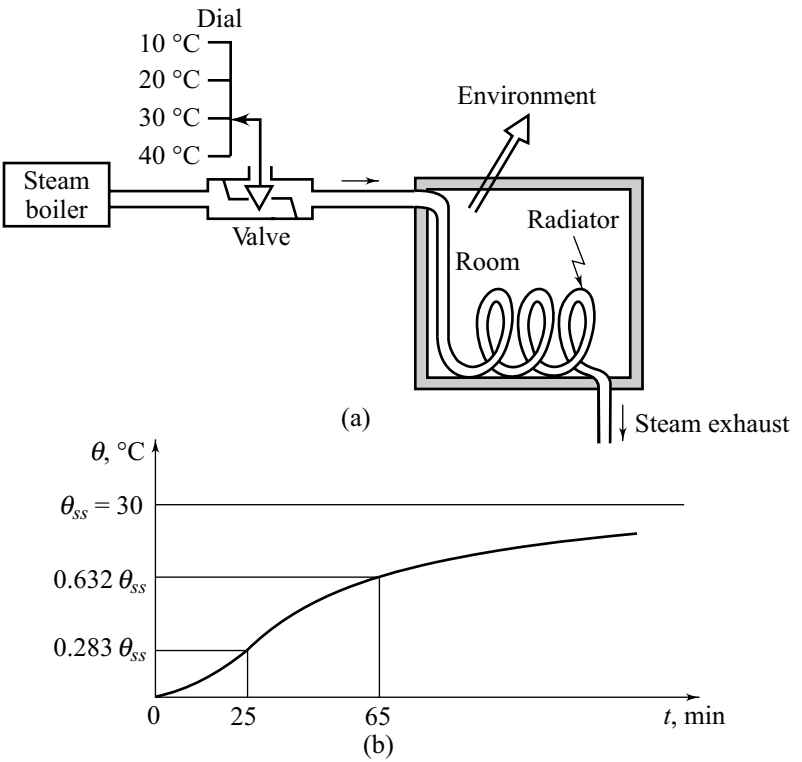


Fig. P3.22

3.23 Consider the liquid-level control system shown in Fig. 1.6. The digital computer was programmed to act as adjustable-gain proportional controller with a sampling period of $T = 10$ sec. The proportional gain was increased in steps. After each increase, the loop was disturbed by introducing a small change in set-point, and the response of the controlled variable (level in the tank) was observed. The proportional gain of 4.75 resulted in oscillatory behavior, with amplitude of oscillations approximately constant. The period of oscillations measured from the response is 800 sec. The PC implements the digital PI control algorithm. Determine tuning parameters for the controller:

$$\Delta u(k) = u(k) - u(k-1) = K_c \left[e(k) - e(k-1) + \frac{T}{T_I} e(k) \right]$$

where

$u(k)$ = output of controller at kth sampling instant;
$\Delta u(k)$ = change in output of controller at kth sampling instant;
$e(k)$ = error at kth sampling instant;
T = sampling time;
T_I = integral time; and
K_c = proportional gain.

3.24 A traffic light controller is to be designed for a road, partly closed to traffic for urgent repair work (Fig. P3.24). North traffic light will go GREEN for 30 sec with South traffic light giving RED signal. For the next 15 sec, both the traffic lights will give RED signals. Thereafter, South traffic light will go GREEN for 30 sec with North traffic light giving RED signal. Both the traffic lights will give RED signal for the next 15 sec. Then this cycle will repeat.
Develop a PLC ladder diagram that accomplishes this objective.

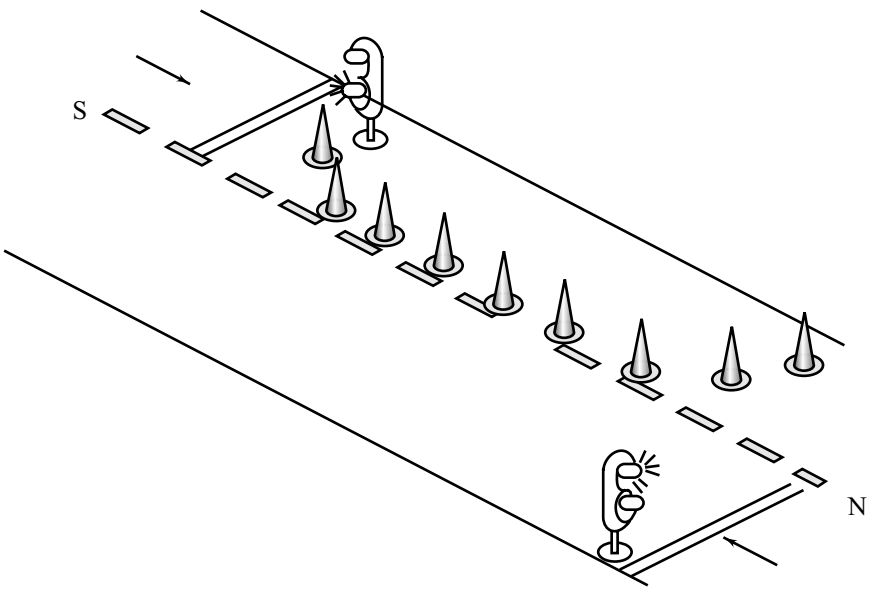


Fig. P3.24

3.25 Consider the tank system of Fig. P3.25. Valve V1 opens on pressing a pushbutton PB1 and liquid begins to fill the tank. At the same time, the stirring motor M starts operations. When the liquid level passes LL2 and reaches LL1, the valve V1 closes and the stirring motor stops. When PB1 is pressed again, the valve V2 opens and starts draining the liquid. When the liquid level drops below LL2, valve V2 closes. This cycle is repeated five times. A buzzer will go high after 5 repetitions. The buzzer will be silenced by pressing pushbutton PB2. The process will now be ready to take up another filling-stirring-draining operation under manual control. Develop a PLC ladder diagram that accomplishes this objective.

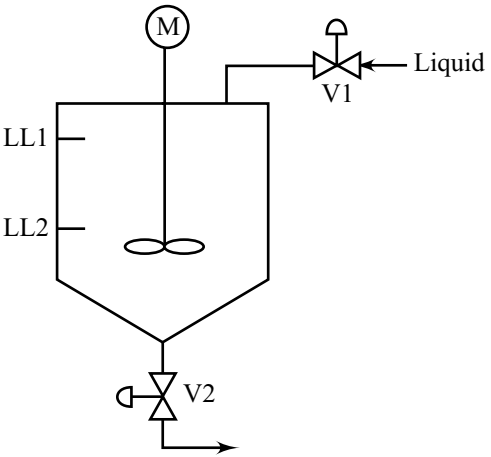


Fig. P3.25

3.26 A control circuit is to be developed to detect and count the number of products being carried on an assembly line (Fig. P3.26). A sensor activates a counter as a product leaves the conveyor and enters the packaging section. When the counter counts five products, the circuit energizes a solenoid. The solenoid remains energized for a period of 2 seconds; the time being measured by a software timer. When the set time has lapsed, the solenoid is deenergized, causing it to retract; and the control circuit is ready for the next cycle.

Develop a suitable PLC ladder diagram.

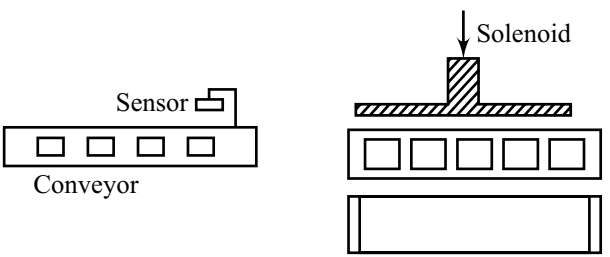


Fig. P3.26

3.27 In the system of Fig. P3.27, a PLC is used to start and stop the motors of a segmented conveyor belt. This allows only belt sections carrying an object to move. Motor M3 is kept ON during the operation. Position of a product is first detected by proximity switch S3, which switches on the motor M2. Sensor S2 switches on the motor M1 upon detection of the product. When the product moves beyond the range of sensor S2, a timer is activated and when the set time of 20 sec has lapsed, motor M2 stops. Similarly, when the product moves beyond the range of sensor S1, another timer is activated and when the set time of 20 sec (for unloading the product) has lapsed, motor M1 stops.

Develop a suitable ladder diagram for control.

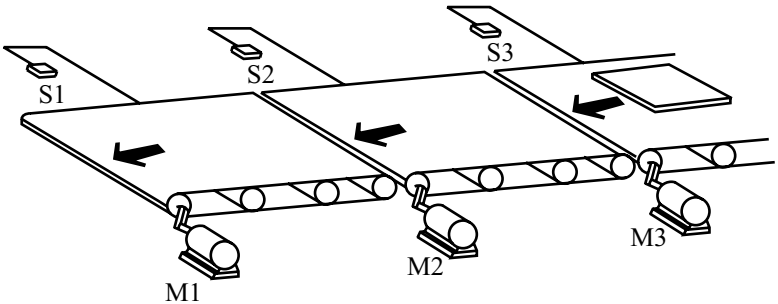


Fig. P3.27

3.28 The system of Fig. P3.28 has the objective of drilling a hole in workpiece moved on a carriage. When the start button PB1 is pushed and LS1 is ON (workpiece loaded), feed carriage motor runs in CW direction, moving the carriage from left to right. When the work comes exactly under the drill, which is sensed by limit switch LS2, the motor is cut-off and the work is ready for drilling operation. A timer with a set time of 7 sec is activated. When the timer set value has lapsed, the motor reverses, moving the carriage from right to left. When the work piece reaches LS1 position, the motor stops. The motor can be stopped by a stop pushbutton while in operation.
Develop a suitable ladder diagram for PLC control.

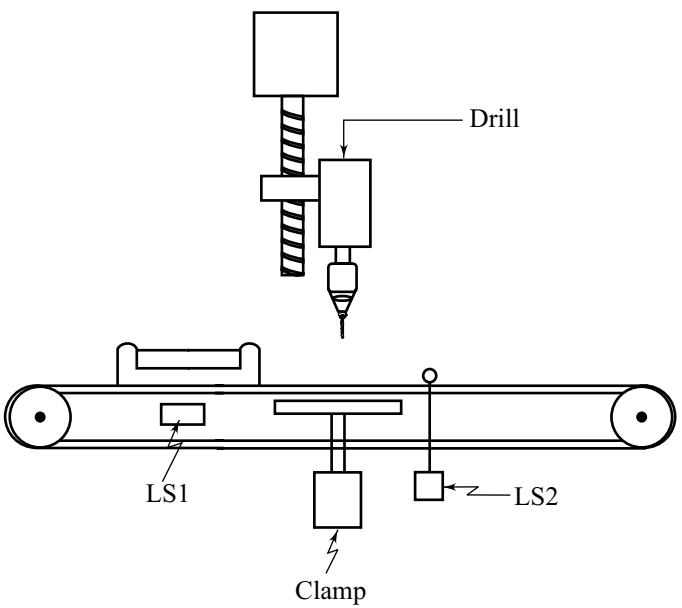


Fig. P3.28

Chapter 4

Design of Digital Control Algorithms

4.1 INTRODUCTION

During recent decades, the design procedures for analog control systems have been well formulated and a large body of knowledge has been accumulated. The analog-design methodology, based on conventional techniques of root locus and Bode plots or the tuning methods of Ziegler and Nichols, may be applied to designing digital control systems. The procedure would be to first design the analog form of the controller, or compensator, to meet a particular set of performance specifications. Having done this, the analog form can be transformed to a discrete-time formulation. This approach is based on the fact that a digital system with a high sampling rate approximates to an analog system. The justification for using digital control under these circumstances must be that the practical limitations of the analog controller are overcome, the implementation cheaper, or that the supervisory control and communications more easily implemented.

However, the use of high sampling rates wastes computer power, and can lead to problems of arithmetic precision, etc. One is, therefore, driven to find methods of design which take account of the sampling process.

The alternative approach is to design controllers directly in the discrete-time domain, based on the specifications of closed-loop system response. The controlled plant is represented by a discrete-time model which is a continuous-time system observed, analyzed, and controlled at discrete intervals of time. This approach provides a direct path to the design of digital controllers. The features of direct digital design are that the sample rates are generally lower than those for discretized analog design, and the design is directly 'performance based'.

Figure 4.1 shows the basic structure of a digital control system. The design problem generally evolves around the choice of the control function $D(z)$, in order to impart a satisfactory form to the closed-loop transfer function. The choice is constrained by the function $G_{h0}G(z)$ representing the fixed process elements.

A wide variety of digital-design procedures is available; these fall into the following two categories:

(i) direct synthesis procedures; and
(ii) iterative design procedures.

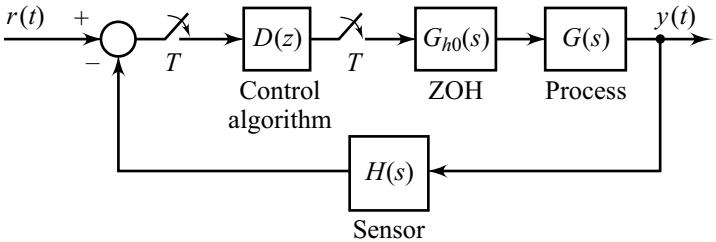


Fig. 4.1 Basic structure of a digital control system

The direct synthesis procedures assume that the control function $D(z)$ is not restricted in any way by hardware or software limitations, and can be allowed to take any form demanded by the nature of the fixed process elements and the specifications of the required system performance. This design approach has found *wider* applications in digital control systems—than has the equivalent technique used with analog systems. In a digital control system, realization of the required $D(z)$ may involve no more than programming a special-purpose software-procedure. With analog systems, the limitation was in terms of the complications involved in designing special purpose analog controllers.

The design obtained by a direct synthesis procedure will give perfect nominal performance. However, the performance may be inadequate in the field because of the sensitivity of the design to plant disturbances and modeling errors. It is important that a control system is robust in its behavior with respect to the discrepancies between the model and the real process, and uncertainties in disturbances acting on the process. Robustness property of some of the standard control structures, such as a three-term (PID) control algorithm, has been very well established. The design of such algorithms calls for an iterative design procedure where the choice of control function $D(z)$ is restricted to using a standard algorithm with variable parameters; the designer must then examine the effect of the choice of controller parameters on the system performance, and make an appropriate final choice. The iterative design procedures for digital control systems are similar to the techniques evolved for analog system design, using root locus and frequency response plots.

Figure 4.2 summarizes the basic routes to the design of digital controllers for continuous-time processes.

The route: continuous-time modeling → continuous-time control design → discrete-time approximation of the controller, was considered in Chapter 2 (refer to Example 2.17). This chapter is devoted to the following route:

- Continuous-time modeling → discrete-time approximation of the model → discrete-time control design.

Plant models can be obtained from the first principles of physics. The designer may, however, turn to the other source of information about plant dynamics, which is the data taken from experiments directly conducted to excite the plant, and to measure the response. The process of constructing models from experimental data is called *system identification*. An introduction to system identification and adaptive control is given in Chapter 10.

One obvious, but fundamental, point is that control design always begins with a sufficiently accurate mathematical model of the process to be controlled. For a typical industrial problem, the effort required for obtaining a mathematical model of the process to be controlled, is often an order of magnitude greater than the effort required for control design proper. Any control-design method that requires only a simple

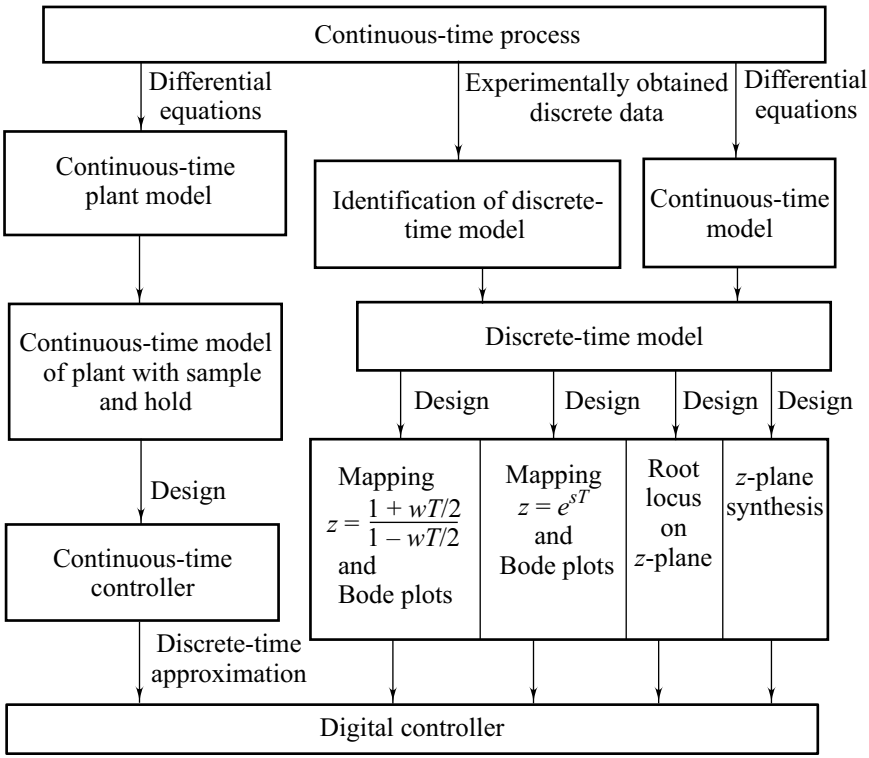


Fig. 4.2 Basic routes to the design of digital controllers

process model, therefore, has a high appeal to those faced with real industrial control problems. Tunable PID controllers which require only simple process models, have already been described in Chapter 3.

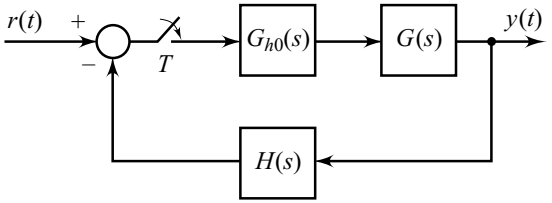
The design approaches discussed in this chapter, use z-transform as the background tool. In Chapters 7 and 8, we will solve the identical problems using state-space formulation.

The discussion in this chapter is based on the material that the student has already studied in a prerequisite, introductory course on control systems. The presentation given in this chapter is not sufficient to learn control system design for the first time; rather, it states, only concisely, the key concepts and relationships from the continuous-time control for ready reference, as we move to the new concepts of discrete-time control. For an in-depth treatment of the principles of control system design, see the companion book [155].

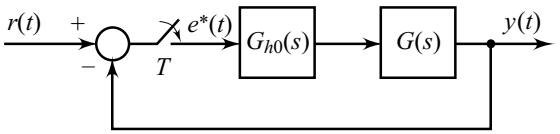
In this chapter, the following classes of feedback systems will be under consideration:

- The system can be represented by a unity-feedback structure shown in Fig. 4.3b.
- The open-loop transfer function $G(s)$ in Fig. 4.3b has no poles in the right half of the s-plane.
- The feedback system of Fig. 4.3b is desired to be an underdamped system.

In Section 1.4, we have described a generalized, operational block diagram of a feedback system (revisiting the section will be helpful). It was shown that the generalized, non-unity-feedback block diagram can be converted to a unity-feedback structure when the sensor transfer function is equal to the transfer function representing the reference input elements. This, in fact, is quite common; the two



(a) A non-unity feedback discrete-time system



(b) A unity feedback discrete-time system

Fig. 4.3

transfer functions in many control system designs are assumed to have zero-order dynamics and are equal.

The majority of the systems we deal with, in practice, belong to the class under consideration in this chapter. However, the control system design techniques presented in the chapter, are not limited to this class. There are control design problems wherein the sensor transfer function $H(s)$ (of first or higher-order), is explicitly present in the feedback path (Fig. 4.3a); the open-loop transfer function $G(s)$ or $G(s)H(s)$, has poles in the right half of the s-plane; and/or the oscillations in feedback system cannot be tolerated, and the system is required to be overdamped. A detailed treatment of such problems is given in the companion book [155]; the principles given therein for continuous-time control, have direct extension to discrete-time control. Our decision to leave these extensions to the reader, is prompted only by the desire to save space for other topics.

4.2 *z*-PLANE SPECIFICATIONS OF CONTROL SYSTEM DESIGN

The central concerns of controller design are for good relative stability and speed of response, good steady-state accuracy, and sufficient robustness. Requirements on time response need to be expressed as constraints on z-plane pole and zero locations, or on the shape of the frequency response in order to permit design in the transform domain. In this section, we give an outline of specifications of controller design in the z-plane.

Our attention will be focused on the unity-feedback systems[1] of the form shown in Fig. 4.3b, with the open-loop transfer function $G_{h0}G(z) = \mathcal{Z}[G_{h0}(s)G(s)]$, having no poles outside the unit circle in the z-plane. Further, the feedback system of Fig. 4.3b is desired to be an underdamped system.

[1] It is assumed that the reader is familiar with the design of unity and non-unity-feedback continuous-time systems. With this background, the results presented in this chapter for unity-feedback discrete-time systems, can easily be extended for the non-unity-feedback case.

The nature of transient response of a linear control system is revealed by any of the standard test signals—impulse, step, ramp, parabola—as this nature is dependent on system poles only and not on the type of the input. It is, therefore, sufficient to specify the transient response to one of the standard test signals; a step is generally used for this purpose. Steady-state response depends on both the system and the type of input. From the steady-state viewpoint, the 'easiest' input is generally a step since it requires only maintaining the output at a constant value, once the transient is over. A more difficult problem is tracking a ramp input. Tracking a parabola is even more difficult since a parabolic function is one degree faster than the ramp function. In practice, we seldom find it necessary to use a signal faster than a parabolic function; characteristics of actual signals which the control systems encounter, are adequately represented by step, ramp, and parabolic functions.

4.2.1 Steady-State Accuracy

Steady-state accuracy refers to the requirement that after all transients become negligible, the error between the reference input r and the controlled output y must be acceptably small. The specification on steady-state accuracy is often based on polynomial inputs of degree k: $r(t) = \frac{t^k}{k!}\mu(t)$. If $k = 0$, the input is a step of unit amplitude; if $k = 1$, the input is a ramp with unit slope; and if $k = 2$, the input is a parabola with unit second derivative. From the common problems of mechanical motion control, these inputs are called, respectively, position, velocity, and acceleration inputs.

For quantitative analysis, we consider the unity-feedback discrete-time system shown in Fig. 4.3b. The steady-state error is the difference between the reference input $r(k)$ and the controlled output $y(k)$, when steady state is reached, i.e., steady-state error

$$e_{ss}^* = \lim_{k \to \infty} e(k) = \lim_{k \to \infty} [r(k) - y(k)] \tag{4.1a}$$

Using the final value theorem (Eqn. (2.52)),

$$e_{ss}^* = \lim_{z \to 1} [(z-1)E(z)] \tag{4.1b}$$

provided that $(z - 1)\,E(z)$ has no poles on the boundary and outside of the unit circle in the z-plane.

For the system shown in Fig. 4.3b, define

$$G_{h0}G(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{G(s)}{s}\right]$$

Then, we have

$$\frac{Y(z)}{R(z)} = \frac{G_{h0}G(z)}{1 + G_{h0}G(z)}$$

and

$$E(z) = R(z) - Y(z) = \frac{R(z)}{1 + G_{h0}G(z)} \tag{4.2}$$

By substituting Eqn. (4.2) into Eqn. (4.1b), we obtain

$$e_{ss}^* = \lim_{z \to 1} [(z-1)E(z)] \tag{4.3a}$$

$$= \lim_{z \to 1} \left[(z-1)\frac{R(z)}{1 + G_{h0}G(z)}\right] \tag{4.3b}$$

Thus, the steady-state error of a discrete-time system with unity feedback, depends on the reference input signal $R(z)$, and the forward-path transfer function $G_{h0}G(z)$. By the nature of the limit in Eqns (4.3),

we see that the result of the limit can be zero, or can be a constant different from zero. Also, the limit may not exist, in which case, the final-value theorem does not apply. However, it is easy to see from basic definition (4.1a) that $e_{ss}^* = \infty$ in this case anyway, because $E(z)$ will have a pole at $z = 1$ that is of order higher than one. Discrete-time systems, having a finite nonzero steady-state error when the reference input is a zero-order polynomial input (a constant), are labeled 'Type-0'. Similarly, a system that has finite nonzero steady-state error to a first-order polynomial input (a ramp), is called a 'Type-1' system, and a system with finite nonzero steady-state error to a second-order polynomial input (a parabola), is called a 'Type-2' system.

Let the reference input to the system of Fig. 4.3b be a step function of magnitude unity. The z-transform of discrete form of $r(t) = \mu(t)$ is (refer to Eqn. (2.40))

$$R(z) = \frac{z}{z-1} \tag{4.4a}$$

Substituting $R(z)$ into Eqn. (4.3b), we have

$$e_{ss}^* = \lim_{z \to 1} \frac{1}{1+G_{h0}G(z)} = \frac{1}{1+\lim\limits_{z \to 1} G_{h0}G(z)}$$

In terms of the *position error constant* K_p, defined as

$$K_p = \lim_{z \to 1} G_{h0}G(z) \tag{4.4b}$$

the steady-state error to unit-step input becomes

$$e_{ss}^* = \frac{1}{1+K_p} \tag{4.4c}$$

For a ramp input $r(t) = t\mu(t)$; the z-transform of its discrete form is (refer to Eqn. (2.42))

$$R(z) = \frac{Tz}{(z-1)^2} \tag{4.5a}$$

Substituting into Eqn. (4.3b), we get

$$e_{ss}^* = \lim_{z \to 1} \frac{T}{(z-1)[1+G_{h0}G(z)]} = \frac{1}{\lim\limits_{z \to 1}\left[\dfrac{z-1}{T} G_{h0}G(z)\right]}$$

In terms of *velocity error constant* K_v, defined as

$$K_v = \frac{1}{T} \lim_{z \to 1} [(z-1)G_{h0}G(z)] \tag{4.5b}$$

the steady-state error to unit-ramp input becomes

$$e_{ss}^* = \frac{1}{K_v} \tag{4.5c}$$

For a parabolic input $r(t) = (t^2/2)\,\mu(t)$; the z-transform of its discrete form is (from Eqns (2.41)–(2.42))

$$R(z) = \frac{T^2 z(z+1)}{2(z-1)^3} \tag{4.6a}$$

Substituting into Eqn. (4.3b), we get

$$e_{ss}^{*} = \lim_{z \to 1} \frac{T^2}{(z-1)^2[1+G_{h0}G(z)]} = \frac{1}{\lim_{z \to 1}\left[\left(\frac{z-1}{T}\right)^2 G_{h0}G(z)\right]}$$

In terms of *acceleration error constant* K_a, defined as

$$K_a = \frac{1}{T^2} \lim_{z \to 1} [(z-1)^2 \, G_{h0}G(z)] \tag{4.6b}$$

the steady-state error to unit-parabolic input becomes

$$e_{ss}^{*} = \frac{1}{K_a} \tag{4.6c}$$

As said earlier, discrete-time systems can be classified on the basis of their steady-state response to polynomial inputs. We can always express the forward-path transfer function $G_{h0}G(z)$ as

$$G_{h0}G(z) = \frac{K \prod_i (z - z_i)}{(z-1)^N \prod_j (z - p_j)} \; ; p_j \neq 1, z_i \neq 1 \tag{4.7}$$

$G_{h0}G(z)$ in Eqn. (4.7) involves the term $(z-1)^N$ in the denominator. As $z \to 1$, this term dominates in determining the steady-state error. Digital control systems are, therefore, classified in accordance with the number of poles at $z = 1$ in the forward-path transfer function, as described below.

Type-0 System

If $N = 0$, the steady-state errors to various standard inputs, obtained from Eqns (4.1)–(4.7), are

$$e_{ss}^{*} = \begin{cases} \dfrac{1}{1+K_p} \text{ in response to unit-step input;} \quad K_p = \left.\dfrac{K \prod_i (z - z_i)}{\prod_j (z - p_j)}\right|_{z=1} \\ \infty \quad \text{in response to unit-ramp input} \\ \infty \quad \text{in response to unit-parabolic input} \end{cases} \tag{4.8a}$$

Thus, a system with $N = 0$, or no pole at $z = 1$ in $G_{h0}G(z)$, has a finite nonzero position error, and infinite velocity and acceleration errors at steady state.

Type-1 System

If $N = 1$, the steady-state errors to various standard inputs are

$$e_{ss}^{*} = \begin{cases} 0 \quad \text{in response to unit-step input;} \\ \dfrac{1}{K_v} \text{ in response to unit-ramp input; } K_v = \left.\dfrac{\frac{K}{T}\prod_i (z - z_i)}{\prod_j (z - p_j)}\right|_{z=1} \\ \infty \quad \text{in response to unit-parabolic input} \end{cases} \tag{4.8b}$$

Thus, a system with $N = 1$, or one pole at $z = 1$ in $G_{h0}G(z)$, has zero position error, a finite nonzero velocity error, and infinite acceleration error at steady state.

Type-2 System

If $N = 2$, the steady-state errors to various standard inputs are

$$e_{ss}^{*} = \begin{cases} 0 & \text{in response to unit-step input} \\ 0 & \text{in response to unit-ramp input} \\ \dfrac{1}{K_a} & \text{in response to unit-parabolic input; } K_a = \left.\dfrac{\dfrac{K}{T^2}\prod\limits_i (z - z_i)}{\prod\limits_j (z - p_j)}\right|_{z=1} \end{cases} \tag{4.8c}$$

Thus, a system with $N = 2$, or two poles at $z = 1$ in $G_{h0}G(z)$, has zero position and velocity errors, and a finite nonzero acceleration error at steady state.

Steady-state errors for various inputs and systems are summarized in Table 4.1.

Table 4.1 Steady-state errors for various inputs and systems

Type of input	*Steady-state error*		
	Type-0 system	Type-1 system	Type-2 system
Unit step	$\dfrac{1}{1+K_p}$	0	0
Unit ramp	∞	$\dfrac{1}{K_v}$	0
Unit parabolic	∞	∞	$\dfrac{1}{K_a}$
$K_p = \lim\limits_{z\to 1} G_{h0}G(z)$;	$K_v = \dfrac{1}{T}\lim\limits_{z\to 1}[(z-1)\,G_{h0}G(z)]$;		$K_a = \dfrac{1}{T^2}\lim\limits_{z\to 1}[(z-1)^2 G_{h0}G(z)]$

The development above indicates that, in general, increased system gain K, and/or addition of poles at $z = 1$ to the open-loop transfer function $G_{h0}G(z)$, tend to decrease steady-state errors. However, as will be seen later in this chapter, both large system gain and the poles at $z = 1$ in the loop transfer function, have destabilizing effects on the system. Thus, a control system design is usually a trade off between steady-state accuracy and acceptable relative stability.

Example 4.1

In the previous chapter, we have shown that sampling usually has a detrimental effect on the transient response and the relative stability of a control system. It is natural to ask what the effect of sampling on

the steady-state error of a closed-loop system will be? In other words, if we start out with a continuous-time system and then add S/H to form a digital control system, how would the steady-state errors of the two systems compare, when subject to the same type of input?

Let us first consider the system of Fig. 4.3b without S/H. Assume that the process $G(s)$ is represented by Type-1 transfer function:

$$G(s) = \frac{K(1+\tau_a s)(1+\tau_b s)\cdots(1+\tau_m s)}{s(1+\tau_1 s)(1+\tau_2 s)\cdots(1+\tau_n s)}$$

having more poles than zeros.

The velocity error constant

$$K_v = \lim_{s \to 0} sG(s) = K$$

The steady-state error of the system to unit-step input is zero, to unit-ramp input is $1/K$, and to unit-parabolic input is ∞.

We now consider the system of Fig. 4.3b with S/H:

$$\begin{aligned} G_{h0}G(z) &= (1-z^{-1})\,\mathscr{Z}\left[\frac{K(1+\tau_a s)(1+\tau_b s)\cdots(1+\tau_m s)}{s^2(1+\tau_1 s)(1+\tau_2 s)\cdots(1+\tau_n s)}\right] \\ &= (1-z^{-1})\,\mathscr{Z}\left[\frac{K}{s^2}+\frac{K_1}{s}+\text{terms due to the nonzero poles}\right] \\ &= (1-z^{-1})\left[\frac{KTz}{(z-1)^2}+\frac{K_1 z}{z-1}+\text{terms due to the nonzero poles}\right] \end{aligned}$$

It is important to note that the terms due to the nonzero poles do not contain the term $(z - 1)$ in the denominator. Thus, the velocity error constant is

$$K_v = \frac{1}{T}\lim_{z \to 1}[(z-1)G_{h0}G(z)] = K$$

The steady-state error of the discrete-time system to unit-step input is zero, to unit-ramp input is $1/K$, and to unit-parabolic input is ∞. Thus, for a Type-1 system, the system with S/H has exactly the same steady-state error as the continuous-time system with the same process transfer function (this, in fact, is true for Type-0 and Type-2 systems also).

Equations (4.5b) and (4.6b) may purport to show that the velocity error constant and the acceleration error constant of a digital control system depend on the sampling period T. However, in the process of evaluation, T gets canceled, and the error depends only on the parameters of the process and the type of inputs.

4.2.2 Transient Accuracy

Transient performance in time domain is defined in terms of parameters of the system response to a step in command input. The most frequently used parameters are rise time, peak time, peak overshoot, and setting time. Figure 4.4 shows a typical unit-step response of a control system.

For underdamped systems, the *rise time*, t_r, is normally defined as the time required for the step response to rise from 0% to 100% of its final value.

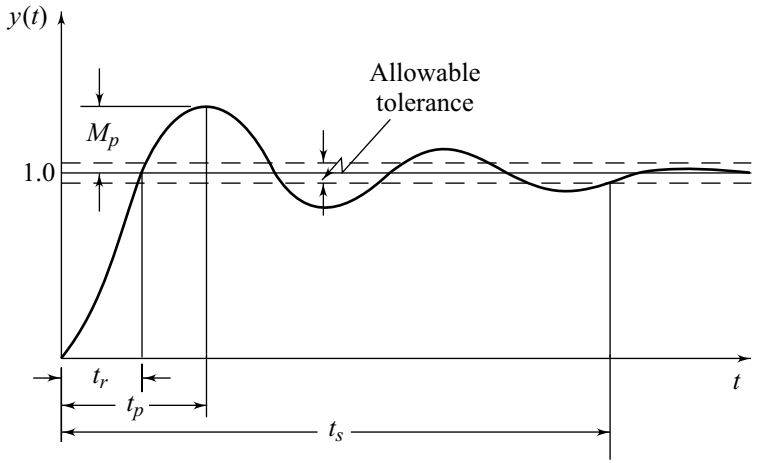


Fig. 4.4 Typical unit-step response of a digital control system

The *peak overshoot*, M_p, is the peak value of the response curve measured from unity. The time at which peak occurs is referred to as the *peak time*, t_p.

The time required for the response to damp out all transients, is called the *settling time*, t_s. Theoretically, the time taken to damp out all transients may be infinity. In practice, however, the transient is assumed to be over when the error is reduced below some acceptable value. Typically, the acceptable level is set at 2% or 5% of the final value.

The use of root locus plots for the design of digital control systems necessitates the translation of time-domain performance specifications into desired locations of closed-loop poles in the z-plane. However, the use of frequency response plots necessitates the translation of time-domain specifications in terms of frequency response features such as bandwidth, phase margin, gain margin, resonance peak, resonance frequency, etc.

Specifications in Terms of Root Locations in the *z*-Plane

Our approach is to first obtain the transient response specifications in terms of characteristic roots in the s-plane, and then use the relation

$$z = e^{sT} \tag{4.9}$$

to map the s-plane characteristic roots to the z-plane.

The transient response of Fig. 4.4 resembles the unit-step response of an underdamped second-order system

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.10}$$

where

ζ = damping ratio, and ω_n = undamped natural frequency.

The transient response specifications in terms of rise time t_r, peak time t_p, peak overshoot M_p, and settling time t_s can be approximated to the parameters ζ and ω_n of the second-order system defined by

Eqn. (4.10), using the following correlations[2]:

$$t_r(0\% \text{ to } 100\%) = \frac{\pi - \cos^{-1}\zeta}{\omega_n\sqrt{1-\zeta^2}} \tag{4.11}$$

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{4.12}$$

$$M_p = \exp\left(-\pi\zeta \Big/ \sqrt{1-\zeta^2}\right) \tag{4.13}$$

$$t_s(2\% \text{ tolerance band}) = \frac{4}{\zeta\omega_n} \tag{4.14}$$

Peak overshoot is used mainly for relative stability. Values in excess of about 40% may indicate that the system is dangerously close to absolute instability. Many systems are designed for 5% to 25% overshoot. No overshoot at all is sometimes desirable. However, this usually penalizes the speed of response needlessly.

The specification on speed of response in terms of t_r, t_p and/or t_s, should be consistent as all these depend on ζ and ω_n. The greater the magnitude of ω_n when ζ is constant, the more rapidly does the response approach the desired steady-state value. The value of ω_n is limited by measurement noise considerations—a system with large ω_n has large bandwidth and will, therefore, allow the high frequency noise signals to affect its performance.

We need to now convert the specifications on ζ and ω_n into guidelines on the placement of poles and zeros in the z-plane, in order to guide the design of digital controls. We do so through the mapping (4.9).

Figure 4.5 illustrates the translation of specifications on ζ and ω_n to the characteristic root locations in the z-plane (referring to Section 2.14 will be helpful). The s-plane poles

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -\zeta\omega_n \pm j\omega_d \tag{4.15a}$$

for constant ζ, lie along a radial line in the s-plane (Fig. 4.5a). In the z-plane,

$$z_{1,2} = e^{-\zeta\omega_n T} e^{\pm j\omega_n T\sqrt{1-\zeta^2}} = r\, e^{\pm j\theta} \tag{4.15b}$$

The magnitude of z (i.e., the distance to the origin) is $r = e^{-\zeta\omega_n T}$ and the angles with the positive real axis of the z-plane, measured positive in the counterclockwise direction, are $\theta = \omega_n T\sqrt{1-\zeta^2}$. It should be observed that the z-plane pole locations depend on the s-plane positions, as well as the sampling interval T.

As ω_n increases for a constant-ζ, the magnitude of z decreases and the phase angle increases; constant-ζ locus is a logarithmic spiral in the z-plane (Fig. 4.5b). Increasing ω_n negatively, gives the mirror image.

In Fig. 4.5a, the s-plane has been divided into strips of width ω_s, where $\omega_s = 2\pi/T$ is the sampling frequency. The primary strip extends from $\omega = -\omega_s/2$ to $+\omega_s/2$, and the complementary strips extend from $-\omega_s/2$ to $-3\omega_s/2$, ... , for negative frequencies, and from $\omega_s/2$ to $3\omega_s/2$,..., for positive frequencies. We will assume that the low-pass analog filtering characteristics of the continuous-time plant and the

[2] Chapter 6 of reference [155].

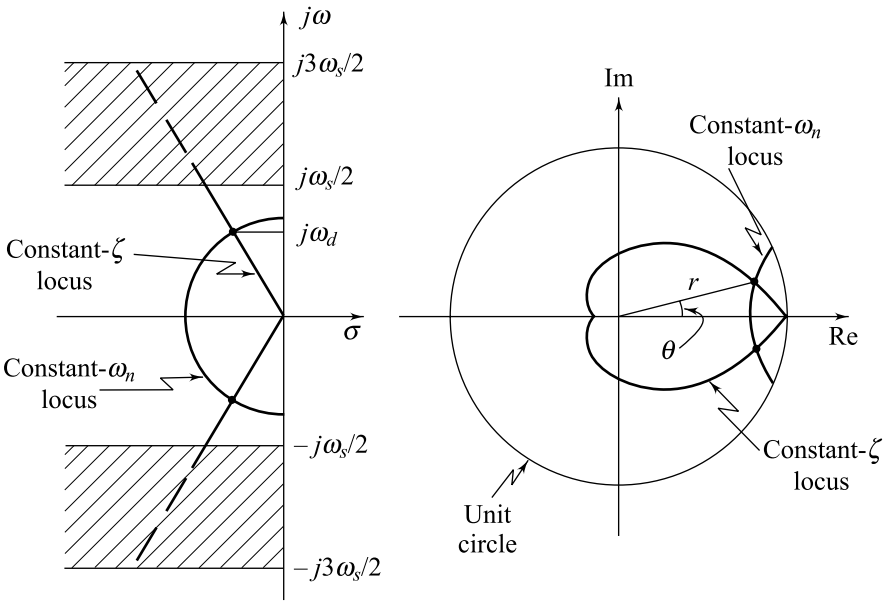


Fig. 4.5 Mapping of *s*-plane patterns on the *z*-plane

ZOH device, attenuate the responses due to the poles in the complementary strips; only the poles in the primary strip, generally, need be considered.

Figure 4.5 illustrates the mapping of constant-ζ locus in the primary strip of the *s*-plane to the *z*-plane. As the imaginary parts $\pm j\omega_d = \pm j\omega_n\sqrt{1-\zeta^2}$ of the *s*-plane poles move closer to the limit $\pm j\omega_s/2$ of the primary strip, the angles $\theta = \pm\,\omega_d T = \pm\,\omega_n T\,\sqrt{1-\zeta^2}$ of the *z*-plane poles approach the direction of the negative real axis. The negative real axis in the *z*-plane, thus, corresponds to the boundaries of the primary strip in the *s*-plane. Figure 4.5 also shows the mapping of a constant-ω_n locus, in the primary strip of the *s*-plane, to the *z*-plane.

In the *z*-plane, the closed-loop poles must lie on the constant-ζ spiral to satisfy peak overshoot requirement, also the poles must lie on constant-ω_n curve to satisfy speed of response requirement. The intersection of the two curves (Fig. 4.5b) provides the preferred pole locations, and the design aim is to make the root locus pass through these locations.

If one chooses the following boundaries for the system response:

$$T = 1 \text{ sec}$$

$$\text{Peak overshoot} \le 15\% \Rightarrow \zeta \ge 0.5$$

$$\text{Settling time} \le 25 \text{ sec} \Rightarrow \omega_n \ge \frac{8}{25},$$

the acceptable boundaries for the closed-loop pole locations in *z*-plane are shown in Fig. 4.6.

In the chart of Fig. 4.6, the patterns are traced for various natural frequencies and damping ratios. Such a chart is a useful aid in root-locus design technique. We will be using this chart in our design examples.

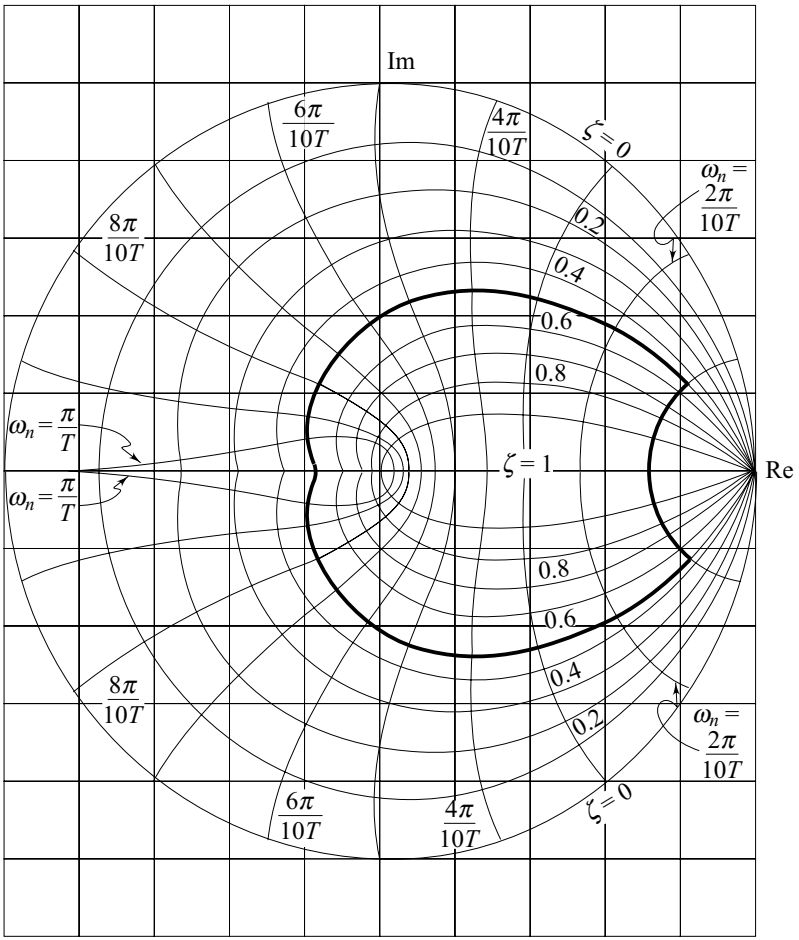


Fig. 4.6 Plot of an acceptable region for poles of a second-order system to satisfy dynamic response specifications

Dominant Poles

Most control systems found in practice are of high order. The preferred locations of closed-loop poles given by Fig. 4.5b, realize the specified transient performance *only* if the other closed-loop poles and zeros of the system have negligible effect on the dynamics of the system, i.e., only if the closed-loop poles corresponding to specified ζ and ω_n, are dominant.

In the following, we examine the relationship between the pole-zero patterns and the corresponding step-responses of discrete-time systems. Our attention will be restricted to the step responses of the discrete-time system with transfer function

$$\frac{Y(z)}{R(z)} = \frac{K(z-z_1)(z-z_2)}{(z-p)(z-re^{j\theta})(z-re^{-j\theta})} = \frac{K(z-z_1)(z-z_2)}{(z-p)(z^2-2r\cos\theta z+r^2)} \tag{4.16}$$

for a selected set of values of the parameters K, z_1, z_2, p, r and θ.

We assume that the roots of the equation

$$z^2 - 2r\cos\theta z + r^2 = 0$$

are the preferred closed-loop poles corresponding to the specified values of ζ and ω_n. Complex-conjugate pole pairs corresponding to $\zeta = 0.5$ with $\theta = 18^o$, 45^o and 72^o, will be considered in our study. The pole pair with $\theta = 18^o$ is shown in Fig. 4.7.

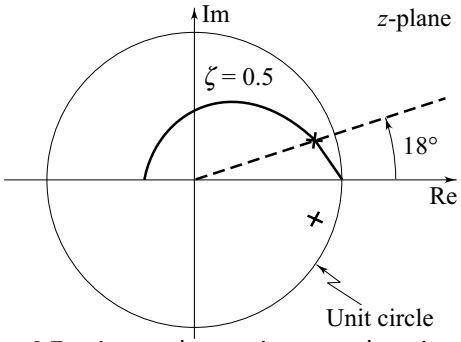


Fig. 4.7 A complex-conjugate pole pair of the system (4.16)

To study the effect of zero location, we let $z_2 = p$ and explore the effect of the (remaining) zero location z_1 on the transient performance. We take the gain K to be such that the steady-state output value equals the step size. For a unit-step input,

$$Y(z) = \left[\frac{K(z - z_1)}{z^2 - 2r\cos\theta\, z + r^2}\right]\left(\frac{z}{z-1}\right) \tag{4.17}$$

with

$$K = \frac{1 - 2r\cos\theta + r^2}{(1 - z_1)}$$

The major effect of the zero z_1 on the step response $y(k)$ is to change the peak overshoot, as may be seen from the step responses plotted in Fig. 4.8a. Figure 4.8b shows plots of peak overshoot *versus* zero location for three different cases of complex-conjugate pole pairs. The major observation from these plots is that the zero has very little influence when on the negative real axis, but its influence is dramatic when it comes near +1.

To study the influence of a third pole on a basically second-order response, we again consider the system (4.16), but this time, we fix $z_1 = z_2 = -1$ and let p vary from near –1 to near +1. In this case, the major influence of the moving singularity is on the rise time of the step response. Figure 4.8c shows plots of rise time *versus* extra pole location, for three different cases of complex-conjugate pole pairs. We see here that the extra pole causes the rise time to get very much longer as the location of p moves towards $z = +1$, and comes to dominate the response.

Our conclusions from these plots are that the addition of a pole, or a zero, to a given system has only a small effect—if the added singularities are in the range 0 to –1. However, a zero moving towards $z = +1$ greatly increases the system overshoot. A pole placed towards $z = +1$ causes the response to slow down and thus, primarily, affects the rise time—which is being progressively increased. The pole pair corresponding to specified ζ and ω_n, is a dominant pole pair of the closed-loop system only if the influence of additional poles and zeros is negligibly small on the dynamic response of the system.

Specifications in Terms of Frequency Response Features

The translation of time-domain specifications into desired locations of pair of dominant closed-loop poles in the z-plane, is useful if the design is to be carried out by using root locus plots. The use of frequency response plots necessitates the translation of time-domain performance specifications in terms of frequency response features.

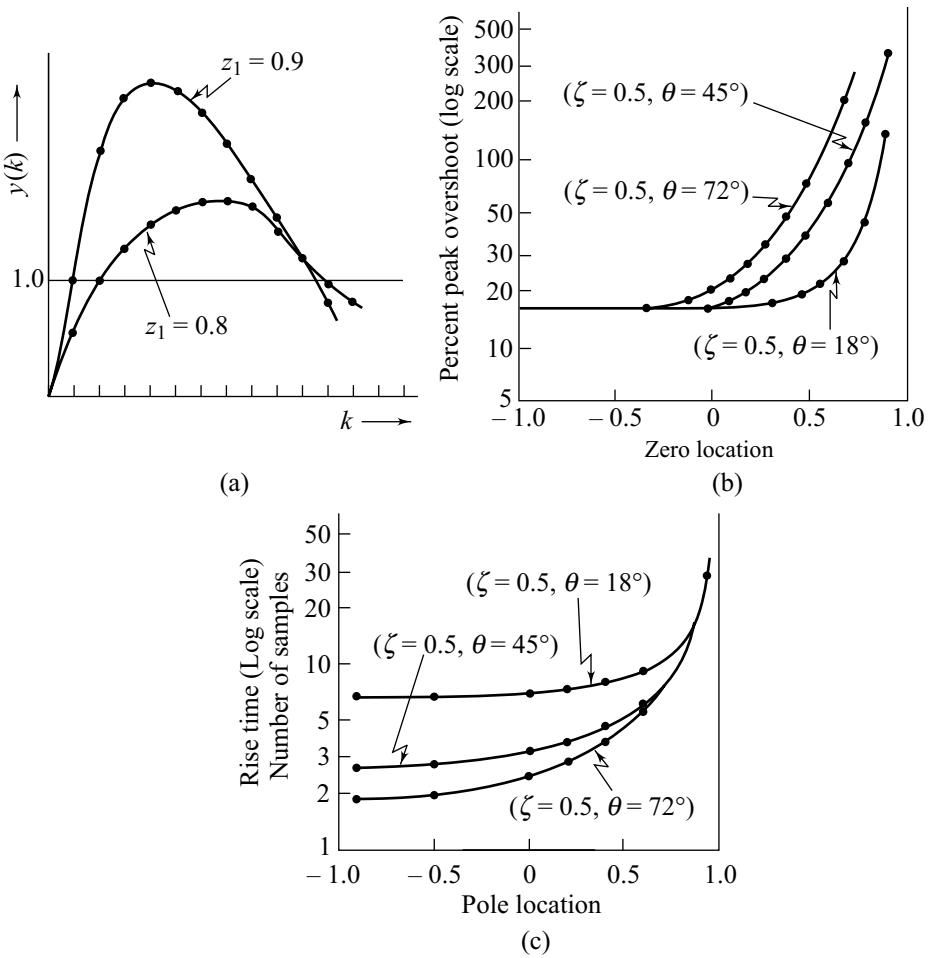


Fig. 4.8 (a) Effect of an extra zero on a discrete-time second-order system, $\zeta = 0.5$, $\theta = 18°$
(b) Effect of an extra zero on a discrete-time second-order system
(c) Effect of an extra pole on a discrete-time second-order system

All the frequency-domain methods of continuous-time systems can be extended for the analysis and design of digital control systems. Consider the system shown in Fig. 4.3b. The closed-loop transfer function of the sampled-data system is

$$\frac{Y(z)}{R(z)} = \frac{G_{h0}G(z)}{1 + G_{h0}G(z)} \tag{4.18}$$

Just as in the case of continuous-time systems, the absolute and relative stability conditions of the closed-loop discrete-time system can be investigated by making the frequency response plots of $G_{h0}G(z)$. The frequency response plots of $G_{h0}G(z)$ are obtained by setting $z = e^{j\omega T}$, and then letting ω vary from $-\omega_s/2$ to $\omega_s/2$. This is equivalent to mapping the unit circle in the z-plane onto the $G_{h0}G(e^{j\omega T})$-plane. Since the unit circle in the z-plane is symmetrical about the real axis, the frequency response plot of $G_{h0}G(e^{j\omega T})$

will also be symmetrical about the real axis, so that only the portion that corresponds to $\omega = 0$ to $\omega = \omega_s/2$ needs to be plotted.

A typical curve of (refer to Eqn. (4.18))

$$\frac{Y}{R}(e^{j\omega T}) = \frac{G_{h0}G(e^{j\omega T})}{1+G_{h0}G(e^{j\omega T})}, \tag{4.19}$$

the closed-loop frequency response, is shown in Fig. 4.9. The amplitude ratio and phase angle will approximate the ideal $1.0 \angle 0^\circ$ for some range of 'low' frequencies, but will deviate for high frequencies. The height M_r (resonance peak) of the peak is a relative stability criterion; the higher the peak, the poorer the relative stability. Many systems are designed to exhibit a resonance peak in the range 1.2 to 1.4. The frequency ω_r (resonance frequency) at which this peak occurs, is a speed of response criterion; the higher the ω_r, the faster the system. For systems that exhibit no peak (sometimes the case), the bandwidth ω_b is used for speed of response specifications. Bandwidth is the frequency at which amplitude ratio has dropped to $1/\sqrt{2}$ times its zero-frequency value. It can, of course, be specified even if there is a peak.

Alternative measures of relative stability and speed of response are stability margins and crossover frequencies. To define these measures, a discussion of the Nyquist stability criterion in the z-plane is

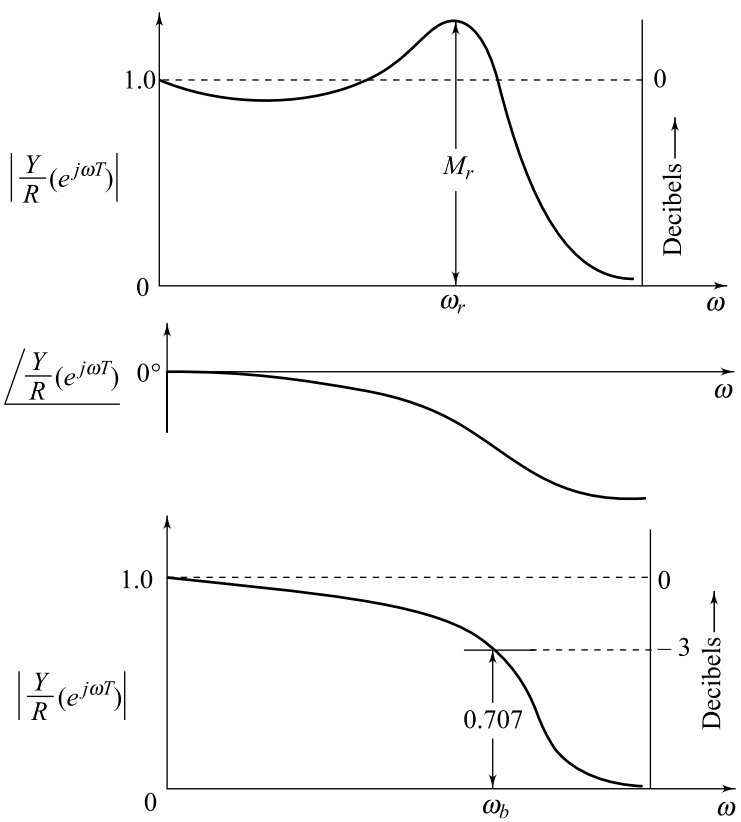


Fig. 4.9 Closed-loop frequency response criteria

required. Given the extensive foundation for the Nyquist criterion for continuous-time systems, that we laid in Chapter 10 of the companion book [155], it will not take us long to present the criterion for the discrete-time case.

4.2.3 Nyquist Stability Criterion in the *z*-Plane

The concepts involved in *z*-plane Nyquist stability criterion, are identical to those for *s*-plane criterion. In the *s*-plane, the region of stability is infinite in extent, namely, the entire left half of the *s*-plane. In the *z*-plane, this is not the case. The region of stability is the interior of the unit circle. This makes drawing the locus of $G_{h0}G(e^{j\omega T})$, the open-loop frequency response on the polar plane, easier because the Nyquist contour Γ_z in the *z*-plane is finite in extent, being simply the unit circle. We treat poles at $z = 1$ in the way we treated poles at $s = 0$, by detouring around them on a contour of arbitrarily small radius.

Figure 4.10a shows a typical Nyquist contour along which we will evaluate $G_{h0}G(z)$. Note that we detour around the pole at $z = 1$, on a portion of a circle of radius ε centered at $z = 1$. A typical Nyquist plot $G_{h0}G(e^{j\omega T})$ is shown in Fig. 4.10b. We see from this figure, that the Nyquist plot is similar to those we obtain for continuous-time functions with a single pole at $s = 0$, with the following exception. The plot does *not* touch the origin in the *z*-plane. The reason is that we evaluate $G_{h0}G(e^{j\omega T})$ over a finite range of values of ω, namely, $0 \le \omega \le \pi/T$, where T is the sampling interval.

We have labeled the segments of Γ_z in the same fashion as we did in the *s*-plane Nyquist analysis [155]. Segment C_1 is the upper half of the unit circle, and segment C_2 is the lower half of the unit circle. Segment C_3 is the portion of a circle with radius ε centered at $z = 1$. There is no segment corresponding to the *s*-plane portion of a circle with infinite radius centered at $s = 0$, because the Nyquist contour Γ_z in the *z*-plane, unlike its counterpart in the *s*-plane, is of finite extent.

Note that the locus $G_{h0}G(C_1)$ in Fig. 4.10b, is directly obtained from $G_{h0}G(e^{j\omega T})$ for $0 \le \omega \le \pi/T$, whereas the locus $G_{h0}G(C_2)$ is the same information, with the phase reflected about 180°; $G_{h0}G(C_3)$ is inferred from Fig. 4.10a based on pole-zero configuration.

In the case of open-loop transfer function $G_{h0}G(z)$ with no poles outside the unit circle, the closed-loop system of Fig. 4.3b is stable if

$N =$ number of clockwise encirclements of the critical point $-1 + j0$ made by the $G_{h0}G(e^{j\omega T})$ locus of the Nyquist plot

$= 0$

Note that the necessary information to determine relative stability is contained in the portion $G_{h0}G(C_1)$ of the Nyquist plot, which corresponds to the frequency response of the open-loop system $G_{h0}G(z)$. This portion of the Nyqusit plot of Fig. 4.10b has been redrawn in Fig. 4.10c. Gain and phase margins are defined so as to provide a two-dimensional measure of how close the Nyquist plot is to encircling the $-1 + j0$ point, and they are identical to the definitions developed for continuous-time systems. The Gain Margin (GM) is the inverse of the amplitude of $G_{h0}G(e^{j\omega T})$ when its phase is 180°, and is a measure of how much gain of the system can be increased before instability results. The Phase Margin (ΦM) is the difference between 180° and the phase of $G_{h0}G(e^{j\omega T})$ when its amplitude is 1. It is a measure of how much additional phase lag, or time delay, can be tolerated before instability results, because the phase of a system is highly related to these characteristics.

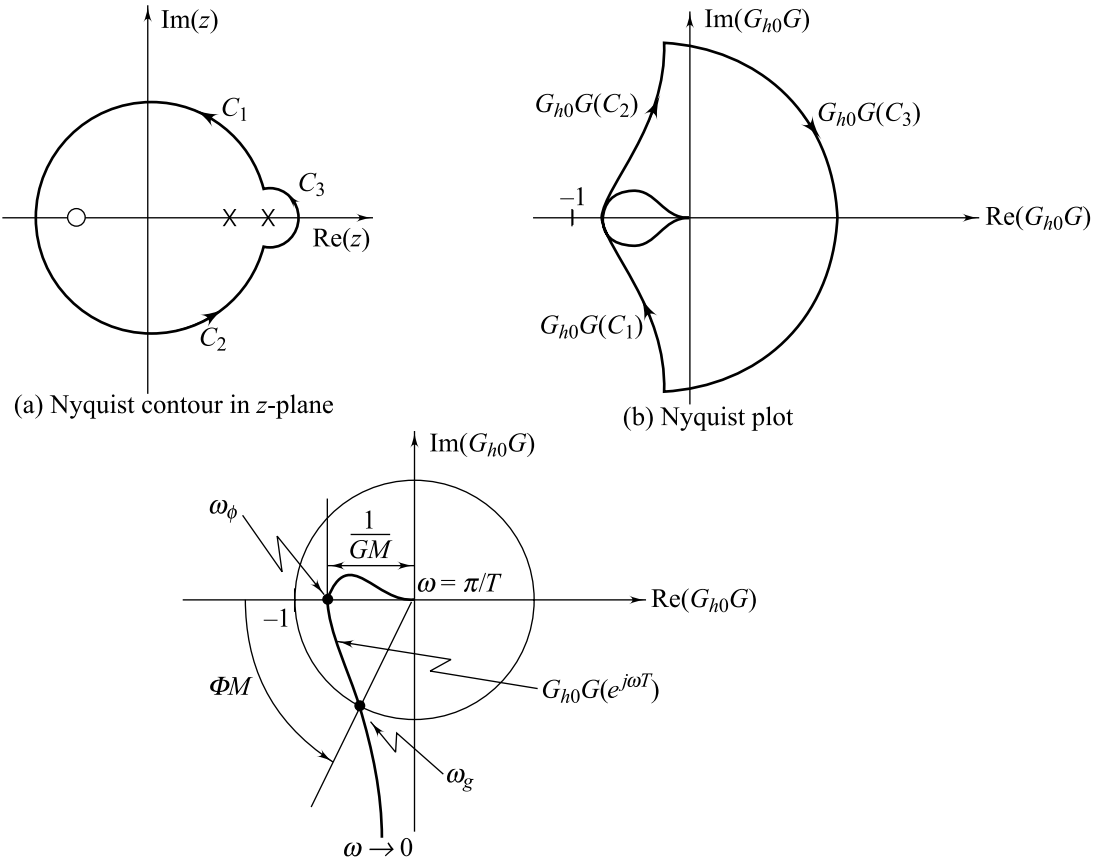


Fig. 4.10

The Nyquist plot in Fig. 4.10c intersects the negative real axis at frequency ω_ϕ. This frequency at which the phase angle of $G_{h0}G(e^{j\omega T})$ is 180°, is referred to as *phase crossover frequency*. The gain margin of the closed-loop system of Fig. 4.3b, is defined as the number

$$GM = \frac{1}{\left|G_{h0}G(e^{j\omega_\phi T})\right|}$$

For stable systems, GM is always a number greater than one.

A unit circle, centered at the origin, has been drawn in Fig. 4.10c in order to identify the point at which the Nyquist plot has unity magnitude. The frequency at this point has been designated ω_g, the *gain crossover frequency*. The phase margin of the closed-loop system of Fig. 4.3b, is defined as

$$\Phi M = 180° + \angle G_{h0}G(e^{j\omega_g T})$$

For stable systems, ΦM is always positive.

The GM and ΦM are both measures of relative stability. General numerical design goals for these margins cannot be given since systems that satisfy other specific performance criteria may exhibit a wide range

of these margins. It is possible, however, to give useful lower bounds—the gain margin should usually exceed 2.5 and the phase margin should exceed 30°.

For continuous-time systems, it is often pointed out that the phase margin is related to the damping ratio ζ for a standard second-order system; the approximate relation being $\zeta = \Phi M/100$. The ΦM, from a z-plane frequency response analysis, carries the same implications about the damping ratio of the closed-loop system.

The gain crossover frequency ω_g is related to the bandwidth of the system. The larger the ω_g, the wider the bandwidth of the closed-loop system, and the faster is its response.

The translation of time-domain specifications in terms of frequency response features, is carried out by using the explicit correlations for second-order system (4.10). The following correlations are valid approximations for higher-order systems dominated by a pair of complex conjugate poles[3].

$$M_r = \frac{1}{2\zeta\sqrt{1-\zeta^2}}\,;\ \zeta \le 0.707 \tag{4.20}$$

$$\omega_r = \omega_n\sqrt{1-2\zeta^2} \tag{4.21}$$

$$\omega_b = \omega_n\left[1-2\zeta^2+\sqrt{(2-4\zeta^2+4\zeta^4)}\right]^{\frac{1}{2}} \tag{4.22}$$

$$\Phi M = \tan^{-1}\left\{2\zeta\Big/\left[\sqrt{1+4\zeta^4}-2\zeta^2\right]^{\frac{1}{2}}\right\} \cong 100\zeta \tag{4.23}$$

4.2.4 Disturbance Rejection

The effectiveness of a system in disturbance signal rejection is readily studied with the topology of Fig. 4.11a. The response $Y(z)$ to disturbance $W(z)$, can be found from the closed-loop transfer function

$$\frac{Y(z)}{W(z)} = \frac{1}{1+D(z)G_{h0}G(z)} \tag{4.24a}$$

We now introduce the function

$$S(z) = \frac{1}{1+D(z)G_{h0}G(z)} \tag{4.24b}$$

which we call the *sensitivity function* of the control system, for reasons to be explained later. To reduce the effects of disturbances, it turns out that $S(e^{j\omega T})$ must be made small over the frequency band of disturbances. If constant disturbances are to be suppressed, $S(1)$ should be made small. If $D(z)G_{h0}G(z)$ includes an integrator (which means that $D(z)$ or $G_{h0}G(z)$ has a pole at $z = 1$), then the steady-state error due to constant disturbance is zero. This may be seen as follows. Since for a constant disturbance of amplitude A, we have

$$W(z) = \frac{Az}{z-1},$$

[3] Chapter 11 of reference [155].

the steady-state value of the output is given by

$$y_{ss} = \lim_{z \to 1} (z-1)Y(z) = \lim_{z \to 1} (z-1)S(z)W(z) = \lim_{z \to 1} AS(z)$$

which is equal to zero if $D(z)G_{h0}G(z)$ has a pole at $z = 1$.

Note that the point where the disturbance enters the system is very important in adjusting the gain of $D(z)G_{h0}G(z)$. For example, consider the system shown in Fig. 4.11b. The closed-loop transfer function for the disturbance is

$$\frac{Y(z)}{W(z)} = \frac{G_{h0}G(z)}{1 + D(z)G_{h0}G(z)}$$

In this case, the steady-state error due to constant disturbance $W(z)$ is not equal to zero when $G_{h0}G(z)$ has a pole at $z = 1$. This may be seen as follows:

Let $$G_{h0}G(z) = Q(z)/(z-1)$$

where $Q(z)$ is a rational polynomial of z, such that $Q(1) \neq 0$ and $Q(1) \neq \infty$; and $D(z)$ is a controller which does not have pole at $z = 1$. Then

$$y_{ss} = \lim_{z \to 1} \frac{(z-1)G_{h0}G(z)}{1 + D(z)G_{h0}G(z)} W(z) = \lim_{z \to 1} \frac{AzQ(z)}{z - 1 + D(z)Q(z)} = \frac{A}{D(1)}$$

Thus, the steady-state error is nonzero; the magnitude of the error can be reduced by increasing the controller gain.

Figure 4.11c gives a block diagram of the situation where measurement noise $W_n(z)$ enters the system through the feedback link. The closed-loop transfer function for this disturbance is

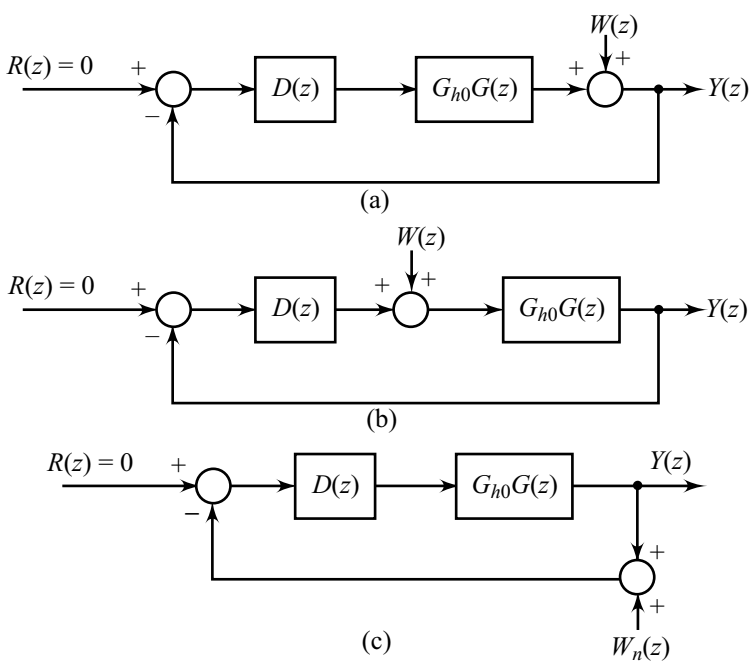


Fig. 4.11 Disturbance rejection

$$\frac{Y(z)}{W_n(z)} = \frac{D(z)G_{h0}G(z)}{1+D(z)G_{h0}G(z)} \tag{4.25}$$

Thus, the measurement noise is transferred to the output whenever $|D(z)G_{h0}G(z)| > 1$. Hence, large gains of $D(z)G_{h0}G(z)$ will lead to large output errors due to measurement noise. This is in conflict with the disturbance-rejection property with respect to configurations of Figs 4.11a and 4.11b. To solve this problem, we can generally examine the measuring instrument and modify the filtering, so that it satisfies the requirements of a particular control problem.

4.2.5 Insensitivity and Robustness

Finally, in our design, we must take into account both the small and, often, the large differences between the derived process model and the real process behavior. The differences may appear due to modeling approximations, and the process behavior changes with time during operation. If, for simplicity, it is assumed that the structure and order of the process model are chosen exactly, and they do not change with time, then these differences are manifested as parameter errors.

Parameter changes with respect to nominal parameter vector $\boldsymbol{\theta}_n$ are assumed. The closed-loop behavior for parameter vector

$$\boldsymbol{\theta} = \boldsymbol{\theta}_n + \Delta\boldsymbol{\theta}$$

is of interest. If the parameter changes are small, then sensitivity methods can be used. For controller design, both good control performance (steady-state accuracy, transient accuracy, and disturbance rejection), and small parameter sensitivity, are required. The resulting controllers are then referred to as *insensitive controllers*. However, for large parameter changes, the sensitivity design is unsuitable. Instead, one has to assume several process models with different parameter vectors $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, ..., $\boldsymbol{\theta}_M$, and try to design a *robust controller* which, for all process models, will maintain stability and certain control performance range.

For the design of insensitive controllers, the situation is very much like the disturbance-signal rejection. The larger the gain of the feedback loop around the offending parameter, the lower the sensitivity of the closed-loop transfer function to changes in that parameter.

Consider the digital control system of Fig. 4.11. The closed-loop input-output behavior corresponding to the nominal parameter vector, is described by

$$M(\boldsymbol{\theta}_n, z) = \frac{Y(z)}{R(z)} = \frac{D(z)G_{h0}G(\boldsymbol{\theta}_n, z)}{1+D(z)G_{h0}G(\boldsymbol{\theta}_n, z)} \tag{4.26}$$

The process parameter vector now changes by an infinitesimal value $\Delta\boldsymbol{\theta}$. For the control loop, it follows that

$$\left.\frac{\partial M(\boldsymbol{\theta}, z)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n} = \frac{D(z)}{[1+D(z)\,G_{h0}G(\boldsymbol{\theta}_n, z)]^2} \left.\frac{\partial G_{h0}G(\boldsymbol{\theta}, z)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n}$$

For $$\Delta G_{h0}G(\boldsymbol{\theta}_n, z) = \left(\left.\frac{\partial G_{h0}G(\boldsymbol{\theta}, z)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n}\right)^T \Delta\boldsymbol{\theta},$$

$$\Delta M(\boldsymbol{\theta}_n, z) = \frac{D(z)}{[1+D(z)G_{h0}G(\boldsymbol{\theta}_n, z)]^2}\left(\left.\frac{\partial G_{h0}G(\boldsymbol{\theta}, z)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n}\right)^T \Delta\boldsymbol{\theta}$$

$$= \left\{\frac{D(z)G_{h0}G(\boldsymbol{\theta}_n, z)}{1+D(z)G_{h0}G(\boldsymbol{\theta}_n, z)}\right\}\left\{\frac{1}{1+D(z)G_{h0}G(\boldsymbol{\theta}_n, z)}\right\}\left\{\frac{1}{G_{h0}G(\boldsymbol{\theta}_n, z)}\right\}\{\Delta G_{h0}G(\boldsymbol{\theta}_n, z)\} \quad (4.27)$$

From Eqns (4.26)–(4.27), it follows that

$$\frac{\Delta M(\boldsymbol{\theta}_n, z)}{M(\boldsymbol{\theta}_n, z)} = S(\boldsymbol{\theta}_n, z)\frac{\Delta G_{h0}G(\boldsymbol{\theta}_n, z)}{G_{h0}G(\boldsymbol{\theta}_n, z)} \quad (4.28a)$$

with the sensitivity function $S(\boldsymbol{\theta}_n, z)$ of the feedback control given as

$$S^M_{G_{h0}G} = S(\boldsymbol{\theta}_n, z) = \frac{1}{1+D(z)G_{h0}G(\boldsymbol{\theta}_n, z)} \quad (4.28b)$$

This sensitivity function shows how relative changes of input/output behavior of a closed loop, depend on changes of the process transfer function. Small parameter-sensitivity of the closed-loop behavior, can be obtained by making $S(\boldsymbol{\theta}_n, e^{j\omega T})$ small in the significant frequency range.

4.2.6 The Case for High-Gain Feedback

Control system design with high-gain feedback results in the following:

(i) good steady-state tracking accuracy;
(ii) good disturbance-signal rejection; and
(iii) low sensitivity to process-parameter variations.

There are, however, factors limiting the gain:

(i) High gain may result in instability problems.
(ii) Input amplitudes limit the gain; excessively large magnitudes of control signals will drive the process to saturation region of its operation, and the control system design, based on linear model of the plant, will no longer give satisfactory performance.
(iii) Measurement noise limits the gain; with high-gain feedback, measurement noise appears unattenuated in the controlled output.

Therefore, in design, we are faced with trade-offs.

4.3 DIGITAL COMPENSATOR DESIGN USING FREQUENCY RESPONSE PLOTS

All the frequency response methods of continuous-time systems[4], are directly applicable for the analysis and design of digital control systems. For a system with closed-loop transfer function

$$\frac{Y(z)}{R(z)} = \frac{G_{h0}G(z)}{1+G_{h0}G(z)} \quad (4.29)$$

[4] Chapters 10–12 of reference [155].

the absolute and relative stability conditions can be investigated by making the frequency response plots of $G_{h0}G(z)$. The frequency response plots of $G_{h0}G(z)$ can be obtained by setting

$$z = e^{j\omega T};\ T = \text{sampling interval} \tag{4.30}$$

and then letting the frequency ω vary from $-\omega_s/2$ to $\omega_s/2$; $\omega_s = 2\pi/T$. Computer assistance is normally required to make the frequency response plots (refer to Problem A.8 in Appendix A).

Since the frequency appears in the form $z = e^{j\omega T}$, the discrete-time transfer functions are typically not rational functions and the simplicity of Bode's design technique is altogether lost in the z-plane. The simplicity can be regained by transforming the discrete-time transfer function in the z-plane, to a different plane (called w) by the bilinear transformation (refer to Eqn. (2.115))

$$z = \frac{1 + wT/2}{1 - wT/2} \tag{4.31a}$$

By solving Eqn. (4.31a) for w, we obtain the inverse relationship

$$w = \frac{2}{T}\frac{z-1}{z+1} \tag{4.31b}$$

Through the z-transformation and the w-transformation, the primary strip of the left half of the s-plane is first mapped into the inside of the unit circle in the z-plane, and then mapped into the entire left half of the w-plane. The two mapping processes are depicted in Fig. 4.12. Notice that as s varies from 0 to $j\omega_s/2$ along the $j\omega$-axis in the s-plane, z varies from 1 to -1 along the unit circle in the z-plane, and w varies from 0 to ∞ along the imaginary axis in the w-plane. The bilinear transformation (4.31) does not have any physical significance in itself and, therefore, all w-plane quantities are fictitious quantities that correspond to the physical quantities of either the s-plane or the z-plane. The correspondence between the real frequency ω, and the fictitious w-plane frequency, denoted as ν, is obtained as follows:

From Eqn. (4.31b),

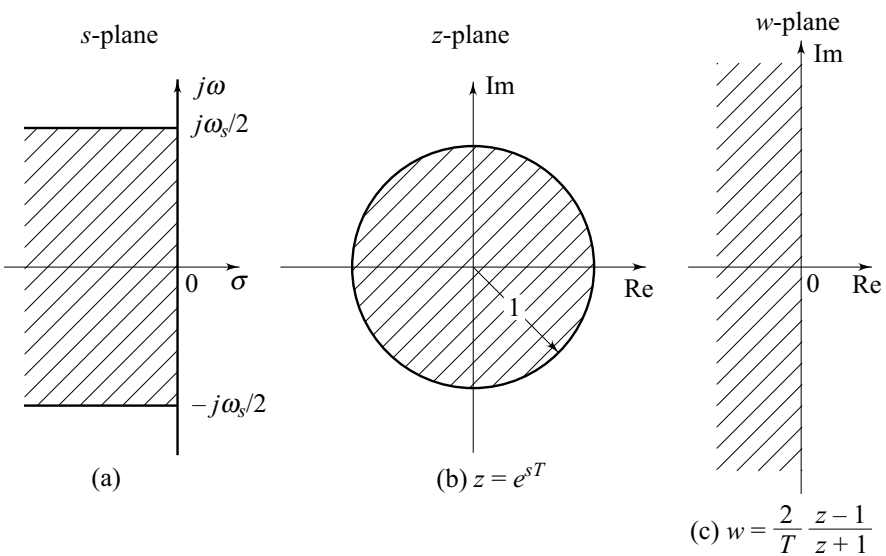


Fig. 4.12 Diagrams showing mappings from s-plane to z-plane and from z-plane to w-plane

$$j\nu = \frac{2}{T}\frac{e^{j\omega T}-1}{e^{j\omega T}+1} = \frac{2}{T}\frac{e^{j\omega T/2}-e^{-j\omega T/2}}{e^{j\omega T/2}+e^{-j\omega T/2}} = \frac{2}{T}j\tan\frac{\omega T}{2}$$

or

$$\nu = \frac{2}{T}\tan\frac{\omega T}{2} \tag{4.32}$$

Thus, a nonlinear relationship or 'warping' exists between the two frequencies ω and ν. As ω moves from 0 to $\omega_s/2$, ν moves from 0 to ∞ (Fig. 4.13).

Note that for relatively small $\frac{\omega T}{2}\left(\frac{\omega T}{2} < 17^{\circ} \text{ or about } 0.3 \text{ rad}\right)$,

$$\nu \cong \frac{2}{T}\left(\frac{\omega T}{2}\right) \cong \omega \tag{4.33}$$

and the 'warping' effect on the frequency response is negligible.

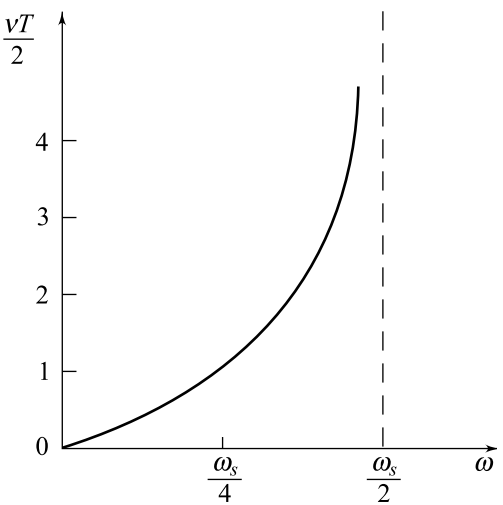


Fig. 4.13 Relationship between the fictitious frequency ν and actual frequency ω

The distortion depicted in Fig. 4.13 may be taken into account in our design of digital compensation by frequency 'prewarping'. The idea of prewarping is simply to adjust the critical frequencies in our design. For example, if the closed-loop bandwidth is specified as ω_b, then the corresponding bandwidth on the w-plane is $\nu_b = \frac{2}{T}\tan\left(\frac{\omega_b T}{2}\right)$. Our design based on the frequency response plots of $G_{h0}G(j\nu)$ attempts to realize a closed-loop bandwidth equal to ν_b.

Example 4.2

Consider a process with transfer function

$$G(s) = \frac{10}{s(\frac{1}{5}s+1)} \tag{4.34}$$

which, when preceded by a ZOH ($T = 0.1$ sec), has the discrete-time transfer function (refer to Table 2.1)

$$G_{h0}G(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{50}{s^2(s+5)}\right] = \frac{0.215(z+0.85)}{(z-1)(z-0.61)} \tag{4.35}$$

By use of the bilinear transformation

$$z = \frac{1+\dfrac{wT}{2}}{1-\dfrac{wT}{2}} = \frac{1+0.05w}{1-0.05w}$$

$G_{h0}G(z)$ can be transformed into $G_{h0}G(w)$ given below.

$$G_{h0}G(w) = \frac{10\left(1-\dfrac{w}{20}\right)\left(1+\dfrac{w}{246.67}\right)}{w\left(1+\dfrac{w}{4.84}\right)} \tag{4.36}$$

Notice that the gain of $G_{h0}G(w)$ is precisely the same as that of $G(s)$—it is 10 in both the cases. This will always be true for a $G_{h0}G(w)$ computed using the bilinear transformation given by Eqns (4.31). The gain of 10 in Eqn. (4.36) is the K_v of the uncompensated system (4.35).

We also note that in Eqn. (4.36), the denominator looks very much similar to that of $G(s)$, and that the denominators will be the same as T approaches zero. This would also have been true for any zeros of $G_{h0}G(w)$ that corresponded to zeros of $G(s)$, but our example does not have any. Our example also shows the creation of a right-half plane zero of $G_{h0}G(w)$ at $2/T$, and the creation of a fast left-half plane zero when compared to the original $G(s)$. The transfer function $G_{h0}G(w)$ is thus a *nonminimum phase* function.

To summarize, the w-transformation maps the inside of the unit circle in the z-plane, into the left half of the w-plane. The magnitude and phase of $G_{h0}G(j\nu)$ correspond to the magnitude and phase of $G_{h0}G(z)$ as z takes on values around the unit circle. Since $G_{h0}G(j\nu)$ is a rational function of ν, we can apply all the standard straight-line approximations to the log-magnitude and phase curves.

To obtain $$G_{h0}G(w) = G_{h0}G(z)\Big|_{z=\frac{1+wT/2}{1-wT/2}}$$

the following ready-to-use formula may be used:

$$G_{h0}G(z) = \frac{K\prod_{i=1}^{m}(z+a_i)}{(z-1)^l\prod_{j=1}^{n}(z+b_j)} \tag{4.37a}$$

$$G_{h0}G(w) = \frac{K\prod_{i=1}^{m}(1+a_i)\left(1-\dfrac{w}{2/T}\right)^{l-m+n}\prod_{i=1}^{m}\left(1+\dfrac{w}{(2/T)\,[(1+a_i)/(1-a_i)]}\right)}{\prod_{j=1}^{n}(1+b_j)T^l w^l\prod_{j=1}^{n}\left(1+\dfrac{w}{(2/T)[(1+b_j)/(1-b_j)]}\right)} \tag{4.37b}$$

The design of analog control systems usually falls into one of the following categories: (1) lead compensation, (2) lag compensation, (3) lag-lead compensation. Other more complex schemes, of course, do exist, but knowing the effects of these three basic types of compensation, gives a designer much insight into the design problem. With reference to the design of digital control systems by Bode plots, the basic forms of compensating network $D(w)$ have also been classified as lead, lag, and lag-lead. In the following paragraphs, we briefly review the fundamental frequency-domain features of these compensators.

Lead Compensation

A simple lead compensator model in the w-plane is described by the transfer function

$$D(w) = \frac{1+w\tau}{1+\alpha w\tau};\ 0<\alpha<1,\ \tau>0 \tag{4.38}$$

The zero-frequency gain of the compensator is found by letting $w = 0$. Thus, in Eqn. (4.38), we are assuming a unity zero-frequency gain for the compensator. Most of the designs require a compensator with a non-unity zero-frequency gain to improve steady-state response, disturbance rejection, etc. A non-unity zero-frequency gain is obtained by multiplying the right side of Eqn. (4.38) by a constant equal to the value of the desired zero-frequency gain. For the purpose of simplifying the design procedure, we normally add the required increase in gain to the plant transfer function, and design the unity zero-frequency gain compensator given by Eqn. (4.38), based on the new plant transfer function. Then the compensator is realized as the transfer function of (4.38) multiplied by the required gain factor.

The Bode plot of the unity zero-frequency gain lead compensator is shown in Fig. 4.14. The maximum phase lead ϕ_m of the compensator is given by the relation

$$\alpha = \frac{1-\sin\phi_m}{1+\sin\phi_m} \tag{4.39}$$

and it occurs at the frequency

$$\nu_m = \sqrt{\left(\frac{1}{\tau}\right)\left(\frac{1}{\alpha\tau}\right)} \tag{4.40}$$

The magnitude of $D(j\nu)$ at $\nu = \nu_m$ is $20\log(1/\sqrt{\alpha})$.

The phase lead is introduced in the vicinity of the gain crossover frequency of the uncompensated system—in order to increase the system's phase margin. Lead compensation increases the system gain at higher frequencies, thereby increasing the system bandwidth and hence the speed of response. However, a system with large bandwidth may be subjected to high-frequency noise problems.

Lag Compensation

A simple lag compensator model in the w-plane is described by the transfer function

$$D(w) = \frac{1+w\tau}{1+\beta w\tau};\ \beta>1,\ \tau>0 \tag{4.41}$$

The Bode plot of this unity zero-frequency gain compensator is shown in Fig. 4.15.

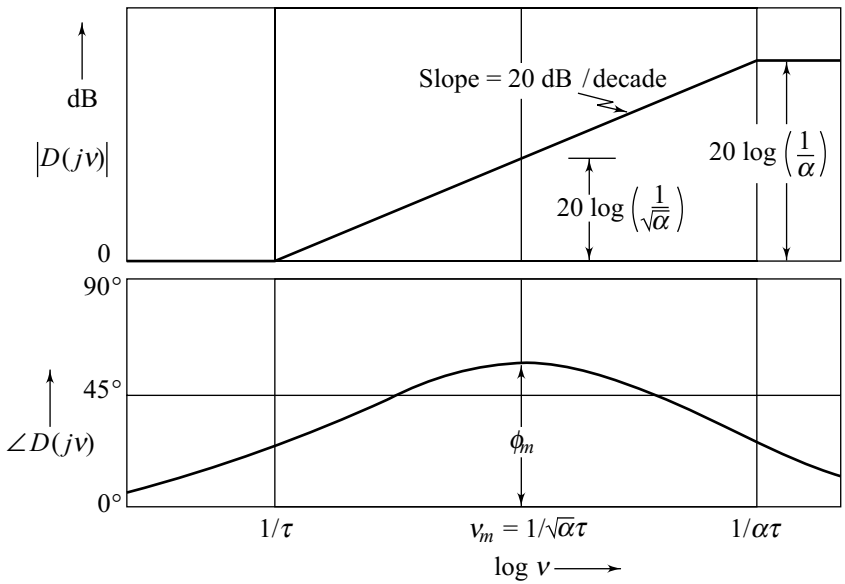


Fig. 4.14 Bode plot of lead compensator

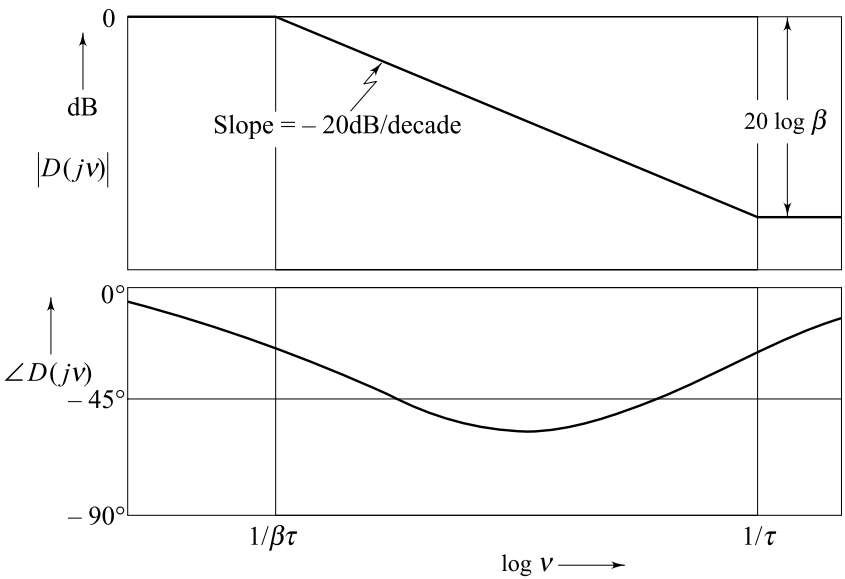


Fig. 4.15 Bode plot of lag compensator

Since the lag compensator reduces the system gain in the high frequency range, without reducing the gain at low frequencies, the total system gain can be appreciably increased by a non-unity zero-frequency gain obtained by multiplying the right side of Eqn. (4.41) by a constant. This is equivalent to increasing the gain for the entire frequency range, and then attenuating the magnitude curve in the high frequency region. This results in an appreciable increase in gain in the low frequency range of the lag-compensated system, thereby improving steady-state accuracy.

In the design method using Bode plots, the attenuation property of lag compensator is utilized; the phase lag characteristic is of no consequence. The attenuation provided by the lag compensator in the high frequency range shifts the gain crossover frequency to a lower value, and gives the system sufficient phase margin. So that a significant phase lag will not be contributed near the new gain crossover, the upper corner frequency $1/\tau$ of $D(w)$ is placed far below the new gain crossover.

With the reduction in system gain at high frequencies, the system bandwidth gets reduced and thus the system has a slower speed of response. This may be an advantage if high frequency noise is a problem.

Lag-Lead Compensation

Equations (4.38) and (4.41) describe simple first-order compensators. In many system design problems, however, the system specifications cannot be satisfied by a first-order compensator. In these cases, higher-order compensators must be used. To illustrate this point, suppose that smaller steady-state errors to ramp inputs are required for a Type-2 system; this requires an increase in the low-frequency gain of the system. If phase-lead compensation is employed, this increase in gain must be reflected at all frequencies. It is then unlikely that one first-order section of phase-lead compensation can be designed to give adequate phase margin. One solution to this problem would be to cascade two first-order lead compensators. However, if the noise in the control system is a problem, this solution may not be acceptable. A different approach is to cascade a lag compensator with a lead compensator. This compensator is usually referred to as a lag-lead compensator.

Example 4.3

Consider the feedback control system shown in Fig. 4.16. The plant is described by the transfer function

$$G(s) = \frac{K}{s(s+5)}$$

Design a digital control scheme for the system to meet the following specifications:

(i) the velocity error constant $K_v \geq 10$;
(ii) peak overshoot M_p to step input $\leq 25\%$; and
(iii) settling time t_s (2% tolerance band) ≤ 2.5 sec.

Solution The design parameters are the sampling interval T, the system gain K, and the parameters of the unity zero-frequency gain compensator $D(z)$.

Let us translate the transient accuracy requirements to frequency response measures. $\zeta = 0.4$ corresponds to a peak overshoot of about 25% (Eqn. (4.13)), and a phase margin of about 40º (Eqn. (4.23)). The requirement of $t_s \cong 2.5$ sec corresponds to $\omega_n = 4$ rad/sec (Eqn. (4.14)) and closed-loop bandwidth $\omega_b \cong 5.5$ rad/sec (Eqn. (4.22)). Taking the sampling frequency about 10 times the bandwidth, we choose the sampling interval

$$T = \frac{2\pi}{10\omega_b} \cong 0.1 \text{ sec}$$

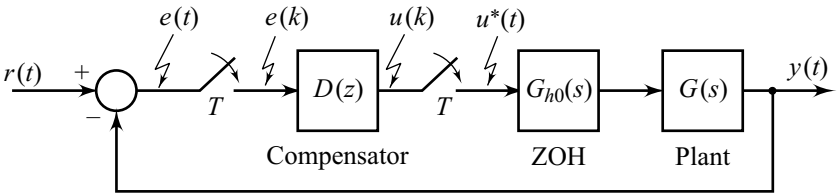


Fig. 4.16 A feedback control system with digital compensation

Our design approach is to first fix the system gain K to a value that results in the desired steady-state accuracy. A unity zero-frequency gain compensator, that satisfies the transient accuracy requirements without affecting the steady-state accuracy, is then introduced.

Since sampling does not affect the error constant of the system, we can relate K with K_v as follows, for the system of Fig. 4.16 with $D(z) = 1$ (i.e., for uncompensated system):

$$K_v = \lim_{s \to 0} sG(s) = \frac{K}{5}$$

Thus, $K = 50$ meets the requirements on steady-state accuracy.

For $T = 0.1$ and $K = 50$, we have

$$G_{h0}G(z) = \mathcal{Z}\left[\frac{1-e^{-Ts}}{s}\left(\frac{50}{s(s+5)}\right)\right] = \frac{0.215(z+0.85)}{(z-1)(z-0.61)} \tag{4.42}$$

$$G_{h0}G(w) = G_{h0}G(z)\Big|_{z=\frac{1+\frac{wT}{2}}{1-\frac{wT}{2}}} = \frac{10\left(1-\frac{w}{20}\right)\left(1+\frac{w}{246.67}\right)}{w\left(1+\frac{w}{4.84}\right)} \tag{4.43}$$

$$G_{h0}G(j\nu) = G_{h0}G(w)\Big|_{w=j\nu} = \frac{10\left(1-\frac{j\nu}{20}\right)\left(1+\frac{j\nu}{246.67}\right)}{j\nu\left(1+\frac{j\nu}{4.84}\right)} \tag{4.44}$$

The Bode plot of $G_{h0}G(j\nu)$ (i.e., the uncompensated system) is shown in Fig. 4.17. We find from this plot, that the uncompensated system has gain crossover frequency $\nu_{c1} = 6.6$ rad/sec and phase margin $\Phi M_1 \cong 20^\circ$. The magnitude versus phase angle curve of the uncompensated system is drawn in Fig. 4.18. The bandwidth[5] of the system is read as

$$\nu_{b1} = 11$$

In terms of the real frequency, the bandwidth (Eqn. (4.32))

[5] The –3dB closed-loop gain contour of the Nichols chart has been used to determine bandwidth. The contour has been constructed using the following table obtained from the Nichols chart.

Degrees	–90	–100	–120	–140	–160	–180	–200	–220
dB	0	–1.5	–4.18	–6.13	–7.28	–7.66	–7.28	–6.13

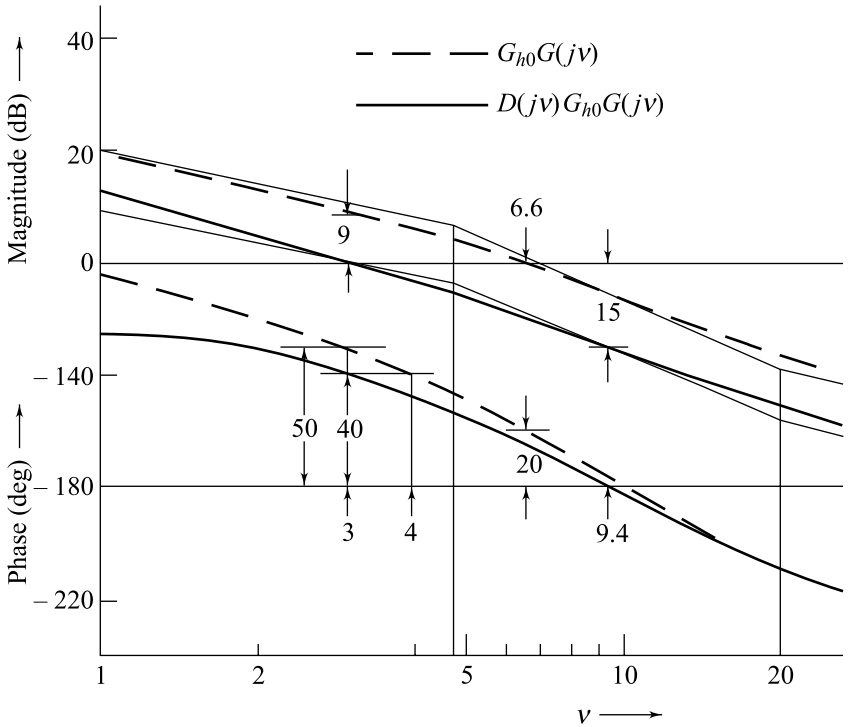


Fig. 4.17 Compensator design (Example 4.3)

$$\omega_{b1} = \frac{2}{T}\tan^{-1}\left(\frac{\nu_{b1}T}{2}\right) = 10 \text{ rad/sec}$$

It is desired to raise the phase margin to 40º without altering K_v. Also the bandwidth should not increase. Obviously, we should first try a lag compensator.

From the Bode plot of uncompensated system, we observe that the phase margin of 40º is obtained if the gain crossover frequency is reduced to 4 rad/sec. The high frequency gain –20 log β of the lag compensator (Fig. 4.15) is utilized to reduce the gain crossover frequency. The upper corner frequency $1/\tau$ of the compensator is placed one octave to one decade below the new gain crossover, so that the phase lag contribution of the compensator, in the vicinity of the new gain crossover, is made sufficiently small. To nullify the small phase lag contribution which will still be present, the gain crossover frequency is reduced to a value ν_{c2} where the phase angle of the uncompensated system is

$$\phi = -180^\circ + \Phi M_s + \varepsilon;$$

ΦM_s is the specified phase margin and ε is allowed a value 5º–15º.

The uncompensated system (Fig. 4.17) has a phase angle

$$\phi = -180^\circ + \Phi M_s + \varepsilon = -180^\circ + 40^\circ + 10^\circ = -130^\circ$$

at $\nu_{c2} = 3$ rad/sec. Placing the upper corner frequency of the compensator two octaves below ν_{c2}, we have

$$\frac{1}{\tau} = \frac{\nu_{c2}}{(2)^2} = \frac{3}{4}$$

To bring the magnitude curve down to 0 dB at ν_{c2}, the lag compensator must provide an attenuation of 9 dB (Fig. 4.17). Therefore,

$$20 \log \beta = 9 \text{ or } \beta = 2.82$$

The lower corner frequency of the compensator is then fixed at

$$\frac{1}{\beta\tau} = 0.266$$

The transfer function of the lag compensator is then

$$D(w) = \frac{1+\tau w}{1+\beta\tau w} = \frac{1+1.33w}{1+3.76w}$$

Phase lag introduced by the compensator at $\nu_{c2} = \tan^{-1}(1.33\ \nu_{c2}) - \tan^{-1}(3.76\nu_{c2}) = 75.93^\circ - 84.93^\circ = -9^\circ$. Therefore, the safety margin of $\varepsilon = 10^\circ$ is justified.

The open-loop transfer function of the compensated system becomes

$$D(w)G_{h0}G(w) = \frac{10\left(1-\frac{w}{20}\right)\left(1+\frac{w}{246.67}\right)\left(1+\frac{w}{0.75}\right)}{w\left(1+\frac{w}{4.84}\right)\left(1+\frac{w}{0.266}\right)}$$

The Bode plot of $D(w)G_{h0}G(w)$ is shown in Fig. 4.17, from where the phase margin of the compensated system is found to be 40° and the gain margin is 15 dB. The magnitude *versus* phase angle curve of the compensated system is shown on Nichols chart in Fig. 4.18. The bandwidth of the compensated system is

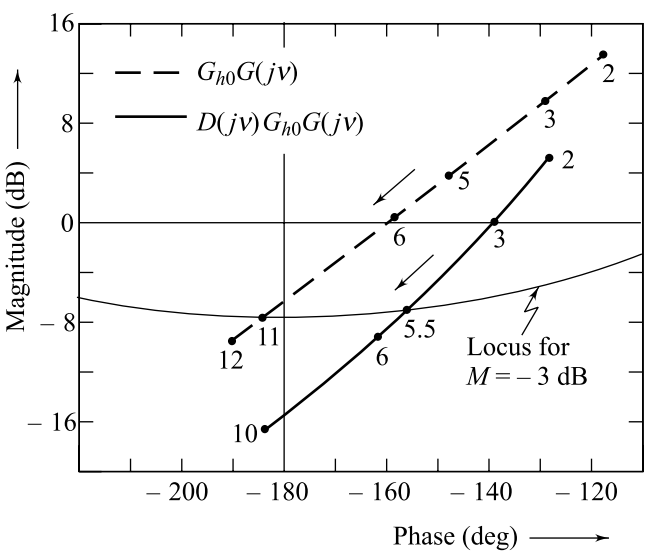


Fig. 4.18 Compensator design (Example 4.3)

$$\nu_{b2} = 5.5 \left(\omega_{b2} = \frac{2}{T} \tan^{-1} \left(\frac{\nu_{b2} T}{2} \right) = 5.36 \text{ rad/sec} \right)$$

Therefore, the addition of the compensator has reduced the bandwidth from 10 rad/sec to 5.36 rad/sec. However, the reduced value lies in the acceptable range.

Substituting $$w = \frac{2}{T} \frac{z-1}{z+1}$$

in $D(w)$, we obtain

$$D(z) = 0.362 \left(\frac{z - 0.928}{z - 0.974} \right) = \frac{0.362z - 0.336}{z - 0.974}$$

Zero-frequency gain of $D(z) = \lim_{z \to 1} \left[0.362 \left(\frac{z - 0.928}{z - 0.974} \right) \right] = 1$

The digital controller $D(z)$ has a pole-zero pair near $z = 1$. This creates a long tail of small amplitude in the step response of the closed-loop system. This behavior of the lag-compensated system will be explained shortly, with the help of root locus plots.

To evaluate the true effectiveness of the design, we write the closed-loop transfer function of the compensated system (Fig. 4.19) and therefrom obtain the response to step input. Computer assistance is usually needed for this analysis.

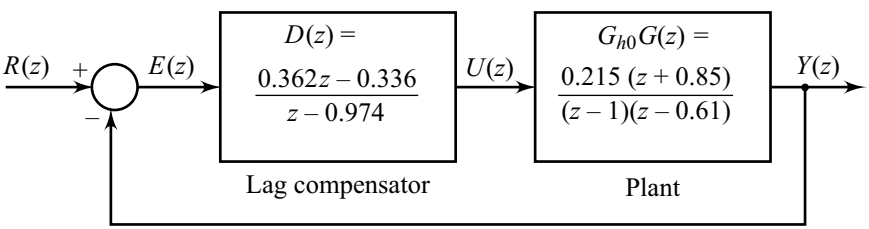


Fig. 4.19 Compensator design (Example 4.3)

Comment We have obtained a digital control algorithm which meets the following objectives: $K_v \cong 10$, $M_p \cong 25\%$, $t_s \cong 2.5$ sec. We may attempt to improve upon this design to obtain $K_v > 10$, $M_p < 25\%$ and $t_s < 2.5$ sec. However, the scope of such an exercise is limited, because the improvement in steady-state accuracy will be at the cost of stability margins and vice versa. Also, the conflicting requirements of limiting the magnitudes of control signals to avoid saturation problems, limiting the bandwidth to avoid high-frequency noise problems, etc., have to be taken into consideration.

Example 4.4

Reconsider the feedback control system of Example 4.3 (Fig. 4.16). We now set the following goal for our design:

(i) $K_v \geq 10$;

(ii) Phase margin ≅ 40°; and
(iii) Bandwidth ≅ 12 rad/sec.

Sampling interval $T = 0.1$ sec corresponds to a sampling frequency which is about five times the closed-loop bandwidth. A smaller value of T is more appropriate for the present design problem which requires higher speed of response; we will, however, take $T = 0.1$ sec to compare our results with those of Example 4.3.

Following the initial design steps of Example 4.3, we find that $K = 50$ meets the requirement on steady-state accuracy. For $K = 50$ and $T = 0.1$ sec, we have (refer to Eqn. (4.44))

$$G_{h0}G(j\nu) = \frac{10\left(1-\frac{j\nu}{20}\right)\left(1+\frac{j\nu}{246.67}\right)}{j\nu\left(1+\frac{j\nu}{4.84}\right)}$$

The uncompensated system has a gain crossover frequency $\nu_{c1} = 6.6$ rad/sec, phase margin $\Phi M_1 \cong 20°$ and bandwidth $\nu_{b1} = 11 (\omega_{b1} = 10$ rad/sec). This follows from Figs 4.20 and 4.21.

It is desired to raise the phase margin to 40° without altering K_v. The bandwidth should also increase. Obviously, we should try a lead compensator for this situation.

The phase margin $\Phi M_1 = 20°$ of the uncompensated system falls short of the specified phase margin $\Phi M_s = 40°$ by 20°. Additional phase margin can be provided by a lead compensator (Fig. 4.14), so placed

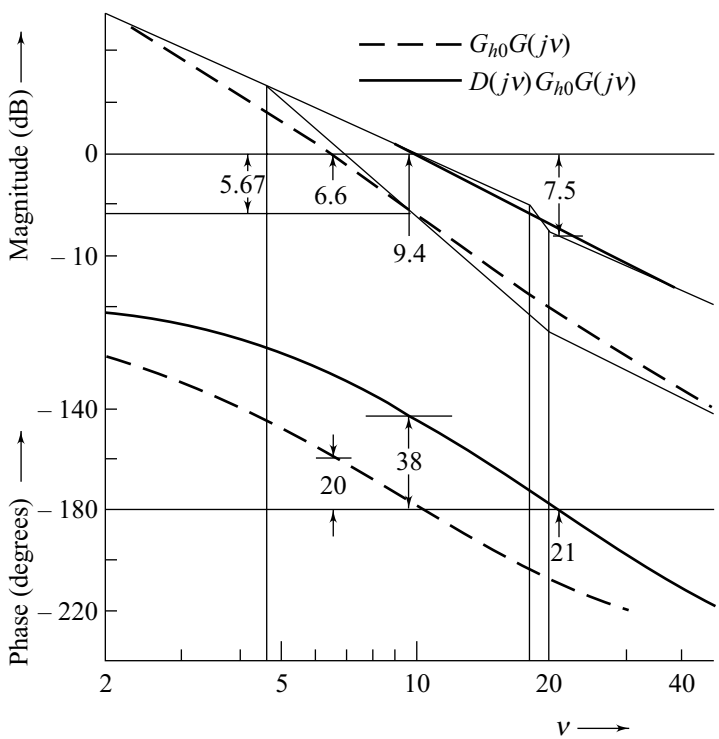


Fig. 4.20 Compensator design (Example 4.4)

that its corner frequencies $1/\tau$ and $1/\alpha\tau$ are on either side of the gain crossover frequency ν_{c1} = 6.6 rad/sec. The compensator so placed will increase the system gain in the vicinity of ν_{c1}; this will cause the gain crossover to shift to the right—to some unknown value ν_{c2}. The phase lead provided by the compensator at ν_{c2} adds to the phase margin of the system.

Phase margin of the uncompensated system at ν_{c1} is ΦM_1. At ν_{c2}, which is expected to be close to ν_{c1}, let us assume the phase margin of the uncompensated system to be $(\Phi M_1 - \varepsilon)$ where ε is allowed a value 5º – 15º. The phase lead required at ν_{c2} to bring the phase margin to the specified value ΦM_s, is given by

$$\phi_l = \Phi M_s - (\Phi M_1 - \varepsilon) = \Phi M_s - \Phi M_1 + \varepsilon$$

In our design, we will force the frequency ν_m of the compensator to coincide with ν_{c2}, so that maximum phase lead ϕ_m of the compensator is added to the phase margin of the system. Thus, we set

$$\nu_{c2} = \nu_m$$

Therefore,

$$\phi_m = \phi_l$$

The α-parameter of the compensator can then be computed from (refer to Eqn. (4.39))

$$\alpha = \frac{1 - \sin\phi_m}{1 + \sin\phi_m}$$

Since at ν_m, the compensator provides a dB-gain of $20 \log(1/\sqrt{\alpha})$, the new crossover frequency $\nu_{c2} = \nu_m$ can be determined as that frequency at which the uncompensated system has a dB-gain of $-20 \log(1/\sqrt{\alpha})$.

For the design problem under consideration,

$$\phi_l = 40^\circ - 20^\circ + 15^\circ = 35^\circ$$

Therefore,

$$\alpha = \frac{1 - \sin 35^\circ}{1 + \sin 35^\circ} = 0.271$$

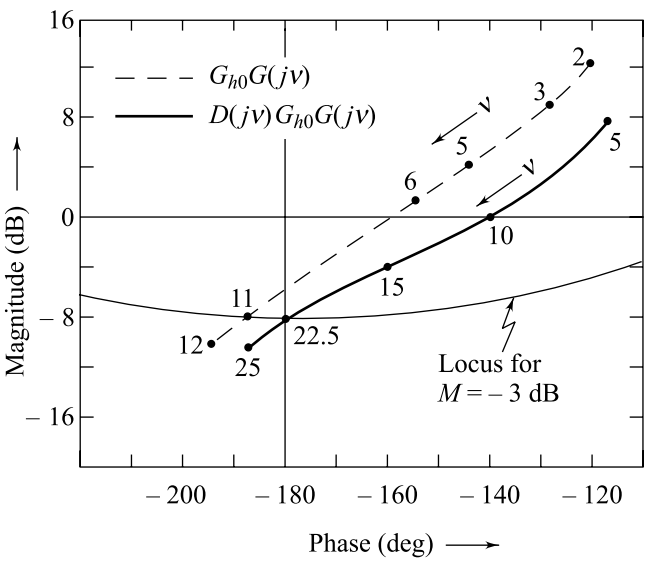


Fig. 4.21 Compensator design (Example 4.4)

The magnitude contribution of the compensator at ν_m is $20 \log (1/\sqrt{0.271}) = 5.67$dB.

From Bode plot of Fig. 4.20, we obtain

$$\nu_{c2} = 9.4 = \nu_m$$

Therefore (refer to Eqn. (4.40))

$$\sqrt{\left(\frac{1}{\tau}\right)\left(\frac{1}{\alpha\tau}\right)} = \nu_m = 9.4$$

or

$$\frac{1}{\tau} = \sqrt{\alpha}(\nu_m) = 4.893 \text{ and } \frac{1}{\alpha\tau} = \frac{4.893}{0.271} = 18.055$$

Since the compensator zero is very close to a pole of the plant, we may cancel the pole with the zero, i.e., we may choose

$$\frac{1}{\tau} = 4.84; \ \frac{1}{\alpha\tau} = 17.86$$

The transfer function of the lead compensator becomes

$$D(w) = \frac{1+\tau w}{1+\alpha\tau w} = \frac{1+0.21w}{1+0.056w}$$

Substituting

$$w = \frac{2}{T}\frac{z-1}{z+1}$$

in $D(w)$, we obtain

$$D(z) = \frac{2.45(z-0.616)}{z-0.057}$$

The open-loop transfer function of the compensated system is

$$D(w)G_{h0}G(w) = \frac{10\left(1-\frac{w}{20}\right)\left(1+\frac{w}{246.67}\right)}{w\left(1+\frac{w}{17.86}\right)}$$

The Bode plot of $D(w)G_{h0}G(w)$ is shown in Fig. 4.20, from where the phase margin of the compensated system is found to be 38º, and gain margin is 7.5 dB. The magnitude versus phase angle curve of the compensated system is shown in Fig. 4.21. The bandwidth of the compensated system is

$$\nu_{b2} = 22.5; \ \omega_{b2} = \frac{2}{T}\tan^{-1}\left(\frac{\nu_{b2}T}{2}\right) = 16.9 \text{ rad/sec}$$

Thus, the addition of the lead compensator has increased the system bandwidth from 10 to 16.9 rad/sec. It may lead to noise problems if the control system is burdened with high frequency noise.

A solution to noise problems involves the use of a lag compensator cascaded with lead compensator. The lag compensation is employed to realize a part of the required phase margin, thus reducing the amount of lead compensation required.

4.4 DIGITAL COMPENSATOR DESIGN USING ROOT LOCUS PLOTS

Design of compensation networks using the root locus plots is a well established procedure in analog control systems. This is essentially a trial-and-error method where, by varying the controller parameters, the roots of the characteristic equation are relocated to favorable locations. In the present section, we shall consider the application of root locus method to the design of digital control systems.

4.4.1 The Root Locus on the *z*-Plane

The characteristic equation of a discrete-time system can always be written in the form

$$1 + F(z) = 0 \tag{4.45}$$

where $F(z)$ is a rational function of z.

From Eqn. (4.45), it is seen that the roots of the characteristic equation (i.e., the closed-loop poles of the discrete-time system), occur only for those values of z where

$$F(z) = -1 \tag{4.46}$$

Since z is a complex variable, Eqn. (4.46) is converted into two conditions given below.

(i) *Magnitude condition*: $|F(z)| = 1$ (4.47a)

(ii) *Angle condition*: $\angle F(z) = \pm 180^{\circ} (2q + 1)$; $q = 0, 1, 2, ...$ (4.47b)

In essence, the construction of the z-plane root loci is to find the points that satisfy these conditions. If we write $F(z)$ in the standard pole-zero form:

$$F(z) = \frac{K \prod_i (z - z_i)}{\prod_j (z - p_j)}; K \geq 0 \tag{4.48a}$$

then the two conditions given in Eqns (4.47) become

$$|F(z)| = \frac{K \prod_i |z - z_i|}{\prod_j |z - p_j|} = 1 \tag{4.48b}$$

and

$$\angle F(z) = \sum_j \angle z - z_i - \sum_j \angle z - p_j = \pm 180^{\circ} (2q + 1); q = 0, 1, 2, \ldots \tag{4.48c}$$

Consequently, given the pole-zero configuration of $F(z)$, the construction of the root loci in the z-plane involves the following steps:

(i) A search for the points on the z-plane that satisfy the angle condition given by Eqn. (4.48c).

(ii) The value of K at a given point on a root locus is determined from the magnitude condition given by Eqn. (4.48b).

The root locus method developed for continuous-time systems can be extended to discrete-time systems without modifications, except that the stability boundary is changed from the $j\omega$ axis in the s-plane, to the unit circle in the z-plane. The reason the root locus method can be extended to discrete-time systems

is that the characteristic equation (4.45) for the discrete-time system, is of exactly the same form as the equation for root locus analysis in the *s*-plane. However, the pole locations for closed-loop systems in the *z*-plane must be interpreted differently from those in the *s*-plane.

We assume that the reader is already familiar with the *s*-plane root locus technique. We shall concentrate on the interpretation of the root loci in the *z*-plane with reference to the system performance, rather than the construction of root loci in the *z*-plane. Rules of construction of root loci are summarized in Table 4.2 for ready reference.[6]

Table 4.2 Rules for construction of Root Locus Plot of $1 + F(z) = 0$

$$F(z) = \frac{K \prod_{i=1}^{m} (z - z_i)}{\prod_{j=1}^{n} (z - p_j)}; \; K \geq 0, \; n \geq m; \; z_i\text{: } m \text{ open-loop zeros; } p_j\text{: } n \text{ open-loop poles}$$

(i) The root locus plot consists of n root loci as K varies from 0 to ∞. The loci are symmetric with respect to the real axis.

(ii) As K increases from zero to infinity, each root locus originates from an open-loop pole with $K = 0$, and terminates either on an open-loop zero or on infinity with $K = \infty$. The number of loci terminating on infinity equals the number of open-loop poles minus zeros.

(iii) The $(n - m)$ root loci which tend to infinity, do so along straight-line asymptotes radiating out from a single point $z = -\sigma_A$ on the real axis (called the centroid), where

$$-\sigma_A = \frac{\Sigma(\text{real parts of open-loop poles}) - \Sigma(\text{real parts of open-loop zeros})}{n - m}$$

These $(n - m)$ asymptotes have angles

$$\phi_A = \frac{(2q+1)180^\circ}{n - m}; \; q = 0, 1, 2, ..., (n - m - 1)$$

(iv) A point on the real axis lies on the locus if the number of open-loop poles plus zeros on the real axis to the right of this point, is odd. By use of this fact, the real axis can be divided into segments *on-locus* and *not-on-locus*; the dividing points being the real open-loop poles and zeros.

(v) The intersections (if any) of root loci with the imaginary axis can be determined by use of the Routh criterion.

(vi) The angle of departure, ϕ_p of a root locus from a complex open-loop pole, is given by

$$\phi_p = 180^\circ + \phi$$

where ϕ is the net angle contribution at this pole of all other open-loop poles and zeros.

(vii) The angle of arrival, ϕ_z of a locus at a complex zero, is given by

$$\phi_z = 180^\circ - \phi$$

where ϕ is the net angle contribution at this zero of all other open-loop poles and zeros.

Contd.

[6] Chapter 7 of reference [155].

Table 4.2 (*Contd.*)

(viii) Points at which multiple roots of the characteristic equation occur (breakaway points of root loci) are the solutions of

$$\frac{dK}{dz} = 0, \text{ where } K = -\frac{\prod_{j=1}^{n}(z - p_j)}{\prod_{i=1}^{m}(z - z_i)}$$

(ix) The gain K at any point z_0 on a root locus, is given by

$$K = \frac{\prod_{j=1}^{n}|z_0 - p_j|}{\prod_{i=1}^{m}|z_0 - z_i|}$$

$$= \frac{[\text{Product of phasor lengths (read to scale) from } z_0 \text{ to poles of } F(z)]}{[\text{Product of phasor lengths (read to scale)from } z_0 \text{ to zeros of } F(z)]}$$

Example 4.5

Consider a process with the transfer function

$$G(s) = \frac{K}{s(s+2)} \tag{4.49a}$$

which, when preceded by a zero-order hold ($T = 0.2$ sec), has the discrete-time transfer function (refer to Table 2.1)

$$G_{h0}G(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{K}{s^2(s+2)}\right] = \frac{K'(z-b)}{(z-a_1)(z-a_2)} \tag{4.49b}$$

where $K' = 0.01758K$, $b = -0.876$, $a_1 = 0.67$, $a_2 = 1$.

The root locus plot of

$$1 + G_{h0}G(z) = 0 \tag{4.50}$$

can be constructed using the rules given in Table 4.2. $G_{h0}G(z)$ has two poles at $z = a_1$ and $z = a_2$, and a zero at $z = b$. From rule (iv), the parts of the real axis between a_1 and a_2, and between $-\infty$ and b constitute sections of the loci. From rule (ii), the loci start from $z = a_1$ and $z = a_2$; one of the loci terminates at $z = b$, and the other locus terminates at $-\infty$. From rule (viii), the breakaway points (there are two) may be obtained by solving for the roots of

$$\frac{dK'}{dz} = 0, \text{ where } K' = -\frac{(z-a_1)(z-a_2)}{(z-b)}$$

However, we can show that for this simple two-pole and one zero configuration, the complex-conjugate section of the root locus plot is a circle. The breakaway points are easily obtained from this result, which is proved as follows:

Let $z = x + jy$

Equation (4.49b) becomes

$$G_{h0}G(z) = \frac{K'(x + jy - b)}{(x + jy - a_1)(x + jy - a_2)} = \frac{K'(x - b + jy)}{(x - a_1)(x - a_2) - y^2 + jy(2x - a_1 - a_2)}$$

On the root loci, z must satisfy Eqn. (4.50).

Therefore,

$$\angle G_{h0}G(z) = \tan^{-1}\frac{y}{x - b} - \tan^{-1}\frac{y(2x - a_1 - a_2)}{(x - a_1)(x - a_2) - y^2} = (2q + 1)\ 180^\circ$$

Taking the tangent of both sides of this equation yields

$$\frac{\dfrac{y}{x - b} - \dfrac{y(2x - a_1 - a_2)}{(x - a_1)(x - a_2) - y^2}}{1 + \dfrac{y}{x - b}\left[\dfrac{y(2x - a_1 - a_2)}{(x - a_1)(x - a_2) - y^2}\right]} = 0$$

or

$$\frac{1}{x - b} - \frac{2x - a_1 - a_2}{(x - a_1)(x - a_2) - y^2} = 0$$

Simplifying, we get

$$(x - b)^2 + y^2 = (b - a_1)(b - a_2) \tag{4.51}$$

which is the equation of a circle with the center at the open-loop zero $z = b$, and the radius equal to $[(b - a_1)(b - a_2)]^{1/2}$.

The root locus plot for the system given by Eqn. (4.49b), is constructed in Fig. 4.22. The limiting value of K for stability may be found by graphical construction or by the Jury stability test. We illustrate the use of graphical construction.

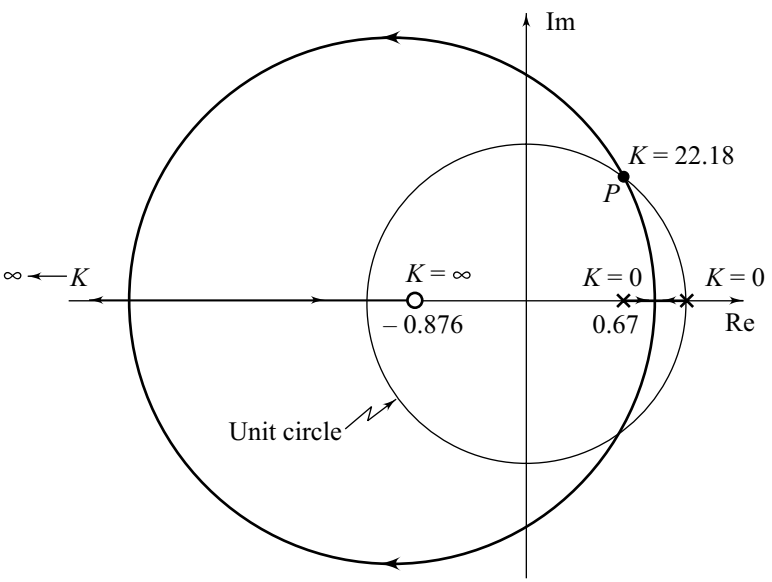


Fig. 4.22 Root locus plot for the system of Example 4.5

By rule (ix) of Table 4.2, the value of K' at point P where the root locus crosses the unit circle is given by

$$K' = \frac{(\text{Phasor length from } P \text{ to pole at } z = 1) \times (\text{Phasor length from } P \text{ to pole at } z = 0.67)}{(\text{Phasor length from } P \text{ to zero at } z = \quad 0.876)}$$

$$= \frac{0.85 \times 0.78}{1.7} = 0.39 = 0.01758K$$

Therefore,
$$K = \frac{0.39}{0.01758} = 22.18$$

The relative stability of the system can be investigated by superimposing the constant-ζ loci on the system root locus plot. This is shown in Fig. 4.23. Inspection of this figure shows that the root locus intersects the $\zeta = 0.3$ locus at point Q. The value of K' at point Q is determined to be 0.1; the gain

$$K = K'/0.01758 = 5.7$$

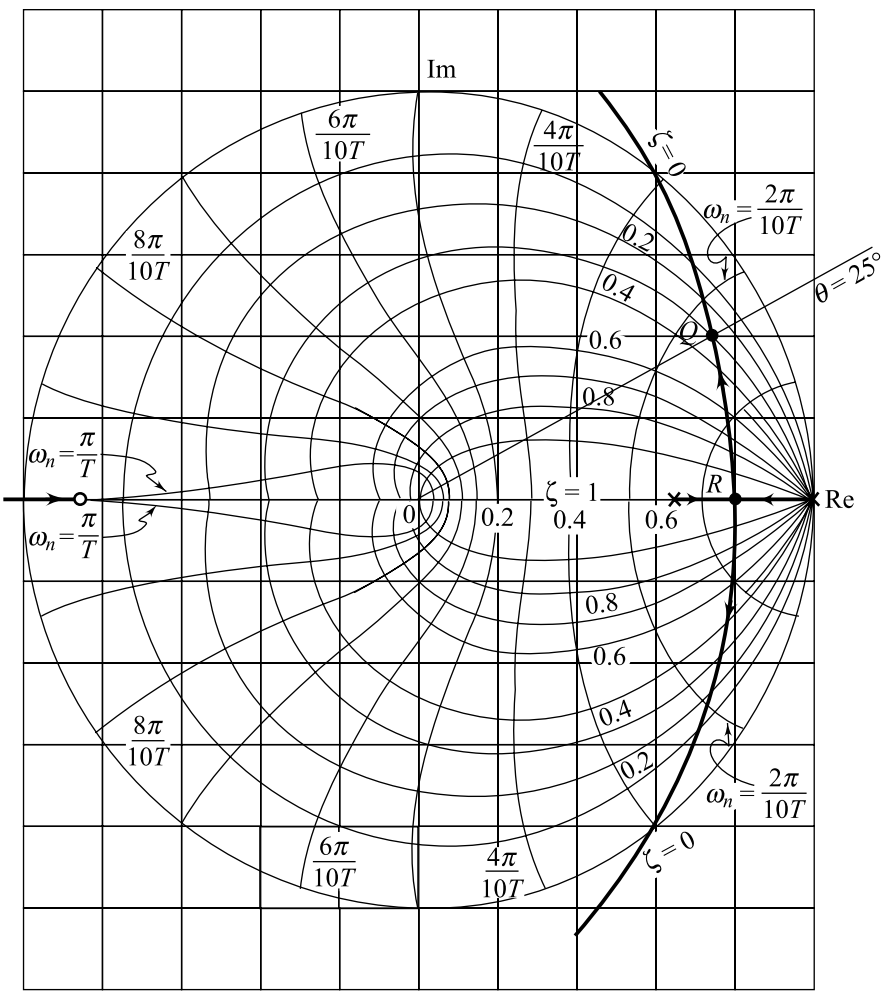


Fig. 4.23 Relative stability analysis

The value of ω_n for $K' = 0.1$, may be obtained by superimposing constant-ω_n loci on the root locus plot and locating the constant-ω_n locus which passes through the point Q. From Fig. 4.23, we observe that none of the constant-ω_n loci on the standard chart passes through the point Q; we have to make a guess for the ω_n value. We can, instead, construct a constant-ω_d locus passing through the point Q and from there obtain ω_n more accurately.

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -\zeta\omega_n \pm j\omega_d$$

are mapped to $$z_{1,2} = e^{-\zeta\omega_n T} e^{\pm j\omega_d T} = re^{\pm j\theta}$$

in the z-plane.

A constant-ω_d locus is thus a radial line passing through the origin at an angle $\theta = \omega_d T$ with the positive real axis of the z-plane, measured positive in the counterclockwise direction.

The radial line passing through the point Q makes an angle $\theta = 25°$ with the real axis (Fig. 4.23). This is a constant-ω_d locus with ω_d given by

$$\omega_d T = \frac{25 \times \pi}{180} \text{ rad}$$

Therefore, $$\omega_n T\sqrt{1-\zeta^2} = \frac{25\pi}{180}$$

This gives $\omega_n = 2.29$ rad/sec.

The value of K' at the breakaway point R, located at $z = 0.824$, is determined to be 0.01594. Therefore, the gain $K = 0.01594/0.01758 = 0.9067$ results in critical damping ($\zeta = 1$) with the two closed-loop poles at $z = 0.824$.

A pole in the s-plane at $s = -a$ has a time constant of $\tau = 1/a$ and an equivalent z-plane location of $e^{-aT} = e^{-T/\tau}$. Thus, for the critically damped case,

$$e^{-0.2/\tau} = 0.824$$

or $$\tau = 1.033 = \text{time constant of the closed-loop poles.}$$

In the frequency-response design procedure described in the previous section, we attempted to reshape the open-loop frequency response to achieve certain stability margins, steady-state response characteristics and so on. A different design technique is presented in this section—the root-locus procedure. In this procedure, we add poles and zeros through a digital controller, so as to shift the roots of the characteristic equation to more appropriate locations in the z-plane. Therefore, it is useful to investigate the effects of various pole-zero configurations of the digital controller on the root locus plots.

Lead Compensation

A simple lead compensator model in the w-plane is described by the transfer function (refer to Eqn. (4.38))

$$D(w) = \frac{1 + w\tau}{1 + \alpha w\tau}; \ \alpha < 1, \ \tau > 0$$

The bilinear transformation

$$w = \frac{2}{T}\frac{z-1}{z+1}$$

transforms $D(w)$ into the following $D(z)$.

$$D(z) = \frac{1+2\tau/T}{1+2\alpha\tau/T}\left[\frac{z+(1-2\tau/T)/(1+2\tau/T)}{z+(1-2\alpha\tau/T)/(1+2\alpha\tau/T)}\right]$$

Since τ and α are both positive numbers and since $\alpha < 1$, the pole and zero of $D(z)$ always lie on the real axis inside the unit circle in the z-plane; the zero is always to the right of the pole. A typical pole-zero configuration of a lead compensator

$$D(z) = K_{c1}\frac{z-\alpha_1}{z-\alpha_2} \tag{4.52}$$

is shown in Fig. 4.24a.

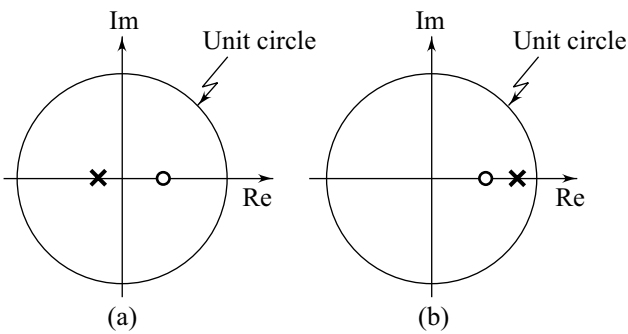


Fig. 4.24 Pole-zero configurations of compensators

For the purpose of simplifying the design procedure, we normally associate the gain K_{c1} with the plant transfer function, and design the lead compensator

$$D(z) = \frac{z-\alpha_1}{z-\alpha_2} \tag{4.53a}$$

based on the new plant transfer function. It may be noted that $D(z)$ given by Eqn. (4.53a) is not a unity gain model; the dc gain of $D(z)$ is given by

$$\lim_{z\to 1}\left(\frac{z-\alpha_1}{z-\alpha_2}\right) = \left(\frac{1-\alpha_1}{1-\alpha_2}\right) \tag{4.53b}$$

To study the effect of a lead compensator on the root loci, we consider a unity-feedback sampled-data system with open-loop transfer function

$$G_{h0}G(z) = \frac{K(z+0.368)}{(z-0.368)(z-0.135)};\ T = 1 \text{ sec} \tag{4.54}$$

The root locus plot of the uncompensated system is shown in Fig. 4.25a. The plot intersects the $\zeta = 0.5$ locus[7] at point P. The value of gain K at this point is determined to be 0.3823.

Constant-ω_d locus passing through point P is a radial line at an angle of 82° with the real axis (Fig. 4.25a). Therefore,

$$\omega_d T = \omega_n T \sqrt{1-\zeta^2} = \frac{82\pi}{180}$$

This gives $\omega_n = 1.65$ rad/sec

Since $G_{h0}G(z)$ given by Eqn. (4.54) is a Type-0 system, we will consider position error constant K_p to study steady-state accuracy. For $K = 0.3823$,

$$K_p = \lim_{z \to 1} G_{h0}G(z) = \frac{0.3823(1+0.368)}{(1-0.368)(1-0.135)} = 0.957$$

We now cancel the pole of $G_{h0}G(z)$ at $z = 0.135$ by the zero of the lead compensator, and add a pole at $z = -0.135$, i.e., we select

$$D(z) = \frac{z-0.135}{z+0.135}$$

Figure 4.25b shows the root locus plot of lead compensated system. The modified locus has moved to the left, towards the more stable part of the plane. The intersection of the locus with the $\zeta = 0.5$ line is at point Q. The value of ω_n at this point is determined to be 2.2 rad/sec. The lead compensator has thus increased ω_n and hence the speed of response of the system. The gain K at point Q is determined to be 0.433. The position error constant of the lead compensated system is given by

$$K_p = \lim_{z \to 1} D(z)G_{h0}G(z) = \lim_{z \to 1} \frac{0.433(z+0.368)}{(z-0.368)(z+0.135)} = 0.82$$

The lead compensator has thus given satisfactory dynamic response, but the position error constant is too low. We will shortly see how K_p can be increased by lag compensation.

---

[7] For a given ζ, the constant-ζ curve may be constructed using Eqn. (4.15b). The following table gives the real and imaginary coordinates of points on some constant-ζ curves.

$\zeta = 0.3$	Re	0.932	0.735	0.360	0	–0.259	–0.380	–0.373
	Im	0.164	0.424	0.623	0.610	0.448	0.220	0
$\zeta = 0.4$	Re	0.913	0.689	0.317	0	–0.201	–0.276	–0.254
	Im	0.161	0.398	0.549	0.504	0.347	0.160	0
$\zeta = 0.5$	Re	0.891	0.640	0.273	0	–0.149	–0.191	–0.163
	Im	0.157	0.370	0.473	0.404	0.259	0.110	0
$\zeta = 0.6$	Re	0.864	0.585	0.228	0	–0.104	–0.122	–0.095
	Im	0.152	0.338	0.395	0.308	0.180	0.070	0
$\zeta = 0.7$	Re	0.830	0.519	0.179	0	–0.064	–0.067	–0.046
	Im	0.146	0.299	0.310	0.215	0.111	0.039	0
$\zeta = 0.8$	Re	0.780	0.431	0.124	0	–0.031	–0.026	–0.015
	Im	0.138	0.249	0.215	0.123	0.053	0.015	0

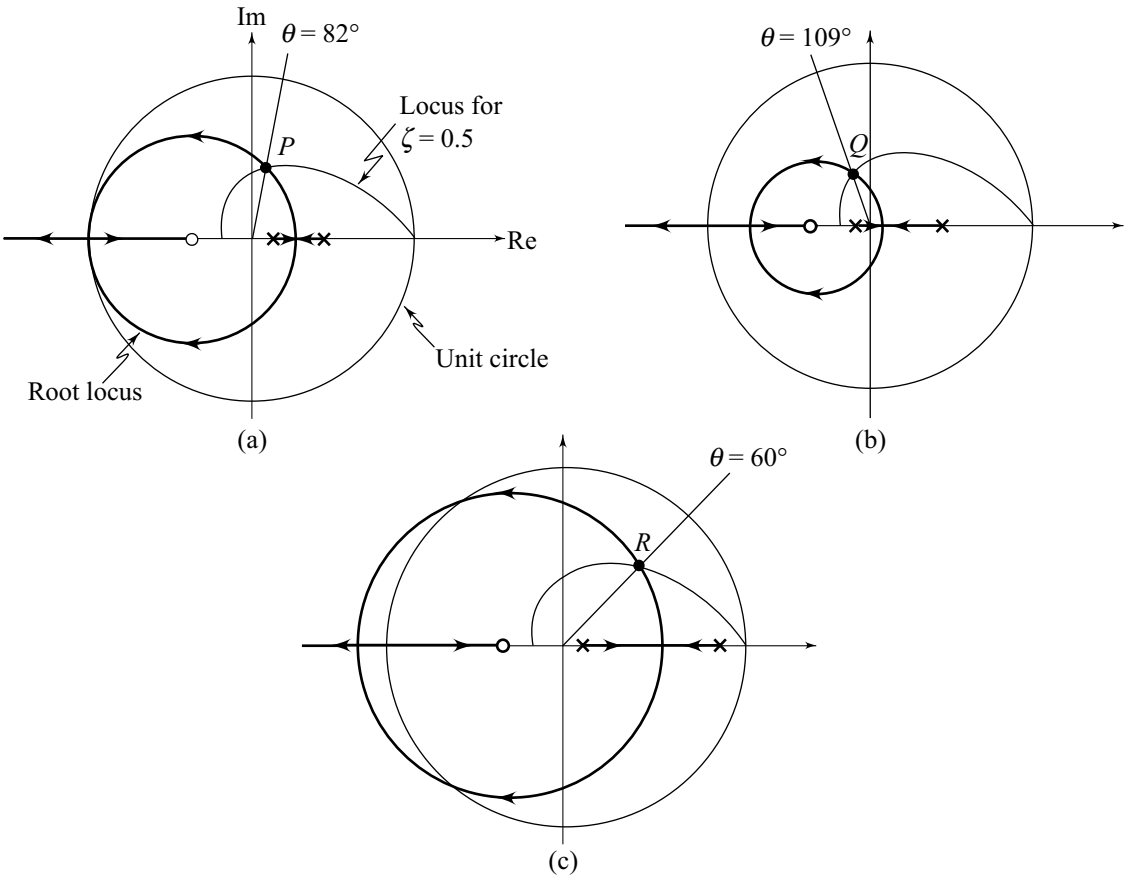


Fig. 4.25 Root locus plot for (a) uncompensated; (b) lead compensated; and (c) lag compensated system.

The selection of the exact values of pole and zero of the lead compensator is done by experience and by trial-and-error. In general, the zero is placed in the neighborhood of the desired dominant closed-loop poles, and the pole is located at a reasonable distance to the left of the zero location.

Lag Compensation

A simple lag compensator model in the w-plane is described by the transfer function (refer to Eqn. (4.41))

$$D(w) = \frac{1 + w\tau}{1 + \beta w\tau}\,;\ \beta > 1,\ \tau > 0$$

The bilinear transformation

$$w = \frac{2}{T}\frac{z-1}{z+1}$$

transforms $D(w)$ into the following $D(z)$.

$$D(z) = \frac{1+2\tau/T}{1+2\beta\tau/T}\left[\frac{z+(1-2\tau/T)/(1+2\tau/T)}{z+(1-2\beta\tau/T)/(1+2\beta\tau/T)}\right]$$

Since τ and β are both positive numbers and since $\beta > 1$, the pole and zero of $D(z)$ always lie on the real axis inside the unit circle; the pole is always to the right of the zero. A typical pole-zero configuration of the lag compensator

$$D(z) = K_{c2}\frac{z-\beta_1}{z-\beta_2} \tag{4.55}$$

is shown in Fig. 4.24b. Note that both the pole and the zero have been shown close to $z = 1$. This, as we shall see, gives better stability properties.

Again, we will associate the gain K_{c2} with the plant transfer function and design the lag compensator,

$$D(z) = \frac{z-\beta_1}{z-\beta_2} \tag{4.56}$$

based on the new plant transfer function. The dc gain of the lag compensator given by (4.56), is equal to

$$\lim_{z\to 1}\frac{z-\beta_1}{z-\beta_2} = \frac{1-\beta_1}{1-\beta_2} \tag{4.57}$$

To study the effect of lag compensator on the root loci, we reconsider the system described by Eqn. (4.54):

$$G_{h0}G(z) = \frac{K(z+0.368)}{(z-0.368)(z-0.135)};\ T = 1 \text{ sec}$$

The root locus plot of the uncompensated system is shown in Fig. 4.25a. At point P, $\zeta = 0.5$, $\omega_n = 1.65$ and $K = 0.3823$ ($K_p = 0.957$).

We now cancel the pole of $G_{h0}G(z)$ at $z = 0.368$ by the zero of the lag compensator, and add a pole at $z = 0.9$, i.e., we select

$$D(z) = \frac{z-0.368}{z-0.9}$$

Figure 4.25c shows the root locus plot of the lag compensated system. The intersection of the locus with $\zeta = 0.5$ line is at point R. The value of ω_n at this point is determined to be 1.2 rad/sec. The lag compensator has thus reduced ω_n and hence the speed of response. The value of the gain K at point R is determined to be 0.478. The position error constant of the lag compensated system is

$$K_p = \lim_{z\to 1} D(z)G_{h0}G(z) = \lim_{z\to 1}\frac{0.478(z+0.368)}{(z-0.135)(z-0.9)} = 7.56$$

Thus, we have been able to increase position error constant appreciably by lag compensation.

If both the pole and the zero of the lag compensator are moved close to $z = 1$, then the root locus plot of the lag compensated system moves back towards its uncompensated shape. Consider the root locus plot of the uncompensated system shown in Fig. 4.25a. The angle contributed at point P by additional pole-zero pair close to $z = 1$ (called a *dipole*), will be negligibly small; therefore, the point P will continue to lie on the lag compensated root locus plot. However, the lag compensator

$$D(z) = \frac{z - \beta_1}{z - \beta_2}$$

will raise the system K_p (refer to Eqn. (4.57)), by a factor of $(1 - \beta_1)/(1 - \beta_2)$.

The following examples illustrate typical digital control system design problems carried out in the z-plane, using the root locus technique. As we shall see, the design of digital compensation using root locus plots is essentially a trial-and-error method. The designer may rely on a digital computer to plot out a large number of root loci by scanning through a wide range of possible values of the compensator parameters, and select the best solution. However, one can still make proper and intelligent initial 'guesses' so that the amount of trial-and-error effort is kept to a minimum.

Example 4.6

Consider the feedback control system shown in Fig. 4.26. The plant is described by the transfer function

$$G(s) = \frac{K}{s(s+2)}$$

Design a digital control scheme for the system to meet the following specifications;

(i) the velocity error constant $K_v = 6$;
(ii) peak overshoot M_p to step input ≤ 15%; and
(iii) settling time t_s (2% tolerance band) ≤ 5 sec.

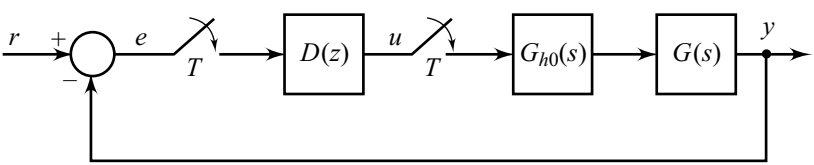


Fig. 4.26 A feedback system with digital compensation

Solution The transient accuracy requirements correspond to $\zeta = 0.5$ and $\omega_n = 1.6$. We select $T = 0.2$ sec. Note that sampling frequency $\omega_s = 2\pi/T$ is about 20 times the natural frequency; therefore, our choice of sampling period is satisfactory.

The transfer function $G_{h0}G(z)$ of the plant, preceded by a ZOH, can be obtained as follows:

$$G_{h0}G(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{K}{s^2(s+2)}\right]$$

$$= \frac{0.01758K(z+0.876)}{(z-1)(z-0.67)} = \frac{K'(z+0.876)}{(z-1)(z-0.67)} \tag{4.58}$$

The root locus plot of this system for $0 \le K' < \infty$ was earlier constructed in Fig. 4.22. Complex-conjugate sections of this plot are shown in Fig. 4.27. The plot intersects the $\zeta = 0.5$ locus at point P. At this point $\omega_n = 1.7$ rad/sec, $K' = 0.0546$.

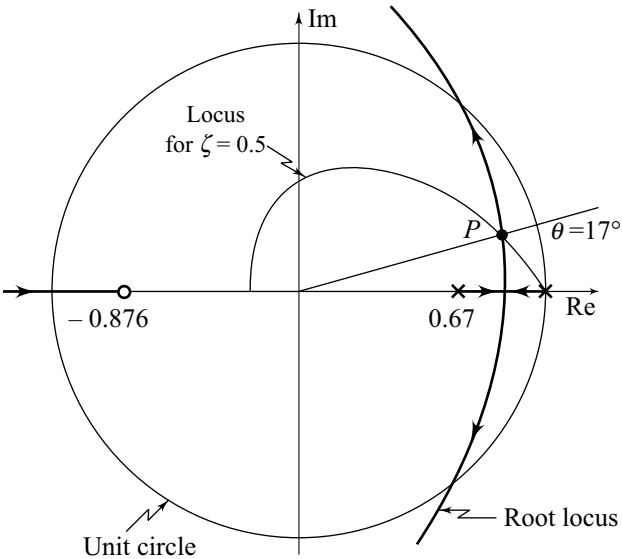


Fig. 4.27 Root locus plot for system (4.58)

Therefore, the transient accuracy requirements ($\zeta = 0.5$, $\omega_n = 1.6$) are almost satisfied by gain adjustment only. Let us now examine the steady-state accuracy of the uncompensated system ($D(z) = 1$) with $K' = 0.0546$.

The velocity error constant K_v of the system is given by

$$K_v = \frac{1}{T}\lim_{z \to 1}(z-1)G_{h0}G(z)$$

$$= \frac{5(0.0546)(1+0.876)}{(1-0.67)} = 1.55$$

The specified value of K_v is 6. Therefore, an increase in K_v by a factor of 3.87 (= 6/1.55) is required. The objective before us now is to introduce a $D(z)$ that raises the system K_v by a factor of 3.87, without appreciably affecting the transient performance of the uncompensated system, i.e., without appreciably affecting the root locus plot in the vicinity of point P. This objective can be realized by a properly designed lag compensator, as is seen below.

We add the compensator pole and zero as shown in Fig. 4.28. Since both the pole and the zero are very close to $z = 1$, the scale in the vicinity of these points has been greatly expanded. The angle contributed by the compensator pole at point P, is almost equal to the angle contributed by the compensator zero. Therefore, the addition of dipole near $z = 1$ does not appreciably disturb the root locus plot in the vicinity of point P. It only slightly reduces ω_n. The lag compensator

$$D(z) = \frac{z-0.96}{z-0.99}$$

raises the system K_v by a factor of $(1 - 0.96)/(1 - 0.99) = 4$.

Note that because of lag compensator, a third closed-loop pole has been added. This pole, as seen from Fig. 4.28, is a real pole lying close to $z = 1$. This pole, fortunately, does not disturb the dominance of the complex conjugate closed-loop poles. The reason is simple.

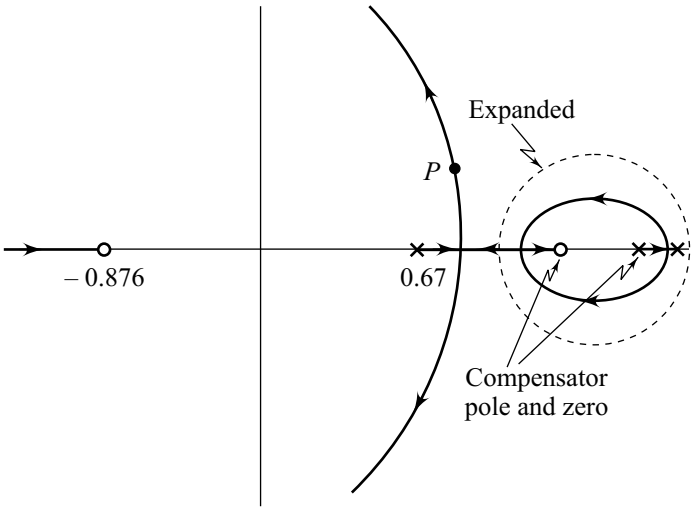


Fig. 4.28 Compensator design (Example 4.6)

The closed-loop pole, close to $z = 1$, has a long time constant. However, there is a zero close to this additional pole. The net effect is that the settling time will increase because of the third pole, but the amplitude of the response term contributed by this pole will be very small. In system response, a long tail of small amplitude will appear which may not appreciably degrade the performance of the system.

Example 4.7

Reconsider the feedback control system of Example 4.6 (Fig. 4.26). We now set the following goal for our design:

(i) $K_v \geq 2.5$;
(ii) $\zeta \cong 0.5$; and
(iii) t_s (2% tolerance band) ≤ 2 sec.

The transient accuracy requirements correspond to $\zeta = 0.5$ and $\omega_n = 4$. For sampling interval $T = 0.2$ sec, the sampling frequency is about eight times the natural frequency. A smaller value of T is more appropriate for the present design problem—which requires higher speed of response. We will, however, take $T = 0.2$ sec to compare our results with those of Example 4.6.

Following the initial design steps of Example 4.6, we find that

$$G_{h0}G(z) = \frac{0.01758K(z+0.876)}{(z-1)(z-0.67)} = \frac{K'(z+0.876)}{(z-1)(z-0.67)}$$

Complex conjugate sections of the root locus plot superimposed on $\zeta = 0.5$ line are shown in Fig. 4.27. The root locus plot intersects the constant-ζ locus at point P. At this point, $\omega_n = 1.7$ rad/sec. The specified value of ω_n is 4. Therefore, the transient accuracy requirements cannot be satisfied by only gain adjustment.

The natural frequency ω_n can be increased by lead compensation. To design a lead compensator, we translate the transient performance specifications into a pair of dominant closed-loop poles, add open-loop poles and zeros through $D(z)$ to reshape the root locus plot, and force it to pass through the desired closed-loop poles.

Point Q in Fig. 4.29 corresponds to the desired closed-loop pole in the upper half of z-plane. It is the point of intersection of the $\zeta = 0.5$ locus and the constant-ω_d locus, with ω_d given by

$$\omega_d = \omega_n \sqrt{1-\zeta^2} = 3.464 \text{ rad/sec}$$

For this value of ω_d, constant-ω_d locus is a radial line at an angle of $\omega_d T \left(\frac{180}{\pi}\right) = 39.7^\circ$ with the real axis.

If the point Q is to lie on the root locus plot of the compensated system, then the sum of the angles contributed by open-loop poles and zeros of the plant, and the pole and zero of the compensator at the point Q, must be equal to $\pm (2q + 1)180^\circ$; $q = 0, 1, 2, \ldots$

The sum of the angle contributions due to open-loop poles and zero of the plant at plant Q, is

$$17.10^\circ - 138.52^\circ - 109.84^\circ = -231.26^\circ$$

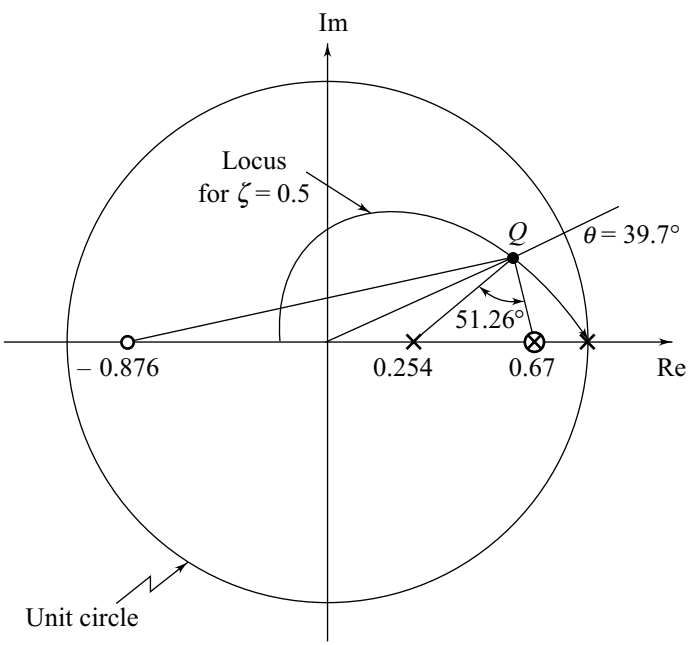


Fig. 4.29 Compensator design (Example 4.7)

Hence, the compensator $D(z)$ must provide +51.26°. The transfer function of the compensator may be assumed to be

$$D(z) = \frac{z-\alpha_1}{z-\alpha_2}$$

If we decide to cancel the pole at $z = 0.67$ by the zero of the compensator at $z = \alpha_1$, then the pole of the compensator can be determined (from the condition that the compensator must provide + 51.26°) as a point at $z = 0.254$ ($\alpha_2 = 0.254$). Thus, the transfer function of the compensator is obtained as

$$D(z) = \frac{z-0.67}{z-0.254}$$

The open-loop transfer function now becomes

$$D(z)G_{h0}G(z) = \frac{0.01758K(z+0.876)(z-0.67)}{(z-0.254)(z-1)(z-0.67)} = \frac{0.01758K(z+0.876)}{(z-0.254)(z-1)} = \frac{K'(z+0.876)}{(z-0.254)(z-1)}$$

The value of K' at point Q, obtained from Fig. 4.29 by graphical construction, is 0.2227. Therefore, $K = 12.67$.

The velocity error constant of the compensated system is given by

$$K_v = \frac{1}{T}\lim_{z\to 1}[(z-1)D(z)G_{h0}G(z)] = 2.8$$

It meets the specification on steady-state accuracy.

If it is required to have a large K_v, then we may include a lag compensator. The lag-lead compensator can satisfy the requirements of high steady-state accuracy and high speed of response.

From the viewpoint of microprocessor implementation of the lag, lead, and lag-lead compensators, the lead compensators present the least coefficient quantization problems, because the locations of poles and zeros are widely separated, and the numerical inaccuracies in realization of these compensators will result in only small deviations in expected system behavior. However, in the case of lag compensators and lag-lead compensators, the lag section may result in considerable coefficient quantization problems, because the locations of poles and zeros are usually close to each other (they are near the point $z = 1$). Numerical problems associated with realization of compensator coefficients, may lead to significant deviations in expected system behavior.

4.5 *z*-PLANE SYNTHESIS

Much of the style of the transform-domain techniques we have been discussing in this chapter, grew out of the limitations of technology which was available for realization of the compensators with pneumatic components, or electric networks and amplifiers. In digital computer, such limitations on realization are, of course, not relevant, and one can ignore these particular constraints. One design method which eliminates these constraints begins from the very direct point of view that we are given a process (plus hold) transfer function, $G_{h0}G(z)$, that we want to construct a desired transfer function, $M(z)$, between

input r and output y and that we have the computer transfer function, $D(z)$, to do the job as per the feedback control structure of Fig. 4.30.

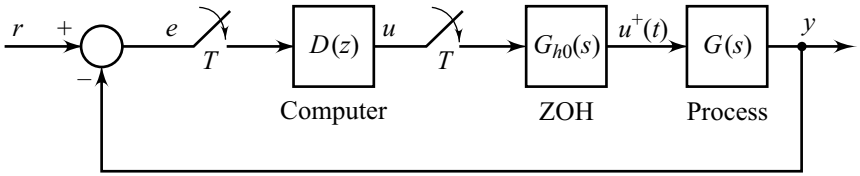


Fig. 4.30 A feedback system with digital compensation

The closed-loop transfer function is given by the formula

$$M(z) = \frac{D(z)G_{h0}G(z)}{1 + D(z)G_{h0}G(z)} \tag{4.59}$$

from which we get the design formula

$$D(z) = \frac{1}{G_{h0}G(z)}\left[\frac{M(z)}{1 - M(z)}\right] \tag{4.60}$$

As is seen from Eqn. (4.60), the controller transfer function consists of the inverse of the plant transfer function and the additional term which depends on the system closed-loop transfer function. Thus, the design procedure, outlined above, looks for a $D(z)$ which will cancel the process effects and add whatever is necessary to give the desired performance.

For prescribing the required closed-loop transfer function $M(z)$, the following restrictions have to be noted.

Realizability of Digital Controller

Assume that a digital controller

$$D(z) = \frac{Q_v(z)}{P_\mu(z)} = \frac{q_0 z^v + q_1 z^{v-1} + \cdots + q_v}{z^\mu + p_1 z^{\mu-1} + \cdots + p_\mu} \tag{4.61}$$

is cascaded with the process

$$G_{ho}G(z) = \frac{B_m(z)}{A_n(z)} = \frac{b_0 z^m + b_1 z^{m-1} + \cdots + b_m}{z^n + a_1 z^{n-1} + \cdots + a_n};\ m \le n \tag{4.62}$$

in the control loop given by Fig. 4.30.

For $D(z)$ to be physically realizable, $v \le \mu$.

The closed-loop transfer function

$$M(z) = \frac{D(z)G_{h0}G(z)}{1 + D(z)G_{h0}G(z)} = \frac{Q_v(z)B_m(z)}{P_\mu(z)A_n(z) + Q_v(z)B_m(z)} = \frac{N_{v+m}(z)}{D_{\mu+n}(z)}$$

The order of the numerator polynomial of $M(z)$ is $v + m$, and the order of the denominator polynomial of $M(z)$ is $\mu + n$. The *pole excess*[8] of $M(z)$ is, therefore, $\{(\mu - v) + (n - m)\}$.

[8] Pole excess of $M(z)$ = {Number of finite poles of $M(z)$ – Number of finite zeros of $M(z)$}.

This means that because of the condition of realizability of digital controller, the pole excess of the closed-loop transfer function $M(z)$ has to be greater than or equal to the pole excess of the process transfer function $G_{h0}G(z)$.

Cancellation of Poles and Zeros

If the digital controller $D(z)$ given by Eqn. (4.60) and the process $G_{h0}G(z)$ are in a closed loop, the poles and zeros of the process are canceled by the zeros and poles of the controller. The cancellation is perfect if the process model $G_{h0}G(z)$ matches the process exactly. Since the process models used for design practically never describe the process behavior exactly, the corresponding poles and zeros will not be canceled exactly; the cancellation will be approximate. For poles and zeros of $G_{h0}G(z)$ which are sufficiently spread in the inner of the unit disc in the z-plane, the approximation in cancellation leads to only small deviations of the assumed behavior $M(z)$ in general. However, one has to be careful if $G_{h0}G(z)$ has poles or zeros on or outside the unit circle. Imperfect cancellation may lead to weakly damped or unstable behavior. Therefore, the design of digital controllers, according to Eqn. (4.60), has to be restricted to cancellation of poles and zeros of $G_{h0}G(z)$ located inside the unit circle. This imposes certain restrictions on the desired transfer function $M(z)$ as is seen below.

Assume that $G_{h0}G(z)$ involves an unstable (or critically stable) pole at $z = \alpha$. Let us define

$$G_{h0}G(z) = \frac{G_1(z)}{z-\alpha}$$

where $G_1(z)$ does not include a term that cancels with $(z - \alpha)$. Then the closed-loop transfer function becomes

$$M(z) = \frac{D(z)\dfrac{G_1(z)}{z-\alpha}}{1+D(z)\dfrac{G_1(z)}{z-\alpha}} \tag{4.63}$$

Since we require that no zero of $D(z)$ cancel the pole of $G_{h0}G(z)$ at $z = \alpha$, we must have

$$1 - M(z) = \frac{1}{1+D(z)\dfrac{G_1(z)}{z-\alpha}} = \frac{z-\alpha}{z-\alpha+D(z)G_1(z)}$$

that is, $1 - M(z)$ must have $z = \alpha$ as a zero. This argument applies equally if $G_{h0}G(z)$ involves two or more unstable (or critically stable) poles.

Also note from Eqn. (4.63) that if poles of $D(z)$ do not cancel zeros of $G_{h0}G(z)$, then the zeros of $G_{h0}G(z)$ become zeros of $M(z)$.

Let us summarize what we have stated concerning cancelation of poles and zeros of $G_{h0}G(z)$.

(i) Since the digital controller $D(z)$ should not cancel unstable (or critically stable) poles of $G_{h0}G(z)$, all such poles of $G_{h0}G(z)$ must be included in $1 - M(z)$ as zeros.

(ii) Zeros of $G_{h0}G(z)$ that lie on or outside the unit circle should not be canceled with poles of $D(z)$; all such zeros of $G_{h0}G(z)$ must be included in $M(z)$ as zeros.

The design procedure, thus, essentially involves the following three steps:

(I) The closed-loop transfer function $M(z)$ of the final system is determined from the performance specifications, and the fixed parts of the system, i.e., $G_{h0}G(z)$.
(II) The transfer function $D(z)$ of the digital controller is found using the design formula (4.60).
(III) The digital controller $D(z)$ is synthesized.

Step (I) is certainly the most difficult one to satisfy. In order to pass step (I), a designer must fulfil the following requirements:

(i) the digital controller $D(z)$ must be physically realizable;
(ii) the poles and zeros of $G_{h0}G(z)$ on or outside the unit circle should not be canceled by $D(z)$; and
(iii) the system specifications on transient and steady-state accuracy should be satisfied.

Example 4.8

The plant of sampled-data system of Fig. 4.30 is described by the transfer function

$$G(s) = \frac{1}{s(10s+1)} \tag{4.64a}$$

The sampling period is 1 sec.

The problem is to design a digital controller $D(z)$ to realize the following specifications:

(i) $K_v \geq 1$;
(ii) $\zeta = 0.5$; and
(iii) t_s (2% tolerance band) ≤ 8 sec.

The selection of a suitable $M(z)$ is described by the following steps.

(i) The z-transfer function of the plant is given by (refer to Table 2.1)

$$G_{h0}G(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{1}{s^2(10s+1)}\right] = 0.04837\,\frac{(z+0.9672)}{(z-1)(z-0.9048)} \tag{4.64b}$$

Since $G_{h0}G(z)$ has one more pole than zero, $M(z)$ must have a pole excess of at least one.

(ii) $G_{h0}G(z)$ has a pole at $z = 1$. This must be included in $1 - M(z)$ as zero, i.e.,

$$1 - M(z) = (z-1)F(z) \tag{4.65}$$

where $F(z)$ is a ratio of polynomials of appropriate dimensions.

(iii) The transient accuracy requirements are specified as $\zeta = 0.5$, $\omega_n = 1 (t_s = 4/\zeta\omega_n)$. With a sampling period $T = 1$ sec, this maps to a pair of dominant closed-loop poles in the z-plane with

$$|z_{1,2}| = e^{-\zeta\omega_n T} = 0.6065$$

$$\angle z_{1,2} = \pm\,\omega_n T\sqrt{1-\zeta^2} = \pm\,\frac{0.866 \times 180}{3.14} = \pm 49.64^{\circ}$$

This corresponds to

$$z_{1,2} = 0.3928 \pm j\,0.4621$$

The closed-loop transfer function, $M(z)$, should have dominant poles at the roots of the equation

$$\Delta(z) = z^2 - 0.7856\,z + 0.3678 = 0 \tag{4.66}$$

The steady-state accuracy requirements demand that steady-state error to unit-step input is zero, and steady-state error to unit-ramp input is less than $1/K_v$.

$$E(z) = R(z) - Y(z) = R(z)[1 - M(z)] = R(z)\,(z-1)F(z)$$

$$e_{ss}^{*}\Big|_{\text{unit step}} = \lim_{z \to 1} z\,(z-1)\,F(z) = 0$$

Thus, with the choice of $M(z)$ given by Eqn. (4.65), the steady-state error to unit-step input is always zero.

$$e_{ss}^{*}\Big|_{\text{unit ramp}} = \lim_{z \to 1} (z-1)\,\frac{Tz}{(z-1)^2}\,(z-1)F(z) = T\,F(1) = 1/K_v$$

For $T = 1$ and $K_v = 1$,

$$F(1) = 1 \tag{4.67}$$

From Eqns (4.65) and (4.66), we observe that

$$F(z) = \frac{z - \alpha}{z^2 - 0.7856z + 0.3678}$$

meets the requirements on realizability of $D(z)$, cancellation of poles and zeros of $G_{h0}G(z)$, and transient accuracy. The requirement on steady-state accuracy is also met if we choose α such that (refer to Eqn. (4.67))

$$\frac{1-\alpha}{1-0.7856+0.3678} = 1$$

This gives $\alpha = 0.4178$

Therefore,

$$F(z) = \frac{z - 0.4178}{z^2 - 0.7856\,z + 0.3678}\,;\; 1 - M(z) = \frac{(z-1)(z-0.4178)}{z^2 - 0.7856\,z + 0.3678}$$

$$M(z) = \frac{0.6322\,z - 0.05}{z^2 - 0.7856\,z + 0.3678} \tag{4.68}$$

Now, turning to the basic design formula (4.60), we compute

$$D(z) = \frac{1}{G_{h0}G(z)}\left[\frac{M(z)}{1-M(z)}\right] = \frac{(z-1)(z-0.9048)}{(0.04837)(z+0.9672)}\left[\frac{0.6322\,z - 0.05}{(z-1)(z-0.4178)}\right]$$

$$= 13.07\,\frac{(z-0.9048)(z-0.079)}{(z+0.9672)(z-0.4178)} \tag{4.69}$$

A plot of the step response of the resulting design is provided in Fig. 4.31, which also shows the control effort. The underdamped response settles within a two percent band of the desired value of unity in less than 8 sec. We can see the oscillation of $u(k)$—associated with the pole of $D(z)$ at $z = -0.9672$, which is quite near the unit circle. Strong oscillations of $u(k)$ are often considered unsatisfactory, even though the process is being controlled as was intended. In the literature, poles near $z = -1$ are often referred to as *ringing poles*.

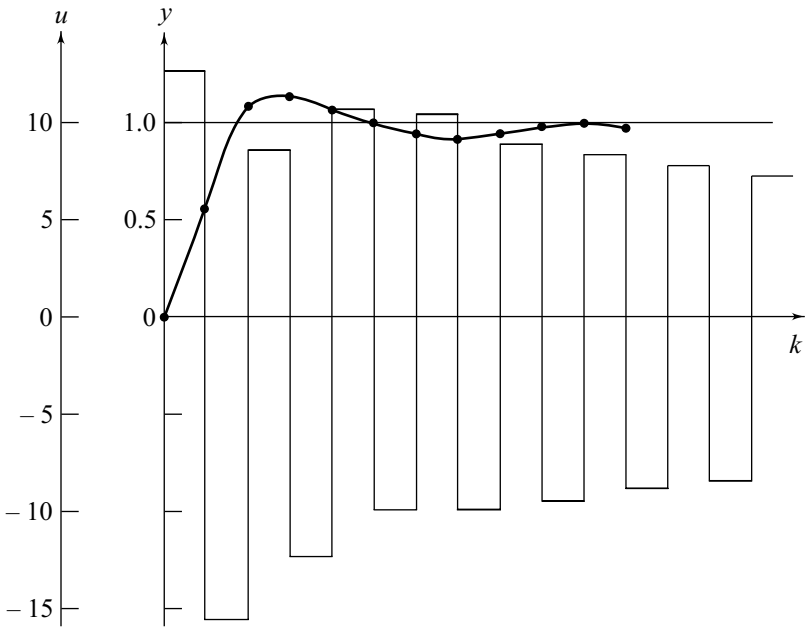


Fig. 4.31 Step response (Example 4.8)

To avoid the ringing effect, we could include the zero of $G_{h0}G(z)$ at $z = -0.9672$ in $M(z)$ as zero, so that this zero of $G_{h0}G(z)$ is not canceled with pole of $D(z)$. $M(z)$ may have additional poles at $z = 0$, where the transient is as short as possible. The result will be a simpler $D(z)$ with a slightly more complicated $M(z)$.

REVIEW EXAMPLES

Review Example 4.1

Consider the digital control system shown in Fig. 4.32. The transfer function of the plant is $G(s) = 1/[s(s + 1)]$. Design a lead compensator $D(z)$ in the w-plane such that the phase margin in 50°, the gain margin is at least 10 dB, and the velocity error constant K_v is 2. Assume that the sampling period is 0.2 sec.

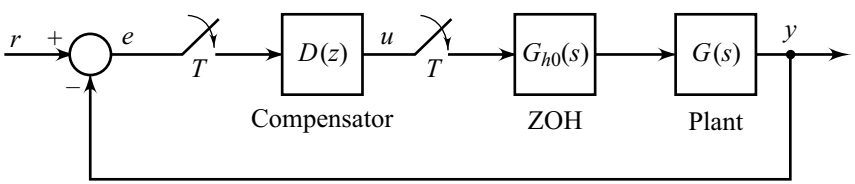


Fig. 4.32 A digital control configuration

Solution The digital controller is assumed to be of the form

$$\frac{U(z)}{E(z)} = KD_1(z) = D(z)$$

To simplify the design procedure, we will associate the gain K of the controller with the plant model. The design problem is, therefore, to obtain compensator $D_1(z)$ for the plant

$$G(s) = \frac{K}{s(s+1)}$$

to meet the specifications on steady-state and transient performance.

We will fix the gain K to a value that realizes the given K_v. A unity dc gain compensator $D_1(z)$ will then be introduced to meet the transient accuracy requirements without affecting the steady-state accuracy.

$$K_v = \lim_{s \to 0} sG(s) = K$$

Therefore, $K = 2$ meets the requirement on steady-state accuracy.

For $T = 0.2$ and $K = 2$, we have (refer to Table 2.1)

$$G_{h0}G(z) = (1 - z^{-1})\, \mathscr{Z}\left[\frac{2}{s^2(s+1)}\right] = 0.03746\left[\frac{z + 0.9356}{(z-1)(z-0.8187)}\right]$$

By use of the bilinear transformation

$$z = \frac{1 + wT/2}{1 - wT/2} = \frac{1 + 0.1w}{1 - 0.1w}$$

$G_{h0}G(z)$ can be transformed to $G_{h0}G(w)$ given below (refer to Eqns (4.37)).

$$G_{h0}G(w) = \frac{2\left(1 - \frac{w}{10}\right)\left(1 + \frac{w}{300.6}\right)}{w\left(1 + \frac{w}{0.997}\right)}$$

The Bode plot of $G_{h0}G(j\nu)$ is shown in Fig. 4.33. The phase margin can be read from the Bode plot as 32º and the gain margin as 14.2 dB.

It is desired to raise the phase margin to 50º without altering K_v. Also the gain margin should be at least 10 dB. We now design a lead compensator

$$D_1(w) = \frac{1 + w\tau}{1 + \alpha w\tau}\,;\ \alpha < 1,\ \tau > 0$$

to meet these objectives. We choose the zero of the compensator at 0.997 (This choice cancels a pole of $G_{h0}G(w)$). Addition of this zero shifts the gain crossover frequency of the uncompensated system to $\nu_c = 1.8$. The phase margin of the uncompensated system at ν_c is $\Phi M_1 = 22^\circ$. The phase lead required at ν_c to bring the phase margin to the specified value $\Phi M_s = 50^\circ$, is given by

$$\phi_l = \Phi M_s - \Phi M_1 + \varepsilon = 50^\circ - 22^\circ + 3^\circ = 31^\circ$$

By using Eqn. (4.39), we obtain

$$\alpha = \frac{1 - \sin 31^\circ}{1 + \sin 31^\circ} = 0.3$$

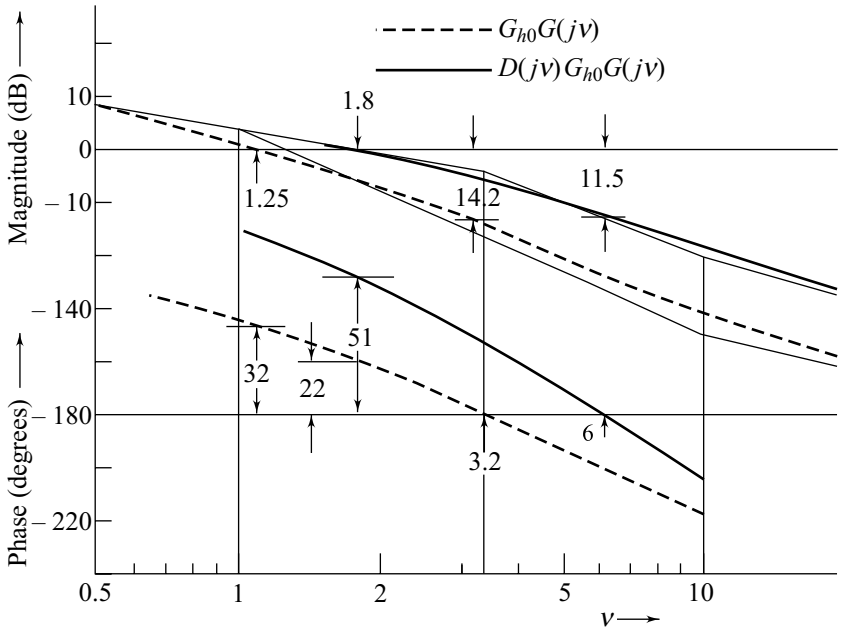


Fig. 4.33 Compensator design (Review Example 4.1)

The phase lead of 31° is provided at the frequency (refer to Eqn. (4.40))

$$\nu_m = \sqrt{\frac{1}{\tau}\left(\frac{1}{\alpha\tau}\right)} = \sqrt{0.997 \times 3.27} = 1.8$$

which is same as the gain crossover frequency.

Thus, the compensator transfer function is

$$D_1(w) = \frac{1+\dfrac{w}{0.997}}{1+\dfrac{w}{3.27}} \tag{4.70}$$

The magnitude and phase angle curves for the compensated open-loop transfer function are shown by solid curves in Fig. 4.33. From these curves, we see that the phase margin is 51º and the gain margin is 11.5 dB.

The compensator transfer function given by Eqn. (4.70) will now be transformed back to the z-plane by the bilinear transformation

$$w = \frac{2}{T}\frac{z-1}{z+1} = 10\,\frac{z-1}{z+1}$$

This gives

$$D_1(z) = 2.718\left(\frac{z-0.8187}{z-0.5071}\right)$$

The system gain K was determined to be 2. Therefore, for the plant of the system of Fig. 4.32, the digital controller is given by

$$\frac{U(z)}{E(z)} = D(z) = 2.718\,K\left(\frac{z-0.8187}{z-0.5071}\right) = 5.436\left(\frac{z-0.8187}{z-0.5071}\right)$$

Review Example 4.2

Consider the digital control configuration shown in Fig. 4.32. The transfer function,

$$G(s) = \frac{e^{-1.5s}}{s+1}$$

describes a process of temperature control *via* mixing (refer to Example 3.3). In the following, we design a digital compensator for the temperature control process; the sampling interval T is assumed to be 1 sec. The transfer function $G_{h0}G(z)$ derived in Example 3.3 is repeated below.

$$G_{h0}G(z) = 0.3935\ \frac{z+0.6066}{z^2(z-0.3679)}$$

Since $G_{h0}G(z)$ is a Type-0 transfer function, the system will have a steady-state error to a constant command or disturbance. If we assume that such a behavior in steady-state is unacceptable, we can correct the problem by including integral control through the transfer function

$$D_1(z) = \frac{Kz}{z-1}$$

The effective plant transfer function is now

$$D_1(z)G_{h0}G(z) = 0.3935K\ \frac{(z+0.6066)}{z(z-1)(z-0.3679)} = \frac{K'(z+0.6066)}{z(z-1)(z-0.3679)}$$

The unity-feedback root locus plot for this transfer function is sketched in Fig. 4.34. The point P on the root locus corresponds to $\zeta = 0.5$, and $\omega_n = 0.423$ $\left(\theta = \omega_n T\sqrt{1-\zeta^2} = \frac{21\pi}{180}\right)$.

The natural frequency ω_n has to be raised to improve the speed of response. We employ a lead compensation which cancels the plant pole at $z = 0.3679$ and the plant zero at $z = -0.6066$. The open-loop transfer function of the lead-compensated system becomes

$$D_2(z)D_1(z)G_{h0}G(z) = \frac{z-0.3679}{z+0.6066}\left[\frac{K'(z+0.6066)}{z(z-1)(z-0.3679)}\right] = \frac{K'}{z(z-1)}$$

The root locus plot is sketched in Fig. 4.35. The point Q on the root locus corresponds to $\zeta = 0.5$, $\omega_n = 0.826$, $K_v = K' = 0.45$.

Suppose we wish to raise K_v to 1. A lag compensator

$$D_3(z) = \frac{z-0.9}{z-0.9545}$$

will raise K_v by a factor of $(1 - 0.9)/(1 - 0.9545)$. The lag pole-zero pair are very close to each other, and do not change the root locus near the dominant roots significantly. However, the lag compensation does introduce a small, but very slow transient, whose effect on dynamic response needs to be evaluated, especially in terms of the response to disturbances.

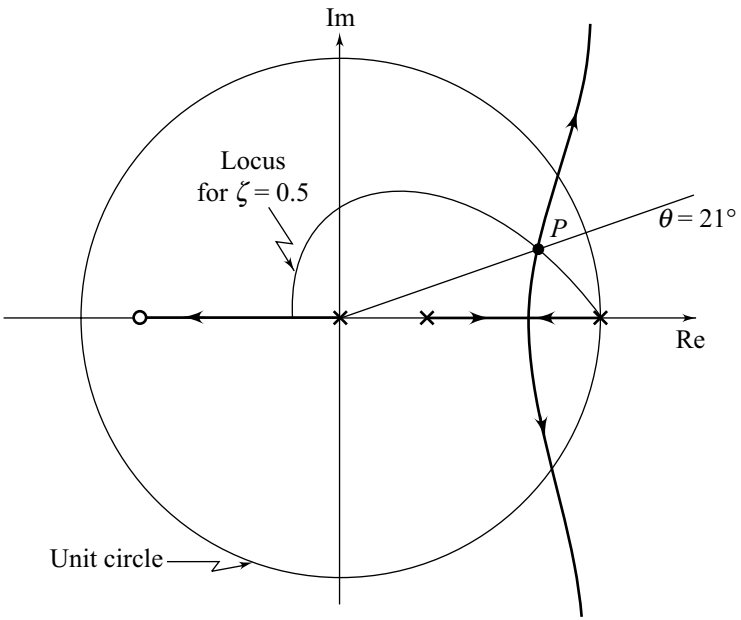


Fig. 4.34 Root locus plot of mixing flow plant with integral control

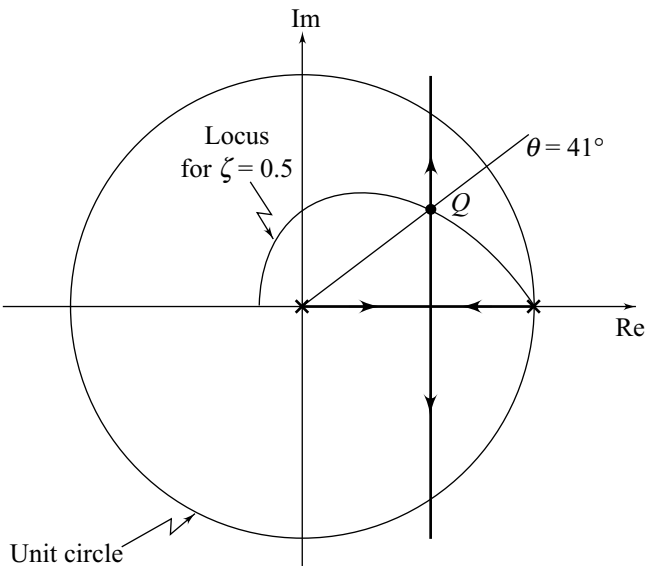


Fig. 4.35 Root locus plot of mixing flow plant with integral control and lead compensation

PROBLEMS

4.1 For the system shown in Fig. P4.1, find

(i) position error constant, K_p;
(ii) velocity error constant, K_v; and
(iii) acceleration error constant, K_a.

Express the results in terms of K_1, K_2, J, and T.

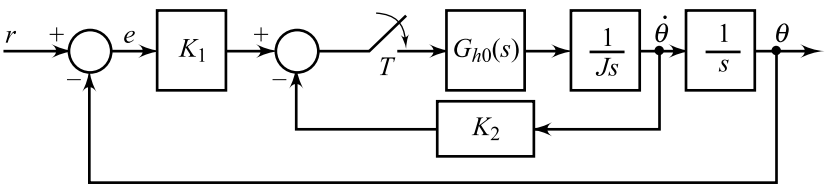
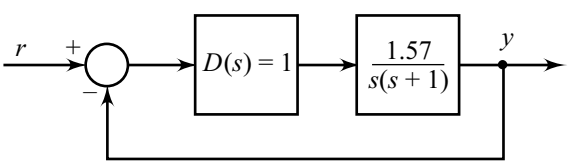


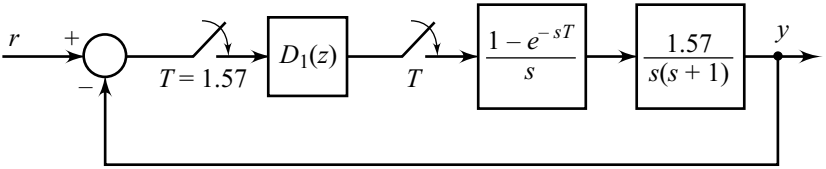
Fig. P4.1

4.2 Consider the analog control system shown in Fig. P4.2a. Show that the phase margin of the system is about 45º.

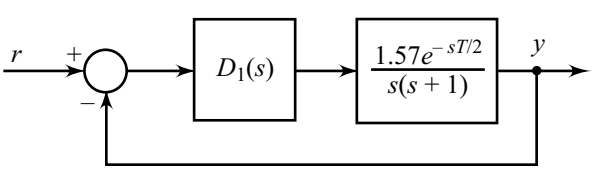
We wish to replace the analog controller by a digital controller as shown in Fig. P4.2b. First, modify the analog controller to take into account the effect of the hold that must be included



(a)



(b)



(c)

Fig. P4.2

in the equivalent digital control system (the zero-order hold may be approximated by a pure time delay of one half of the sampling period T (Fig. P4.2c), and then a lag compensator $D_1(s)$ may be designed to realize the phase margin of 45º). Then, by using the bilinear transformation, determine the equivalent digital controller.

Compare the velocity error constants of the original analog system of Fig. P4.2a, and the equivalent digital system of Fig. P4.2b.

4.3 A unity-feedback system is characterized by the open-loop transfer function

$$G_{h0}G(z) = \frac{0.2385(z+0.8760)}{(z-1)(z-0.2644)}$$

The sampling period $T = 0.2$ sec.

Determine steady-state errors for unit-step, unit-ramp, and unit-acceleration inputs.

4.4 Predict the nature of the transient response of a discrete-time system whose characteristic equation is given by

$$z^2 - 1.9z + 0.9307 = 0$$

The sampling interval $T = 0.02$ sec.

4.5 The system of Fig. P4.5 contains a disturbance input $W(s)$, in addition to the reference input $R(s)$.

(a) Express $Y(z)$ as a function of the two inputs.

(b) Suppose that $D_2(z)$ and $D_3(z)$ are chosen such that $D_3(z) = D_2(z)G_{h0}G(z)$. Find $Y(z)$ as a function of the two inputs.

(c) What is the advantage of the choice in part (b) if it is desired to minimize the response $Y(z)$ to the disturbance $W(s)$?

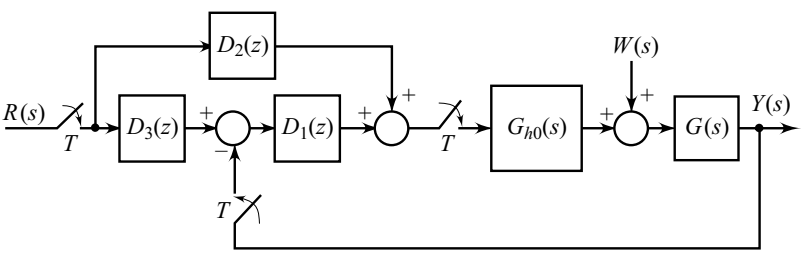


Fig. P4.5

4.6 Consider the system of Fig. P4.6. The design specifications for the system require that

(i) the steady-state error to a unit-ramp reference input be less than 0.01; and

(ii) a constant disturbance w should not affect the steady-state value of the output.

Show that these objectives can be met if $D(z)$ is a proportional-plus-integral compensator.

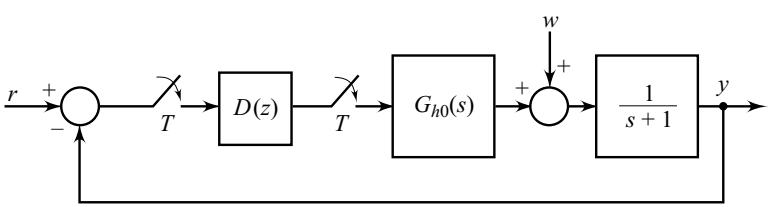


Fig. P4.6

4.7 Consider the feedback system shown in Fig. P4.7. The nominal values of the parameters K and τ of the plant $G(s)$ are both equal to 1. Find an expression for the sensitivity $S(z)$ of the closed-loop transfer function $M(z)$, with respect to incremental changes in open-loop transfer function $G_{h0}G(z)$. Plot $|S(e^{j\omega T})|$ for $0 \le \omega \le \omega_s/2$, where ω_s is the sampling frequency. Determine the bandwidth of the system if it is designed to have $|S(e^{j\omega T})| < 1$.

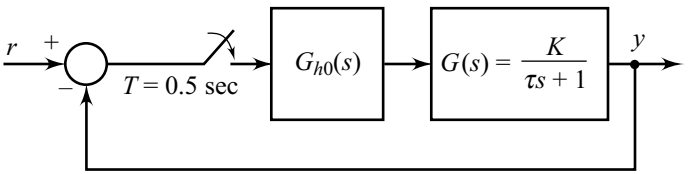


Fig. P4.7

4.8 A unity-feedback digital control system has open-loop transfer function

$$G_{h0}G(z) = \frac{0.368z + 0.264}{z^2 - 1.368z + 0.368}\ ;\ T = 1 \text{ sec}$$

The function $G_{h0}G(e^{j\omega T})$ may be used to obtain frequency response plots of the system. This function is, however, irrational. Prove that the relation

$$\omega = \frac{2}{T}\tan^{-1}\frac{\nu T}{2}$$

approximates $G_{h0}G(e^{j\omega T})$ to a rational function $G_{h0}G(j\nu)$.

For $G_{h0}G(j\nu)$, construct the Bode plot, and the log-magnitude versus phase angle plot and obtain the gain margin, the phase margin and the bandwidth ν_b. What is the corresponding value of ω_b? The –3 dB contour of the Nichols chart may be constructed using the following table:

Phase, degrees	0	–30	–60	–90	–120	–150	–180	–210
Magnitude, dB	7.66	6.8	4.18	0	–4.18	–6.8	–7.66	–6.8

4.9 Consider the control system of Fig. P4.9, where the plant transfer function

$$G(s) = \frac{1}{s(s+2)}\text{, and } T = 0.1 \text{ sec}$$

(a) Increase the plant gain to the value that results in $K_v = 5$. Then find the phase margin.
(b) Design a lead compensator that results in 55º phase margin with $K_v = 5$.
(c) Design a lag compensator that results in 55º phase margin with $K_v = 5$.
(d) Obtain the bandwidth realized by the three designs corresponding to parts (a), (b) and (c). Comment on the result.
(e) Is the selection of $T = 0.1$ sec justified from closed-loop bandwidth considerations?

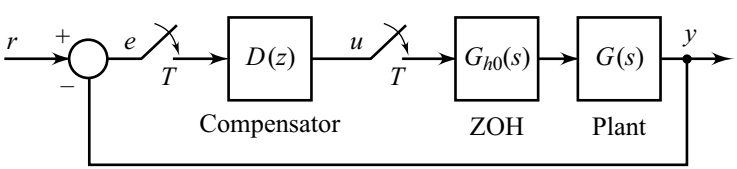


Fig. P4.9

4.10 Consider the control system of Fig. P4.9, where the plant transfer function is $G(s) = 1/s^2$, and $T = 0.1$ sec. Design a lead compensator such that the phase margin is 50º and the gain margin is at least 10 dB. Obtain the velocity error constant K_v of the compensated system.

Can the design be achieved using a lag compensator? Justify your answer.

4.11 Consider the control system of Fig. P4.9, where the plant transfer function is

$$G(s) = \frac{K}{s(s+5)}, \text{ and } T = 0.1 \text{ sec}$$

The performance specifications are given as

(i) velocity error constant $K_v \geq 10$;

(ii) phase margin $\Phi M \geq 60^\circ$; and

(iii) bandwidth $\omega_b = 8$ rad/sec.

(a) Find the value of K that gives $K_v = 10$. Determine the phase margin and the bandwidth of the closed-loop system.

(b) Show that if lead compensation is employed, the system bandwidth will increase beyond the specified value, and if lag compensation is attempted, the bandwidth will decrease sufficiently so as to fall short of the specified value.

(c) Design a lag section of a lag-lead compensator to provide partial compensation for the phase margin. Add a lead section to realize phase margin of 60º. Check the bandwidth of the lag-lead compensated system.

(d) Find the transfer function $D(z)$ of the lag-lead compensator and suggest a realization scheme.

4.12 Shown in Fig. P4.12a is a closed-loop temperature control system. Controlled electric heaters maintain the desired temperature of the liquid in the tank. The computer output controls electronic switches (triacs), to vary the effective voltage supplied to the heaters, from 0 V to 230 V. The temperature is measured by a thermocouple whose output is amplified to give a voltage in the range required by A/D converter. A simplified block diagram of the system, showing perturbation dynamics, is given in Fig. P4.12b.

(a) Consider the analog control loop of Fig. P4.12c. Determine K that gives 2% steady-state error to a step input.

(b) Let $D(z) = K$ obtained in part (a). Is the sampled-data system of Fig. P4.12b stable for this value of $D(z)$?

(c) Design a lag compensator for the system of part (b), such that 2% steady-state error is realized, the phase margin is greater than 40º and the gain margin is greater than 6 dB. Give the total transfer function $D(z)$ of the compensator.

(d) Can the design of part (c) be achieved using a lead compensator? Justify your answer.

4.13 (a) Consider a unity-feedback system with open-loop transfer function

$$G_{h0}G(z) = \frac{K(z-z_1)}{(z-p_1)(z-p_2)}; \; 0 \leq K < \infty$$

The poles and zero of this second-order transfer function lie on the real axis; the poles are adjacent or congruent, with the zero to their left. Prove that the complex-conjugate section of the root locus plot is a circle with the center at $z = z_1$, and the radius equal to $\sqrt{(z_1 - p_1)(z_1 - p_2)}$.

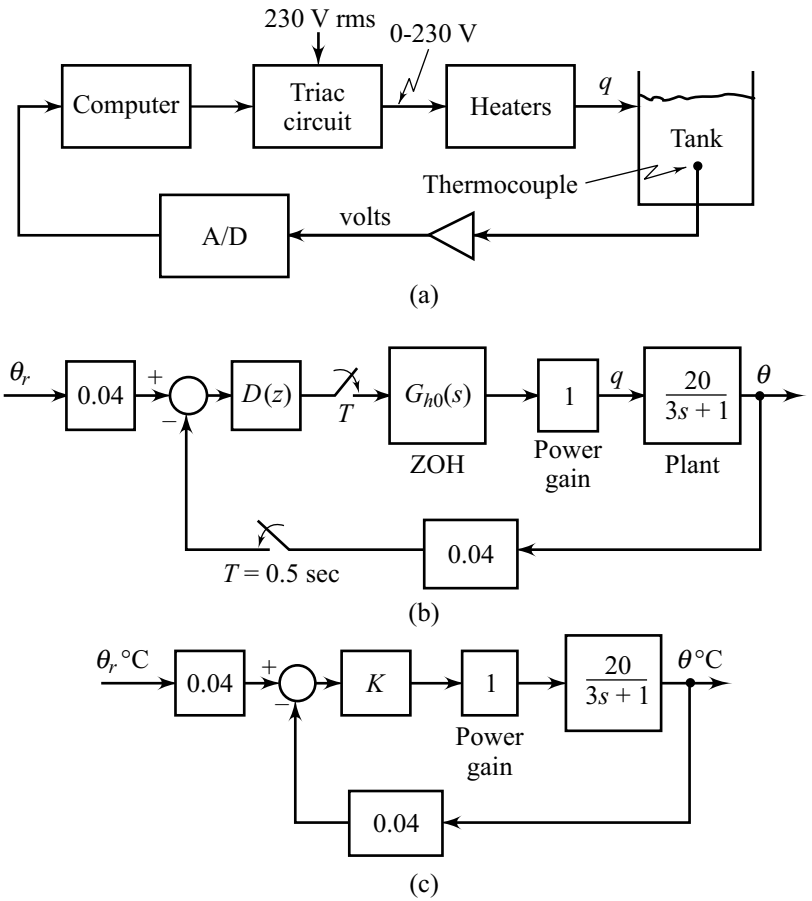


Fig. P4.12

(b) Given $$G_{h0}G(z) = \frac{K(z-0.9048)}{(z-1)^2}$$

Sketch the root locus plot for $0 \le K < \infty$. Using the information in the root locus plot, determine the range of values of K for which the closed-loop system is stable. Also determine the value of K for which the system closed-loop poles are real and multiple.

4.14 A sampled-data feedback control system is shown in Fig. P4.14. The controlled process of the system is described by the transfer function

$$G(s) = \frac{K}{s(s+1)}\ ;\ 0 \le K < \infty$$

The sampling period $T = 1$ sec.

(a) Sketch the root locus plot for the system on the z-plane and from there obtain the value of K that results in marginal stability.

(b) Repeat part (a) for (i) $T = 2$ sec, (ii) $T = 4$ sec, and compare the stability properties of the system with different values of sampling interval.

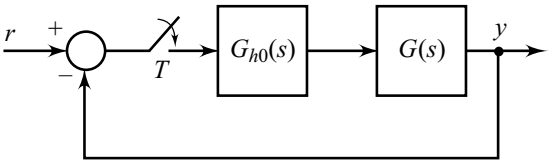


Fig. P4.14

4.15 The digital process of a unity-feedback system is described by the transfer function

$$G_{h0}G(z) = \frac{K(z+0.717)}{(z-1)(z-0.368)}\,;\; T = 1 \text{ sec}$$

Sketch the root locus plot for $0 \le K < \infty$ and from there obtain the following information:

(a) The value of K that results in marginal stability. Also find the frequency of oscillations.

(b) The value of K that results in $\zeta = 1$. What are the time constants of the closed-loop poles?

(c) The value of K that results in $\zeta = 0.5$. Also find the natural frequency ω_n for this value of K. You may use the following table to construct a constant-ζ locus on the z-plane corresponding to $\zeta = 0.5$.

Re	0.891	0.64	0.389	0.169	0	–0.113	–0.174	–0.188	–0.163
Im	0.157	0.37	0.463	0.464	0.404	0.310	0.207	0.068	0

4.16 The characteristic equation of a feedback control system is

$$z^2 + 0.2A\,z - 0.1\,A = 0$$

Sketch the root loci for $0 \le A < \infty$, and therefrom obtain the range of parameter A for which the system is stable.

4.17 The block diagram of a sampled-data system using a dc motor for speed control is shown in Fig. P4.17. The encoder senses the motor speed, and the output of the encoder is compared with the speed command. Sketch the root locus plot for $0 \le K < \infty$.

(a) For $K = 1$, find the time constant of the closed-loop pole.

(b) Find the value of K which results in a closed-loop pole whose time constant is less than or equal to one fourth of the value found in part (a).

Use the parameter values:

$$K_m = 1,\ \tau_m = 1,\ T = 0.1 \text{ sec},\ P = 60 \text{ pulses/revolution}.$$

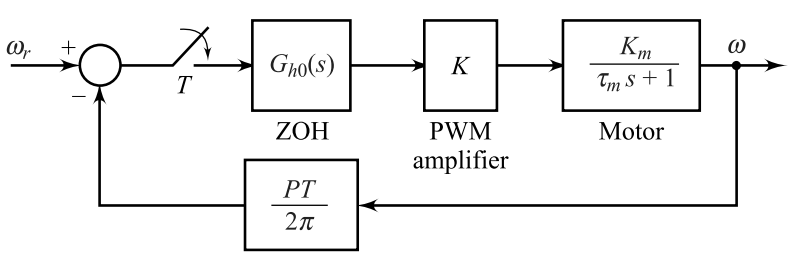


Fig. P4.17

4.18 Consider the system shown in Fig. P4.9 with $G(s) = \dfrac{1}{s(s+1)}$ and $T = 0.2$ sec.

(a) Design a lead compensator so that the dominant closed-loop poles of the system will have $\zeta = 0.5$ and $\omega_n = 4.5$.

(b) Obtain the velocity error constant K_v of the lead compensated system.

(c) Add a lag compensator in cascade so that K_v is increased by a factor of 3. What is the effect of the lag compensator on the transient response of the system?

(d) Obtain the transfer function $D(z)$ of the lag-lead compensator, and suggest a realization scheme.

Use root locus method.

4.19 Consider the system shown in Fig. P4.9 with

$$G(s) = \frac{1}{(s+1)(s+2)}\ ;\ T = 1 \text{ sec}$$

Design a compensator $D(z)$ that meets the following specifications on system performance:

(a) $\zeta = 0.5$;

(b) $\omega_n = 1.5$; and

(c) $K_p \geq 7.5$.

Use root locus method.

4.20 The block diagram of a digital control system is shown in Fig. P4.9. The controlled process is described by the transfer function

$$G(s) = \frac{K}{s^2}\ ;\ T = 1 \text{ sec}$$

which may represent a pure inertial load.

(a) The dominant closed-loop poles of the system are required to have $\zeta = 0.7$, $\omega_n = 0.3$ rad/sec. Mark the desired dominant closed-loop pole locations in the z-plane. The root loci must pass through these points.

(b) Place the zero of the compensator $D(z)$ below the dominant poles and find the location of pole of $D(z)$, so that the angle criterion at the dominant poles is satisfied. Find the value of K, so that the magnitude criterion at the dominant poles is satisfied.

(c) Find the acceleration error constant, K_a.

(d) Your design will result in specified values of ζ and ω_n for the closed-loop system response, only if the dominance condition is satisfied. Find the third pole of the closed-loop system and comment on the effectiveness of your design.

4.21 The configuration of a commercial broadcast videotape positioning system is shown in Fig. P4.21. The relationship between the armature voltage (applied to drive motor) and tape speed at the recording and playback heads, is approximated by the transfer function $G(s)$. The delay term involved, accounts for the propagation of speed changes along the tape, over the distance of physical separation of the tape drive mechanism and the recording and playback heads. The tape position is sensed by a recorded signal on the tape itself.

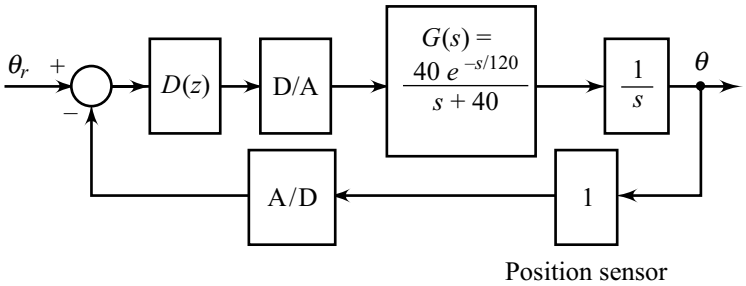


Fig. P4.21

Design the digital controller that should result in zero steady-state error to any step change in the desired tape position. The closed-loop poles of the system are required to lie within a circle of radius 0.56. Take the sampling interval $T = 1/120$ sec.

4.22 Consider the sampled-data system shown in Fig. P4.22; the plant is known to have the transfer function

$$G(s) = \frac{1}{s(s+1)}$$

A sampling period of $T = 0.1$ sec is to be used.

(a) Design a digital controller to realize the following specifications:
   (i) $\zeta = 0.8$;
   (ii) $\omega_n = 2\pi/10T$; and
   (iii) $K_v \geq 5$.

(b) Design a digital controller so that the response to unit-step input is

$$y(k) = 0, 0.5, 1, 1, \ldots$$

Find the steady-state error to unit-ramp input.

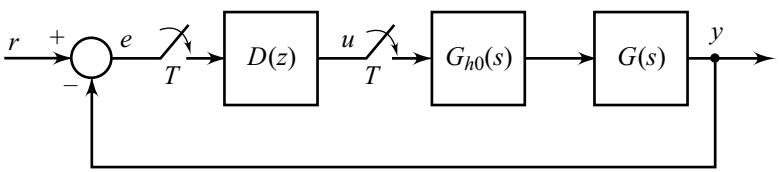


Fig. P4.22

4.23 In the control configuration of Fig. P4.22, find the control algorithm $D(z)$ so that the response to a unit-step function will be $y(t) = 1 - e^{-t}$. The plant transfer function is

$$G(s) = \frac{1}{10s+1}$$

Assume that the sampling interval $T = 2$ sec.

Part II

State Variable Methods in Automatic Control: Continuous-Time and Sampled-Data Systems

In Part I of the book, we developed some general procedures for the design of controllers. Our discussion was basically centered around the generalized operational block diagram of a feedback system, shown in Fig. 1.8.

We have assumed in our presentation, that the dynamic behavior of the plant can be represented (or approximated with 'sufficient' accuracy) by a linear time-invariant nth-order system, which is described by a strictly proper, minimal (controllable and observable) rational transfer function $G_P(s)$. We have also assumed that any *external disturbances* that affect the plant, can be represented by a single, additive signal $w(t)$, with known dynamic properties (refer to Fig.1.8). The dynamics in the feedback path (often attributed to the sensor), was assumed to be characterized by the proper minimal transfer function $H(s)$, which produces a continuous measure of the potentially noisy output $y(t)$.

We placed a restriction on the design of controllers: the controller can be represented by a linear time-invariant system, whose single output (for the single-input single-output (SISO) systems) $u(t)$ is produced by the input $r(t) - b(t) = \hat{e}(t)$. Therefore, its dynamic behavior can be described by

$$U(s) = D(s)[R(s) - B(s)]$$

where $D(s)$ is the proper minimal transfer function of the controller, whose degree defines the order of the controller.

We have observed that in many cases involving the so-called classical control techniques, the transfer function $A(s)$ (corresponding to the reference-input elements (Fig.1.8)), is assumed to be equal to $H(s)$. This implies the more restrictive unity-feedback configuration depicted in Fig. 1.12. However, the choice of $A(s) \neq H(s)$ would imply non-unity-feedback structure; the design procedures for this structure have been developed in our companion book [155]. In the vast majority of applications, the unity-feedback configuration is preferred because the error ($e(t) = r(t) - y(t) = y_r(t) - y(t)$) is explicitly present—both to drive the controller, and to be zeroed via feedback.

In this part of the book, we intend to relax the restrictions we have so far imposed on the development of general procedures for the design of controllers. We know that the output $y(t)$ does not represent the complete dynamic state of the plant at time t; it is the state vector $\mathbf{x}(t)=[x_1(t), \ldots, x_n(t)]^T$ which carries complete knowledge on the dynamics at time t. In the output-feedback configurations of the form shown in Fig. 1.8, only partial information on the dynamical state of the plant is fed back. We will relax this restriction and allow the complete state $\mathbf{x}(t)$ to be fed back.

In the classical configuration of Fig. 1.8, the controller output $u(t)$ is produced by one input: $[r(t) - b(t)]$. We will relax this restriction also, and allow the controller $u(t)$ to be a function of $r(t)$, and $b(t)$ independently.

The control law will take the form

$$u(t) = k_R\, r(t) - \mathbf{k}\,\mathbf{x}(t)$$

where k_R (scalar) is the reference control gain, and $\mathbf{k}$ ($1 \times n$ vector) is the state feedback gain.

Design procedures for such a control law will require state-space formulation for the dynamic systems in the feedback loop. We, therefore, begin our discussion with development of state variable models, and their analysis (Chapters 5 and 6). This will be followed by development of design procedures (Chapters 7 and 8).

Chapter 5

Control System Analysis using State Variable Methods

5.1 INTRODUCTION

In Part I of this book, we have seen that the root-locus method and the frequency-response method are quite powerful for the analysis and design of feedback control systems. The analysis and design are carried out using transfer functions, together with a variety of graphical tools such as root-locus plots, Nyquist plots, Bode plots, Nichols chart, etc. These techniques of the so-called *classical control theory* have been greatly enhanced by the availability, and low cost, of digital computers for system analysis and simulation. The graphical tools can now be more easily used with computer graphics.

The classical design methods suffer from certain limitations, due to the fact that the transfer function model is applicable only to linear time-invariant systems, and that, there too, it is generally restricted to Single-Input, Single-Output (SISO) systems. This is because the classical design approach becomes highly cumbersome for use in Multi-Input, Multi-Output (MIMO) systems. Another limitation of the transfer function technique is that it reveals only the system output for a given input and provides no information about the internal behavior of the system. There may be situations where the output of a system is stable and yet some of the system elements may have a tendency to exceed their specified ratings. In addition to this, it may sometimes be necessary, and advantageous, to provide a feedback proportional to the internal variables of a system, rather than the output alone, for the purpose of stabilizing and improving the performance of a system.

The limitations of classical methods, based on transfer function models, have led to the development of state variable approach of analysis and design. It is a direct time-domain approach which provides a basis for *modern control theory*. It is a powerful technique for the analysis and design of linear and nonlinear, time-invariant or time-varying MIMO systems. The organization of the state variable approach is such that it is easily amenable to solution through digital computers.

It will be incorrect to conclude from the foregoing discussion, that the state variable design methods can completely replace the classical design methods. In fact, the classical control theory, comprising a large body of use-tested knowledge, is still going strong. State variable design methods prove their mettle in applications that are intractable by classical methods.

The state variable formulation contributes to the application areas of classical control theory in a different way. To compute the response of $G(s)$ to an input $R(s)$, requires the expansion of

$\{G(s)R(s)\}$ into partial fractions; which, in turn, requires computation of all the poles of $\{G(s)R(s)\}$, or all the roots of a polynomial. The roots of a polynomial are very sensitive to their coefficients (refer to Review Example 3.3). Furthermore, to develop a computer program to carry out partial fraction expansion is not simple. On the other hand, the response of state variable equations is easy to program. Its computation does not require the computation of roots or eigenvalues. Therefore, it is less sensitive to parameter variations. For these reasons, it is desirable to compute the response of $G(s)$ through state variable equations. State variable formulation is thus the most efficient form of system representation—from the standpoint of computer simulation. For this reason, many Computer-Aided-Design (CAD) packages, handling both the classical and the modern tools of control system design, use this notation. It is, therefore, helpful for the control engineer to be familiar with state variable methods of system representation and analysis.

Part-II of this text presents an introduction to a range of topics which fall within the domain of state variable analysis and design. Our approach is to build on, and complement, the classical methods of analysis and design. State variable analysis and design methods use vector and matrix algebra and are, to some extent, different from those based on transfer functions. For this reason, we have not integrated the state variable approach with the frequency-domain approach based on transfer functions.

We have been mostly concerned with SISO systems in the text so far. In the remaining chapters also, our emphasis will be on the control of SISO systems. However, many of the analysis and design methods based on state variable concepts are applicable to both SISO and MIMO systems with almost equal convenience; the only difference being the additional computational effort for MIMO systems, which is taken care of by CAD packages. A specific reference to such results will be made at appropriate places in these chapters.

5.2 VECTORS AND MATRICES

This section is intended to be a concise summary of facts about vectors and matrices. Having them all at hand will minimize the need to consult a book on matrix theory. It also serves to define the notation and terminology which are, regrettably, not entirely standard.

No attempt has been made at proving every statement made in this section. The interested reader is urged to consult a suitable book (for example [27, 28]) for details of proofs.

5.2.1 Matrices[1]

Basic definitions and algebraic operations associated with matrices are given below.

Matrix

The matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} = [a_{ij}] \tag{5.1}$$

[1] We will use upper case bold letters to represent matrices and lower case bold letters to represent vectors.

is a rectangular array of *nm* elements. It has *n* rows and *m* columns. a_{ij} denotes (i, j)th element, i.e., the element located in *i*th row and *j*th column. **A** is said to be a *rectangular matrix* of order $n \times m$.

When $m = n$, i.e., the number of columns is equal to that of rows, the matrix is said to be a *square matrix* of order *n*.

A $n \times 1$ matrix, i.e., a matrix having only one column is called a *column matrix*. A $1 \times n$ matrix, i.e., a matrix having only one row is called a *row matrix*.

Diagonal Matrix

A diagonal matrix is a square matrix whose elements off the *principal diagonal* are all zeros ($a_{ij} = 0$ for $i \neq j$). The following matrix is a diagonal matrix.

$$\mathbf{\Lambda} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} = \text{diag}\,[a_{11} \quad a_{12} \quad \cdots \quad a_{nn}] \tag{5.2}$$

Unit (Identity) Matrix

A unit matrix **I** is a diagonal matrix whose diagonal elements are all equal to unity ($a_{ii} = 1$, $a_{ij} = 0$ for $i \neq j$).

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Whenever necessary, an $n \times n$ unit matrix will be denoted by $\mathbf{I}_n$.

Null (Zero) Matrix

A null matrix **0** is a matrix whose elements are all equal to zero.

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Whenever necessary, the dimensions of the null matrix will be indicated by two subscripts: $\mathbf{0}_{nm}$.

Lower-Triangular Matrix

A lower-triangular matrix **L** has all its elements *above* the principal diagonal equal to zero; $l_{ij} = 0$ if $i < j$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nm} \end{bmatrix}$$

Upper-Triangular Matrix

An upper-triangular matrix **U** has all its elements *below* the principal diagonal equal to zero; $u_{ij} = 0$ if $i > j$ for $1 \le i \le n$ and $1 \le j \le m$.

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ 0 & u_{22} & \cdots & u_{2m} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & u_{nm} \end{bmatrix}$$

Matrix Transpose

If the rows and columns of an $n \times m$ matrix **A** are interchanged, the resulting $m \times n$ matrix, denoted as $\mathbf{A}^T$, is called the *transpose* of the matrix **A**. Namely, if **A** is given by Eqn. (5.1), then

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix}$$

Some properties of the matrix transpose are

(i) $(\mathbf{A}^T)^T = \mathbf{A}$

(ii) $(k\mathbf{A})^T = k\mathbf{A}^T$, where k is a scalar

(iii) $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$

(iv) $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

Symmetric Matrix

If a square matrix **A** is equal to its transpose;

$$\mathbf{A} = \mathbf{A}^T,$$

then the matrix **A** is called a symmetric matrix.

Skew-Symmetric Matrix

If a square matrix **A** is equal to the negative of its transpose;

$$\mathbf{A} = -\mathbf{A}^T,$$

then the matrix **A** is called a skew-symmetric matrix.

Conjugate Matrix

If the complex elements of a matrix **A** are replaced by their respective conjugates, then the resulting matrix is called the conjugate of **A**.

Conjugate Transpose

The conjugate transpose is the conjugate of the transpose of a matrix. Given a matrix **A**, the conjugate transpose is denoted by **A***, and is equal to conjugate of $\mathbf{A}^T$.

Hermitian Matrix

If a square matrix **A** is equal to its conjugate transpose;

$$\mathbf{A} = \mathbf{A}^*,$$

then the matrix **A** is called a Hermitian matrix. For matrices whose elements are all real (*real matrices*), symmetric and Hermitian mean the same thing.

Skew-Hermitian Matrix

If a square matrix **A** is equal to the negative of its conjugate transpose;

$$\mathbf{A} = -\mathbf{A}^*,$$

then the matrix **A** is called a skew-Hermitian matrix. For real matrices, skew-symmetric and skew-Hermitian mean the same thing.

Determinant of a Matrix

Determinants are defined for square matrices only. The determinant of the $n \times n$ matrix **A**, written as $|\mathbf{A}|$, or *det* **A**, is a scalar-valued function of **A**. It is found through the use of minors and cofactors.

The *minor* m_{ij} of the element a_{ij} is the determinant of a matrix of order $(n-1)\times(n-1)$, obtained from **A** by removing the row and the column containing a_{ij}.

The *cofactor* c_{ij} of the element a_{ij} is defined by the equation

$$c_{ij} = (-1)^{i+j} m_{ij}$$

Determinants can be evaluated by the method of *Laplace expansion*. If **A** is an $n \times n$ matrix, any arbitrary row k can be selected and $|\mathbf{A}|$ is then given by

$$|\mathbf{A}| = \sum_{j=1}^{n} a_{kj}\, c_{kj}$$

Similarly, Laplace expansion can be carried out with respect to any arbitrary column l, to obtain

$$|\mathbf{A}| = \sum_{i=1}^{n} a_{il}\, c_{il}$$

Laplace expansion reduces the evaluation of an $n \times n$ determinant down to the evaluation of a string of $(n-1)\times(n-1)$ determinants, namely, the cofactors.

Some properties of determinants are

(i) $det\,\mathbf{AB} = (det\,\mathbf{A})(det\,\mathbf{B})$
(ii) $det\,\mathbf{A}^T = det\,\mathbf{A}$

(iii) $det\, k\mathbf{A} = k^n\, det\, \mathbf{A}$; $\mathbf{A}$ is $n \times n$ matrix and k is scalar

(iv) The determinant of any diagonal or triangular matrix is the product of its diagonal elements.

Singular Matrix

A square matrix is called singular if the associated determinant is zero.

Nonsingular Matrix

A square matrix is called nonsingular if the associated determinant is nonzero.

Adjoint Matrix

The adjoint matrix of a square matrix $\mathbf{A}$ is found by replacing each element a_{ij} of matrix **A,** by its cofactor c_{ij} and then transposing.

$$adj\, \mathbf{A} = \mathbf{A}^{+} = \begin{bmatrix} c_{11} & c_{21} & \cdots & c_{n1} \\ c_{12} & c_{22} & \cdots & c_{n2} \\ \vdots & \vdots & & \vdots \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{bmatrix} = [c_{ji}]$$

Note that

$$\mathbf{A}(adj\, \mathbf{A}) = (adj\, \mathbf{A})\mathbf{A} = |\mathbf{A}|\, \mathbf{I} \tag{5.3}$$

Matrix Inverse

The inverse of a square matrix is written as $\mathbf{A}^{-1}$, and is defined by the relation

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

From Eqn. (5.3) and the definition of the inverse matrix, we have

$$\mathbf{A}^{-1} = \frac{adj\, \mathbf{A}}{|\mathbf{A}|} \tag{5.4}$$

Some properties of matrix inverse are

(i) $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$ (ii) $(\mathbf{A}^{T})^{-1} = (\mathbf{A}^{-1})^{T}$ (iii) $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$

(iv) $det\, \mathbf{A}^{-1} = \dfrac{1}{det\, \mathbf{A}}$ (v) $det\, \mathbf{P}^{-1}\mathbf{A}\mathbf{P} = det\, \mathbf{A}$

(vi) Inverse of diagonal matrix given by Eqn. (5.2) is

$$\mathbf{\Lambda}^{-1} = \begin{bmatrix} 1/a_{11} & 0 & \cdots & 0 \\ 0 & 1/a_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1/a_{nn} \end{bmatrix} = \text{diag} \left[\frac{1}{a_{11}} \quad \frac{1}{a_{22}} \cdots \frac{1}{a_{nn}} \right]$$

Rank of a Matrix

The rank $\rho(\mathbf{A})$ of a matrix $\mathbf{A}$ is the dimension of the largest array in $\mathbf{A}$ with a nonzero determinant. Some properties of rank are

(i) $\rho(\mathbf{A}^T) = \rho(\mathbf{A})$

(ii) The rank of a rectangular matrix cannot exceed the lesser of the number of rows and the number of columns. A matrix whose rank is equal to the lesser of the number of rows and number of columns is said to be of *full rank*.

$\rho(\mathbf{A}) \leq \min(n, m)$; $\mathbf{A}$ is $n \times m$ matrix

(iii) The rank of a product of two matrices cannot exceed the rank of the either:

$\rho(\mathbf{AB}) \leq \min[\rho(\mathbf{A}), \rho(\mathbf{B})]$

Trace of a Matrix

The trace of a square matrix $\mathbf{A}$ is the sum of the elements on the principal diagonal.

$$tr\,\mathbf{A} = \sum_i a_{ii} \tag{5.5}$$

Some properties of trace are

(i) $tr\,\mathbf{A}^T = tr\,\mathbf{A}$

(ii) $tr(\mathbf{A} + \mathbf{B}) = tr\,\mathbf{A} + tr\,\mathbf{B}$

(iii) $tr\,\mathbf{AB} = tr\,\mathbf{BA}$; $tr\,\mathbf{AB} \neq (tr\,\mathbf{A})(tr\,\mathbf{B})$

(iv) $tr\,\mathbf{P}^{-1}\mathbf{AP} = tr\,\mathbf{A}$

Partitioned Matrix

A matrix can be partitioned into submatrices or vectors. Broken lines are used to show the partitioning when the elements of the submatrices are explicitly shown. For example,

$$\mathbf{A} = \left[\begin{array}{cc:c} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \hdashline a_{31} & a_{32} & a_{33} \end{array}\right]$$

The broken lines indicating the partitioning are sometimes omitted when the context makes it clear that partitioned matrices are being considered. For example, the matrix $\mathbf{A}$ given above may be expressed as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

We will be frequently using the following forms of partitioning.

(i) Matrix $\mathbf{A}$ partitioned into its columns:

$$\mathbf{A} = [\,\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_m]$$

where

$$\mathbf{a}_i = \begin{bmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{ni} \end{bmatrix} = i\text{th column in } \mathbf{A}$$

(ii) Matrix **A** partitioned into its rows:

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha}_n \end{bmatrix}$$

where

$$\boldsymbol{\alpha}_i = [a_{i1} \quad a_{i2} \quad \cdots \quad a_{im}] = i\text{th row in } \mathbf{A}$$

(iii) A *block diagonal matrix* is a square matrix that can be partitioned so that the nonzero elements are contained only in square submatrices along the main diagonal,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_m \end{bmatrix} = \text{diag}\,[\mathbf{A}_1 \quad \mathbf{A}_2 \quad \cdots \quad \mathbf{A}_m]$$

For this case

(i) $|\mathbf{A}| = |\mathbf{A}_1||\mathbf{A}_2| \cdots |\mathbf{A}_m|$

(ii) $\mathbf{A}^{-1} = \text{diag}\,[\mathbf{A}_1^{-1} \quad \mathbf{A}_2^{-1} \cdots \mathbf{A}^{-1}{}_m]$, provided that $\mathbf{A}^{-1}$ exists.

5.2.2 Vectors

We will be mostly concerned with vectors and matrices that have *real* elements. We, therefore, restrict our discussion to these cases only. An extension of the results to the situations where the vectors/matrices have complex elements is quite straightforward.

Scalar (Inner) Product of Vectors

The scalar product of two $n \times 1$ constant vectors **x** and **y** is defined as

$$\begin{aligned} < \mathbf{x}, \mathbf{y} > &= \mathbf{x}^T\mathbf{y} \\ &= [x_1 \quad x_2 \quad \cdots \quad x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = x_1y_1 + x_2\,y_2 + \quad \cdots \quad + x_n\,y_n \\ &= \mathbf{y}^T\mathbf{x} \end{aligned}$$

Vector Norm

The concept of norm of a vector is a generalization of the idea of length. For the vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

the *Euclidean vector norm* $\|\mathbf{x}\|$ is defined by

$$\|\mathbf{x}\| = (x_1^2 + x_2^2 + \cdots + x_n^2)^{1/2} = (\mathbf{x}^T\mathbf{x})^{1/2} \tag{5.6a}$$

In two or three dimensions, it is easy to see that this definition for the length of $\mathbf{x}$ satisfies the conditions of Euclidean geometry. It is a generalization to n dimensions of the theorem of Pythagoras.

For any nonsingular matrix $\mathbf{P}$, the vector

$$\mathbf{y} = \mathbf{P}\mathbf{x}$$

has the Euclidean norm

$$\|\mathbf{y}\| = [(\mathbf{P}\mathbf{x})^T(\mathbf{P}\mathbf{x})]^{1/2} = (\mathbf{x}^T\mathbf{P}^T\mathbf{P}\mathbf{x})^{1/2}$$

Letting $\mathbf{Q} = \mathbf{P}^T\mathbf{P}$, we write

$$\|\mathbf{y}\| = (\mathbf{x}^T\mathbf{Q}\mathbf{x})^{1/2}$$

or

$$\|\mathbf{x}\|_{\mathbf{Q}} = (\mathbf{x}^T\mathbf{Q}\mathbf{x})^{1/2} \tag{5.6b}$$

We call $\|\mathbf{x}\|_{\mathbf{Q}}$ the norm of $\mathbf{x}$ with respect to $\mathbf{Q}$. It is, in fact, a generalization of the norm defined in (5.6a) in that it is a measure of the size of $\mathbf{x}$ 'weighted' by the matrix $\mathbf{Q}$.

Matrix Norm

The norm of a matrix is a measure of the 'size' of the matrix (not its dimension). For the matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

the *Euclidean matrix norm* $\|\mathbf{A}\|$ is defined by

$$\|\mathbf{A}\| = \left[\sum_{i,j=1}^{n} a_{ij}^2\right]^{1/2} = [tr\,(\mathbf{A}^T\mathbf{A}]^{1/2} \tag{5.6c}$$

We can also describe the size of $\mathbf{A}$ by

$$\|\mathbf{A}\| = \max_{\mathbf{x}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|};\ \mathbf{x} \neq \mathbf{0}$$

i.e., the largest value of the ratio of the length $\|\mathbf{A}\mathbf{x}\|$ to the length $\|\mathbf{x}\|$.

Using the Euclidean norm for vectors, we obtain

$$\|\mathbf{A}\| = \max_{\mathbf{x}} \frac{\left(\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}\right)^{1/2}}{\left(\mathbf{x}^T\mathbf{x}\right)^{1/2}} = \max_{\mathbf{x}} \left(\frac{\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}\right)^{1/2}$$

The maximum value of the ratio in this expression can be determined in terms of the *eigenvalues*[2] of the matrix $\mathbf{A}^T\mathbf{A}$. The *real symmetric* matrix $\mathbf{A}^T\mathbf{A}$ has all real and nonnegative eigenvalues and the maximum value of the ratio $(\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x})/(\mathbf{x}^T\mathbf{x})$ is equal to the maximum eigenvalue of $\mathbf{A}^T\mathbf{A}$ (for proof, refer to [107]). Therefore,

$$\|\mathbf{A}\| = (\text{Maximum eigenvalue of } \mathbf{A}^T\mathbf{A})^{1/2} \tag{5.6d}$$

This definition of the matrix norm is known as the *spectral norm*[3] of $\mathbf{A}$.

The square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ are called the *singular values* of $\mathbf{A}$. The spectral norm of $\mathbf{A}$ is equal to its largest singular value.

Singular values of a matrix are useful in numerical analysis. The ratio of the largest to the smallest singular values of $\mathbf{A}$, called the *condition number* of $\mathbf{A}$, is a measure of how close the matrix $\mathbf{A}$ comes to being singular. The matrix $\mathbf{A}$ is, therefore, 'ill-conditioned' if its condition number is large.

Orthogonal Vectors

Any two vectors which have a zero scalar product are said to be orthogonal vectors. Two $n \times 1$ vectors $\mathbf{x}$ and $\mathbf{y}$ are orthogonal if

$$\mathbf{x}^T\mathbf{y} = 0$$

A set of vectors is said to be orthogonal if, and only if, every two vectors from the set are orthogonal: $\mathbf{x}^T\mathbf{y} = 0$ for all $\mathbf{x} \neq \mathbf{y}$ in the set.

Unit Vector

A unit vector $\hat{\mathbf{x}}$ is, by definition, a vector whose norm is unity; $\|\hat{\mathbf{x}}\| = 1$. Any nonzero vector $\mathbf{x}$ can be normalized to form a unit vector.

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Orthonormal Vectors

A set of vectors is said to be orthonormal if, and only if, the set is orthogonal and each vector in this orthogonal set is a unit vector.

Orthogonal Matrix

Suppose that $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is an orthogonal set:

$$\mathbf{x}_i^T\mathbf{x}_i = 1, \qquad \text{for all } i$$

---

[2] The roots of the equation

$$|\lambda\mathbf{I} - \mathbf{A}| = 0$$

are called the eigenvalues of matrix $\mathbf{A}$. Detailed discussion is given in Section 5.6.

[3] Refer to [105] for other valid vector and matrix norms.

and

$$\mathbf{x}_i^T \mathbf{x}_j = 0 \qquad \text{for all } i \text{ and } j \text{ with } i \neq j.$$

If we form the $n \times n$ matrix

$$\mathbf{P} = [\, \mathbf{x}_1 \;\; \mathbf{x}_2 \cdots \mathbf{x}_n],$$

it follows from partitioned multiplication that

$$\mathbf{P}^T\mathbf{P} = \mathbf{I}$$

That is,

$$\mathbf{P}^T = \mathbf{P}^{-1}$$

Such a matrix **P** is called an orthogonal matrix.

Linearly Dependent Vectors

Consider a set of m vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, each of which has n components. If there exists a set of m scalars α_i, at least one of which is not zero, which satisfies

$$\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \cdots + \alpha_m\mathbf{x}_m = \mathbf{0},$$

then the set of vectors $\{\mathbf{x}_i\}$ is said to be linearly dependent.

Linearly Independent Vectors

Any set of vectors $\{\mathbf{x}_i\}$ which is not linearly dependent is said to be linearly independent. That is, if

$$\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \cdots + \alpha_m\mathbf{x}_m = \mathbf{0}$$

implies that each $\alpha_i = 0$, then $\{\mathbf{x}_i\}$ are linearly independent vectors.

Test for Linear Independence

Consider the set of m vectors $\{\mathbf{x}_i\}$, each of which has n components, with $m \neq n$. Assume that this set is linearly dependent so that

$$\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \cdots + \alpha_m\mathbf{x}_m = \mathbf{0}$$

with at least one nonzero α_i.

Premultiplying both sides of this equation by $\mathbf{x}_i^T$, gives a set of m simultaneous equations:

$$\alpha_1\mathbf{x}_i^T\mathbf{x}_1 + \alpha_2\,\mathbf{x}_i^T\mathbf{x}_2 + \cdots + \alpha_m\,\mathbf{x}_i^T\mathbf{x}_m = 0;\ i = 1, 2, \ldots, m$$

These equations can be written in the matrix form as

$$\begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \cdots & \mathbf{x}_1^T\mathbf{x}_m \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \cdots & \mathbf{x}_2^T\mathbf{x}_m \\ \vdots & \vdots & & \vdots \\ \mathbf{x}_m^T\mathbf{x}_1 & \mathbf{x}_m^T\mathbf{x}_2 & \cdots & \mathbf{x}_m^T\mathbf{x}_m \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.7a}$$

or

$$\mathbf{G}\boldsymbol{\alpha} = \mathbf{0}$$

If the $m \times m$ matrix $\mathbf{G}$ has a nonzero determinant, then $\mathbf{G}^{-1}$ exists, and

$$\boldsymbol{\alpha} = \mathbf{G}^{-1}\mathbf{0} = \mathbf{0} \tag{5.7b}$$

This contradicts the assumption of at least one nonzero α_i. The matrix $\mathbf{G}$ is called the *Grammian matrix*.

A necessary and sufficient condition for the set $\{\mathbf{x}_i\}$ to be linearly dependent is that $|\mathbf{G}| = 0$.

Linear Independence and Rank

The column rank of a matrix $\mathbf{A}$ is equal to maximum number of linearly independent columns in $\mathbf{A}$.

The maximum number of linearly independent columns of a matrix is equal to the maximum number of linearly independent rows. Therefore, the column rank of $\mathbf{A}$ = the row rank of $\mathbf{A} = \rho(\mathbf{A})$, which is, in turn, equal to the order of the largest square array in $\mathbf{A}$ whose determinant does not vanish.

Vector Functions

In Section 5.8, we will require a test for the linear independence of the rows of a matrix whose elements are functions of time.

Consider a matrix

$$\mathbf{F}(t) = \begin{bmatrix} f_{11}(t) & f_{12}(t) & \cdots f_{1m}(t) \\ \vdots & \vdots & \vdots \\ f_{n1}(t) & f_{n2}(t) & \cdots f_{nm}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1(t) \\ \vdots \\ \mathbf{f}_n(t) \end{bmatrix}$$

$\mathbf{f}_i(t)$; $i = 1, \ldots, n$ are the n row vectors of matrix $\mathbf{F}$; each vector has m components.

The scalar product of two $1 \times m$ vector functions $\mathbf{f}_i(t)$ and $\mathbf{f}_j(t)$ on $[t_0, t_1]$ is by definition,

$$< \mathbf{f}_i, \mathbf{f}_j > = \int_{t_0}^{t_1} \mathbf{f}_i(t)\mathbf{f}_j^T(t)\, dt$$

The set of n row-vector functions $\{\mathbf{f}_1(t), \ldots, \mathbf{f}_n(t)\}$ are linearly dependent if there exists a set of n scalars α_i, at least one of which is not zero, which satisfies

$$\alpha_1\mathbf{f}_1(t) + \alpha_2\mathbf{f}_2(t) + \cdots + \alpha_n\mathbf{f}_n(t) = \mathbf{0}_{1\times m}$$

or

$$\alpha_1 \begin{bmatrix} f_{11}(t) \\ \vdots \\ f_{1m}(t) \end{bmatrix} + \alpha_2 \begin{bmatrix} f_{21}(t) \\ \vdots \\ f_{2m}(t) \end{bmatrix} + \cdots + \alpha_n \begin{bmatrix} f_{n1}(t) \\ \vdots \\ f_{nm}(t) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

Equivalently, the n rows $\mathbf{f}_i(t)$ are linearly dependent if

$$\boldsymbol{\alpha}^T\mathbf{F}(t) = \mathbf{0} \tag{5.8a}$$

for some

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \neq \mathbf{0}$$

The Grammian matrix of functions $\mathbf{f}_i(t)$, $i = 1, \ldots, n$; where $\mathbf{f}_i(t)$ is the ith row of the matrix $\mathbf{F}(t)$, is given by (refer to Eqns (5.7))

$$\mathbf{W}(t_0, t_1) = \int_{t_0}^{t_1} \mathbf{F}(t)\mathbf{F}^T(t)\, dt \tag{5.8b}$$

The functions $\mathbf{f}_1(t), \ldots, \mathbf{f}_n(t)$, which are the rows of the matrix $\mathbf{F}(t)$, are linearly dependent on $[t_0, t_1]$ if, and only if, the $n \times n$ constant matrix $\mathbf{W}(t_0, t_1)$ is singular.

5.2.3 Quadratic Forms and Definiteness of Matrices

An expression such as

$$V(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}\, x_i\, x_j$$

involving terms of second degree in x_i and x_j, is known as the *quadratic form* of n variables. Such scalar-valued functions are extensively used in stability analysis and modern control design.

In practice, one is usually concerned with quadratic forms $V(x_1, x_2, \ldots, x_n)$ that assume only *real* values. When x_i, x_j, and q_{ij} are all real, the value of V is real, and the quadratic form can be expressed in the vector-matrix notation as

$$V(\mathbf{x}) = [x_1 \quad x_2 \quad \cdots \quad x_n] \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

or $$V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x}$$

Any real square matrix $\mathbf{Q}$ may be written as the sum of a symmetric matrix $\mathbf{Q}_s$ and a skew-symmetric matrix $\mathbf{Q}_{sk}$, as shown below.

Let

$$\mathbf{Q} = \mathbf{Q}_s + \mathbf{Q}_{sk}$$

Taking transpose of both sides,

$$\mathbf{Q}^T = \mathbf{Q}_s^T + \mathbf{Q}_{sk}^T = \mathbf{Q}_s - \mathbf{Q}_{sk}$$

Solving for $\mathbf{Q}_s$ and $\mathbf{Q}_{sk}$, we obtain

$$\mathbf{Q}_s = \frac{\mathbf{Q} + \mathbf{Q}^T}{2}; \quad \mathbf{Q}_{sk} = \frac{\mathbf{Q} - \mathbf{Q}^T}{2}$$

For a real matrix $\mathbf{Q}$, the quadratic function $V(\mathbf{x})$ is, therefore, given by

$$V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} = \mathbf{x}^T(\mathbf{Q}_s + \mathbf{Q}_{sk})\mathbf{x} = \mathbf{x}^T\mathbf{Q}_s\mathbf{x} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}^T\mathbf{x}$$

Since $\mathbf{x}^T\mathbf{Q}\mathbf{x} = (\mathbf{x}^T\mathbf{Q}\mathbf{x})^T = \mathbf{x}^T\mathbf{Q}^T\mathbf{x}$, we have

$$V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}_s\mathbf{x}$$

Thus, in quadratic function $V(\mathbf{x})$, only the symmetric portion of $\mathbf{Q}$ is of importance. We shall, therefore, tacitly assume that $\mathbf{Q}$ is symmetric.

It may be noted that real vector $\mathbf{x}$ and real matrix $\mathbf{Q}$ do not constitute necessary requirements for $V(\mathbf{x})$ to be real. $V(\mathbf{x})$ can be real when $\mathbf{Q}$ and $\mathbf{x}$ are possibly complex; it can easily be established that for a Hermitian matrix $\mathbf{Q}$,

$$V(\mathbf{x}) = \mathbf{x}^*\mathbf{Q}\mathbf{x}$$

has real values.

Our discussion will be restricted to *real symmetric* matrices $\mathbf{Q}$.

If for all $\mathbf{x} \neq \mathbf{0}$,

(i) $V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} \geq 0$,
then $V(\mathbf{x})$ is called a positive semidefinite function and $\mathbf{Q}$ is called a *positive semidefinite matrix*;

(ii) $V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} > 0$,
then $V(\mathbf{x})$ is called a positive definite function and $\mathbf{Q}$ is called a *positive definite matrix*;

(iii) $V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} \leq 0$,
then $V(\mathbf{x})$ is called a negative semidefinite function and $\mathbf{Q}$ is called a *negative semidefinite matrix*; and

(iv) $V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} < 0$,
then $V(\mathbf{x})$ is called a negative definite function and $\mathbf{Q}$ is called a *negative definite matrix.*

Tests for Definiteness

(i) *Eigenvalues of* $\mathbf{Q}$ *and the nature of quadratic form*

A real symmetric matrix $\mathbf{Q}$ has all real eigenvalues, and the signs of the eigenvalues of $\mathbf{Q}$ determine the nature of the quadratic form $\mathbf{x}^T\mathbf{Q}\mathbf{x}$, as summarized in Table 5.1.

Table 5.1

Eigenvalues of $\mathbf{Q}$	*Nature of quadratic form* $\mathbf{x}^T\mathbf{Q}\mathbf{x}$
All $\lambda_i > 0$	Positive definite
All $\lambda_i \geq 0$	Positive semidefinite
All $\lambda_i < 0$	Negative definite
All $\lambda_i \leq 0$	Negative semidefinite
Some $\lambda_i \geq 0$, some $\lambda_j \leq 0$	Indefinite

(ii) *Sylvester's Criterion*

The Sylvester's criterion states that the necessary and sufficient conditions for

$$V(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} = [x_1 \quad x_2 \cdots x_n] \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

to be positive definite are that all the successive *principal minors* of **Q** be positive, i.e.,

$$q_{11} > 0; \begin{vmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{vmatrix} > 0; \begin{vmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{vmatrix} > 0; \cdots ; |\mathbf{Q}| > 0 \tag{5.9}$$

The necessary and sufficient conditions for $V(\mathbf{x})$ to be positive semidefinite are that **Q** is singular and all the other principal minors of **Q** are non-negative.

$V(\mathbf{x})$ is negative definite if $[-V(\mathbf{x})]$ is positive definite. Similarly, $V(\mathbf{x})$ is negative semidefinite if $[-V(\mathbf{x})]$ is positive semidefinite.

5.3 STATE VARIABLE REPRESENTATION

We will be mostly concerned with SISO system configurations of the type shown in the block diagram of Fig. 5.1. The plant in the figure is a physical process, characterized by the state variables $x_1, x_2, \ldots, x_n$, the output variable y and the input variable u.

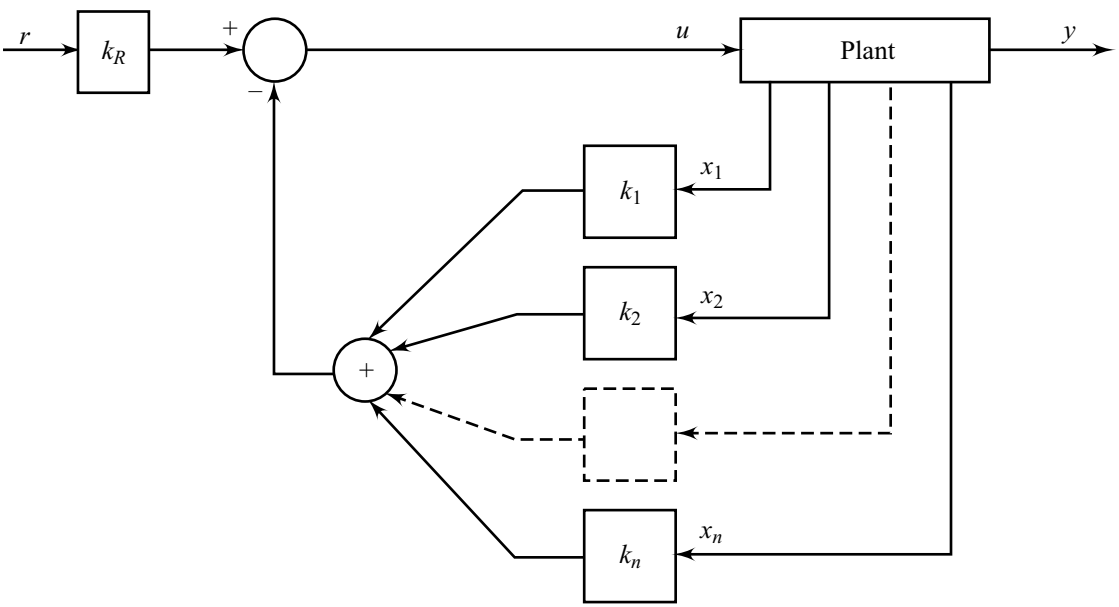


Fig. 5.1 A state-feedback control configuration

5.3.1 State Variable Concepts

The modeling process of linear systems involves setting up a chain of cause-effect relationships, beginning from the input variable and ending at the output variable. This cause-effect chain includes a number of internal variables. These variables are eliminated, both in the differential equation model and in the transfer function model, to obtain the final relationship between the input and the output.

Analysis of systems with the input-output model will not give any information about the behavior of the internal variables for different operating conditions. For a better understanding of the system behavior, its mathematical model should include the internal variables also. The state variable techniques of system representation and analysis, make the internal variables an integral part of the system model, and thus provide more complete information about the system behavior. In order to appreciate how the internal variables are included in the system representation, let us examine the modeling process by means of a simple example.

Consider the network shown in Fig. 5.2a. The set of voltages and currents associated with all the branches of the network at any time t, represents the *status* of the network at that time. Application of Kirchhoff's current law at nodes 1 and 2 of the network gives the following equations:

$$\frac{de_1}{dt} + 2\frac{de_2}{dt} = \frac{u - e_1}{2}$$

$$2\frac{de_3}{dt} = 2\frac{de_2}{dt}$$

Application of Kirchhoff's *voltage law* to the loop consisting of the three capacitors yields

$$e_1(t) - e_2(t) = e_3(t)$$

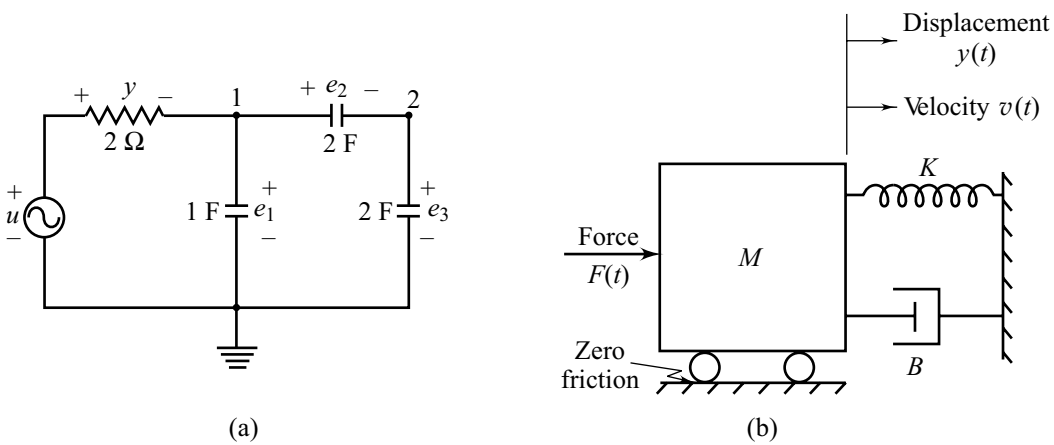


Fig. 5.2

All other voltage and current variables associated with the network are related to e_1, e_2, e_3 and input u, through linear algebraic equations. This means that their values (at all instants of time) can be obtained from the knowledge of the network variables e_1, e_2, e_3 and the input variable u, merely by linear combinations. In other words, the reduced set $\{e_1(t), e_2(t), e_3(t)\}$ of network variables with the input variable $u(t)$, completely represents the status of the network at time t.

For the purpose of finding a mathematical model to represent a system, we will naturally choose a minimal set of variables that describes the status of the system. Such a set would be obtained when none of the selected variables is related to other variables and the input, through linear algebraic equations. A little consideration shows that there is redundancy in the set$\{e_1(t), e_2(t), e_3(t)\}$ for the network of Fig. 5.2a; a set of two variables, say, $\{e_1(t), e_2(t)\}$, with the input $u(t)$ represents the network completely at time t.

Manipulation of the network equations obtained earlier, gives

$$\frac{de_1(t)}{dt} = -\frac{1}{4}e_1(t) + \frac{1}{4}u(t)$$

$$\frac{de_2(t)}{dt} = -\frac{1}{8}e_1(t) + \frac{1}{8}u(t)$$

This set of equations constitutes a mathematical model for the system. It is a set of two first-order differential equations. Its complete solution for any given $u(t)$ applied at $t = t_0$, will require a knowledge of the value of the selected variables $\{e_1, e_2\}$ at $t = t_0$. To put it differently, we can say that if the values of $\{e_1, e_2\}$ at $t = t_0$ are known, then the values of these variables at any time $t > t_0$, in response to a given input $u_{(t_0, t]}$, can be obtained by the solution of the two first-order differential equations. A set of system variables having this property, is called a set of *state variables*. The set of values of these variables at any time t is called the *state* of the system at time t. The set of first-order differential equations relating the first derivative of the state variables with the variables themselves and the input, is called a set of *state equations*. It is also to be noted that the number of state variables needed to form a correct and complete model of the system is equal to the *order of the system*.

An important point regarding the concept of state of a system is that the choice of state variables is not unique. In the network of Fig. 5.2a, instead of voltages $\{e_1, e_2\}$, the voltages $\{e_2, e_3\}$ or $\{e_3, e_1\}$ may be taken as state variables to define the state of the system. In fact, any set of variables $x_1(t)$ and $x_2(t)$, given by

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix}$$

where p_{ij} are constants such that the matrix

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

is nonsingular, is qualified to describe the state of the system of Fig. 5.2a because we can express the capacitor voltages in terms of the selected variables $x_1(t)$ and $x_2(t)$. This brings out additional facts that there are *infinitely many choices of state variables for any given system, and that the selected state variables may not correspond to physically measurable quantities of the system*. Later in this chapter, we will see that all the choices of state variables are not equally convenient. Usually, state variables are chosen so that they correspond to physically measurable quantities, or lead to particularly simplified calculations.

For a particular goal of study of a given physical system, we may not be interested in the total information about the system at a particular time. We may be interested only in a part of the total information. This is called the *output of the system*, which can be obtained algebraically from the information of the system state and the input. The output is, by definition, a physical attribute of the system and is measurable. For example, in the electric network of Fig. 5.2a, the information of interest may be the voltage across the resistor. The output

$$y(t) = -e_1(t) + u(t)$$

As another example of the state of a system, consider the mechanical network shown in Fig. 5.2b. The force $F(t)$ is the input variable. Defining the displacement $y(t)$ of the mass as the output variable, we obtain the following input-output model for the system:

$$M\frac{d^2y(t)}{dt^2} + B\frac{dy(t)}{dt} + Ky(t) = F(t)$$

An alternative form of the input-output model is the transfer function model:

$$\frac{Y(s)}{F(s)} = \frac{1}{Ms^2 + Bs + K}$$

The set of forces, velocities, and displacements, associated with all the elements of the mechanical network at any time t, represents the status of the network at that time. A little consideration shows that values of all the system variables (at all instants of time) can be obtained from the knowledge of the system variables $y(t)$ and $v(t)$, and the input variable $F(t)$, merely by linear combinations. The dynamics of $y(t)$ and $v(t)$ are given by the following first-order differential equations:

$$\frac{dy(t)}{dt} = v(t)$$

$$\frac{dv(t)}{dt} = -\frac{K}{M}y(t) - \frac{B}{M}v(t) + \frac{1}{M}F(t)$$

The variables $\{y(t), v(t)\}$ are, therefore, the state variables of the system of Fig. 5.2b, and the two first-order differential equations given above, are the state equations of the system. Using standard symbols for state variables and input variable, we can write the state equations as

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{K}{M}x_1 - \frac{B}{M}x_2 + \frac{1}{M}u$$

$$x_1(t) \triangleq y(t);\ x_2(t) \triangleq v(t);\ u(t) \triangleq F(t)$$

Defining $y(t)$ as the output variable, the output equation becomes

$$y = x_1$$

We can now appreciate the following definitions:

State

The *state* of a dynamic system is the smallest set of variables (called *state variables*) such that the knowledge of these variables at $t = t_0$, together with the knowledge of the input for $t \geq t_0$, completely determines the behavior of the system for any time $t \geq t_0$.

State Vector

If n state variables $x_1, x_2, \ldots, x_n$, are needed to completely describe the behavior of a given system, then these n state variables can be considered the n components of a vector **x**. Such a vector is called a *state vector*.

State Space

The n-dimensional space whose coordinate axes consist of the x_1-axis, x_2-axis, …, x_n-axis, is called a *state space*.

State Trajectory

At any time t_0, the state vector (and hence the state of the system) defines a point in the state space. As time progresses and the system state changes, a set of points will be defined. This set of points, the locus of the tip of the state vector as time progresses, is called the *state trajectory* of the system.

State space and state trajectory in two-dimensional cases are referred to as the *phase plane* and *phase trajectory*, respectively.

5.3.2 State Variable Modeling

We know through our modeling experience that the application of physical laws to mechanical, electrical, thermal, liquid-level, and other physical processes results in a set of first-order and second-order differential equations.[4] Linear time-invariant differential equations can be rearranged in the following form:

$$\begin{aligned}
\dot{x}_1(t) &= a_{11}x_1(t) + a_{12}x_2(t) + \cdots + a_{1n}x_n(t) + b_1 u(t) \\
\dot{x}_2(t) &= a_{21}x_1(t) + a_{22}x_2(t) + \cdots + a_{2n}x_n(t) + b_2 u(t) \\
&\vdots \\
\dot{x}_n(t) &= a_{n1}x_1(t) + a_{n2}x_2(t) + \cdots + a_{nn}x_n(t) + b_n u(t)
\end{aligned} \tag{5.10a}$$

where the coefficients a_{ij} and b_i are constants. These n first-order differential equations are called *state equations* of the system.

Integration of Eqn. (5.10a) gives

$$x_i(t) = x_i(t_0) + \int_{t_0}^{t} [a_{i1}x_1(t) + \cdots + a_{in}x_n(t) + b_i u(t)]dt;\ i = 1, 2, \ldots, n$$

Thus, the n state variables and hence the state of the system can be determined uniquely at any $t > t_0$, if each state variable is known at $t = t_0$, and the control force $u(t)$ is known throughout the interval t_0 to t.

The output $y(t)$ at any time t will be a function of $x_1(t), x_2(t), \ldots, x_n(t)$. However, in some cases, the output may also depend upon the instantaneous value of the input $u(t)$. For linear systems, the output is a linear combination of the state variables and the input:

$$y(t) = c_1 x_1(t) + c_2 x_2(t) + \cdots + c_n x_n(t) + d\, u(t) \tag{5.10b}$$

where c_i and d are constants. The algebraic equation (5.10b) is called *output equation* of the system.

Since every real-world system has some nonlinearity, a mathematical model of the form (5.10) is an approximation to reality. Many real-world nonlinearities involve a 'smooth' curvelinear relation between independent and dependent variables. Nonlinear functions $f_i(\cdot)$ and $g(\cdot)$ of the form

$$\dot{x}_i(t) = f_i(x_1(t), x_2(t), \ldots, x_n(t), u(t));\ x_i(t_0) \triangleq x_i^0 \tag{5.11a}$$

$$y(t) = g(x_1(t), x_2(t), \ldots, x_n(t), u(t)) \tag{5.11b}$$

[4] Chapter 2 of reference [155]

may be linearized about a selected operating point using the multivariable form of the Taylor series:

$$f(x_1, x_2, x_3, \cdots) = f(x_{10}, x_{20}, \ldots) + \left[\left.\frac{\partial f}{\partial x_1}\right|_{x_{10},\, x_{20},\, \ldots}\right](x_1 - x_{10}) + \left[\left.\frac{\partial f}{\partial x_2}\right|_{x_{10},\, x_{20},\, \ldots}\right](x_2 - x_{20}) + \cdots \quad (5.11c)$$

One of the advantages of state variable formulation is that an extremely compact vector-matrix notation can be used for the mathematical model. Using the laws of matrix algebra, it becomes much less cumbersome to manipulate the equations.

In the vector-matrix notation, we may write Eqns (5.10) as

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} u(t); \quad \begin{bmatrix} x_1(t_0) \\ x_2(t_0) \\ \vdots \\ x_n(t_0) \end{bmatrix} \triangleq \begin{bmatrix} x_1^0 \\ x_2^0 \\ \vdots \\ x_n^0 \end{bmatrix} \quad (5.12a)$$

$$y(t) = [c_1 \quad c_2 \cdots c_n] \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} + d\, u(t) \quad (5.12b)$$

In compact notation, Eqns (5.12) may be expressed as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t);\ \mathbf{x}(t_0) \triangleq \mathbf{x}^0 : \textit{State equation} \quad (5.13a)$$

$$y(t) = \mathbf{c}\mathbf{x}(t) + du(t) : \textit{Output equation} \quad (5.13b)$$

where

$\mathbf{x}(t)$ = $n \times 1$ state vector of nth-order dynamic system
$u(t)$ = system input
$y(t)$ = defined output
$\mathbf{A}$ = $n \times n$ matrix
$\mathbf{b}$ = $n \times 1$ column matrix
$\mathbf{c}$ = $1 \times n$ row matrix
d = scalar, representing direct coupling between input and output (direct coupling is rare in control systems, i.e., usually $d = 0$)

Example 5.1 *Two very usual applications of dc motors are in speed and position control systems.*

Figure 5.3 gives the basic block diagram of a speed control system. A separately excited dc motor drives the load. A dc tachogenerator is attached to the motor shaft; speed signal is fed back and the error signal is used to control the armature voltage of the motor.

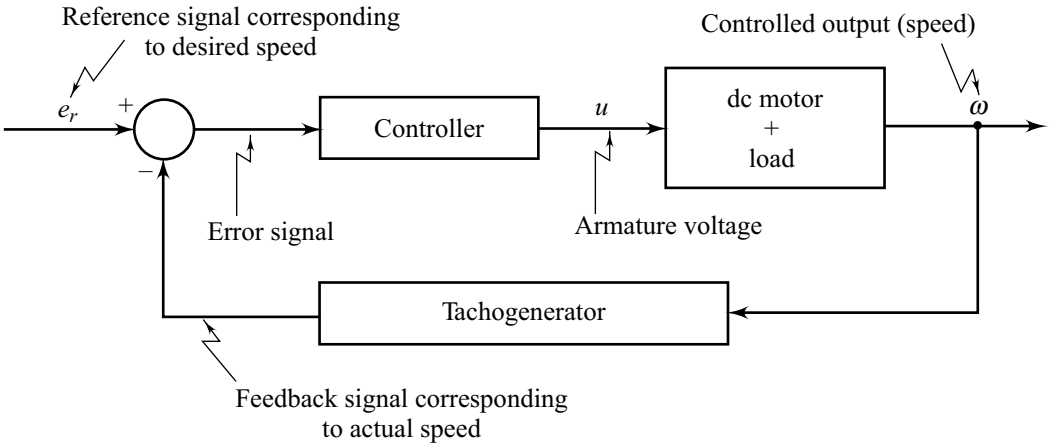


Fig. 5.3 Basic block diagram of a closed-loop speed control system

In the following, we derive the plant model for the speed control system. A separately excited dc motor with armature voltage control, is shown in Fig. 5.4.

The voltage loop equation is

$$u(t) = L_a \frac{di_a(t)}{dt} + R_a\, i_a(t) + e_b(t) \tag{5.14a}$$

where

L_a = inductance of armature winding (henrys);
R_a = resistance of armature winding (ohms);
i_a = armature current (amperes);
e_b = back emf (volts); and
u = applied armature voltage (volts).

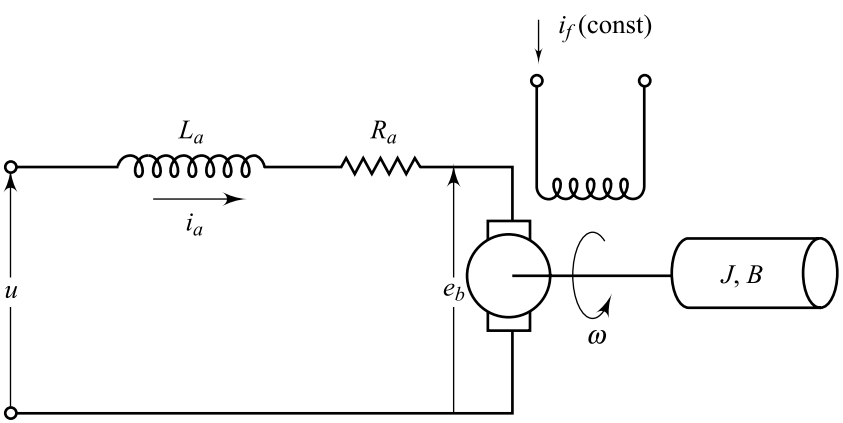


Fig. 5.4 Model of a separately excited dc motor

The torque balance equation is

$$T_M(t) = J\,\frac{d\omega(t)}{dt} + B\omega(t) \tag{5.14b}$$

where

T_M = torque developed by the motor (newton-m);
J = equivalent moment of inertia of motor and load referred to motor shaft (kg-m^2);
B = equivalent viscous friction coefficient of motor and load referred to motor shaft ((newton-m)/(rad/sec)); and
ω = angular velocity of motor shaft (rad/sec).

In servo applications, the dc motors are generally used in the linear range of the magnetization curve. Therefore, the air gap flux ϕ is proportional to the field current. For the armature controlled motor, the field current i_f is held constant. Therefore, the torque T_M developed by the motor, which is proportional to the product of the armature current and the air gap flux, can be expressed as

$$T_M(t) = K_T\, i_a(t) \tag{5.14c}$$

where

K_T = motor torque constant ((newton-m/amp))

The counter electromotive force e_b, which is proportional to ϕ and ω, can be expressed as

$$e_b(t) = K_b\, \omega(t) \tag{5.14d}$$

where

K_b = back emf constant[5] (volts/(rad/sec))

Equations (5.14) can be reorganized as

$$\begin{aligned} \frac{di_a(t)}{dt} &= -\frac{R_a}{L_a}\, i_a(t) - \frac{K_b}{L_a}\,\omega(t) + \frac{1}{L_a}\,u(t) \\ \frac{d\omega(t)}{dt} &= \frac{K_T}{J}\, i_a(t) - \frac{B}{J}\,\omega(t) \end{aligned} \tag{5.15}$$

$x_1(t) = \omega(t)$, and $x_2(t) = i_a(t)$ is the obvious choice for state variables. The output variable is $y(t) = \omega(t)$.

The plant model of the speed control system, organized into the vector-matrix notation, is given below.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{B}{J} & \frac{K_T}{J} \\ -\frac{K_b}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix} u(t)$$

$$y(t) = x_1(t)$$

Let us assign numerical values to the system parameters.

[5] In MKS units, $K_b = K_T$; Section 3.2 of reference [155].

For the parameters[6]

$$R_a = 1 \text{ ohm}, L_a = 0.1 \text{ H}, J = 0.1 \text{ kg-m}^2, B = 0.1 \text{ (newton-m)/(rad/sec)}, K_b = K_T = 0.1, \tag{5.16}$$

the plant model becomes

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t) \end{aligned} \tag{5.17}$$

where $\mathbf{A} = \begin{bmatrix} -1 & 1 \\ -1 & -10 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}; \mathbf{c} = [1 \quad 0]$

Example 5.2

Figure 5.5 gives the basic block diagram of a position control system. The controlled variable is now the angular position $\theta(t)$ of the motor shaft:

$$\frac{d\theta(t)}{dt} = \omega(t) \tag{5.18}$$

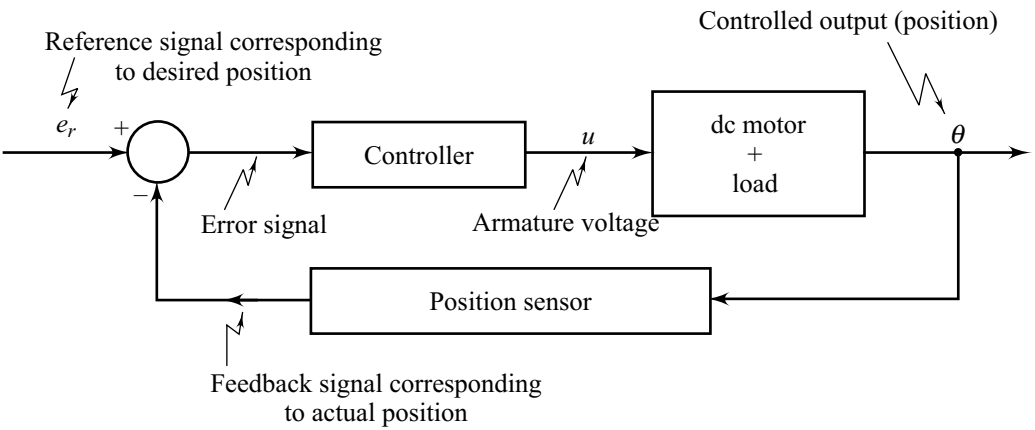


Fig. 5.5 Basic block diagram of a closed-loop position control system

We make the following choice for state and output variables.

$$x_1(t) = \theta(t), x_2(t) = \omega(t), x_3(t) = i_a(t), y(t) = \theta(t)$$

For this choice, we obtain the following plant model from Eqns (5.15) and (5.18).

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{B}{J} & \frac{K_T}{J} \\ 0 & -\frac{K_b}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L_a} \end{bmatrix} u(t)$$

$$y(t) = x_1(t)$$

[6] These parameters have been chosen for computational convenience

For the system parameters given by (5.16), the plant model for position control system becomes

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t) \end{aligned} \tag{5.19}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & -1 & -10 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix}; \mathbf{c} = [1 \quad 0 \quad 0]$$

In Examples 5.1 and 5.2 discussed above, the selected state variables are the physical quantities of the systems which can be measured.

We will see in Chapter 7 that in a physical system, in addition to output, other state variables could be utilized for the purpose of feedback. The implementation of design with state variable feedback becomes straightforward if the state variables are available for feedback. The choice of physical variables of a system as state variables, therefore, helps in the implementation of design. Another advantage of selecting physical variables for state variable formulation is that the solution of state equation gives time variation of variables which have direct relevance to the physical system.

5.3.3 Transformation of State Variables

It frequently happens that the state variables used in the original formulation of the dynamics of a system are not as convenient as another set of state variables. Instead of having to reformulate the system dynamics, it is possible to transform the set $\{\mathbf{A}, \mathbf{b}, \mathbf{c}, d\}$ of the original formulation (5.13), to a new set $\{\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}, \bar{d}\}$. The change of variables is represented by a linear transformation

$$\mathbf{x} = \mathbf{P}\bar{\mathbf{x}} \tag{5.20a}$$

where $\bar{\mathbf{x}}$ is a state vector in the new formulation, and $\mathbf{x}$ is the state vector in the original formulation. It is assumed that the transformation matrix $\mathbf{P}$ is a nonsingular $n \times n$ matrix, so that we can always write

$$\bar{\mathbf{x}} = \mathbf{P}^{-1}\mathbf{x} \tag{5.20b}$$

We assume, moreover, that $\mathbf{P}$ is a constant matrix.

The original dynamics are expressed by

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t); \mathbf{x}(t_0) \triangleq \mathbf{x}^0 \tag{5.21a}$$

and the output by

$$y(t) = \mathbf{c}\mathbf{x}(t) + du(t) \tag{5.21b}$$

Substitution of **x,** as given by Eqn. (5.20a), into these equations gives

$$\begin{aligned} \mathbf{P}\dot{\bar{\mathbf{x}}}(t) &= \mathbf{A}\mathbf{P}\bar{\mathbf{x}}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{P}\bar{\mathbf{x}}(t) + du(t) \end{aligned}$$

or

$$\dot{\bar{\mathbf{x}}}(t) = \bar{\mathbf{A}}\,\bar{\mathbf{x}}(t) + \bar{\mathbf{b}}\,u(t); \; \bar{\mathbf{x}}(t_0) = \mathbf{P}^{-1}\mathbf{x}(t_0) \tag{5.22a}$$

$$y(t) = \bar{\mathbf{c}}\,\bar{\mathbf{x}}(t) + \bar{d}\,u(t) \tag{5.22b}$$

with

$$\bar{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P},\ \bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b},\ \bar{\mathbf{c}} = \mathbf{c}\mathbf{P},\ \bar{d} = d$$

In the next section, we will prove that both the linear systems (5.21) and (5.22) have identical output responses for the same input. The linear system (5.22) is said to be *equivalent* to the linear system (5.21), and **P** is called an *equivalence* or *similarity transformation.*

It is obvious that there exist an infinite number of equivalent systems since the transformation matrix **P** can be arbitrarily chosen. Some transformations have been extensively used for the purposes of analysis and design. Five of such special (*canonical*) transformations will be used in the present and the next two chapters.

Example 5.3 *Example 5.1 Revisited*

For the system of Fig. 5.4, we have taken angular velocity $\omega(t)$ and armature current $i_a(t)$ as state variables:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \omega \\ i_a \end{bmatrix}$$

We now define new state variables as

$$\bar{x}_1 = \omega,\ \bar{x}_2 = -\omega + i_a$$

or

$$\bar{\mathbf{x}} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ -x_1 + x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

We can express velocity $x_1(t)$ and armature current $x_2(t)$ in terms of the variables $\bar{x}_1(t)$ and $\bar{x}_2(t)$:

$$\mathbf{x} = \mathbf{P}\,\bar{\mathbf{x}} \tag{5.23}$$

with

$$\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Using Eqns (5.22) and (5.17), we obtain the following state variable model for the system of Fig. 5.4, in terms of the transformed state vector $\bar{\mathbf{x}}(t)$:

$$\begin{aligned} \dot{\bar{\mathbf{x}}}(t) &= \bar{\mathbf{A}}\bar{\mathbf{x}}(t) + \bar{\mathbf{b}}\,u(t) \\ y(t) &= \bar{\mathbf{c}}\,\bar{\mathbf{x}}(t) \end{aligned} \tag{5.24}$$

where

$$\bar{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\,\mathbf{P} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ -1 & -10 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -11 & -11 \end{bmatrix}$$

$$\bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 10 \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$$

$$\bar{\mathbf{c}} = \mathbf{cP} = [1 \quad 0]\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = [1 \quad 0]$$

$$\bar{x}_1(t_0) = x_1(t_0);\ \bar{x}_2(t_0) = -x_1(t_0) + x_2(t_0)$$

Equations (5.24) give an alternative state variable model of the system previously represented by Eqns (5.17). $\bar{\mathbf{x}}(t)$ and $\mathbf{x}(t)$ both qualify to be state vectors of the given system (the two vectors individually characterize the system completely at time t), and the output $y(t)$, as we shall see shortly, is uniquely determined from either of the models (5.17) and (5.24). State variable model (5.24) is thus *equivalent* to the model (5.17), and the matrix **P** given by Eqn. (5.23) is an *equivalence* or *similarity transformation.*

The state variable model given by Eqns (5.24) is in a *canonical* (special) form. In Chapter 7, we will use this form of model for pole-placement design by state feedback.

5.3.4 State Diagrams

An important advantage of state variable formulation is that it is a straightforward method to obtain a simulation diagram for the state equations. This is extremely useful if we wish to use computer simulation methods to study dynamic systems. In the following, we give an example of analog simulation diagram. Examples of digital simulation will appear in Chapter 6.

For brevity, we consider a second-order system:

$$\begin{aligned} \dot{x}_1(t) &= a_{11}\, x_1(t) + a_{12}\, x_2(t) + b_1 u(t) \\ \dot{x}_2(t) &= a_{21}\, x_1(t) + a_{22}\, x_2(t) + b_2 u(t) \\ y(t) &= c_1 x_1(t) + c_2 x_2(t) \end{aligned} \tag{5.25}$$

It is evident that if we knew $\dot{x}_1$ and $\dot{x}_2$, we could obtain x_1 and x_2 by simple integration. Hence $\dot{x}_1$ and $\dot{x}_2$ should be the inputs to two integrators. The corresponding integrator outputs are x_1 and x_2. This leaves only the problem of obtaining $\dot{\mathbf{x}}_1$ and $\dot{\mathbf{x}}_2$ for use as inputs to the integrators. In fact, this is already specified by state equations. The completed state diagram is shown in Fig. 5.6. This diagram is essentially an analog-computer program for the given system.

5.4 CONVERSION OF STATE VARIABLE MODELS TO TRANSFER FUNCTIONS

We shall derive the transfer function of a SISO system from the Laplace-transformed version of the state and output equations. Refer to Section 5.2 for the vector and matrix operations used in the derivation.

Consider the state variable model (Eqns (5.13)):

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t);\ \mathbf{x}(t_0) \triangleq \mathbf{x}^0 \\ y(t) &= \mathbf{c}\mathbf{x}(t) + du(t) \end{aligned} \tag{5.26}$$

Taking the Laplace transform of Eqns (5.26), we obtain

$$\begin{aligned} s\mathbf{X}(s) - \mathbf{x}^0 &= \mathbf{A}\mathbf{X}(s) + \mathbf{b}U(s) \\ Y(s) &= \mathbf{c}\mathbf{X}(s) + dU(s) \end{aligned}$$

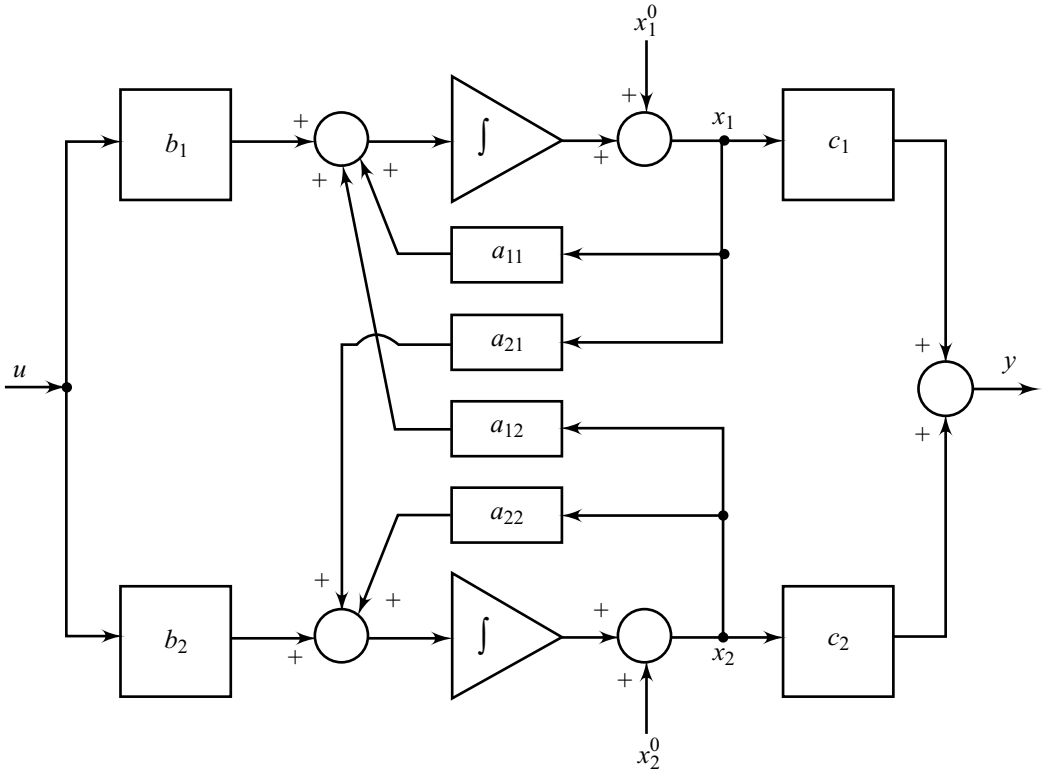


Fig. 5.6 State diagram for the system (5.25)

where

$$\mathbf{X}(s) \triangleq \mathscr{L}[\mathbf{x}(t)];\ U(s) \triangleq \mathscr{L}[u(t)];\ Y(s) \triangleq \mathscr{L}[y(t)]$$

Manipulation of these equations gives

$$(s\mathbf{I} - \mathbf{A})\mathbf{X}(s) = \mathbf{x}^0 + \mathbf{b}U(s);\ \mathbf{I} \text{ is } n \times n \text{ identity matrix}$$

or

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}^0 + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}U(s) \tag{5.27a}$$

$$Y(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}^0 + [\mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d]U(s) \tag{5.27b}$$

Equations (5.27) are algebraic equations. If $\mathbf{x}^0$ and $U(s)$ are known, $\mathbf{X}(s)$ and $Y(s)$ can be computed from these equations.

In the case of a zero initial state (i.e., $\mathbf{x}^0 = \mathbf{0}$), the input-output behavior of the system (5.26) is determined entirely by the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d \tag{5.28}$$

We can express the inverse of the matrix $(s\mathbf{I} - \mathbf{A})$ as

$$(s\mathbf{I} - \mathbf{A})^{-1} = \frac{(s\mathbf{I} - \mathbf{A})^{+}}{|s\mathbf{I} - \mathbf{A}|} \tag{5.29}$$

where

$|s\mathbf{I} - \mathbf{A}|$ = determinant of the matrix $(s\mathbf{I} - \mathbf{A})$

$(s\mathbf{I} - \mathbf{A})^{+}$= adjoint of the matrix $(s\mathbf{I} - \mathbf{A})$

Using Eqn. (5.29), the transfer function $G(s)$ given by Eqn. (5.28) can be written as

$$G(s) = \frac{\mathbf{c}(s\mathbf{I} - \mathbf{A})^{+}\mathbf{b}}{|s\mathbf{I} - \mathbf{A}|} + d \tag{5.30}$$

For a general nth-order matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},$$

the matrix $(s\mathbf{I} - \mathbf{A})$ has the following appearance:

$$(s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s-a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & s-a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & & \vdots \\ -a_{n1} & -a_{n2} & \cdots & s-a_{nn} \end{bmatrix}$$

If we imagine calculating $det(s\mathbf{I} - \mathbf{A})$, we see that one of the terms will be the product of diagonal elements of $(s\mathbf{I} - \mathbf{A})$:

$$(s - a_{11})(s - a_{22}) \cdots (s - a_{nn}) = s^n + \alpha_1' s^{n-1} + \cdots + \alpha_n',$$

a polynomial of degree n with the leading coefficient of unity. There will be other terms coming from the off-diagonal elements of $(s\mathbf{I} - \mathbf{A})$, but none will have a degree as high as n. Thus $|s\mathbf{I} - \mathbf{A}|$ will be of the following form:

$$|s\mathbf{I} - \mathbf{A}| = \Delta(s) = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n \tag{5.31}$$

where α_i are constant scalars.

This is known as the *characteristic polynomial* of the matrix **A**. It plays a vital role in the dynamic behavior of the system. The roots of this polynomial are called the *characteristic roots* or *eigenvalues* of matrix **A**. These roots, as we shall see in Section 5.7, determine the essential features of the unforced dynamic behavior of the system (5.26).

The adjoint of an $n \times n$ matrix is itself an $n \times n$ matrix, whose elements are the cofactors of the original matrix. Each cofactor is obtained by computing the determinant of the matrix that remains when a row and a column of the original matrix are deleted. It thus follows that each element in $(s\mathbf{I} - \mathbf{A})^{+}$ is a polynomial in s of maximum degree $(n - 1)$. Adjoint of $(s\mathbf{I} - \mathbf{A})$ can, therefore, be expressed as

$$(s\mathbf{I} - \mathbf{A})^{+} = \mathbf{Q}_1 s^{n-1} + \mathbf{Q}_2 s^{n-2} + \cdots + \mathbf{Q}_{n-1} s + \mathbf{Q}_n \tag{5.32}$$

where $\mathbf{Q}_i$ are constant $n \times n$ matrices.

We can express transfer function $G(s)$ given by Eqn. (5.30) in the following form:

$$G(s) = \frac{\mathbf{c}[\mathbf{Q}_1 s^{n-1} + \mathbf{Q}_2 s^{n-2} + \cdots + \mathbf{Q}_{n-1} s + \mathbf{Q}_n]\mathbf{b}}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1} s + \alpha_n} + d \tag{5.33}$$

$G(s)$ is thus a rational function of s. When $d = 0$, the degree of numerator polynomial of $G(s)$ is strictly less than the degree of the denominator polynomial and, therefore, the resulting transfer function is a *strictly proper transfer function*. When $d \neq 0$, the degree of numerator polynomial of $G(s)$ will be equal to the degree of the denominator polynomial, giving a *proper transfer function*. Further,

$$d = \lim_{s \to \infty} [G(s)] \tag{5.34}$$

From Eqns (5.31) and (5.33) we observe that the characteristic polynomial of matrix **A** of the system (5.26) is same as the denominator polynomial of the corresponding transfer function $G(s)$. If there are no cancellations between the numerator and denominator polynomials of $G(s)$ in Eqn. (5.33), the *eigenvalues* of matrix **A** are same as the *poles* of $G(s)$. We will take up in Section 5.9, this aspect of the correspondence between state variable models and transfer functions. It will be proved that for a completely controllable and completely observable state variable model, the eigenvalues of matrix **A** are same as the poles of the corresponding transfer function.

5.4.1 Invariance Property

It is recalled that the state variable model for a system is not unique, but depends on the choice of a set of state variables. A transformation

$$\mathbf{x}(t) = \mathbf{P}\,\bar{\mathbf{x}}(t);\ \mathbf{P} \text{ is a nonsingular matrix} \tag{5.35}$$

results in the following alternative state variable model (refer to Eqns (5.22)) for the system (5.26):

$$\dot{\bar{\mathbf{x}}}(t) = \bar{\mathbf{A}}\bar{\mathbf{x}}(t) + \bar{\mathbf{b}}u(t);\ \bar{\mathbf{x}}(t_0) = \mathbf{P}^{-1}\mathbf{x}(t_0) \tag{5.36a}$$

$$y(t) = \bar{\mathbf{c}}\,\bar{\mathbf{x}}(t) + du(t) \tag{5.36b}$$

where $\bar{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P},\ \bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b},\ \bar{\mathbf{c}} = \mathbf{c}\mathbf{P}$

The definition of new set of internal state variables should, evidently, not affect the eigenvalues or input-output behavior. This may be verified by evaluating the characteristic polynomial and the transfer function of the transformed system.

(i) $$|s\mathbf{I} - \bar{\mathbf{A}}| = |s\mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P}| = |s\mathbf{P}^{-1}\mathbf{P} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P}| = |\mathbf{P}^{-1}(s\mathbf{I} - \mathbf{A})\mathbf{P}| = |\mathbf{P}^{-1}|\,|s\mathbf{I} - \mathbf{A}|\,|\mathbf{P}| = |s\mathbf{I} - \mathbf{A}| \tag{5.37}$$

(ii) System output in response to input $u(t)$ is given by the transfer function

$$\begin{aligned}\bar{\mathbf{G}}(s) &= \bar{\mathbf{c}}(s\mathbf{I} - \bar{\mathbf{A}})^{-1}\,\bar{\mathbf{b}} + d = \mathbf{c}\mathbf{P}(s\mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P})^{-1}\mathbf{P}^{-1}\mathbf{b} + d \\ &= \mathbf{c}\mathbf{P}(s\mathbf{P}^{-1}\mathbf{P} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P})^{-1}\,\mathbf{P}^{-1}\mathbf{b} + d = \mathbf{c}\mathbf{P}[\mathbf{P}^{-1}(s\mathbf{I} - \mathbf{A})\mathbf{P}]^{-1}\mathbf{P}^{-1}\mathbf{b} + d \\ &= \mathbf{c}\mathbf{P}\mathbf{P}^{-1}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{P}\mathbf{P}^{-1}\mathbf{b} + d = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d = G(s)\end{aligned} \tag{5.38}$$

(iii) System output in response to initial state $\bar{\mathbf{x}}(t_0)$ is given by (refer to Eqn. (5.27b))

$$\bar{\mathbf{c}}(s\mathbf{I} - \bar{\mathbf{A}})^{-1}\,\bar{\mathbf{x}}(t_0) = \mathbf{c}\mathbf{P}(s\mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P})^{-1}\mathbf{P}^{-1}\mathbf{x}(t_0) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(t_0) \tag{5.39}$$

The input-output behavior of the system (5.26) is, thus, *invariant* under the transformation (5.35).

Example 5.4

Consider the position control system of Example 5.2. The plant model of the system is reproduced below:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t)\end{aligned} \tag{5.40}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & -1 & -10 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix}; \mathbf{c} = [1 \quad 0 \quad 0]$$

The characteristic polynomial of matrix **A** is

$$|s\mathbf{I} - \mathbf{A}| = \begin{vmatrix} s & -1 & 0 \\ 0 & s+1 & -1 \\ 0 & 1 & s+10 \end{vmatrix} = s(s^2 + 11s + 11)$$

The transfer function

$$\begin{aligned} G(s) &= \frac{Y(s)}{U(s)} = \frac{\mathbf{c}(s\mathbf{I} - \mathbf{A})^+ \mathbf{b}}{|\, s\mathbf{I} - \mathbf{A}\,|} \\ &= \frac{[1 \quad 0 \quad 0]\begin{bmatrix} s^2+11s+11 & s+10 & 1 \\ 0 & s(s+10) & s \\ 0 & -s & s(s+1) \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix}}{s(s^2+11s+11)} \\ &= \frac{10}{s(s^2+11s+11)} \end{aligned} \tag{5.41}$$

Alternatively, we can draw the state diagram of the plant model in signal-flow graph form and from there, obtain the transfer function using Mason's gain formula. For the plant model (5.40), the state diagram is shown in Fig. 5.7. Application of Mason's gain formula[7] yields

$$\begin{aligned} \frac{Y(s)}{U(s)} &= G(s) = \frac{10s^{-3}}{1-(-10s^{-1} - s^{-1} - s^{-2}) + 10s^{-2}} \\ &= \frac{10}{s^3 + 11s^2 + 11s} = \frac{10}{s(s^2+11s+11)} \end{aligned}$$

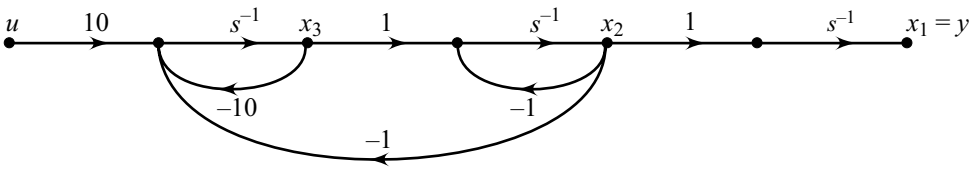


Fig. 5.7 State diagram for the system (5.40)

5.4.2 Resolvent Algorithm

The matrix

$$\mathbf{\Phi}(s) = (s\mathbf{I} - \mathbf{A})^{-1} = \frac{(s\mathbf{I} - \mathbf{A})^+}{|\, s\mathbf{I} - \mathbf{A}\,|} \tag{5.42}$$

[7] Section 2.12 of reference [155]

is known in mathematical literature as the *resolvent* of **A**. Resolvent matrix $\mathbf{\Phi}(s)$ can be expressed in the following form (refer to Eqns (5.31) and (5.32)):

$$\mathbf{\Phi}(s) = (s\mathbf{I} - \mathbf{A})^{-1} = \frac{\mathbf{Q}_1 s^{n-1} + \mathbf{Q}_2 s^{n-2} + \cdots + \mathbf{Q}_{n-1} s + \mathbf{Q}_n}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1} s + \alpha_n} \tag{5.43}$$

where $\mathbf{Q}_i$ are constant $(n \times n)$ matrices and α_j are constant scalars.

An interesting and useful relationship for the coefficient matrices $\mathbf{Q}_i$ of the adjoint matrix, can be obtained by multiplying both sides of Eqn. (5.43) by $|s\mathbf{I} - \mathbf{A}|(s\mathbf{I} - \mathbf{A})$. The result is

$$|s\mathbf{I} - \mathbf{A}|\,\mathbf{I} = (s\mathbf{I} - \mathbf{A})(\mathbf{Q}_1 s^{n-1} + \mathbf{Q}_2 s^{n-2} + \cdots + \mathbf{Q}_{n-1} s + \mathbf{Q}_n)$$

or

$$s^n\mathbf{I} + \alpha_1 s^{n-1}\mathbf{I} + \cdots + \alpha_n\mathbf{I} = s^n\mathbf{Q}_1 + s^{n-1}(\mathbf{Q}_2 - \mathbf{A}\mathbf{Q}_1) + \cdots + s(\mathbf{Q}_n - \mathbf{A}\mathbf{Q}_{n-1}) - \mathbf{A}\mathbf{Q}_n$$

Equating the coefficients of s^i on both the sides gives

$$\begin{aligned}
\mathbf{Q}_1 &= \mathbf{I} \\
\mathbf{Q}_2 &= \mathbf{A}\mathbf{Q}_1 + \alpha_1\mathbf{I} \\
\mathbf{Q}_3 &= \mathbf{A}\mathbf{Q}_2 + \alpha_2\mathbf{I} \\
&\vdots \\
\mathbf{Q}_n &= \mathbf{A}\mathbf{Q}_{n-1} + \alpha_{n-1}\mathbf{I} \\
\mathbf{0} &= \mathbf{A}\mathbf{Q}_n + \alpha_n\mathbf{I}
\end{aligned} \tag{5.44a}$$

We have thus determined that the leading coefficient of $(s\mathbf{I} - \mathbf{A})^{+}$ is the identity matrix, and that the subsequent coefficients can be obtained recursively. The last equation in (5.44a) is redundant, but can be used as a check when these recursion equations are used as the basis of a numerical algorithm.

An algorithm based on Eqns (5.44a) requires the coefficients α_i (i = 1, 2, ..., n) of the characteristic polynomial. Fortunately, the determination of these coefficients can be included in the algorithm, for it can be shown that[8]

$$\alpha_i = -\frac{1}{i}\, tr(\mathbf{A}\mathbf{Q}_i);\ i = 1, 2, ..., n \tag{5.44b}$$

where $tr(\mathbf{M})$, the trace of **M**, is the sum of all the diagonal elements of the matrix **M**.

The algorithm given by Eans (5.44), called the *resolvent algorithm*, is convenient for hand calculation and also easy to implement on a digital computer.

Example 5.5

Here we again compute $(s\mathbf{I} - \mathbf{A})^{-1}$ which appeared in Example 5.4, but this time using the resolvent algorithm (5.44).

$$\mathbf{Q}_1 = \mathbf{I},\ \alpha_1 = -tr(\mathbf{A}) = 11$$

$$\mathbf{Q}_2 = \mathbf{A} + \alpha_1\mathbf{I}$$

[8] The proof of relation (5.44b) is quite involved and will not be presented here. Refer to [108].

$$= \begin{bmatrix} 11 & 1 & 0 \\ 0 & 10 & 1 \\ 0 & -1 & 1 \end{bmatrix}; \alpha_2 = -\tfrac{1}{2} tr(\mathbf{AQ}_2) = 11$$

$$\mathbf{Q}_3 = \mathbf{AQ}_2 + \alpha_2 \mathbf{I}$$

$$= \begin{bmatrix} 11 & 10 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \alpha_3 = -\tfrac{1}{3} tr(\mathbf{AQ}_3) = 0$$

As a numerical check, we see that the relation

$$\mathbf{0} = \mathbf{AQ}_3 + \alpha_3 \mathbf{I}$$

is satisfied. Therefore,

$$(s\mathbf{I} - \mathbf{A})^{-1} = \mathbf{\Phi}(s) = \frac{\mathbf{Q}_1 s^2 + \mathbf{Q}_2 s + \mathbf{Q}_3}{s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3}$$

$$= \frac{1}{s(s^2 + 11s + 11)} \begin{bmatrix} s^2 + 11s + 11 & s + 10 & 1 \\ 0 & s(s+10) & s \\ 0 & -s & s(s+1) \end{bmatrix}$$

Cayley–Hamilton Theorem

Using resolvent algorithm, we develop here a fundamental property of the characteristic equation. To this end, we write from Eqns (5.44a)

$$\mathbf{Q}_2 = \mathbf{A} + \alpha_1 \mathbf{I}$$

$$\mathbf{Q}_3 = \mathbf{AQ}_2 + \alpha_2 \mathbf{I} = \mathbf{A}^2 + \alpha_1 \mathbf{A} + \alpha_2 \mathbf{I}$$

$$\vdots$$

$$\mathbf{Q}_n = \mathbf{A}^{n-1} + \alpha_1 \mathbf{A}^{n-2} + \cdots + \alpha_{n-1} \mathbf{I}$$

$$\mathbf{AQ}_n = \mathbf{A}^n + \alpha_1 \mathbf{A}^{n-1} + \cdots + \alpha_{n-1} \mathbf{A} = -\alpha_n \mathbf{I}$$

Therefore,

$$\mathbf{A}^n + \alpha_1 \mathbf{A}^{n-1} + \cdots + \alpha_{n-1} \mathbf{A} + \alpha_n \mathbf{I} = \mathbf{0} \tag{5.45}$$

This is the well-known result known as the *Cayley–Hamilton theorem*. Note that this equation is same as the characteristic equation

$$s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1} s + \alpha_n = 0 \tag{5.46}$$

with the scalar s^i in the latter replaced by the matrix $\mathbf{A}^i$ ($i = 1, 2, \ldots, n$).

Thus, another way of stating the Cayley–Hamilton theorem is as follows: *Every matrix satisfies its own characteristic equation.*

Later we will use the resolvent algorithm and the Cayley–Hamilton theorem for evaluation of the state transition matrix required for the solution of the state equations.

5.5 CONVERSION OF TRANSFER FUNCTIONS TO CANONICAL STATE VARIABLE MODELS

In the last section, we studied the problem—finding the transfer function from the state variable model of a system. The converse problem—finding a state variable model from the transfer function of a system, is the subject of discussion in this section. This problem is quite important because of the following reasons:

(i) Quite often the system dynamics is determined experimentally using standard test signals like a step, impulse, or sinusoidal signal. A transfer function is conveniently fitted to the experimental data in some best possible manner.

There are, however, many design techniques developed exclusively for state variable models. In order to apply these techniques, experimentally obtained transfer function descriptions must be realized into state variable models.

(ii) Realization of transfer functions into state variable models is needed even if the control system design is based on frequency-domain design methods. In these cases, the need arises for the purpose of transient response simulation. Many algorithms and numerical integration computer programs designed for solution of systems of first-order equations are available, but there is not much software for the numerical inversion of Laplace transforms. Thus, if a reliable method is needed for calculating the transient response of a system, one may be better off converting the transfer function of the system to state variable description, and numerically integrating the resulting differential equations, rather than attempting to compute the inverse Laplace transform by numerical methods.

We shall discuss here the problem of realization of transfer function into state variable models. Note the use of the term 'realization'. A state variable model that has a prescribed rational function $G(s)$ as its transfer function, is the *realization* of $G(s)$. The term 'realization' is justified by the fact that by using the state diagram corresponding to the state variable model, the system with the transfer function $G(s)$ can be built in the real world by an op amp circuit.[9]

The following three problems are involved in the realization of a given transfer function into state variable models:

(i) Is it possible at all, to obtain state variable description from the given transfer function?

(ii) If yes, is the state variable description *unique* for a given transfer function?

(iii) How do we obtain the state variable description from the given transfer function?

The answer to the first problem has been given in the last section. A rational function $G(s)$ is realizable by a finite dimensional linear time-invariant state model if, and only if, $G(s)$ is a proper rational function. A proper rational function will have state model of the form:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t) + du(t) \end{aligned} \tag{5.47}$$

where $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ and d are constant matrices of appropriate dimensions. A strictly proper rational function will have state model of the form

[9] Section 7.9 of reference [155]

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t)\\ y(t) &= \mathbf{c}\mathbf{x}(t)\end{aligned} \tag{5.48}$$

Let us now turn to the second problem. In the last section, we saw that there are innumerable systems that have the same transfer function. Hence, the representation of a transfer function in state variable form is obviously, not unique. However, all these representations will be equivalent.

In the remaining part of this section, we deal with the third problem. We shall develop three standard, or 'canonical' representations of transfer functions.

A linear time-invariant SISO system is described by transfer function of the form

$$G(s) = \frac{\beta_0 s^m + \beta_1 s^{m-1} + \cdots + \beta_m}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n}; m \le n$$

where the coefficients α_i and β_i are real constant scalars. Note that there is no loss in generality to assume the coefficient of s^n to be unity.

In the following, we derive results for $m = n$; these results may be used for the case $m < n$ by setting appropriate β_i coefficients equal to zero. Therefore, our problem is to obtain a state variable model corresponding to the transfer function

$$G(s) = \frac{\beta_0 s^n + \beta_1 s^{n-1} + \cdots + \beta_n}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n} \tag{5.49}$$

5.5.1 First Companion Form

Our development starts with a transfer function of the form

$$\frac{Z(s)}{U(s)} = \frac{1}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n} \tag{5.50}$$

which can be written as

$$(s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n)\, Z(s) = U(s)$$

The corresponding differential equation is

$$p^n z(t) + \alpha_1 p^{n-1} z(t) + \cdots + \alpha_n z(t) = u(t)$$

where

$$p^k z(t) \triangleq \frac{d^k z(t)}{dt^k}$$

Solving for highest derivative of $z(t)$, we obtain

$$p^n z(t) = -\alpha_1 p^{n-1} z(t) - \alpha_2 p^{n-2} z(t) - \cdots - \alpha_n z(t) + u(t) \tag{5.51}$$

Now consider a chain of n integrators as shown in Fig. 5.8. Suppose that the output of the last integrator is $z(t)$; then, the output of the just previous integrator is $pz = dz/dt$, and so forth. The output from the first integrator is $p^{n-1}z(t)$, and thus, the input to this integrator is $p^n z(t)$. This leaves only the problem of obtaining $p^n z(t)$ for use as input to the first integrator. In fact, this is already specified by Eqn. (5.51). Realization of this equation is shown in Fig. 5.8.

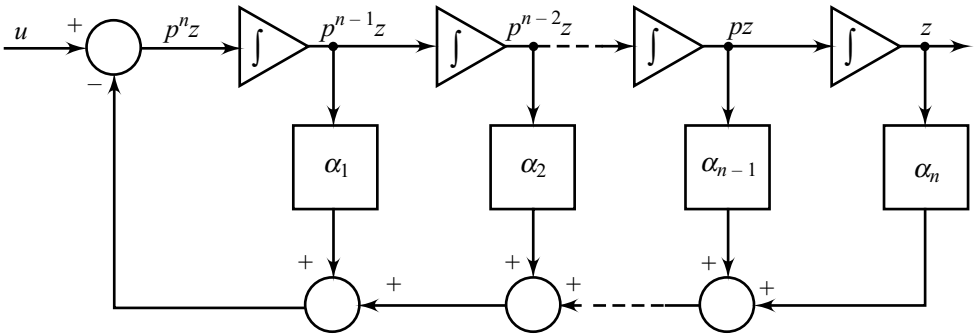


Fig. 5.8 Realization of the system (5.51)

Having developed a realization of the simple transfer function (5.50), we are now in a position to consider the more general transfer function (5.49). We decompose this transfer function into two parts, as shown in Fig. 5.9. The output $Y(s)$ can be written as

$$Y(s) = (\beta_0 s^n + \beta_1 s^{n-1} + \cdots + \beta_n)\, Z(s) \tag{5.52a}$$

where $Z(s)$ is given by

$$\frac{Z(s)}{U(s)} = \frac{1}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n} \tag{5.52b}$$

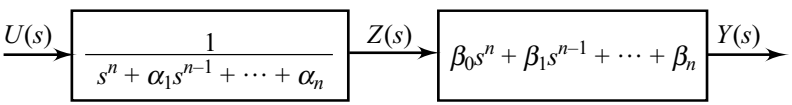


Fig. 5.9 Decomposition of the transfer function (5.49)

A realization of the transfer function (5.52b) has already been developed. Figure 5.8 shows this realization. The output of the last integrator is $z(t)$ and the inputs to the integrators in the chain—from the last to the first— are the n successive derivatives of $z(t)$.

Realization of the transfer function (5.52a) is now straightforward. The output

$$y(t) = \beta_0 p^n z(t) + \beta_1 p^{n-1} z(t) + \cdots + \beta_n z(t),$$

is the sum of the scaled versions of the inputs to the n integrators. Figure 5.10 shows complete realization of the transfer function (5.49). All that remains to be done is to write the corresponding differential equations.

To get one state variable model of the system, we identify the output of each integrator in Fig. 5.10 with a state variable starting at the right and proceeding to the left. The corresponding differential equations, using this identification of state variables, are

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
&\vdots \\
\dot{x}_{n-1} &= x_n \\
\dot{x}_n &= -\alpha_n x_1 - \alpha_{n-1} x_2 - \cdots - \alpha_1 x_n + u
\end{aligned} \tag{5.53a}$$

The output equation is found by careful examination of the block diagram of Fig. 5.10. Note that there are two paths from the output of each integrator to the system output—one path upward through the box labeled β_i, and a second path down through the box labeled α_i and hence, through the box labeled β_0. As a consequence,

$$y = (\beta_n - \alpha_n\beta_0)\, x_1 + (\beta_{n-1} - \alpha_{n-1}\beta_0)\, x_2 + \cdots + (\beta_1 - \alpha_1\beta_0)\, x_n + \beta_0 u \tag{5.53b}$$

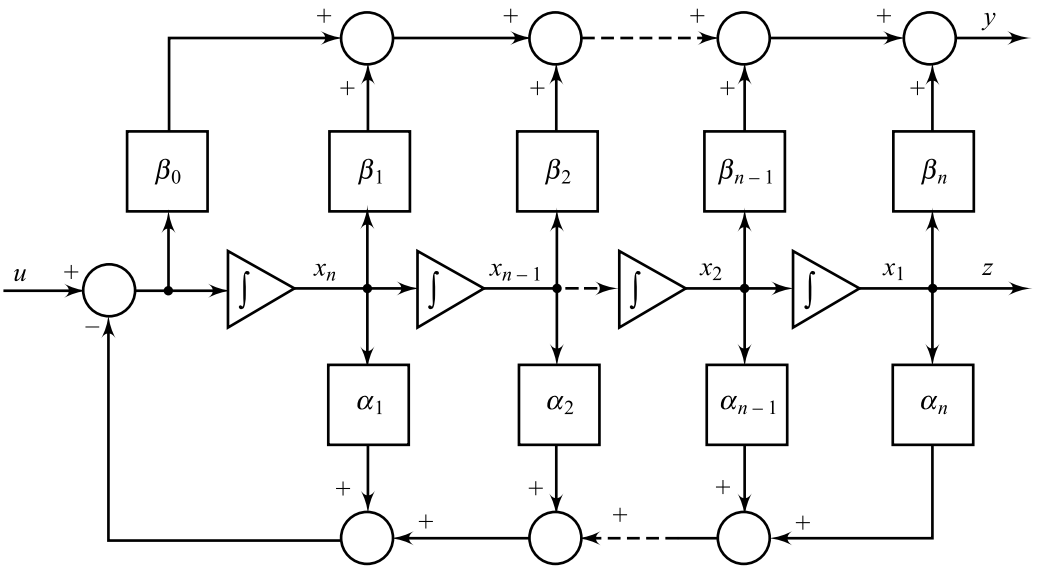


Fig. 5.10 Realization of the system (5.49)

The state and output equations (5.53), organized in vector-matrix form, are given below.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t) + du(t) \end{aligned} \tag{5.54}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_n & -\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_1 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{c} = [\beta_n - \alpha_n\beta_0, \quad \beta_{n-1} - \alpha_{n-1}\,\beta_0, \ldots, \beta_1 - \alpha_1\beta_0]; \; d = \beta_0$$

If the direct path through β_0 is absent (refer to Fig. 5.10), then the scalar d is zero and the row matrix $\mathbf{c}$ contains only the β_i coefficients.

The matrix $\mathbf{A}$ in Eqns (5.54) has a very special structure: the coefficients of the denominator of the transfer function preceded by minus signs, form a string along the bottom row of the matrix. The rest of the matrix is zero except for the 'superdiagonal' terms which are all unity. In matrix theory, a matrix

with this structure is said to be in *companion form*. For this reason, we identify the realization (5.54) as *companion-form realization* of the transfer function (5.49). We call this the *first companion form*; another companion form, second companion from, is discussed in the following section.

5.5.2 Second Companion Form

In the first companion form, the coefficients of the denominator of the transfer function appear in one of the rows of the **A** matrix. There is another companion form in which the coefficients appear in a column of the **A** matrix. This can be obtained by writing Eqn. (5.49) as

$$(s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n)\, Y(s) = (\beta_0 s^n + \beta_1 s^{n-1} + \cdots + \beta_n)\, U(s)$$

or
$$s^n\,[Y(s) - \beta_0 U(s)] + s^{n-1}\,[\alpha_1 Y(s) - \beta_1 U(s)] + \cdots + [\alpha_n Y(s) - \beta_n U(s)] = 0$$

On dividing by s^n and solving for $Y(s)$, we obtain

$$Y(s) = \beta_0 U(s) + \frac{1}{s}\,[\beta_1 U(s) - \alpha_1 Y(s)] + \cdots + \frac{1}{s^n}\,[\beta_n U(s) - \alpha_n Y(s)] \tag{5.55}$$

Note that $1/s^n$ is the transfer function of a chain of n integrators. Realization of $\frac{1}{s^n}\,[\beta_n U(s) - \alpha_n Y(s)]$ requires a chain of n integrators with input $[\beta_n u - \alpha_n y]$ to the first integrator in the chain from left-to-right. Realization of $\frac{1}{s^{n-1}}\,[\beta_{n-1} U(s) - \alpha_{n-1} Y(s)]$, requires a chain of $(n–1)$ integrators, with input $[\beta_{n-1} u - \alpha_{n-1} y]$ to the second integrator in the chain, from left-to-right, and so forth. This immediately leads to the structure shown in Fig. 5.11. The signal y is fed back to each of the integrators in the chain, and the signal u is fed forward. Thus the signal $[\beta_n u - \alpha_n y]$ passes through n integrators; the signal $[\beta_{n-1} u - \alpha_{n-1} y]$ passes through $(n - 1)$ integrators, and so forth—to complete the realization of Eqn. (5.55). The structure retains the ladder-like shape of the first companion form, but the feedback paths are in different directions.

We can now write differential equations for the realization given by Fig. 5.11. To get one state variable model, we identify the output of each integrator in Fig. 5.11 with a state variable starting at the left and proceeding to the right. The corresponding differential equations are

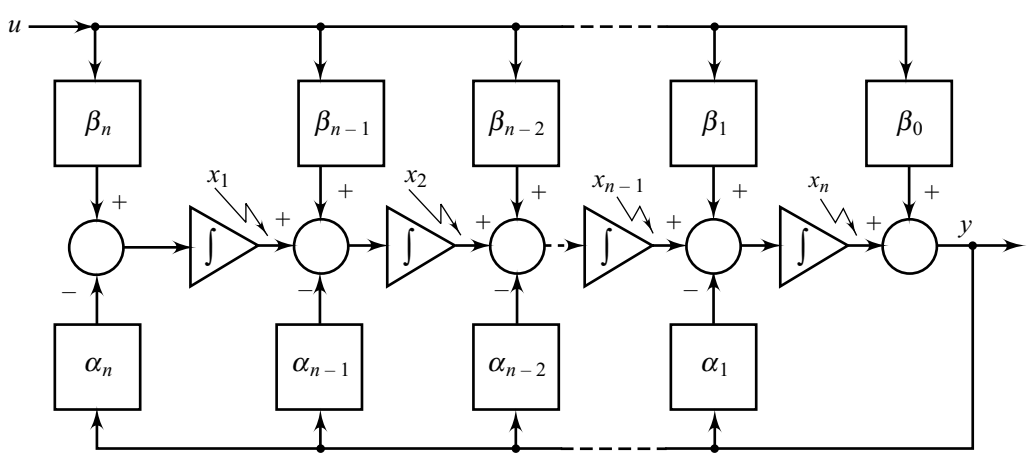


Fig. 5.11 Realization of Eqn. (5.55)

$$\begin{aligned}
\dot{x}_n &= x_{n-1} - \alpha_1 (x_n + \beta_0 u) + \beta_1 u \\
\dot{x}_{n-1} &= x_{n-2} - \alpha_2 (x_n + \beta_0 u) + \beta_2 u \\
&\vdots \\
\dot{x}_2 &= x_1 - \alpha_{n-1} (x_n + \beta_0 u) + \beta_{n-1} u \\
\dot{x}_1 &= -\alpha_n (x_n + \beta_0 u) + \beta_n u
\end{aligned}$$

and the output equation is

$$y = x_n + \beta_0 u$$

The state and output equations, organized in vector-matrix form, are given below.

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\
y(t) &= \mathbf{c}\mathbf{x}(t) + du(t)
\end{aligned} \tag{5.56}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -\alpha_n \\ 1 & 0 & \cdots & 0 & -\alpha_{n-1} \\ 0 & 1 & \cdots & 0 & -\alpha_{n-2} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_1 \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} \beta_n - \alpha_n \beta_0 \\ \beta_{n-1} - \alpha_{n-1}\beta_0 \\ \vdots \\ \beta_1 - \alpha_1 \beta_0 \end{bmatrix}$$

$$\mathbf{c} = [0\ 0\ \cdots\ 0\ 1];\ d = \beta_0$$

Compare **A**, **b**, and **c** matrices of the second companion form with that of the first. We observe that **A**, **b**, and **c** matrices of one companion form correspond to the transpose of the **A**, **c**, and **b** matrices, respectively, of the other.

There are many benefits derived from the companion forms of state variable models. One obvious benefit is that both the companion forms lend themselves easily to simple analog computer models. Both the companion forms also play an important role in pole-placement design through state feedback. This will be discussed in Chapter 7.

5.5.3 Jordan Canonical Form

In the two canonical forms (5.54) and (5.56), the coefficients of the denominator of the transfer function appear in one of the rows or columns of matrix **A**. In another of the canonical forms, the poles of the transfer function form a string along the main diagonal of the matrix. This canonical form follows directly from the partial fraction expansion of the transfer function.

The general transfer function under consideration is (refer to Eqn. (5.49))

$$G(s) = \frac{\beta_0 s^n + \beta_1 s^{n-1} + \cdots + \beta_n}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n}$$

By long division, $G(s)$ can be written as

$$G(s) = \beta_0 + \frac{\beta_1' s^{n-1} + \beta_2' s^{n-2} + \cdots + \beta_n'}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n} = \beta_0 + G'(s)$$

The results are simplest when the poles of the transfer function are all distinct. The partial fraction expansion of the transfer function, then has the form

$$G(s) = \frac{Y(s)}{U(s)} = \beta_0 + \frac{r_1}{s - \lambda_1} + \frac{r_2}{s - \lambda_2} + \cdots + \frac{r_n}{s - \lambda_n} \tag{5.57}$$

The coefficients $r_i (i = 1, 2, \ldots, n)$ are the residues of the transfer function $G'(s)$ at the corresponding poles at $s = \lambda_i$ $(i = 1, 2, \ldots, n)$. In the form of Eqn. (5.57), the transfer function consists of a direct path with gain β_0, and n first-order transfer functions in parallel. A block diagram representation of Eqn. (5.57) is shown in Fig. 5.12. The gains, corresponding to the residues, have been placed at the outputs of the integrators. This is quite arbitrary. They could have been located on the input side, or indeed, split between the input and the output.

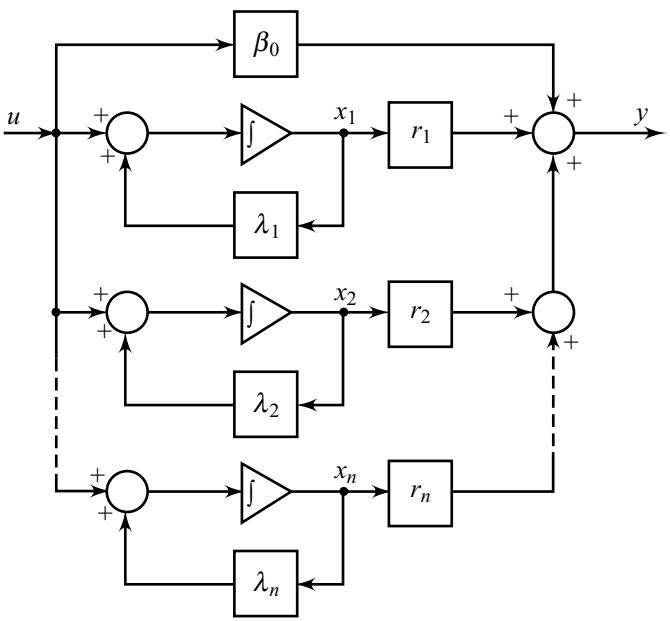


Fig. 5.12 Realization of $G(s)$ in Eqn. (5.57)

Identifying the outputs of the integrators with the state variables results in the following state and output equations:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{\Lambda x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{cx}(t) + du(t) \end{aligned} \tag{5.58}$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}; \mathbf{c} = [r_1 \quad r_2 \quad \cdots \quad r_n]; d = \beta_0$$

It is observed that for this canonical state variable model, the matrix $\mathbf{\Lambda}$ is a diagonal matrix with the poles of $G(s)$ as its diagonal elements. The unique *decoupled* nature of the canonical model is obvious from Eqns (5.58); the n first-order differential equations are independent of each other:

$$\dot{x}_i(t) = \lambda_i x_i(t) + u(t);\ i = 1, 2, \ldots, n \tag{5.59}$$

This decoupling feature, as we shall see later in this chapter, greatly helps in system analysis.

The block diagram representation of Fig. 5.12 can be turned into hardware *only* if all the poles at $s = \lambda_1, \lambda_2, ..., \lambda_n$ are real. If they are complex, the feedback gains and the gains corresponding to the residues, are complex. In this case, the representation must be considered as being purely conceptual; valid for theoretical studies, but not physically realizable. A realizable representation can be obtained by introducing an equivalence transformation.

Suppose that $s = \sigma + j\omega$, $s = \sigma - j\omega$ and $s = \lambda$ are the three poles of a transfer function. The residues at the pair of complex conjugate poles must be themselves complex conjugates. Partial fraction expansion of the transfer function, with a pair of complex conjugate poles and a real pole, has the form

$$G(s) = d + \frac{p + jq}{s - (\sigma + j\omega)} + \frac{p - jq}{s - (\sigma - j\omega)} + \frac{r}{s - \lambda}$$

A state variable model for this transfer function is given below (refer to Eqns (5.58)):

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{\Lambda x} + \mathbf{b}u \\ y &= \mathbf{cx} + du \end{aligned} \tag{5.60}$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \sigma + j\omega & 0 & 0 \\ 0 & \sigma - j\omega & 0 \\ 0 & 0 & \lambda \end{bmatrix};\ \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix};\ \mathbf{c} = [p + jq \quad p - jq \quad r]$$

Introducing an equivalence transformation

$$\mathbf{x} = \mathbf{P}\,\bar{\mathbf{x}}$$

with

$$\mathbf{P} = \begin{bmatrix} 1/2 & -j1/2 & 0 \\ 1/2 & j1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

we obtain (refer to Eqns (5.22))

$$\begin{aligned} \dot{\bar{\mathbf{x}}}(t) &= \bar{\mathbf{A}}\,\bar{\mathbf{x}}(t) + \bar{\mathbf{b}}\,u(t) \\ y(t) &= \bar{\mathbf{c}}\,\bar{\mathbf{x}}(t) + du(t) \end{aligned} \tag{5.61}$$

where

$$\bar{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{\Lambda P} = \begin{bmatrix} 1 & 1 & 0 \\ j & -j & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma + j\omega & 0 & 0 \\ 0 & \sigma - j\omega & 0 \\ 0 & 0 & \lambda \end{bmatrix} \begin{bmatrix} 1/2 & -j1/2 & 0 \\ 1/2 & j1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sigma & \omega & 0 \\ -\omega & \sigma & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$

$$\bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix};\ \bar{\mathbf{c}} = \mathbf{cP} = [p \quad q \quad r]$$

When the transfer function $G(s)$ has repeated poles, the partial fraction expansion will not be as simple as Eqn. (5.57). Assume that $G(s)$ has m distinct poles at $s = \lambda_1, \lambda_2, \ldots, \lambda_m$ of multiplicity $n_1, n_2, \ldots, n_m$, respectively; $n = n_1 + n_2 + \cdots + n_m$. That is, $G(s)$ is of the form

$$G(s) = \beta_0 + \frac{\beta_1' s^{n-1} + \beta_2' s^{n-2} + \cdots + \beta_n'}{(s-\lambda_1)^{n_1}(s-\lambda_2)^{n_2}\cdots(s-\lambda_m)^{n_m}} \tag{5.62}$$

The partial fraction expansion of $G(s)$ is of the form.

$$G(s) = \beta_0 + H_1(s) + \cdots + H_m(s) = \frac{Y(s)}{U(s)} \tag{5.63}$$

where

$$H_i(s) = \frac{r_{i1}}{(s-\lambda_i)^{n_i}} + \frac{r_{i2}}{(s-\lambda_i)^{n_i-1}} + \cdots + \frac{r_{in_i}}{(s-\lambda_i)} = \frac{Y_i(s)}{U(s)}$$

The first term in $H_i(s)$ can be synthesized as a chain of n_i identical, first-order systems, each having transfer function $1/(s-\lambda_i)$. The second term can be synthesized by a chain of $(n_i - 1)$ first-order systems, and so forth. The entire $H_i(s)$ can be synthesized by the system having the block diagram shown in Fig. 5.13.

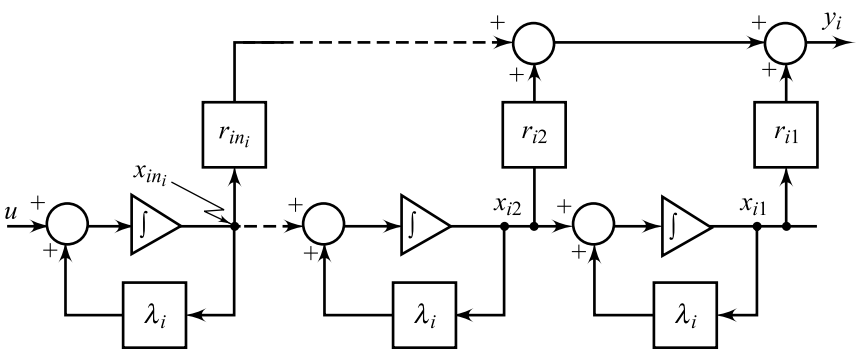


Fig. 5.13 Realization of $H_i(s)$ in Eqn. (5.63)

We can now write differential equations for the realization of $H_i(s)$, given by Fig. 5.13. To get one state variable formulation, we identify the output of each integrator with a state variable—starting at the right and proceeding to the left. The corresponding differential equations are

$$\begin{aligned} \dot{x}_{i1} &= \lambda_i x_{i1} + x_{i2} \\ \dot{x}_{i2} &= \lambda_i x_{i2} + x_{i3} \\ &\vdots \\ \dot{x}_{in_i} &= \lambda_i\ x_{in_i} + u \end{aligned} \tag{5.64a}$$

and the output is given by

$$y_i = r_{i1} x_{i1} + r_{i2} x_{i2} + \cdots + r_{in_i} x_{in_i} \tag{5.64b}$$

If the state vector for the subsystem is defined by

$$\mathbf{x}_i = [x_{i1} \quad x_{i2} \cdots x_{in_i}]^T$$

then Eqns (5.64) can be written in the standard form

$$\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{\Lambda}_i \mathbf{x}_i + \mathbf{b}_i u \\ y_i &= \mathbf{c}_i \mathbf{x}_i \end{aligned} \tag{5.65}$$

where

$$\mathbf{\Lambda}_i = \begin{bmatrix} \lambda_i & 1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_i & 1 \\ 0 & 0 & 0 & \cdots & 0 & \lambda_i \end{bmatrix}; \mathbf{b}_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}; \mathbf{c}_i = [r_{i1} \quad r_{i2} \quad \cdots \quad r_{in_i}]$$

Note that matrix $\mathbf{\Lambda}_i$ has two diagonals—the principal diagonal has the corresponding characteristic root (pole), and the superdiagonal has all 1s. In matrix theory, a matrix having this structure is said to be in *Jordan form*. For this reason, we identify the realization (5.65) as *Jordan canonical form*.

According to Eqn. (5.63), the overall transfer function $G(s)$ consists of a direct path with gain β_0 and m subsystems, each of which is in the Jordan canonical form, as shown in Fig. 5.14. The state vector of the overall system consists of the concatenation of the state vectors of each of the *Jordan blocks*:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{bmatrix} \tag{5.66a}$$

Since there is no coupling between any of the subsystems, the $\mathbf{\Lambda}$ matrix of the overall system is 'block diagonal':

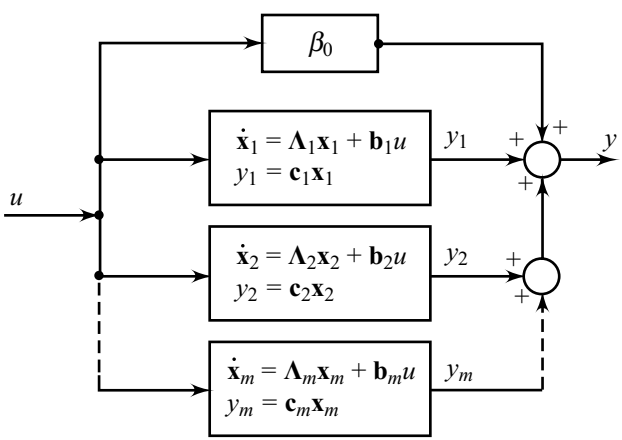


Fig. 5.14 Subsystems of Jordan canonical form combined into overall system

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{\Lambda}_m \end{bmatrix} \tag{5.66b}$$

where each of the submatrices $\mathbf{\Lambda}_i$ is in the Jordan canonical form (5.65). The **b** and **c** matrices of the overall system are the concatenations of the $\mathbf{b}_i$ and $\mathbf{c}_i$ matrices, respectively, of each of the subsystems:

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_m \end{bmatrix}; \mathbf{c} = [\mathbf{c_1} \quad \mathbf{c_2} \quad \ldots \mathbf{c}_{\boldsymbol{m}}]; d = \beta_0 \tag{5.66c}$$

The state variable model (5.58) derived for the case of distinct poles, is a special case of Jordan canonical form (5.66) where each Jordan block is of 1×1 dimension.

Example 5.6

In the following, we obtain three different realizations for the transfer function

$$G(s) = \frac{s+3}{s^3 + 9s^2 + 24s + 20} = \frac{Y(s)}{U(s)}$$

First Companion Form Note that the given $G(s)$ is a strictly proper fraction; the realization will, therefore, be of the form (5.48), i.e., the parameter d in the realization $\{\mathbf{A}, \mathbf{b}, \mathbf{c}, d\}$ is zero.

The state variable formulation in the first companion form, can be written just by inspection of the given transfer function. Referring to Eqns (5.54), we obtain

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -20 & -24 & -9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$y = [3 \quad 1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Figure 5.15a shows the state diagram in signal flow graph form.

Second Companion Form Referring to Eqns (5.56), we obtain

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -20 \\ 1 & 0 & -24 \\ 0 & 1 & -9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} u$$

$$y = x_3$$

Figure 5.15b shows the state diagram.

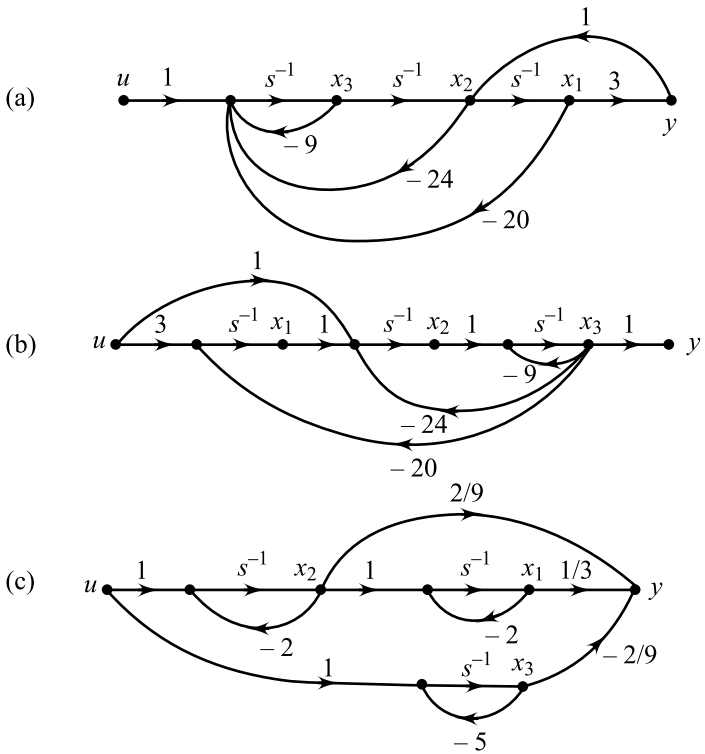


Fig. 5.15 Three realizations of given $G(s)$

Jordan Canonical Form The given transfer function $G(s)$ in the factored form:

$$G(s) = \frac{s+3}{(s+2)^2(s+5)}$$

Using partial fraction expansion, we obtain

$$G(s) = \frac{1/3}{(s+2)^2} + \frac{2/9}{s+2} + \frac{-2/9}{s+5}$$

A matrix of the state variable model in Jordan canonical form will be block-diagonal; consisting of two Jordan blocks (refer to Eqns (5.65)):

$$\mathbf{\Lambda}_1 = \begin{bmatrix} -2 & 1 \\ 0 & -2 \end{bmatrix}; \mathbf{\Lambda}_2 = [-5]$$

The corresponding $\mathbf{b}_i$ and $\mathbf{c}_i$ vectors are (refer to Eqns (5.65)):

$$\mathbf{b}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \mathbf{c}_1 = [\tfrac{1}{3} \quad \tfrac{2}{9}]; \mathbf{b}_2 = [1]; \mathbf{c}_2 = [-\tfrac{2}{9}]$$

The state variable model of the given $G(s)$ in Jordan canonical form is, therefore, given by (refer to Eqns (5.66))

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \left[\begin{array}{cc:c} -2 & 1 & 0 \\ 0 & -2 & 0 \\ \hdashline 0 & 0 & -5 \end{array}\right] + \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} u$$

$$y = \left[\begin{array}{ccc} \frac{1}{3} & \frac{2}{9} & \vdots \; \frac{-2}{9} \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Figure 5.15c shows the state diagram. We note that Jordan canonical state variables are not completely decoupled. The decoupling is blockwise; state variables of one block are independent of state variables of all other blocks. However, the state variables of one block, among themselves, are coupled; the coupling is unique and simple.

5.6 EIGENVALUES AND EIGENVECTORS

The last section was concerned with the derivation of state variable models for a given transfer function. Out of infinitely many realizations possible for a given transfer function, we have derived the following three 'standard' or canonical forms:

(i) First companion form
(ii) Second companion form
(iii) Jordan form

Consider now the situation where the system dynamics is already known in the form of a state variable model. For example, state equations representing the dynamics of a physical system may be obtained by the application of physical laws. However, state variables in such a formulation may not be as convenient as some other canonical state variables. Transformation of an original state variable model to a canonical form may, therefore, be helpful in solving analysis and design problems.

In this section, we deal with the problem of transformation of a given state variable model to Jordan canonical form (transformation of a given model to other canonical forms will be taken up in Section 5.9, and to companion forms in Chapter 7).

Given state variable model:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t) + du(t) \end{aligned} \tag{5.67}$$

where $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ and d are constant matrices of dimensions $n \times n$, $n \times 1$, $1 \times n$ and 1×1, respectively.

The problem is to find an equivalence transformation

$$\mathbf{x} = \mathbf{P}\,\overline{\mathbf{x}} \tag{5.68}$$

such that the equivalent model (refer to Eqns (5.22))

$$\begin{aligned} \dot{\overline{\mathbf{x}}}(t) &= \mathbf{P}^{-1}\mathbf{A}\mathbf{P}\,\overline{\mathbf{x}}(t) + \mathbf{P}^{-1}\mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{P}\,\overline{\mathbf{x}}(t) + du(t) \end{aligned} \tag{5.69}$$

is in Jordan canonical form.

5.6.1 Eigenvalues

For a general *n*th-order matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

the determinant

$$|\lambda \mathbf{I} - \mathbf{A}| = \begin{vmatrix} \lambda - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \lambda - a_{nn} \end{vmatrix}$$

On expanding the determinant we find that $|\lambda \mathbf{I} - \mathbf{A}|$, called the *characteristic polynomial* of the matrix **A,** is a polynomial of degree *n*:

$$|\lambda \mathbf{I} - \mathbf{A}| = \Delta(\lambda) = \lambda^n + \alpha_1 \lambda^{n-1} + \cdots + \alpha_{n-1}\lambda + \alpha_n$$

where α_i are constant scalars.

The equation

$$\Delta(\lambda) = \lambda^n + \alpha_1 \lambda^{n-1} + \cdots + \alpha_{n-1}\lambda + \alpha_n = 0 \tag{5.70}$$

is called the *characteristic equation* of the matrix **A**, and its *n* roots are called *characteristic roots*, or *characteristic values*, or *eigenvalues* of the matrix **A**. When **A** represents the dynamic matrix of a linear system, the eigenvalues determine the dynamic response of the system (the next section will establish this fact), and also turn out to be the poles of the corresponding transfer function (refer to Eqn. (5.31)).

Eigenvalues of a matrix **A** are invariant under equivalence transformation (refer to Eqn. (5.37)), i.e.,

$$|\lambda \mathbf{I} - \mathbf{A}| = |\lambda \mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P}|$$

for any nonsingular matrix **P**.

5.6.2 Eigenvectors

Consider an $n \times n$ matrix **A** with eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. We start with the assumption of distinct eigenvalues; later we will relax this assumption.

Case I: All Eigenvalues are Distinct State transformation to Jordan canonical form requires a transformation matrix **P** such that

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{5.71}$$

Let the transformation matrix **P** required to transform **A** to **Λ**, be of the form

$$\mathbf{P} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n]; \tag{5.72a}$$

$$\mathbf{v}_i = \begin{bmatrix} v_{1i} \\ v_{2i} \\ \vdots \\ v_{ni} \end{bmatrix} = i\text{th column of } \mathbf{P} \tag{5.72b}$$

Equation (5.71) shows that

$$\mathbf{AP} = \mathbf{P\Lambda}$$

or

$$\mathbf{A}[\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n] = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

By equating the ith columns, we obtain

$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

or

$$(\lambda_i \mathbf{I} - \mathbf{A})\mathbf{v}_i = \mathbf{0} \tag{5.73}$$

This is a set of n homogeneous equations in n unknowns $v_{1i}, v_{2i}, \ldots, v_{ni}$.

There are two questions of interest with regard to Eqn. (5.73):

(i) whether a solution to Eqn. (5.73) exists; and
(ii) if the answer to the first question is yes, how many linearly independent solutions occur?

We consider an example to answer these questions. Refer to Section 5.2 for the basic definitions from linear algebra used in the sequel.

Example 5.7

The matrix

$$\mathbf{A} = \begin{bmatrix} -4 & 1 & 0 \\ 0 & -3 & 1 \\ 0 & 0 & -2 \end{bmatrix}$$

has the characteristic equation

$$|\lambda \mathbf{I} - \mathbf{A}| = \begin{vmatrix} \lambda + 4 & -1 & 0 \\ 0 & \lambda + 3 & -1 \\ 0 & 0 & \lambda + 2 \end{vmatrix}$$

$$= (\lambda + 4)(\lambda + 3)(\lambda + 2) = 0$$

Therefore, the eigenvalues of **A** are $\lambda_1 = -2$, $\lambda_2 = -3$ and $\lambda_3 = -4$.

Consider a set of homogeneous equations

$$(\lambda_1 \mathbf{I} - \mathbf{A})\mathbf{v}_1 = \mathbf{0}$$

or
$$\begin{bmatrix} 2 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{21} \\ v_{31} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.74}$$

It is easy to check that rank of the matrix $(\lambda_1\mathbf{I} - \mathbf{A})$ is two, i.e.,
$$\rho(\lambda_1\mathbf{I} - \mathbf{A}) = 2$$

A highest-order array having a nonvanishing determinant, is
$$\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix},$$

which is obtained from $(\lambda_1\mathbf{I} - \mathbf{A})$ by omitting the third row and the third column. Consequently, a set of linearly independent equations is
$$2v_{11} - v_{21} = 0$$
$$v_{21} = v_{31}$$

or
$$\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ v_{31} \end{bmatrix}$$

Therefore,
$$\begin{bmatrix} v_{11} \\ v_{21} \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ v_{31} \end{bmatrix} = \begin{bmatrix} v_{31}/2 \\ v_{31} \end{bmatrix}$$

There are three components in $\mathbf{v}_1$ and two equations governing them; therefore, one of the three components can be arbitrarily chosen. For $v_{31} = 2$, a solution to Eqn. (5.74) is
$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

A different choice for v_{31} leads to a different solution to Eqn. (5.74). In fact, this set of equations has infinite solutions as demonstrated below.

For $v_{31} = 2\alpha$ (with α arbitrary), the solution
$$\mathbf{v}_1 = \alpha \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

Obviously, this solution is non-unique. However, all nontrivial solutions have a unique direction, and they differ only in terms of a scalar multiplier. There is, thus, only one independent solution.

Corresponding to the eigenvalue $\lambda_2 = -3$, a linearly independent solution to homogeneous equations
$$(\lambda_2\mathbf{I} - \mathbf{A})\mathbf{v}_2 = \mathbf{0}$$

is given by
$$\mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

And for $\lambda_3 = -4$, the equations

$$(\lambda_3\mathbf{I} - \mathbf{A})\mathbf{v}_3 = \mathbf{0}$$

have a linearly independent solution

$$\mathbf{v}_3 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

In general, the number of equations that the vector $\mathbf{v}_i$ in (5.73) has to obey, is equal to $\rho(\lambda_i\mathbf{I} - \mathbf{A})$ where $\rho(\mathbf{M})$ denotes the rank of matrix $\mathbf{M}$. There are n components in $\mathbf{v}_i$ (n = number of columns of $(\lambda_i\mathbf{I} - \mathbf{A})$); therefore, $(n - \rho(\lambda_i\mathbf{I} - \mathbf{A}))$ components of $\mathbf{v}_i$ can be arbitrarily chosen. Thus, the number of linearly independent solutions of the homogeneous equation (5.73) = $[n - \rho(\lambda_i\mathbf{I} - \mathbf{A})] = \gamma(\lambda_i\mathbf{I} - \mathbf{A})$, where $\gamma(\mathbf{M})$ denotes the *nullity* of matrix $\mathbf{M}$.

We have the following answers to the two questions raised earlier with regard to Eqn. (5.73):

(i) For Eqn. (5.73) to have a nontrivial solution, rank of $(\lambda_i\mathbf{I} - \mathbf{A})$ must be less than n, or, equivalently, *det* $(\lambda_i\mathbf{I} - \mathbf{A}) = 0$. This condition is satisfied by virtue of the fact that λ_i is an eigenvalue.

(ii) The number of linearly independent solutions to Eqn. (5.73), is equal to nullity of $(\lambda_i\mathbf{I} - \mathbf{A})$.

The nullity of matrix $(\lambda_i\mathbf{I} - \mathbf{A})$ does not exceed the multiplicity of the eigenvalue λ_i (refer to Lancaster and Tismenetsky [28] for proof of the result). Therefore, for distinct eigenvalue λ_i, there is one, and only one, linearly independent solution to Eqn. (5.73). This solution is called the *eigenvector* of $\mathbf{A}$ associated with the eigenvalue λ_i.

Theorem 5.1 Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ be the eigenvectors associated with the distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, respectively, of matrix $\mathbf{A}$. The vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are linearly independent and the nonsingular matrix

$$\mathbf{P} = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_n]$$

transforms matrix $\mathbf{A}$ into Jordan canonical form.

Proof Let $\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_n\mathbf{v}_n = \mathbf{0}$ (5.75)

If it can be shown that this implies that $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$, then the set $\{\mathbf{v}_i\}$ is linearly independent. Define

$$\mathbf{T}_i = \lambda_i\mathbf{I} - \mathbf{A}$$

Note that $\mathbf{T}_i\mathbf{v}_i = \mathbf{0}$

and $\mathbf{T}_i\mathbf{v}_j = (\lambda_i - \lambda_j)\mathbf{v}_j$ if $i \neq j$

Multiplying Eqn. (5.75) by $\mathbf{T}_1$ gives

$$\alpha_2(\lambda_1 - \lambda_2)\mathbf{v}_2 + \alpha_3(\lambda_1 - \lambda_3)\,\mathbf{v}_3 + \cdots + \alpha_n(\lambda_1 - \lambda_n)\mathbf{v}_n = \mathbf{0}$$

Multiplying this in turn by $\mathbf{T}_2, \mathbf{T}_3, \ldots, \mathbf{T}_{n-1}$ gives

$$\alpha_3 (\lambda_1 - \lambda_3)(\lambda_2 - \lambda_3)\mathbf{v}_3 + \cdots + \alpha_n(\lambda_1 - \lambda_n)(\lambda_2 - \lambda_n)\mathbf{v}_n = \mathbf{0}$$

$$\vdots$$

$$\alpha_{n-1} (\lambda_1 - \lambda_{n-1})(\lambda_2 - \lambda_{n-1}) \cdots (\lambda_{n-2} - \lambda_{n-1})\mathbf{v}_{n-1}$$

$$+ \alpha_n (\lambda_1 - \lambda_n)(\lambda_2 - \lambda_n) \cdots (\lambda_{n-2} - \lambda_n)\mathbf{v}_n = \mathbf{0} \tag{5.76}$$

$$\alpha_n(\lambda_1 - \lambda_n)(\lambda_2 - \lambda_n) \cdots (\lambda_{n-2} - \lambda_n)(\lambda_{n-1} - \lambda_n)\mathbf{v}_n = \mathbf{0} \tag{5.77}$$

Since $\mathbf{v}_n \neq \mathbf{0}$ and $\lambda_n \neq \lambda_i$ for $i \neq n$, Eqn. (5.77) requires that $\alpha_n = 0$. This, plus Eqn. (5.76), requires that $\alpha_{n-1} = 0$.

Continuing this reasoning shows that Eqn. (5.75) requires $\alpha_i = 0$ for $i = 1, 2, \ldots, n$; so the eigenvectors $\mathbf{v}_i$ are linearly independent.

The matrix $\mathbf{P}$, constructed by placing the eigenvectors (columns) together, is therefore a nonsingular matrix.

As per Eqns (5.71)–(5.73), $\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{\Lambda}$.

Example 5.8

Consider the matrix

$$\mathbf{A} = \begin{bmatrix} -4 & 1 & 0 \\ 0 & -3 & 1 \\ 0 & 0 & -2 \end{bmatrix}$$

for which we found, in Example 5.7, the eigenvalues and eigenvectors to be

$$\lambda_1 = -2, \mathbf{v}_1 = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}; \lambda_2 = -3, \mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}; \lambda_3 = -4, \mathbf{v}_3 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

The transformation matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

This gives

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \frac{1}{4}\begin{bmatrix} 0 & 0 & 2 \\ 0 & 4 & -4 \\ 2 & -2 & 1 \end{bmatrix}\begin{bmatrix} -4 & 1 & 0 \\ 0 & -3 & 1 \\ 0 & 0 & -2 \end{bmatrix}\begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & -4 \end{bmatrix} = \mathbf{\Lambda}$$

which is the diagonal matrix (a special case of Jordan canonical form) with eigenvalues of $\mathbf{A}$ as its diagonal elements. In fact, $\mathbf{\Lambda}$ could be written down directly without computing $\mathbf{P}^{-1}\mathbf{A}\mathbf{P}$.

Computation of Eigenvectors The eigenvectors $\mathbf{v}_i$ which satisfy the equations

$$(\lambda_i\mathbf{I} - \mathbf{A})\mathbf{v}_i = \mathbf{0} \tag{5.78}$$

can be computed by solving the set of linear algebraic equations. The method of Gauss elimination is a straightforward and powerful procedure for reducing systems of linear equations to a simple *reduced form*, easily solved by substitution (refer to Noble and Daniel [27]). High quality software is available commercially; for example, the MATLAB system from the Math Works [152].

In the following, we give an analytical procedure of computing the eigenvectors. This procedure is quite useful for hand calculations.

Using the property (refer to Eqn. (5.3))

$$\mathbf{M}\ adj\ \mathbf{M} = |\mathbf{M}|\mathbf{I}$$

and letting $\mathbf{M} = (\lambda_i\mathbf{I} - \mathbf{A})$ yields

$$(\lambda_i\mathbf{I} - \mathbf{A})\ adj\,(\lambda_i\mathbf{I} - \mathbf{A}) = |\lambda_i\mathbf{I} - \mathbf{A}|\,\mathbf{I}$$

Since $|\lambda_i\mathbf{I} - \mathbf{A}|$ is the characteristic polynomial and λ_i is an eigenvalue, this equation becomes

$$(\lambda_i\mathbf{I} - \mathbf{A})\ adj\,(\lambda_i\mathbf{I} - \mathbf{A}) = \mathbf{0} \tag{5.79}$$

A comparison of Eqn. (5.78) with (5.79) shows that $\mathbf{v}_i$ is proportional to any nonzero column of $adj\,(\lambda_i\mathbf{I} - \mathbf{A})$.

Example 5.9

Consider the state variable model

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

$$y = \mathbf{c}\mathbf{x}$$

with $\mathbf{A} = \begin{bmatrix} -9 & 1 & 0 \\ -26 & 0 & 1 \\ -24 & 0 & 0 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 2 \\ 5 \\ 0 \end{bmatrix}$; $\mathbf{c} = [1 \quad 2 \quad -1]$

The characteristic equation

$$|\lambda\mathbf{I} - \mathbf{A}| = 0$$

yields the roots $\lambda_1 = -2$, $\lambda_2 = -3$, and $\lambda_3 = -4$.

$$adj(\lambda\mathbf{I} - \mathbf{A}) = adj \begin{bmatrix} \lambda+9 & -1 & 0 \\ 26 & \lambda & -1 \\ 24 & 0 & \lambda \end{bmatrix} = \begin{bmatrix} \lambda^2 & \lambda & 1 \\ -26\lambda-24 & \lambda^2+9\lambda & \lambda+9 \\ -24\lambda & -24 & \lambda^2+9\lambda+26 \end{bmatrix}$$

For $\lambda_1 = -2$, $adj(\lambda_1\mathbf{I} - \mathbf{A}) = \begin{bmatrix} 4 & -2 & 1 \\ 28 & -14 & 7 \\ 48 & -24 & 12 \end{bmatrix}$; $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 7 \\ 12 \end{bmatrix}$

For $\lambda_2 = -3$, $adj(\lambda_2\mathbf{I} - \mathbf{A}) = \begin{bmatrix} 9 & -3 & 1 \\ 54 & -18 & 6 \\ 72 & -24 & 8 \end{bmatrix}$; $\mathbf{v}_2 = \begin{bmatrix} 1 \\ 6 \\ 8 \end{bmatrix}$

For $\lambda_3 = -4$, $adj(\lambda_3\mathbf{I} - \mathbf{A}) = \begin{bmatrix} 16 & -4 & 1 \\ 80 & -20 & 5 \\ 96 & -24 & 6 \end{bmatrix}$; $\mathbf{v}_3 = \begin{bmatrix} 1 \\ 5 \\ 6 \end{bmatrix}$

In each case, the columns of $adj(\lambda_i\mathbf{I} - \mathbf{A})$ are linearly related. In practice, it is necessary to calculate only one (nonzero) column of the adjoint matrix.

The transformation matrix

$$\mathbf{P} = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3] = \begin{bmatrix} 1 & 1 & 1 \\ 7 & 6 & 5 \\ 12 & 8 & 6 \end{bmatrix}$$

State transformation

$$\mathbf{x} = \mathbf{P}\,\bar{\mathbf{x}}$$

results in the following model (refer to Eqns (5.22)):

$$\dot{\bar{\mathbf{x}}} = \mathbf{\Lambda}\,\bar{\mathbf{x}} + \bar{\mathbf{b}}u$$
$$y = \bar{\mathbf{c}}\,\bar{\mathbf{x}}$$

with

$$\mathbf{\Lambda} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P} = -\frac{1}{2}\begin{bmatrix} -4 & 2 & -1 \\ 18 & -6 & 2 \\ -16 & 4 & -1 \end{bmatrix}\begin{bmatrix} -9 & 1 & 0 \\ -26 & 0 & 1 \\ -24 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \\ 7 & 6 & 5 \\ 12 & 8 & 6 \end{bmatrix} = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & -4 \end{bmatrix}$$

$$\bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b} = \begin{bmatrix} -1 \\ -3 \\ 6 \end{bmatrix};\ \bar{\mathbf{c}} = \mathbf{c}\mathbf{P} = [3\ \ 5\ \ 5]$$

Case II: ***Some Eigenvalues are Multiple Roots of the Characteristic Equation*** For notational convenience, we assume that matrix $\mathbf{A}$ has an eigenvalue λ_1 of multiplicity n_1, and all other eigenvalues λ_{n_1+1}, …, λ_n are distinct, i.e.,

$$|\lambda\mathbf{I} - \mathbf{A}| = (\lambda - \lambda_1)^{n_1}(\lambda - \lambda_{n_1+1})\ \cdots\ (\lambda - \lambda_n)$$

Recall the result stated earlier: the nullity γ of matrix $(\lambda_i\mathbf{I} - \mathbf{A})$ does not exceed the multiplicity of λ_i. Therefore,

$$1 \le \gamma(\lambda_1\mathbf{I} - \mathbf{A}) \le n_1$$
$$\gamma(\lambda_{n_1+1}\mathbf{I} - \mathbf{A}) = 1$$
$$\vdots$$
$$\gamma(\lambda_n\mathbf{I} - \mathbf{A}) = 1$$

We know that the number of linearly independent eigenvectors associated with an eigenvalue λ_i is equal to the nullity γ of the matrix $(\lambda_i\mathbf{I} - \mathbf{A})$. Thus, when one or more eigenvalues is a repeated root of the characteristic equation, a full set of n linearly independent eigenvectors may, or may not, exist.

It is convenient to consider three subclassifications for Case II.

Case II$_1$: ***Nullity of*** $(\lambda_1\mathbf{I} - \mathbf{A}) = n_1$ In this case, the vector equation

$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v} = \mathbf{0}$$

has n_1 linearly independent solutions, say, $\mathbf{v}_1$, $\mathbf{v}_2$, …, $\mathbf{v}_{n1}$. We have thus, a full set of n_1 eigenvectors associated with multiple eigenvalue λ_1.

The remaining $(n - n_1)$ eigenvectors are obtained from the vector equations

$$(\lambda_j\mathbf{I} - \mathbf{A})\mathbf{v}_j = \mathbf{0}, j = n_1 + 1, \ldots, n$$

Each of these vector equations has only *one* linearly independent solution.

The matrix

$$\mathbf{P} = [\mathbf{v}_1 \quad \mathbf{v}_2 \cdots \mathbf{v}_{n1} \quad \mathbf{v}_{n_1+1} \cdots \mathbf{v}_n]$$

gives

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \lambda_1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \lambda_{n_1+1} & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

Case II$_2$: ***Nullity of*** $(\lambda_1\mathbf{I} - \mathbf{A}) = 1$ For this case, there is only one eigenvector associated with λ_1, regardless of multiplicity n_1. This eigenvector is given by the linearly independent solution of the vector equation

$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v} = \mathbf{0}$$

The solution to this equation may be found as in Case I.

We have seen in Cases I and II$_1$, that the transformation matrix $\mathbf{P}$ yields a diagonal matrix $\mathbf{\Lambda}$ if, and only if, $\mathbf{P}$ has a set of n linearly independent eigenvectors. When nullity of the matrix $(\lambda_1\mathbf{I} - \mathbf{A})$ is one, n linearly independent eigenvectors cannot be constructed and, therefore, the transformation to a diagonal matrix is not possible.

The simplest form to which matrix $\mathbf{A}$, having a multiple eigenvalue λ_1 of multiplicity n_1 with $\gamma(\lambda_1\mathbf{I} - \mathbf{A}) = 1$ and all other distinct eigenvalues, can be reduced is the Jordan canonical form:

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_{n_1+1} & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{\Lambda}_n \end{bmatrix}$$

where the Jordan blocks $\mathbf{\Lambda}_i$ are

$$\mathbf{\Lambda}_1 = \begin{bmatrix} \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_1 \end{bmatrix}$$

$$\mathbf{\Lambda}_{n_1+1} = [\lambda_{n_1+1}]; \cdots; \mathbf{\Lambda}_n = [\lambda_n]$$

The transformation matrix **P** is given by

$$\mathbf{P} = [\mathbf{v}_1 \quad \mathbf{v}_2 \cdots \mathbf{v}_{n_1} \quad \mathbf{v}_{n_1+1} \cdots \mathbf{v}_n]$$

with $\mathbf{v}_1\ \mathbf{v}_2 \cdots, \mathbf{v}_{n_1}$ determined as follows:

$$\mathbf{A}\,[\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_{n_1}] = [\mathbf{v}_1\ \mathbf{v}_2 \cdots \mathbf{v}_{n_1}] \begin{bmatrix} \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_1 \end{bmatrix}$$

or

$$\begin{aligned} \mathbf{A}\mathbf{v}_1 &= \lambda_1 \mathbf{v}_1 \\ \mathbf{A}\mathbf{v}_2 &= \mathbf{v}_1 + \lambda_1 \mathbf{v}_2 \\ &\vdots \\ \mathbf{A}\mathbf{v}_{n_1} &= \mathbf{v}_{n_1-1} + \lambda_1 \mathbf{v}_{n_1} \end{aligned}$$

Rearranging these equations, we obtain

$$\begin{aligned} (\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_1 &= \mathbf{0} \\ (\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_2 &= -\mathbf{v}_1 \\ &\vdots \\ (\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_{n_1} &= -\mathbf{v}_{n_1-1} \end{aligned}$$

It can easily be established that each of these vector equations gives one linearly independent solution, and the solutions $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{n_1}$ form a linearly independent set of vectors. We shall call the set of vectors $\{\mathbf{v}_1, ..., \mathbf{v}_{n_1}\}$ the chain of *generalized eigenvectors*. Note that the vector $\mathbf{v}_1$ in the chain is, in fact, the eigenvector associated with multiple eigenvalue λ_1.

Eigenvectors for the Jordan blocks $\mathbf{\Lambda}_{n_1+1}, ..., \mathbf{\Lambda}_n$ are given by the solution of the vector equations

$$(\lambda_j\mathbf{I} - \mathbf{A})\mathbf{v}_j = \mathbf{0}; j = n_1 + 1, ..., n$$

The eigenvectors corresponding to distinct eigenvalues, and the chains of generalized eigenvectors corresponding to multiple eigenvalues, form the transformation matrix **P**.

Case II$_3$: $\mathbf{1 < \gamma(\lambda_1 I - A) < n_1}$ For this case, there are γ eigenvectors associated with λ_1. There will be one Jordan block for each eigenvector; that is, λ_1 will have γ blocks associated with it. This case is just a combination of the Cases II$_1$ and II$_2$; there is only one ambiguity—the knowledge of n_1 and γ does not directly give the information about the dimension of each of the Jordan blocks associated with λ_1.

Assume that λ_1 is a fourth-order root of the characteristic equation and $\gamma\,(\lambda_1\mathbf{I} - \mathbf{A}) = 2$. The two eigenvectors associated with λ_1 satisfy

$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_a = \mathbf{0}, (\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_b = \mathbf{0}$$

To form the transformation matrix, we require two generalized eigenvectors—but it is still uncertain whether the generalized eigenvectors are both associated with $\mathbf{v}_a$, or both with $\mathbf{v}_b$, or one with each. That is, the two Jordan blocks could take one of the following forms:

$$\mathbf{\Lambda}_1 = \begin{bmatrix} \lambda_1 & 1 & 0 \\ 0 & \lambda_1 & 1 \\ 0 & 0 & \lambda_1 \end{bmatrix}, \mathbf{\Lambda}_2 = [\lambda_1]$$

or

$$\mathbf{\Lambda}_1 = \begin{bmatrix} \lambda_1 & 1 \\ 0 & \lambda_1 \end{bmatrix}, \mathbf{\Lambda}_2 = \begin{bmatrix} \lambda_1 & 1 \\ 0 & \lambda_1 \end{bmatrix}$$

The first pair corresponds to the equations

$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_1 = \mathbf{0}$$
$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_2 = -\mathbf{v}_1$$
$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_3 = -\mathbf{v}_2$$
$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_4 = \mathbf{0}$$

The second pair corresponds to the equations

$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_1 = \mathbf{0}$$
$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_2 = -\mathbf{v}_1$$
$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_3 = \mathbf{0}$$
$$(\lambda_1\mathbf{I} - \mathbf{A})\mathbf{v}_4 = -\mathbf{v}_3$$

Ambiguities such as this, can be resolved by the trial-and-error procedure.

An n-dimensional SISO system with m distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_m$, of multiplicity $n_1, n_2, ..., n_m$, respectively $\left(n = \sum_{i=1}^{m} n_i \right)$, has the following Jordan canonical representation:

$$\dot{\mathbf{x}} = \mathbf{\Lambda}\mathbf{x} + \mathbf{b}u$$
$$y = \mathbf{c}\mathbf{x} + du$$

where $\mathbf{\Lambda}$ is a block diagonal matrix with Jordan blocks $\mathbf{\Lambda}_1, ..., \mathbf{\Lambda}_m$ corresponding to the eigenvalues $\lambda_1, ..., \lambda_m$, respectively, on its principal diagonal; each Jordan block $\mathbf{\Lambda}_i$ corresponding to the eigenvalue λ_i is again a block diagonal matrix with $\gamma(i)$ sub-blocks on its principal diagonal; $\gamma(i)$ being the number of linearly independent eigenvectors associated with the eigenvalue λ_i:

$$\underset{(n \times n)}{\mathbf{\Lambda}} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{\Lambda}_m \end{bmatrix}$$

$$\underset{(n_i \times n_i)}{\mathbf{\Lambda}_i} = \begin{bmatrix} \mathbf{\Lambda}_{1i} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_{2i} & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{\Lambda}_{\gamma(i)i} \end{bmatrix}; i = 1, 2, \ldots, m$$

$$\mathbf{\Lambda}_{ki} = \begin{bmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_i \end{bmatrix}; k = 1, 2, \ldots, \gamma(i)$$

The topic of computation of eigenvectors and generalized eigenvectors for systems with multiple eigenvalues is much too detailed and specialized for this book to treat (Refer to Gopal [105] and Brogan [106]). Over the years, experts have developed excellent general-purpose computer programs for the efficient and accurate determination of eigenvectors and generalized eigenvectors [152-154].

In this book, the usefulness of the transformation of state variable models to Jordan canonical form will be illustrated through system examples having distinct eigenvalues.

5.7 SOLUTION OF STATE EQUATIONS

In this section, we investigate the solution of the state equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t); \mathbf{x}(t_0) \triangleq \mathbf{x}^0 \tag{5.80}$$

where $\mathbf{x}$ is $n \times 1$ state vector, u is a scalar input, $\mathbf{A}$ is $n \times n$ constant matrix, and $\mathbf{b}$ is $n \times 1$ constant vector.

5.7.1 Matrix Exponential

Functions of square matrices arise in connection with the solution of vector differential equations. Of immediate interest to us are matrix infinite series.

Consider the infinite series in a scalar variable x:

$$f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots = \sum_{i=0}^{\infty} \alpha_i x^i \tag{5.81a}$$

with the radius of convergence r.

We can define infinite series in a matrix variable $\mathbf{A}$, as

$$f(\mathbf{A}) = \alpha_0 \mathbf{I} + \alpha_1 \mathbf{A} + \alpha_2 \mathbf{A}^2 + \cdots = \sum_{i=0}^{\infty} \alpha_i \mathbf{A}^i \tag{5.81b}$$

An important relation between the scalar power series (5.81a) and the matrix power series (5.81b) is that if the absolute values of eigenvalues of $\mathbf{A}$ are smaller than r, then the matrix power series (5.81b) converges (for proof, refer to Lefschetz [33]).

Consider, in particular, the scalar power series

$$f(x) = 1 + x + \frac{1}{2!}x^2 + \cdots + \frac{1}{k!}x^k + \cdots = \sum_{i=0}^{\infty} \frac{1}{i!} x^i \tag{5.82a}$$

It is well-known that this power series converges on to the exponential e^x for all finite x, so that

$$f(x) = e^x \tag{5.82b}$$

It follows from this result that the matrix power series

$$f(\mathbf{A}) = \mathbf{I} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \cdots + \frac{1}{k!}\mathbf{A}^k + \cdots = \sum_{i=0}^{\infty} \frac{1}{i!}\mathrm{A}^i$$

converges for all **A**. By analogy with the power series in Eqns (5.82) for the ordinary exponential function, we adopt the following nomenclature:

If **A** is an $n \times n$ matrix, the *matrix exponential* of **A** is

$$e^{\mathbf{A}} \triangleq \mathbf{I} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \cdots + \frac{1}{k!}\mathbf{A}^k + \cdots = \sum_{i=0}^{\infty} \frac{1}{i!}\mathrm{A}^i$$

The following matrix exponential will appear in the solution of state equations:

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{1}{2!}\mathbf{A}^2t^2 + \cdots + \frac{1}{k!}\mathbf{A}^kt^k + \cdots = \sum_{i=0}^{\infty} \frac{1}{i!}\mathbf{A}^it^i \tag{5.83}$$

It converges for all **A** and all finite t.

In the following, we examine some of the properties of the matrix exponential.

(i) $$e^{\mathbf{A}0} = \mathbf{I} \tag{5.84}$$

This is easily verified by setting $t = 0$ in Eqn. (5.83).

(ii) $$e^{\mathbf{A}(t+\tau)} = e^{\mathbf{A}t}e^{\mathbf{A}\tau} = e^{\mathbf{A}\tau}e^{\mathbf{A}t} \tag{5.85}$$

This is easily verified by multiplying out the first few terms for $e^{\mathbf{A}t}$ and $e^{\mathbf{A}\tau}$.

(iii) $$(e^{\mathbf{A}t})^{-1} = e^{-\mathbf{A}t} \tag{5.86}$$

Setting $\tau = -t$ in Eqn. (5.85), we obtain

$$e^{\mathbf{A}t}e^{-\mathbf{A}t} = e^{\mathbf{A}0} = \mathbf{I}$$

Thus the inverse of $e^{\mathbf{A}t}$ is $e^{-\mathbf{A}t}$.

Since the inverse of $e^{\mathbf{A}t}$ always exists, the matrix exponential is nonsingular for all finite values of t.

(iv) $$\frac{d}{dt}e^{\mathbf{A}t} = \mathbf{A}e^{\mathbf{A}t} = e^{\mathbf{A}t}\mathbf{A} \tag{5.87}$$

Term-by-term differentiation of Eqn. (5.83) gives

$$\begin{aligned}\frac{d}{dt}e^{\mathbf{A}t} &= \mathbf{A} + \mathbf{A}^2t + \frac{1}{2!}\mathbf{A}^3t^2 + \cdots + \frac{1}{(k-1)!}\mathbf{A}^kt^{k-1} + \cdots \\ &= \mathbf{A}[\mathbf{I} + \mathbf{A}t + \frac{1}{2!}\mathbf{A}^2t^2 + \cdots + \frac{1}{(k-1)!}\mathbf{A}^{k-1}t^{k-1} + \cdots] = \mathbf{A}e^{\mathbf{A}t} \\ &= [\mathbf{I} + \mathbf{A}t + \frac{1}{2!}\mathbf{A}^2t^2 + \cdots + \frac{1}{(k-1)!}\mathbf{A}^{k-1}t^{k-1} + \cdots]\mathbf{A} = e^{\mathbf{A}t}\mathbf{A}\end{aligned}$$

5.7.2 Solution of Homogeneous State Equation

The simplest form of the general differential equation (5.80) is the homogeneous, i.e., unforced equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t);\ \mathbf{x}(t_0) \triangleq \mathbf{x}^0 \tag{5.88}$$

We assume a solution $\mathbf{x}(t)$ of the form

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{k} \tag{5.89}$$

where $e^{\mathbf{A}t}$ is the matrix exponential function defined in Eqn. (5.83), and $\mathbf{k}$ is a suitably chosen constant vector.

The assumed solution is, in fact, the true solution since it satisfies the differential equation (5.88) as is seen below.

$$\dot{\mathbf{x}}(t) = \frac{d}{dt}[e^{\mathbf{A}t}\mathbf{k}] = \frac{d}{dt}[e^{\mathbf{A}t}]\mathbf{k}$$

Using property (5.87) of the matrix exponential, we obtain

$$\dot{\mathbf{x}}(t) = \mathbf{A}e^{\mathbf{A}t}\,\mathbf{k} = \mathbf{A}\mathbf{x}(t)$$

To evaluate the constant vector $\mathbf{k}$ in terms of the known initial state $\mathbf{x}(t_0)$, we substitute $t = t_0$ in Eqn. (5.89):

$$\mathbf{x}(t_0) = e^{\mathbf{A}t_0}\,\mathbf{k}$$

Using property (5.86) of the matrix exponential, we obtain

$$\mathbf{k} = (e^{\mathbf{A}t_0})^{-1}\mathbf{x}(t_0) = e^{-\mathbf{A}t_0}\mathbf{x}(t_0)$$

Thus, the general solution to Eqn. (5.88) for the state $\mathbf{x}(t)$ at time t, given the state $\mathbf{x}(t_0)$ at time t_0, is

$$\mathbf{x}(t) = e^{\mathbf{A}t}\,e^{-\mathbf{A}t_0}\,\mathbf{x}(t_0) = e^{\mathbf{A}(t-t_0)}\,\mathbf{x}(t_0) \tag{5.90a}$$

We have used the property (5.85) of the matrix exponential to express the solution in this form.

If the initial time $t_0 = 0$, i.e., the initial state $\mathbf{x}^0$ is known at $t = 0$, we have from Eqn. (5.90a):

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) \tag{5.90b}$$

From Eqn. (5.90b), it is observed that the initial state $\mathbf{x}(0) \triangleq \mathbf{x}^0$ at $t = 0$ is driven to a state $\mathbf{x}(t)$ at time t. This transition in state is carried out by the matrix exponential $e^{\mathbf{A}t}$. Due to this property, $e^{\mathbf{A}t}$ is known as the *state transition matrix*, and is denoted by $\phi(t)$.

Properties of State Transition Matrix

Properties of the matrix exponential, given earlier in Eqns (5.84)–(5.87), are restated below in terms of state transition matrix $\phi(t)$.

(i) $$\frac{d}{dt}\phi(t) = \mathbf{A}\phi(t);\ \phi(0) = \mathbf{I}$$

(ii) $$\phi(t_2 - t_1)\phi(t_1 - t_0) = \phi(t_2 - t_0) \text{ for any } t_0, t_1, t_2$$

This property of the state transition matrix is important since it implies that a state transition process can be divided into a number of sequential transitions. The transition from t_0 to t_2:

$$\mathbf{x}(t_2) = \phi(t_2 - t_0)\mathbf{x}(t_0);$$

is equal to the transition from t_0 to t_1 and then from t_1 to t_2:

$$\mathbf{x}(t_1) = \boldsymbol{\phi}(t_1 - t_0)\mathbf{x}(t_0)$$
$$\mathbf{x}(t_2) = \boldsymbol{\phi}(t_2 - t_1)\mathbf{x}(t_1)$$

(iii) $\boldsymbol{\phi}^{-1}(t) = \boldsymbol{\phi}(-t)$

(iv) $\boldsymbol{\phi}(t)$ is a nonsingular matrix for all finite t.

Evaluation of State Transition Matrix

The state transition matrix $\boldsymbol{\phi}(t) = e^{\mathbf{A}t}$ of an $n \times n$ matrix $\mathbf{A}$, is given by the infinite series (5.83). The series converges for all $\mathbf{A}$ and all finite t. Hence, $e^{\mathbf{A}t}$ can be evaluated within prescribed accuracy by truncating the series at, say, $i = N$. An algorithm for evaluation of matrix series is given in Section 6.3.

In the following, we discuss the commonly used methods for evaluating $e^{\mathbf{A}t}$ in closed form.

Evaluation Using Inverse Laplace Transforms Taking the Laplace transform on both sides of Eqn. (5.88) yields

$$s\mathbf{X}(s) - \mathbf{x}^0 = \mathbf{A}\mathbf{X}(s)$$

where $$\mathbf{X}(s) \triangleq \mathscr{L}[\mathbf{x}(t)];\ \mathbf{x}^0 \triangleq \mathbf{x}(0)$$

Solving for $\mathbf{X}(s)$, we get

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}^0$$

The state vector $\mathbf{x}(t)$ can be obtained by inverse transforming $\mathbf{X}(s)$:

$$\mathbf{x}(t) = \mathscr{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}]\mathbf{x}^0$$

Comparing this equation with Eqn. (5.90b), we get

$$e^{\mathbf{A}t} = \boldsymbol{\phi}(t) = \mathscr{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}] \tag{5.91}$$

The matrix $(s\mathbf{I} - \mathbf{A})^{-1} = \boldsymbol{\Phi}(s)$ is known in mathematical literature as the *resolvent* of $\mathbf{A}$. The entries of the *resolvent matrix* $\boldsymbol{\Phi}(s)$ are rational functions of s. Resolvent matrix $\boldsymbol{\Phi}(s)$ can be expressed in the following form (refer to Eqn. (5.43)):

$$\boldsymbol{\Phi}(s) = \frac{\mathbf{Q}(s)}{\Delta(s)} = \frac{\mathbf{Q}_1 s^{n-1} + \mathbf{Q}_2 s^{n-2} + \cdots + \mathbf{Q}_{n-1} s + \mathbf{Q}_n}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1} s + \alpha_n} \tag{5.92a}$$

where $\mathbf{Q}_i$ are constant $(n \times n)$ matrices and α_j are constant scalars. The coefficients of the scalar polynomial $\Delta(s)$ and the matrix polynomial $\mathbf{Q}(s)$ may be determined sequentially by *resolvent algorithm* (convenient for digital computer) given in Eqns (5.44).

The inverse transform

$$\mathscr{L}^{-1}[\mathbf{Q}(s)/\Delta(s)] = e^{\mathbf{A}t} \tag{5.92b}$$

can be expressed as a power series in t.

Example 5.10

Consider the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 3 \end{bmatrix} \mathbf{x};\ \mathbf{x}(0) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

By direct computation, we have

$$(s\mathbf{I} - \mathbf{A})^{-1} = \begin{bmatrix} s & 0 & 2 \\ 0 & s-1 & 0 \\ -1 & 0 & s-3 \end{bmatrix}^{-1} = \frac{(s\mathbf{I} - \mathbf{A})^{+}}{|\, s\mathbf{I} - \mathbf{A}\,|}$$

$$|s\mathbf{I} - \mathbf{A}| = (s-1)^2\,(s-2);\ (s\mathbf{I} - \mathbf{A})^{+} = \begin{bmatrix} (s-1)(s-3) & 0 & -2(s-1) \\ 0 & (s-1)(s-2) & 0 \\ (s-1) & 0 & s(s-1) \end{bmatrix}$$

$$e^{\mathbf{A}t} = \mathscr{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}] = \mathscr{L}^{-1} \begin{bmatrix} \dfrac{(s-3)}{(s-1)(s-2)} & 0 & \dfrac{-2}{(s-1)(s-2)} \\ 0 & \dfrac{1}{(s-1)} & 0 \\ \dfrac{1}{(s-1)(s-2)} & 0 & \dfrac{s}{(s-1)(s-2)} \end{bmatrix}$$

$$= \begin{bmatrix} 2e^{t} - e^{2t} & 0 & 2e^{t} - 2e^{2t} \\ 0 & e^{t} & 0 \\ -e^{t} + e^{2t} & 0 & 2e^{2t} - e^{t} \end{bmatrix}$$

Consequently, the free response of the system is

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) = \begin{bmatrix} 0 \\ e^{t} \\ 0 \end{bmatrix}$$

Note that $\mathbf{x}(t)$ could be more easily computed by taking the inverse Laplace transform of

$$\mathbf{X}(s) = [(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0)].$$

Evaluation Using Similarity Transformation Suppose that $\mathbf{A}$ is an $n \times n$ nondiagonal matrix with distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$. We define the diagonal matrix $\mathbf{\Lambda}$ as

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

A and **Λ** are similar matrices; there exists a nonsingular transformation matrix **P** such that (refer to Eqns (5.22))

$$\mathbf{\Lambda} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$$

Now

$$\mathbf{P}^{-1}e^{\mathbf{A}t}\mathbf{P} = \mathbf{P}^{-1}[\mathbf{I} + \mathbf{A}t + \frac{1}{2!}\mathbf{A}^2t^2 + \cdots]\mathbf{P} = \mathbf{I} + \mathbf{P}^{-1}\mathbf{A}\mathbf{P}t + \frac{1}{2!}\mathbf{P}^{-1}\mathbf{A}^2\mathbf{P}t^2 + \cdots$$

$$= \mathbf{I} + \mathbf{P}^{-1}\mathbf{A}\mathbf{P}t + \frac{1}{2!}\mathbf{P}^{-1}\mathbf{A}\mathbf{P}\mathbf{P}^{-1}\mathbf{A}\mathbf{P}t^2 + \cdots = \mathbf{I} + \mathbf{\Lambda}t + \frac{1}{2!}\mathbf{\Lambda}^2t^2 + \cdots = e^{\mathbf{\Lambda}t}$$

Thus the matrices $e^{\mathbf{A}t}$ and $e^{\mathbf{\Lambda}t}$ are similar. Since **Λ** is diagonal, $e^{\mathbf{\Lambda}t}$is given by

$$e^{\mathbf{\Lambda}t} = \begin{bmatrix} e^{\lambda_1 t} & 0 & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix}$$

The matrix exponential $e^{\mathbf{A}t}$ of matrix **A** with distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ may, therefore, be evaluated using the following relation:

$$e^{\mathbf{A}t} = \mathbf{P}\,e^{\mathbf{\Lambda}t}\,\mathbf{P}^{-1} = \mathbf{P}\begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix}\mathbf{P}^{-1} \tag{5.93}$$

where **P** is a transformation matrix that transforms **A** into the diagonal form.

(For the general case wherein matrix **A** has multiple eigenvalues, refer to [105]. Also refer to Review Example 5.3 given at the end of this chapter).

Example 5.11

Consider the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}\mathbf{x};\ \mathbf{x}(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The characteristic equation for this system is

$$|\lambda\mathbf{I} - \mathbf{A}| = \begin{vmatrix} \lambda & -1 \\ 2 & (\lambda + 3) \end{vmatrix} = 0$$

or

$$(\lambda + 1)(\lambda + 2) = 0$$

Therefore, the eigenvalues of system matrix **A** are

$$\lambda_1 = -1,\ \lambda_2 = -2$$

Eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ corresponding to the eigenvalues λ_1 and λ_2, respectively, can be determined from the adjoint matrix $(\lambda\mathbf{I} - \mathbf{A})^+$ (refer to Eqn. (5.79)).

$$(\lambda \mathbf{I} - \mathbf{A})^{+} = \begin{bmatrix} (\lambda + 3) & 1 \\ -2 & \lambda \end{bmatrix}$$

For $\lambda = \lambda_1 = -1$,

$$(\lambda_1 \mathbf{I} - \mathbf{A})^{+} = \begin{bmatrix} 2 & 1 \\ -2 & -1 \end{bmatrix}; \mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

For $\lambda = \lambda_2 = -2$,

$$(\lambda_2 \mathbf{I} - \mathbf{A})^{+} = \begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix}; \mathbf{v}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

The transformation matrix **P** that transforms **A** into diagonal form, is

$$\mathbf{P} = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix}$$

The matrix exponential

$$e^{\mathbf{A}t} = \mathbf{P} \begin{bmatrix} e^{-t} & 0 \\ 0 & e^{-2t} \end{bmatrix} \mathbf{P}^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} e^{-t} & 0 \\ 0 & e^{-2t} \end{bmatrix} \begin{bmatrix} 2 & 1 \\ -1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 2e^{-t} - e^{-2t} & e^{-t} - e^{-2t} \\ -2e^{-t} + 2e^{-2t} & -e^{-t} + 2e^{-2t} \end{bmatrix}$$

Consequently, the free response of the system is

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) = \begin{bmatrix} e^{-t} - e^{-2t} \\ -e^{-t} + 2e^{-2t} \end{bmatrix}$$

Evaluation Using Cayley–Hamilton Technique The state transition matrix may be evaluated using a technique based on the Cayley–Hamilton theorem. To begin with, we restate the theorem proved earlier in Section 5.4 (refer to Eqns (5.45)–(5.46)).

Every square matrix **A** *satisfies its own characteristic equation*.

Thus if we have, for an $n \times n$ matrix **A**, the characteristic equation

$$\Delta(\lambda) = |\lambda \mathbf{I} - \mathbf{A}| = \lambda^n + \alpha_1 \lambda^{n-1} + \cdots + \alpha_{n-1}\lambda + \alpha_n = 0,$$

then, according to this theorem

$$\Delta(\mathbf{A}) = \mathbf{A}^n + \alpha_1 \mathbf{A}^{n-1} + \cdots + \alpha_{n-1}\mathbf{A} + \alpha_n \mathbf{I} = \mathbf{0}$$

where **I** is an identity matrix and **0** is a null matrix.

This theorem provides a simple procedure for evaluating the function of a matrix. In the study of linear systems, we are mostly concerned with functions which can be represented as a series of the powers of a matrix. Consider the matrix polynomial

$$f(\mathbf{A}) = a_0\mathbf{I} + a_1\mathbf{A} + a_2\mathbf{A}^2 + \cdots + a_n\mathbf{A}^n + a_{n+1}\mathbf{A}^{n+1} + \cdots \quad (5.94a)$$

This matrix polynomial, which is of degree higher than the order of **A**, can be computed by consideration of the scalar polynomial

$$f(\lambda) = a_0 + a_1\lambda + a_2\lambda^2 + \cdots + a_n\lambda^n + a_{n+1}\lambda^{n+1} + \cdots \tag{5.94b}$$

Dividing $f(\lambda)$ by the characteristic polynomial $\Delta(\lambda)$, we get

$$\frac{f(\lambda)}{\Delta(\lambda)} = q(\lambda) + \frac{g(\lambda)}{\Delta(\lambda)} \tag{5.95a}$$

where $g(\lambda)$ is the remainder polynomial of the following form:

$$g(\lambda) = \beta_0 + \beta_1\lambda + \cdots + \beta_{n-1}\lambda^{n-1} \tag{5.95b}$$

Equation (5.95a) may be written as

$$f(\lambda) = q(\lambda)\Delta(\lambda) + g(\lambda) \tag{5.96}$$

Assume that the $n \times n$ matrix **A** has n distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$;

$$\Delta(\lambda_i) = 0; \quad i = 1, 2, ..., n$$

If we evaluate $f(\lambda)$ in Eqn. (5.96) at the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$, we have

$$f(\lambda_i) = g(\lambda_i),\ i = 1, 2, ..., n \tag{5.97}$$

The coefficients $\beta_0, \beta_1, ... , \beta_{n-1}$ in Eqn. (5.95b) can be computed by solving the set of n simultaneous equations obtained by successively substituting $\lambda_1, \lambda_2, ... , \lambda_n$ in Eqn. (5.97).

Substituting **A** for λ in Eqn. (5.96), we get

$$f(\mathbf{A}) = q(\mathbf{A})\Delta(\mathbf{A}) + g(\mathbf{A})$$

Since $\Delta(\mathbf{A})$ is identically zero, it follows that

$$f(\mathbf{A}) = g(\mathbf{A}) = \beta_0\mathbf{I} + \beta_1\mathbf{A} + \cdots + \beta_{n-1}\,\mathbf{A}^{n-1}$$

If **A** possesses an eigenvalue λ_k of multiplicity n_k, then only one independent equation can be obtained by substituting λ_k into Eqn. (5.97). The remaining $(n_k - 1)$ linear equations, which must be obtained in order to solve for β_i's, can be found by differentiating both sides of Eqn. (5.97).

Since

$$\left[\frac{d^j}{d\lambda^j}\Delta(\lambda)\right]\Bigg|_{\lambda=\lambda_k} = 0\ ;\ j = 0, 1, ..., (n_k - 1),$$

it follows that

$$\left[\frac{d^j}{d\lambda^j}f(\lambda)\right]\Bigg|_{\lambda=\lambda_k} = \left[\frac{d^j}{d\lambda^j}g(\lambda)\right]\Bigg|_{\lambda=\lambda_k};\ j = 0, 1, ..., (n_k - 1)$$

The formal procedure of evaluation of the matrix polynomial $f(\mathbf{A})$ is given below.

(i) Compute $\Delta(\lambda) \triangleq |\lambda\mathbf{I} - \mathbf{A}|$

(ii) Find the roots of $\Delta(\lambda) = 0$, say,

$$\Delta(\lambda) = (\lambda - \lambda_1)^{n_1}\,(\lambda - \lambda_2)^{n_2} \cdots (\lambda - \lambda_m)^{n_m} \tag{5.98a}$$

where $n_1 + n_2 + \cdots + n_m = n$. In other words, $\Delta(\lambda)$ has root λ_i with multiplicity n_i. If λ_i is a complex number, then its complex conjugate is also a root of $\Delta(\lambda)$.

(iii) Form a polynomial $g(\lambda)$ of degree $(n-1)$; i.e.,

$$g(\lambda) = \beta_0 + \beta_1\lambda + \cdots + \beta_{n-1}\lambda^{n-1} \tag{5.98b}$$

where the unknown parameters $\beta_0, \beta_1, ..., \beta_{n-1}$ are to be solved in Step (v).

(iv) Form the following n equations:

$$\left[\frac{d^j}{d\lambda^j} f(\lambda)\right]\Bigg|_{\lambda=\lambda_i} = \left[\frac{d^j}{d\lambda^j} g(\lambda)\right]\Bigg|_{\lambda=\lambda_i} \quad \begin{matrix} j = 0,1,\ldots,(n_i - 1) \\ ;\ i = 1,2,\ldots,m \end{matrix} \tag{5.98c}$$

(v) Solve for the n unknown parameters $\beta_0, \beta_1, ..., \beta_{n-1}$ from the n equations in Step (iv). Then

$$f(\mathbf{A}) = g(\mathbf{A}) = \beta_0\mathbf{I} + \beta_1\mathbf{A} + \cdots + \beta_{n-1}\mathbf{A}^{n-1} \tag{5.98d}$$

Example 5.12

Find $f(\mathbf{A}) = \mathbf{A}^{10}$ for

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

Solution The characteristic polynomial is

$$\Delta(\lambda) = |\lambda\mathbf{I} - \mathbf{A}| = \begin{vmatrix} \lambda & -1 \\ 2 & \lambda+3 \end{vmatrix} = (\lambda + 1)(\lambda + 2)$$

The roots of $\Delta(\lambda) = 0$ are $\lambda_1 = -1$, $\lambda_2 = -2$.

Since $\mathbf{A}$ is of second order, the polynomial $g(\lambda)$ will be of the following form:

$$g(\lambda) = \beta_0 + \beta_1\lambda$$

The coefficients β_0 and β_1 are evaluated from equations

$$f(\lambda_1) = (\lambda_1)^{10} = g(\lambda_1) = \beta_0 + \beta_1\lambda_1$$
$$f(\lambda_2) = (\lambda_2)^{10} = g(\lambda_2) = \beta_0 + \beta_1\lambda_2$$

The result is

$$\beta_0 = -1022,\ \beta_1 = -1023$$

Therefore,

$$f(\mathbf{A}) = \mathbf{A}^{10} = \beta_0\mathbf{I} + \beta_1\mathbf{A} = \begin{bmatrix} -1022 & -1023 \\ 2046 & 2047 \end{bmatrix}$$

The Cayley–Hamilton technique allows us to solve the problem of evaluation of $e^{\mathbf{A}t}$, where $\mathbf{A}$ is a constant $n \times n$ matrix. Since the matrix power series

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{\mathbf{A}^2t^2}{2!} + \cdots + \frac{\mathbf{A}^nt^n}{n!} + \cdots$$

converges for all $\mathbf{A}$ and for all finite t, the matrix polynomial $f(\mathbf{A}) = e^{\mathbf{A}t}$ can be expressed as a polynomial $g(\mathbf{A})$ of degree $(n-1)$. This is illustrated below with the help of an example.

Example 5.13

Consider the system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -2 \\ 0 & 1 & 0 \\ 1 & 0 & 3 \end{bmatrix}; \mathbf{x}(0) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

In the following, we evaluate the function

$$f(\mathbf{A}) = e^{\mathbf{A}t}$$

using the Cayley–Hamilton technique.

The characteristic polynomial of matrix **A** is

$$\Delta(\lambda) = |\lambda\mathbf{I} - \mathbf{A}| = \begin{vmatrix} \lambda & 0 & 2 \\ 0 & (\lambda - 1) & 0 \\ -1 & 0 & (\lambda - 3) \end{vmatrix} = (\lambda - 1)^2 (\lambda - 2)$$

The characteristic equation $\Delta(\lambda) = 0$ has a second-order root at $\lambda_1 = 1$ and a simple root at $\lambda_2 = 2$.

Since **A** is of third order, the polynomial $g(\lambda)$ will be of the form

$$g(\lambda) = \beta_0 + \beta_1\lambda + \beta_2\lambda^2$$

The coefficients β_0, β_1, and β_2 are evaluated using the following relations:

$$f(\lambda_1) = g(\lambda_1)$$

$$\frac{d}{d\lambda} f(\lambda)\Big|_{\lambda = \lambda_1} = \frac{d}{d\lambda} g(\lambda)\Big|_{\lambda = \lambda_1}$$

$$f(\lambda_2) = g(\lambda_2)$$

These relations yield the following set of simultaneous equations:

$$e^t = \beta_0 + \beta_1 + \beta_2$$

$$te^t = \beta_1 + 2\beta_2$$

$$e^{2t} = \beta_0 + 2\beta_1 + 4\beta_2$$

Solving these equations, we obtain

$$\beta_0 = -2te^t + e^{2t}$$

$$\beta_1 = 3te^t + 2e^t - 2e^{2t}, \text{ and}$$

$$\beta_2 = e^{2t} - e^t - te^t$$

Hence, we have

$$e^{\mathbf{A}t} = g(\mathbf{A}) = \beta_0\mathbf{I} + \beta_1\mathbf{A} + \beta_2\mathbf{A}^2$$

$$= (-2te^t + e^{2t})\mathbf{I} + (3te^t + 2e^t - 2e^{2t})\mathbf{A} + (e^{2t} - e^t - te^t)\mathbf{A}^2$$

$$= \begin{bmatrix} 2e^{t} - e^{2t} & 0 & 2e^{t} - 2e^{2t} \\ 0 & e^{t} & 0 \\ -e^{t} + e^{2t} & 0 & 2e^{2t} - e^{t} \end{bmatrix}$$

Consequently, the free response ($u(t) = 0$) of the system is

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) = \begin{bmatrix} 0 \\ e^{t} \\ 0 \end{bmatrix}$$

This result is identical to the one obtained earlier in Example 5.10.

5.7.3 Solution of Nonhomogeneous State Equation

When an input $u(t)$ is present, the complete solution $\mathbf{x}(t)$ is obtained from the nonhomogeneous equation (5.80).

By writing Eqn. (5.80) as

$$\dot{\mathbf{x}}(t) - \mathbf{A}\mathbf{x}(t) = \mathbf{b}u(t)$$

and premultiplying both sides of this equation by $e^{-\mathbf{A}t}$, we obtain

$$e^{-\mathbf{A}t}[\dot{\mathbf{x}}(t) - \mathbf{A}\mathbf{x}(t)] = e^{-\mathbf{A}t}\mathbf{b}u(t) \tag{5.99}$$

By applying the rule for the derivative of the product of two matrices, we can write (refer to Eqn. (5.87))

$$\frac{d}{dt}[e^{-\mathbf{A}t}\mathbf{x}(t)] = e^{-\mathbf{A}t}\frac{d}{dt}(\mathbf{x}(t)) + \frac{d}{dt}(e^{-\mathbf{A}t})\mathbf{x}(t) = e^{-\mathbf{A}t}\dot{\mathbf{x}}(t) - e^{-\mathbf{A}t}\mathbf{A}\mathbf{x}(t)$$
$$= e^{-\mathbf{A}t}[\dot{\mathbf{x}}(t) - \mathbf{A}\mathbf{x}(t)]$$

Use of this equality in Eqn. (5.99) gives

$$\frac{d}{dt}[e^{-\mathbf{A}t}\mathbf{x}(t)] = e^{-\mathbf{A}t}\mathbf{b}u(t)$$

Integrating both sides with respect to t between the limits 0 and t, we get

$$e^{-\mathbf{A}t}\mathbf{x}(t)\Big|_{0}^{t} = \int_{0}^{t} e^{-\mathbf{A}\tau}\mathbf{b}u(\tau)\,d\tau$$

or

$$e^{-\mathbf{A}t}\mathbf{x}(t) - \mathbf{x}(0) = \int_{0}^{t} e^{-\mathbf{A}\tau}\mathbf{b}u(\tau)\,d\tau$$

Now, premultiplying both sides by $e^{\mathbf{A}t}$, we have

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_{0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{b}u(\tau)\,d\tau \tag{5.100}$$

If the initial state is known at $t = t_0$, rather than $t = 0$, Eqn. (5.100) becomes

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\,\mathbf{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{b}u(\tau)d\tau \tag{5.101}$$

Equation (5.101) can also be written as

$$\mathbf{x}(t) = \boldsymbol{\phi}(t-t_0)\,\mathbf{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\phi}(t-\tau)\,\mathbf{b}u(\tau)\,d\tau \tag{5.102}$$

where

$$\boldsymbol{\phi}(t) = e^{\mathbf{A}t}$$

Equation (5.102) is the solution of Eqn. (5.80). This equation is called the *state transition equation*. It describes the change of state relative to the initial conditions $\mathbf{x}(t_0)$ and the input $u(t)$.

Example 5.14

For the speed control system of Fig. 5.3, the following plant model was derived in Example 5.1 (refer to Eqns (5.17)):

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$
$$y = \mathbf{c}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} -1 & 1 \\ -1 & -10 \end{bmatrix};\ \mathbf{b} = \begin{bmatrix} 0 \\ 10 \end{bmatrix};\ \mathbf{c} = [\,1 \quad 0\,]$$

State variables x_1 and x_2 are the physical variables of the system:

$x_1(t) = \omega(t)$, angular velocity of the motor shaft
$x_2(t) = i_a(t)$, armature current

The output

$$y(t) = x_1(t) = \omega(t)$$

In the following, we evaluate the response of this system to a unit-step input, under zero initial conditions.

$$(s\mathbf{I} - \mathbf{A})^{-1} = \begin{bmatrix} s+1 & -1 \\ 1 & s+10 \end{bmatrix}^{-1} = \frac{1}{s^2 + 11s + 11}\begin{bmatrix} s+10 & 1 \\ -1 & s+1 \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{s+10}{(s+a_1)(s+a_2)} & \dfrac{1}{(s+a_1)(s+a_2)} \\ \dfrac{-1}{(s+a_1)(s+a_2)} & \dfrac{s+1}{(s+a_1)(s+a_2)} \end{bmatrix};\ a_1 = 1.1125,\ a_2 = 9.8875$$

$$e^{\mathbf{A}t} = \mathscr{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}]$$

$$= \begin{bmatrix} 1.0128e^{-a_1 t} - 0.0128e^{-a_2 t} & 0.114e^{-a_1 t} - 0.114e^{-a_2 t} \\ -0.114e^{-a_1 t} + 0.114e^{-a_2 t} & -0.0128e^{-a_1 t} + 1.0128e^{-a_2 t} \end{bmatrix}$$

$$u(t) = 1;\ t \geq 0;\ \mathbf{x}(0) = \mathbf{0}$$

Therefore,

$$\mathbf{x}(t) = \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{b}\,d\tau = \int_0^t \begin{bmatrix} 1.14\left(e^{-a_1(t-\tau)} - e^{-a_2(t-\tau)}\right) \\ 1.14\left(-0.1123e^{-a_1(t-\tau)} + 8.8842e^{-a_2(t-\tau)}\right) \end{bmatrix} d\tau$$

$$= \begin{bmatrix} 0.9094 - 1.0247\,e^{-a_1 t} + 0.1153\,e^{-a_2 t} \\ -0.0132 + 0.1151\,e^{-a_1 t} - 0.1019\,e^{-a_2 t} \end{bmatrix}$$

The output

$$y(t) = \omega(t) = 0.9094 - 1.0247\,e^{-1.1125t} + 0.1153e^{-9.8875t};\ t \geq 0$$

5.8 CONCEPTS OF CONTROLLABILITY AND OBSERVABILITY

Controllability and observability are properties which describe structural features of a dynamic system. These properties play an important role in modern control system design theory; the conditions on controllability and observability often govern the control solution.

To illustrate the motivation of investigating controllability and observability properties, we consider the problem of the stabilization of an inverted pendulum on a motor-driven cart.

Example 5.15

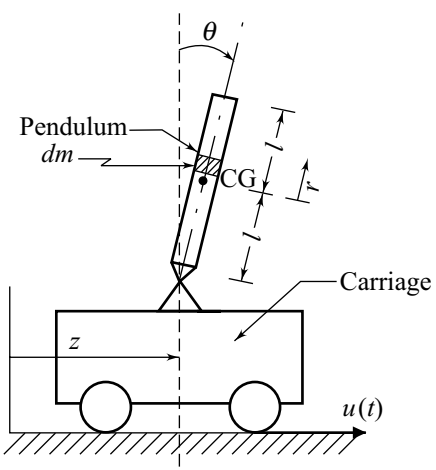


Fig. 5.16 Inverted pendulum system

Figure 5.16 shows an inverted pendulum with its pivot mounted on a cart. The cart is driven by an electric motor. The motor drives a pair of wheels of the cart; the whole cart and the pendulum become the 'load' on the motor. The motor at time t exerts a torque $T(t)$ on the wheels. The linear force applied to the cart is $u(t)$; $T(t) = Ru(t)$, where R is the radius of the wheels.

The pendulum is obviously unstable. It can, however, be kept upright by applying a proper control force $u(t)$. This somewhat artificial system example represents a dynamic model of a space booster on take off—the booster is balanced on top of the rocket engine thrust vector.

From inspection of Fig. 5.16, we construct the differential equations describing the dynamics of the inverted pendulum and the cart. The horizontal displacement of the pivot on the cart with respect to the fixed nonrotating frame, is $z(t)$, while the rotational angle of the pendulum is $\theta(t)$. The parameters of the system are as follows:

M = the mass of the cart;
L = the length of the pendulum = $2l$;
m = the mass of the pendulum; and

J = the moment of inertia of the pendulum with respect to center of gravity (CG).

$$J = \int_{-l}^{l} r^2 dm = \int_{-l}^{l} r^2 (\rho A dr) = \rho A \left[\frac{r^3}{3}\right]_{-l}^{l} = \rho A \left(\frac{2l^3}{3}\right) = \rho A (2l)\left(\frac{l^2}{3}\right)$$

$$= \frac{ml^2}{3}$$

where A = area of cross section, and ρ = density.

The horizontal and vertical positions of the CG of the pendulum are given by $(z + l\sin\theta)$ and $(l\cos\theta)$, respectively.

The forces exerted on the pendulum are—the force mg on the center of gravity, a horizontal reaction force H and a vertical reaction force V (Fig. 5.17a). H is the horizontal reaction force that the cart exerts on the pendulum, whereas $-H$ is the force exerted by the pendulum on the cart. Similar convention applies to forces V and $-V$.

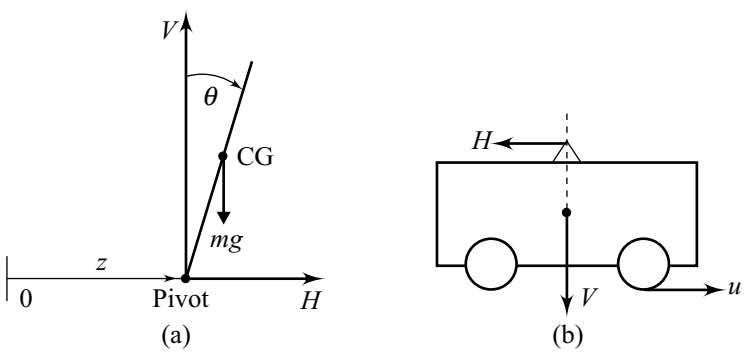


Fig. 5.17

Taking moments around CG of the pendulum, we get

$$J\frac{d^2\theta(t)}{dt^2} = V(t)\,l\sin\theta(t) - H(t)\,l\cos\theta(t) \tag{5.103a}$$

Summing up all forces on the pendulum in vertical and horizontal directions, we obtain

$$m\frac{d^2}{dt^2}(l\cos\theta(t)) = V(t) - mg \tag{5.103b}$$

$$m\frac{d^2}{dt^2}(z(t) + l\sin\theta(t)) = H(t) \tag{5.103c}$$

Summing up all the forces on the cart in the horizontal direction (Fig. 5.17b), we get

$$M\frac{d^2z(t)}{dt^2} = u(t) - H(t) - F_c \tag{5.103d}$$

where

$$F_c = B_c\, sign\,(\dot{z}) \tag{5.103e}$$

is the model of the frictional force of the cart wheels on the track; B_c is the cart friction coefficient.

Performing the required differentiation in Eqns (5.103b) and (5.103c), we get

$$ml(-\dot{\theta}^2 \cos\theta - \ddot{\theta}\cos\theta) = V - mg \tag{5.104a}$$

$$m\left(\ddot{z} + l(-\dot{\theta}^2 \sin\theta + \ddot{\theta}\cos\theta)\right) = H \tag{5.104b}$$

Substituting (5.103d) into (5.104b) gives

$$m\ddot{z} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2 \sin\theta + F_c = u - M\ddot{z} \tag{5.104c}$$

Now, substituting Eqns (5.104a) and (5.103d) into (5.103a) yields

$$J\ddot{\theta} = \left[mg - ml\dot{\theta}^2 \cos\theta - ml\ddot{\theta}\sin\theta\right] l\sin\theta + F_c l\cos\theta - (u - M\ddot{z})l\cos\theta \tag{5.104d}$$

We next substitute $(u - M\ddot{z})$ from (5.104c) into (5.104d) and perform manipulations to get

$$J\ddot{\theta} = mgl\sin\theta - ml^2\ddot{\theta} - m\ddot{z}l\cos\theta \tag{5.104e}$$

Let

$$a = \frac{1}{m+M}$$

Then, we can represent (5.104e) as

$$\ddot{z} = -\ mal\ddot{\theta}\cos\theta + mal\dot{\theta}^2 \sin\theta - aF_c + au \tag{5.104f}$$

We substitute (5.104f) into (5.104e), to obtain

$$\ddot{\theta} = \frac{mgl\sin\theta - (m^2l^2a\dot{\theta}^2 \sin 2\theta)/2 + (mal\cos\theta)F_c - (mal\cos\theta)u}{J - m^2l^2a\cos^2\theta + ml^2} \tag{5.104g}$$

We next substitute $\ddot{\theta}$ from (5.104e) into (5.104f) to get

$$\ddot{z} = \frac{-(m^2l^2ag\sin 2\theta)/2 + (mal\dot{\theta}^2 \sin\theta + a(u - F_c))(J + ml^2)}{J + ml^2 - m^2l^2a\cos^2\theta} \tag{5.104h}$$

Since $J = \frac{1}{3}ml^2$, Eqns (5.104g) and (5.104h) reduce to the following nonlinear set of equations.

$$\ddot{\theta} = \frac{g\sin\theta - (mla\dot{\theta}^2 \sin 2\theta)/2 + a\cos\theta(F_c - u)}{4l/3 - mla\cos^2\theta} \tag{5.105a}$$

$$\ddot{z} = \frac{-(mag\sin 2\theta)/2 + (a\theta\sin\theta)4ml/3 + (u - F_c)4a/3}{4/3 - ma\cos^2\theta} \tag{5.105b}$$

Suppose that the system parameters are as follows:

$M = 1$ kg; $m = 0.15$ kg; and $l = 0.5$ m.

Recall that $g = 9.81$ m/sec^2.

In our problem, since the objective is to keep the pendulum upright, it seems reasonable to assume that $\dot{\theta}(t)$ and $\theta(t)$ will remain close to zero. In view of this, we can set with sufficient accuracy $\sin\theta \simeq \theta$; $\cos\theta \simeq 1$. Also, the second-order deviations $\theta \times \theta \simeq 0$; $\dot{\theta} \times \dot{\theta} \simeq 0$. We further assume, for simplified analysis, that $F_c = 0$.

With these assumptions, we have from Eqns (5.105)

$$\ddot{\theta}(t) = 16.3106\ \theta(t) - 1.4458\ u(t)$$

$$\ddot{z}(t) = -\ 1.0637\ \theta(t) + 0.9639\ u(t)$$

Choosing the states $x_1 = \theta, x_2 = \dot{\theta}$, $x_3 = z$, and $x_4 = \dot{z}$, we obtain the following state model for the inverted pendulum on moving cart:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u \tag{5.106}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 16.3106 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1.0637 & 0 & 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ -1.4458 \\ 0 \\ 0.9639 \end{bmatrix}$$

The plant (5.106) is said to be *completely controllable* if every state $\mathbf{x}(t_0)$ can be affected or controlled to reach a desired state in finite time, by some unconstrained control $u(t)$. Shortly, we will see that the plant (5.106) satisfies this condition, and therefore, a solution exists to the following control problem:

Move the cart from one location to another without causing the pendulum to fall.

The solution to this control problem is not unique. We normally look for a feedback control scheme so that the destabilizing effects of disturbance forces (due to wind, for example) are filtered out. Figure 5.18a shows a state-feedback control scheme for stabilizing the inverted pendulum. The closed-loop system is formed by feeding back the state variables through a real constant matrix **k**:

$$u(t) = -\mathbf{k}\mathbf{x}(t)$$

The closed-loop system is thus described by

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{b}\mathbf{k})\mathbf{x}(t)$$

The design objective in this case is to find the feedback matrix **k** such that the closed-loop system is stable. The existence of a solution to this design problem is directly based on the controllability property of the plant (5.106). This will be established in Chapter 7.

Implementation of the state-feedback control solution requires access to all the state variables of the plant model. In many control situations of interest, it is possible to install sensors to measure all the state variables. This may not be possible or practical in some cases. For example, if the plant model includes nonphysical state variables, measurement of these variables using physical sensors is not possible. Accuracy requirements or cost considerations may prohibit the use of sensors for some physical variables also.

The input and the output of a system are always physical quantities, and are normally easily accessible to measurement. We, therefore, need a subsystem that performs the estimation of state variables based on the information received from the input $u(t)$ and the output $y(t)$. This subsystem is called an *observer* whose design is based on *observability property* of the controlled system.

The plant (5.106) is said to be *completely observable* if all the state variables in $\mathbf{x}(t)$ can be observed from the measurements of the output $y(t) = \theta(t)$ and the input $u(t)$. Shortly, we will see that the plant (5.106) does not satisfy this condition and therefore, a solution to the observer-design problem does not exist when the inputs to the observer subsystem are $u(t)$ and $\theta(t)$.

Cart position $z(t)$ is easily accessible to measurement and as we shall see, the observability condition is satisfied with this choice of input information to the observer subsystem. Figure 5.18b shows the block

diagram of the closed-loop system with an observer that estimates the state vector from measurements of $u(t)$ and $z(t)$. The observed or estimated state vector, designated as $\hat{\mathbf{x}}$, is then used to generate the control u through the feedback matrix $\mathbf{k}$.

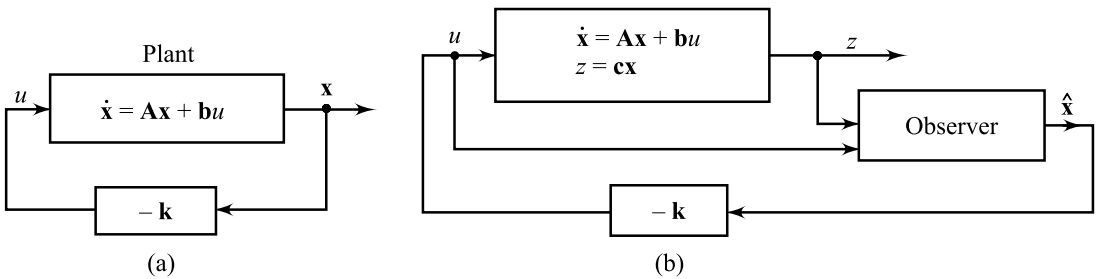


Fig. 5.18 Control system with state feedback

A study of controllability and observability properties, presented in this section, provides a basis for the state-feedback design problems discussed in Chapter 7. Further, these properties establish the conditions for complete equivalence between the state variable and transfer function representations.

5.8.1 Definitions of Controllability and Observability

In this section, we study the controllability and observability of linear time-invariant systems described by state variable model of the following form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \tag{5.107a}$$

$$y(t) = \mathbf{c}\mathbf{x}(t) + du(t) \tag{5.107b}$$

where $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ and d are respectively $n \times n$, $n \times 1$, $1 \times n$ and 1×1 matrices, $\mathbf{x}(t)$ is $n \times 1$ state vector, $y(t)$ and $u(t)$ are, respectively, output and input variables.

Controllability

For the linear system given by Eqns (5.107), if there exists an input $u_{[0,\, t_1]}$ which transfers the initial state $\mathbf{x}(0) \triangleq \mathbf{x}^0$ to the state $\mathbf{x}^1$ in a finite time t_1, the state $\mathbf{x}^0$ is said to be controllable. If all initial states are controllable, the system is said to be *completely controllable*, or simply *controllable*. Otherwise, the system is said to be *uncontrollable*.

From Eqn. (5.100), the solution of Eqn. (5.107a) is

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}^0 + \int_0^t e^{\mathbf{A}(t-\tau)}\,\mathbf{b}u(\tau)\,d\tau$$

To study the controllability property, we may assume, without loss of generality, that $\mathbf{x}^1 \equiv \mathbf{0}$. Therefore, if the system (5.107) is controllable, there exists an input $u_{[0,\, t_1]}$ such that

$$-\mathbf{x}^0 = \int_0^{t_1} e^{-\mathbf{A}\tau}\,\mathbf{b}u(\tau)\,d\tau \tag{5.108}$$

From this equation, we observe that complete controllability of a system depends on **A** and **b**, and is independent of output matrix **c**. The controllability of the system (5.107) is frequently referred to as the controllability of the pair {**A**, **b**}.

It may be noted that according to the definition of controllability, there is no constraint imposed on the input or on the trajectory that the state should follow. Further, the system is said to be uncontrollable although it may be 'controllable in part'.

From the definition of controllability, we observe that by complete controllability of a plant we mean that we can make the plant do whatever we please. Perhaps this definition is too restrictive in the sense that we are asking too much of the plant. But if we are able to show that system equations satisfy this definition, certainly there can be no intrinsic limitation on the design of the control system for the plant. However, if the system turns out to be uncontrollable, it does not necessarily mean that the plant can never be operated in a satisfactory manner. Provided that a control system will maintain the important variables in an acceptable region, the fact that the plant is not completely controllable, is immaterial.

Another important point which the reader must bear in mind, is that almost all physical systems are nonlinear in nature to a certain extent, and a linear model is obtained after making certain approximations. Small perturbations of the elements of **A** and **b** may cause an uncontrollable system to become controllable. It may also be possible to increase the number of control variables and make the plant completely controllable (controllability of multi-input systems is discussed in Section 5.10).

A common source of uncontrollable state variable models arises when redundant state variables are defined. No one would intentionally use more state variables than the minimum number needed to characterize the behavior of a dynamic system. In a complex system with unfamiliar physics, one may be tempted to write down differential equations for everything in sight and, in doing so, may write down more equations than are necessary. This will invariably result in an uncontrollable model for the system.

Observability

For the linear system given by Eqns (5.107), if the knowledge of the output y and the input u over a finite interval of time $[0, t_1]$ suffices to determine the state $\mathbf{x}(0) \triangleq \mathbf{x}^0$, the state $\mathbf{x}^0$ is said to be observable. If all initial states are observable, the system is said to be *completely observable*, or simply *observable* otherwise, the system is said to be *unobservable*.

The output of the system (5.107) is given by

$$y(t) = \mathbf{c}e^{\mathbf{A}t}\mathbf{x}^0 + \mathbf{c}\int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{b}u(\tau)d\tau + du(t)$$

The output and the input can be measured and used, so that the following signal $\eta(t)$ can be obtained from u and y.

$$\eta(t) \triangleq y(t) - \mathbf{c}\int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{b}u(\tau)d\tau - du(t) = \mathbf{c}e^{\mathbf{A}t}\mathbf{x}^0 \tag{5.109}$$

Premultiplying by $e^{\mathbf{A}^T t}\mathbf{c}^T$ and integrating from 0 to t_1 gives

$$\left\{\int_0^{t_1} e^{\mathbf{A}^T t}\mathbf{c}^T\mathbf{c}e^{\mathbf{A}t}dt\right\}\mathbf{x}^0 = \int_0^{t_1} e^{\mathbf{A}^T t}\mathbf{c}^T\eta(t)dt \tag{5.110}$$

When the signal $\eta(t)$ is available over a time interval $[0, t_1]$, and the system (5.107) is observable, then the initial state $\mathbf{x}^0$ can be uniquely determined from Eqn. (5.110).

From Eqn. (5.110) we see that complete observability of a system depends on **A** and **c**, and is independent of **b**. The observability of the system (5.107) is frequently referred to as the observability of the pair $\{\mathbf{A}, \mathbf{c}\}$.

Note that the system is said to be unobservable, although it may be 'observable in part'. Plants that are not completely observable can often be made observable by making more measurements (observability of multi-output systems will be discussed in Section 5.10). Alternately, one may examine feedback control schemes which do not require complete state feedback.

5.8.2 Controllability Tests

It is difficult to guess whether a system is controllable or not from the defining equation (5.108). Some simple mathematical tests which answer the question of controllability, have been developed. The following theorem gives two controllability tests.

Theorem 5.2 The necessary and sufficient condition for the system (5.107) to be completely controllable is given by any one of the following:

I. $$\mathbf{W}(0, t_1) = \int_0^{t_1} e^{-\mathbf{A}t}\mathbf{b}\,\mathbf{b}^T e^{-\mathbf{A}^T t}\,dt \tag{5.111}$$
is nonsingular.

II. The $n \times n$ *controllability matrix*
$$\mathbf{U} \triangleq [\mathbf{b} \quad \mathbf{Ab} \quad \mathbf{A}^2\mathbf{b} \cdots \mathbf{A}^{n-1}\mathbf{b}] \tag{5.112}$$
has rank equal to n, i.e., $\rho(\mathbf{U}) = n$.
Since Test II can be computed without integration, it allows the controllability of a system to be easily checked.

Proof of Test I

Sufficiency: If $\mathbf{W}(0, t_1)$ given in Eqn. (5.111) is nonsingular, the input
$$u(t) = -\mathbf{b}^T e^{-\mathbf{A}^T t}\,\mathbf{W}^{-1}(0, t_1)\mathbf{x}^0 \tag{5.113}$$
can be applied to the system. This input satisfies the condition given in Eqn. (5.108):

$$\int_0^{t_1} e^{-\mathbf{A}t}\mathbf{b}u(t)\,dt = -\int_0^{t_1} e^{-\mathbf{A}t}\mathbf{b}\mathbf{b}^T e^{-\mathbf{A}^T t}\mathbf{W}^{-1}(0, t_1)\mathbf{x}^0\,dt$$
$$= -\left\{\int_0^{t_1} e^{-\mathbf{A}t}\mathbf{b}\,\mathbf{b}^T e^{-\mathbf{A}^T t}dt\right\}\mathbf{W}^{-1}(0, t_1)\mathbf{x}^0 = -\mathbf{x}^0$$

Necessity: Assume that the system is controllable, though $\mathbf{W}(0, t_1)$ is singular for any t_1. Then, as per the results given in Eqns (5.8), the n rows of $e^{-\mathbf{A}t}\mathbf{b}$ are linearly dependent, i.e., there exists a nonzero $n \times 1$ vector $\boldsymbol{\alpha}$ such that
$$\boldsymbol{\alpha}^T e^{-\mathbf{A}t}\,\mathbf{b} = 0 \tag{5.114}$$

From the assumption of controllability, there exists an input u satisfying Eqn. (5.108); therefore, from Eqns (5.108) and (5.114),

$$-\boldsymbol{\alpha}^T\mathbf{x}^0 = \int_0^{t_1} \boldsymbol{\alpha}^T e^{-\mathbf{A}t}\mathbf{b}\, u(t)\, dt = 0 \tag{5.115}$$

holds for any initial state $\mathbf{x}^0$. By choosing $\mathbf{x}^0 = \boldsymbol{\alpha}$, Eqn. (5.115) gives (refer to Eqn. (5.6a)),

$$\boldsymbol{\alpha}^T\boldsymbol{\alpha} = [\|\boldsymbol{\alpha}\|]^2 = 0$$

This is true only for $\boldsymbol{\alpha} = \mathbf{0}$, which contradicts the nonzero property of $\boldsymbol{\alpha}$. Therefore, the nonsingularity of $\mathbf{W}(0, t_1)$ is proved.

Proof of Test II

Sufficiency: It is first assumed that though $\rho(\mathbf{U}) = n$, the system is not controllable, and by showing that this is a contradiction, the controllability of the system is proved.

By the above assumption,

$$\rho(\mathbf{U}) = n \quad \text{and} \quad \mathbf{W}(0, t_1) \text{ is singular.}$$

Therefore, Eqn. (5.114) holds, i.e.,

$$\boldsymbol{\alpha}^T e^{-\mathbf{A}t}\,\mathbf{b} = 0;\ t \geq 0,\ \boldsymbol{\alpha} \neq \mathbf{0}$$

Derivatives of the above equation at $t = 0$, yield (refer to Eqn. (5.87)),

$$\boldsymbol{\alpha}^T\mathbf{A}^k\mathbf{b} = 0;\ k = 0, 1, ..., (n-1)$$

which is equivalent to

$$\boldsymbol{\alpha}^T [\mathbf{b} \quad \mathbf{Ab} \cdots \mathbf{A}^{n-1}\mathbf{b}] = \boldsymbol{\alpha}^T\mathbf{U} = \mathbf{0}$$

Therefore, n rows of controllability matrix $\mathbf{U}$ are linearly dependent (refer to Eqn. (5.8a)). This contradicts the assumption that $\rho(\mathbf{U}) = n$; hence the system is completely controllable.

Necessity: It is assumed that the system is completely controllable but $\rho(\mathbf{U}) < n$. From this assumption, there exists nonzero vector $\boldsymbol{\alpha}$ satisfying

$$\boldsymbol{\alpha}^T\mathbf{U} = \mathbf{0}$$

or

$$\boldsymbol{\alpha}^T\mathbf{A}^k\,\mathbf{b} = 0;\ k = 0, 1, ..., (n-1) \tag{5.116a}$$

Also from the Cayley–Hamilton theorem, $e^{-\mathbf{A}t}$ can be expressed as a linear combination of $\mathbf{I}$, $\mathbf{A}$, ..., $\mathbf{A}^{n-1}$:

$$e^{-\mathbf{A}t} = \beta_0\mathbf{I} + \beta_1\mathbf{A} + \cdots + \beta_{n-1}\,\mathbf{A}^{n-1} \tag{5.116b}$$

From Eqns (5.116a) and (5.116b), we obtain

$$\boldsymbol{\alpha}^T e^{-\mathbf{A}t}\mathbf{b} = 0,\ t \geq 0,\ \boldsymbol{\alpha} \neq \mathbf{0}$$

and therefore (refer to Eqns (5.8)),

$$\int_0^{t_1} \boldsymbol{\alpha}^T e^{-\mathbf{A}t}\mathbf{b}\,\mathbf{b}^T\, e^{-\mathbf{A}^T t}\,\boldsymbol{\alpha}\, dt = \boldsymbol{\alpha}^T\mathbf{W}(0, t_1)\,\boldsymbol{\alpha} = 0$$

Since the system is completely controllable, $\mathbf{W}(0, t_1)$ should be nonsingular from Test I; this contradicts the assumption that $\boldsymbol{\alpha}$ is nonzero. Therefore, $\rho(\mathbf{U}) = n$.

Example 5.16

Recall the inverted pendulum of Example 5.15, shown in Fig. 5.16, in which the object is to apply a force $u(t)$ so that the pendulum remains balanced in the vertical position. We found the linearized equations governing the system to be

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

where

$$\mathbf{x} = [\theta \quad \dot{\theta} \quad z \quad \dot{z}]^T$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 16.3106 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1.0637 & 0 & 0 & 0 \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 0 \\ -1.4458 \\ 0 \\ 0.9639 \end{bmatrix}$$

$z(t)$ = horizontal displacement of the pivot on the cart; and

$\theta(t)$ = rotational angle of the pendulum.

To check the controllability of this system, we compute the controllability matrix **U**:

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab} \quad \mathbf{A}^2\mathbf{b} \quad \mathbf{A}^3\mathbf{b}] = \begin{bmatrix} 0 & -1.4458 & 0 & -23.5816 \\ -1.4458 & 0 & -23.5816 & 0 \\ 0 & 0.9639 & 0 & 1.5379 \\ 0.9639 & 0 & 1.5379 & 0 \end{bmatrix}$$

Since $|\mathbf{U}| = 420.4851$, **U** has full rank, and by Theorem 5.2, the system is completely controllable. Thus, if the angle θ departs from equilibrium by a small amount, a control always exists which will drive it back to zero.[10] Moreover, a control also exists which will drive both θ and z, as well as their derivatives, to zero.

It may be noted that Eqn. (5.113) suggests a control law to prove the sufficiency of the controllability test. It does not necessarily give an acceptable solution to the control problem. The open-loop control given by Eqn. (5.113) is normally, not acceptable. In Chapter 7, we will derive a state-feedback control law for the inverted pendulum. As we shall see, for such a control to exist, complete controllability of the plant is a *necessary requirement.*

Example 5.17

Consider the electrical network shown in Fig. 5.19. Differential equations governing the dynamics of this network, can be obtained by various standard methods. By use of nodal analysis, for example, we get

$$C_1 \frac{de_1}{dt} + \frac{e_1 - e_2}{R_3} + \frac{e_1 - e_0}{R_1} = 0$$

$$C_2 \frac{de_2}{dt} + \frac{e_2 - e_1}{R_3} + \frac{e_2 - e_0}{R_2} = 0$$

[10] This justifies the assumption that $\theta(t) \cong 0$, provided we choose an appropriate control strategy.

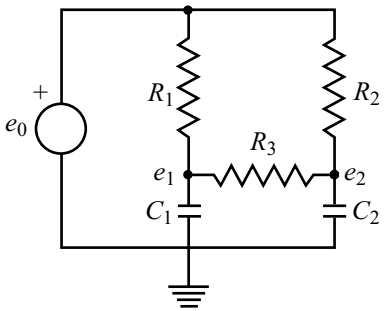


Fig. 5.19

The appropriate state variables for the network are the capacitor voltages e_1 and e_2. Thus, the state equations of the network are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}e_0$$

where $\mathbf{x} = [e_1 \quad e_2]^T$

$$\mathbf{A} = \begin{bmatrix} -\left(\frac{1}{R_1}+\frac{1}{R_3}\right)\frac{1}{C_1} & \frac{1}{R_3C_1} \\ \frac{1}{R_3C_2} & -\left(\frac{1}{R_2}+\frac{1}{R_3}\right)\frac{1}{C_2} \end{bmatrix}; \mathbf{b} = \begin{bmatrix} \frac{1}{R_1C_1} \\ \frac{1}{R_2C_2} \end{bmatrix}$$

The controllability matrix of the system is

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab}] = \begin{bmatrix} \frac{1}{R_1C_1} & -\frac{1}{(R_1C_1)^2}+\frac{1}{R_3C_1}\left(\frac{1}{R_2C_2}-\frac{1}{R_1C_1}\right) \\ \frac{1}{R_2C_2} & -\frac{1}{(R_2C_2)^2}+\frac{1}{R_3C_2}\left(\frac{1}{R_1C_1}-\frac{1}{R_2C_2}\right) \end{bmatrix}$$

We see that under the condition

$$R_1C_1 = R_2C_2$$

$\rho(\mathbf{U}) = 1$ and the system becomes 'uncontrollable'. This condition is the one required to balance the bridge, and in this case, the voltage across the terminals of R_3 cannot be influenced by the input e_0.

5.8.3 Observability Tests

The following theorem gives two observability tests.

Theorem 5.3 The necessary and sufficient condition for the system (5.107) to be completely observable, is given by any one of the following:

I. $$\mathbf{M}(0,t_1) = \int_0^{t_1} e^{\mathbf{A}^T t}\mathbf{c}^T\mathbf{c}e^{\mathbf{A}t}dt \tag{5.117}$$

is nonsingular.

II. The $n \times n$ *observability matrix*

$$\mathbf{V} \triangleq \begin{bmatrix} \mathbf{c} \\ \mathbf{cA} \\ \vdots \\ \mathbf{cA}^{n-1} \end{bmatrix} \tag{5.118}$$

has rank equal to n, i.e., $\rho(\mathbf{V}) = n$.

Proof Using the defining equation (5.110), this theorem can be proved in a manner similar to Theorem 5.2.

Example 5.18

We now return to the inverted pendulum of Example 5.16. Assuming that the only output variable to be measured is $\theta(t)$, the position of the pendulum, then the linearized equations governing the system are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

$$y = \mathbf{c}\mathbf{x}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 16.3106 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1.0637 & 0 & 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ -1.4458 \\ 0 \\ 0.9639 \end{bmatrix}$$

$$\mathbf{c} = [\,1 \quad 0 \quad 0 \quad 0\,]$$

The observability matrix

$$\mathbf{V} = \begin{bmatrix} \mathbf{c} \\ \mathbf{cA} \\ \mathbf{cA}^2 \\ \mathbf{cA}^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 16.3106 & 0 & 0 & 0 \\ 0 & 16.3106 & 0 & 0 \end{bmatrix}$$

$|\mathbf{V}| = 0$, and therefore, by Theorem 5.3, the system is not completely observable.

Consider now, the displacement $z(t)$ of the cart as the output variable. Then

$$\mathbf{c} = [0 \quad 0 \quad 1 \quad 0]$$

and the observability matrix

$$\mathbf{V} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1.0637 & 0 & 0 & 0 \\ 0 & -1.0637 & 0 & 0 \end{bmatrix}$$

$|\mathbf{V}| = 1.1315 \neq 0$; the system is, therefore, completely observable. The values of $\dot{z}(t)$, $\theta(t)$ and $\dot{\theta}(t)$ can all be determined by observing $z(t)$ over an arbitrary time interval. Observer design for the inverted-pendulum system is given in Chapter 7.

5.8.4 Invariance Property

It is recalled that the state variable model for a system is not unique, but depends on the choice of a set of state variables. A transformation

$$\mathbf{x}(t) = \mathbf{P}\,\overline{\mathbf{x}}\,(t);\ \mathbf{P} \text{ is a nonsingular constant matrix,}$$

results in the following alternative state variable model (refer to Eqns (5.22)) for the system (5.107):

$$\dot{\bar{\mathbf{x}}}(t) = \bar{\mathbf{A}}\,\bar{\mathbf{x}}(t) + \bar{\mathbf{b}}u(t);\ \bar{\mathbf{x}}(t_0) = \mathbf{P}^{-1}\mathbf{x}(t_0)$$
$$y(t) = \bar{\mathbf{c}}\,\bar{\mathbf{x}}(t) + du(t)$$

where

$$\bar{\mathbf{A}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P},\ \bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b},\ \bar{\mathbf{c}} = \mathbf{c}\mathbf{P}$$

The definition of a new set of internal state variables should, evidently, not affect the controllability and observability properties. This may be verified by evaluating the controllability and observability matrices of the transformed system.

I. $$\bar{\mathbf{U}} = [\bar{\mathbf{b}} \quad \bar{\mathbf{A}}\,\bar{\mathbf{b}} \quad \cdots \quad (\bar{\mathbf{A}})^{n-1}\,\bar{\mathbf{b}}] \tag{5.119a}$$

$$\bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{b}$$
$$\bar{\mathbf{A}}\,\bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}\mathbf{P}^{-1}\mathbf{b} = \mathbf{P}^{-1}\mathbf{A}\mathbf{b}$$
$$(\bar{\mathbf{A}})^2\,\bar{\mathbf{b}} = \bar{\mathbf{A}}(\bar{\mathbf{A}}\,\bar{\mathbf{b}}) = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}\mathbf{P}^{-1}\mathbf{A}\mathbf{b} = \mathbf{P}^{-1}\mathbf{A}^2\mathbf{b}$$
$$\vdots$$
$$(\bar{\mathbf{A}})^{n-1}\,\bar{\mathbf{b}} = \mathbf{P}^{-1}\mathbf{A}^{n-1}\mathbf{b}$$

Therefore,

$$\bar{\mathbf{U}} = [\mathbf{P}^{-1}\mathbf{b} \quad \mathbf{P}^{-1}\mathbf{A}\mathbf{b} \cdots \mathbf{P}^{-1}\mathbf{A}^{n-1}\mathbf{b}] = \mathbf{P}^{-1}\mathbf{U}$$

where $$\mathbf{U} = [\mathbf{b} \quad \mathbf{A}\mathbf{b} \cdots \mathbf{A}^{n-1}\mathbf{b}] \tag{5.119b}$$

Since $\mathbf{P}^{-1}$ is nonsingular,

$$\rho(\bar{\mathbf{U}}) = \rho(\mathbf{U}) \tag{5.119c}$$

II. A similar relationship can be shown for the observability matrices.

5.8.5 Controllability and Observability of State Variable Model in Jordan Canonical Form

If the system equations are known in Jordan canonical form, then one need not resort to controllability and observability tests given by Theorems 5.2 and 5.3. These properties can be determined almost by inspection of the system equations, as will be shown below.

Consider a SISO system with distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$. The Jordan canonical state model of this system is of the form

$$\dot{\mathbf{x}} = \mathbf{\Lambda}\mathbf{x} + \mathbf{b}u$$
$$y = \mathbf{c}\mathbf{x} + du \tag{5.120}$$

with $$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix};\ \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix};\ \mathbf{c} = [c_1\ c_2 \cdots c_n]$$

Theorem 5.4 The system (5.120) is completely controllable if, and only if, none of the elements of the column matrix **b** is zero, and (5.120) is completely observable if, and only if, none of the elements of the row matrix **c** is zero.

Proof The controllability matrix

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{\Lambda b} \cdots \mathbf{\Lambda}^{n-1}\mathbf{b}]$$

$$= \begin{bmatrix} b_1 & b_1\lambda_1 & \cdots & b_1\lambda_1^{n-1} \\ b_2 & b_2\lambda_2 & \cdots & b_2\lambda_2^{n-1} \\ \vdots & \vdots & & \vdots \\ b_n & b_n\lambda_n & \cdots & b_n\lambda_n^{n-1} \end{bmatrix}$$

$$|\mathbf{U}| = b_1 \times b_2 \times \cdots \times b_n \begin{vmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{n-1} \\ 1 & \lambda_2 & \cdots & \lambda_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_n & \cdots & \lambda_n^{n-1} \end{vmatrix} \neq 0 \text{ if } b_i \neq 0,\ i = 1, 2, ..., n.$$

This proves the first part of the theorem. The second part can be proved in a similar manner.[11]

5.9 EQUIVALENCE BETWEEN TRANSFER FUNCTION AND STATE VARIABLE REPRESENTATIONS

In frequency-domain analysis, it is tacitly assumed that the dynamic properties of a system are completely determined by the transfer function of the system. That this is not always the case is illustrated by the following examples.

Example 5.19

Consider the system

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{b}u$$
$$y = \mathbf{cx} \tag{5.121}$$

with $\mathbf{A} = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; $\mathbf{c} = [0 \quad 1]$

The controllability matrix

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab}] = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

[11] Refer to Gopal [105] for controllability and observability tests using Jordan canonical representation of systems with multiple eigenvalues

Since $\rho(\mathbf{U}) = 1$, the second-order system (5.121) is not completely controllable. The eigenvalues of matrix **A** are the roots of the characteristic equation

$$|s\mathbf{I} - \mathbf{A}| = \begin{vmatrix} s+2 & -1 \\ -1 & s+2 \end{vmatrix} = 0$$

The eigenvalues are obtained as $-1, -3$. The *modes* of the transient response are, therefore, e^{-t} and e^{-3t}.

The transfer function of the system (5.121) is calculated as

$$G(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} = [0 \quad 1] \begin{bmatrix} s+2 & -1 \\ -1 & s+2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$= [0 \quad 1] \begin{bmatrix} \dfrac{s+2}{(s+1)(s+3)} & \dfrac{1}{(s+1)(s+3)} \\ \dfrac{1}{(s+1)(s+3)} & \dfrac{s+2}{(s+1)(s+3)} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{s+1}$$

We find that because of pole-zero cancellation, both the eigenvalues of matrix **A** do not appear as poles in $G(s)$. The dynamic mode e^{-3t} of the system (5.121), does not show up in input-output characterization given by the transfer function $G(s)$. Note that the system under consideration is not a completely controllable system.

Example 5.20

Consider the system

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u \\ y &= \mathbf{c}\mathbf{x} \end{aligned} \tag{5.122}$$

with $\mathbf{A} = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$; $\mathbf{c} = [1 \quad -1]$

The observability matrix

$$\mathbf{V} = \begin{bmatrix} \mathbf{c} \\ \mathbf{cA} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -3 & 3 \end{bmatrix}$$

Since $\rho(\mathbf{V}) = 1$, the second-order system (5.122) is not completely observable.

The eigenvalues of matrix **A** are $-1, -3$. The transfer function of the system (5.122) is calculated as

$$G(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}$$

$$= [1 \quad -1] \begin{bmatrix} \dfrac{s+2}{(s+1)(s+3)} & \dfrac{1}{(s+1)(s+3)} \\ \dfrac{1}{(s+1)(s+3)} & \dfrac{s+2}{(s+1)(s+3)} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{s+3}$$

The dynamic mode e^{-t} of the system (5.122), does not show up in the input-output characterization given by the transfer function $G(s)$. Note that the system under consideration is not a completely observable system.

In the following, we give two specific state transformations to reveal the underlying structure imposed upon a system by its controllability and observability properties (for proof, refer to [105]). These results are then used to establish equivalence between transfer function and state variable representations.

Theorem 5.5 Consider an nth-order system

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u \\ y &= \mathbf{c}\mathbf{x} \end{aligned} \tag{5.123a}$$

Assume that

$$\rho(\mathbf{U}) = \rho[\mathbf{b} \quad \mathbf{A}\mathbf{b} \cdots \mathbf{A}^{n-1}\mathbf{b}] = m < n$$

Consider the equivalence transformation

$$\mathbf{x} = \mathbf{P}\,\bar{\mathbf{x}} = [\,\mathbf{P}_1 \;\; \mathbf{P}_2]\,\bar{\mathbf{x}} \tag{5.123b}$$

where $\mathbf{P}_1$ is composed of m linearly independent columns of $\mathbf{U}$, and $(n - m)$ columns of $\mathbf{P}_2$ are chosen arbitrarily so that matrix $\mathbf{P}$ is nonsingular.

The equivalence transformation (5.123b) transforms the system (5.123a) to the following form:

$$\begin{aligned} \begin{bmatrix} \dot{\bar{\mathbf{x}}}_1 \\ \dot{\bar{\mathbf{x}}}_2 \end{bmatrix} &= \begin{bmatrix} \bar{\mathbf{A}}_c & \bar{\mathbf{A}}_{12} \\ \mathbf{0} & \bar{\mathbf{A}}_{22} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{b}}_c \\ \mathbf{0} \end{bmatrix} u \\ &= \bar{\mathbf{A}}\;\bar{\mathbf{x}} + \bar{\mathbf{b}}u \\ y &= [\bar{\mathbf{c}}_1 \quad \bar{\mathbf{c}}_2] \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \end{bmatrix} = \bar{\mathbf{c}}\,\bar{\mathbf{x}} \end{aligned} \tag{5.123c}$$

where the m-dimensional subsystem

$$\dot{\bar{\mathbf{x}}}_1 = \bar{\mathbf{A}}_c\,\bar{\mathbf{x}}_1 + \bar{\mathbf{b}}_c\,u + \bar{\mathbf{A}}_{12}\,\bar{\mathbf{x}}_2$$

is controllable from u (the additional driving term $\bar{\mathbf{A}}_{12}\bar{\mathbf{x}}_2$ has no effect on controllability); the $(n - m)$ dimensional subsystem

$$\dot{\bar{\mathbf{x}}}_2 = \bar{\mathbf{A}}_{22}\,\bar{\mathbf{x}}_2$$

is not affected by the input and is, therefore, entirely uncontrollable.

This theorem shows that any system which is not completely controllable, can be decomposed into controllable and uncontrollable subsystems shown in Fig. 5.20. The state model (5.123c) is said to be in *controllability canonical form*.

In Section 5.4, it was shown that the characteristic equations and transfer functions of equivalent systems are identical. Thus, the set of eigenvalues of matrix $\mathbf{A}$ of system (5.123a) is same as the set of eigenvalues of matrix $\mathbf{A}$ of system (5.123c), which is a union of the subsets of eigenvalues of matrices $\bar{\mathbf{A}}_c$ and $\bar{\mathbf{A}}_{22}$. Also the transfer function of system (5.123a) must be the same as that of (5.123c). The transfer function of (5.123a) is calculated from Eqn. (5.123c) as[12]

---

[12] $\left[\begin{array}{c:c} \mathbf{A}_1 & \mathbf{A}_2 \\ \hdashline \mathbf{0} & \mathbf{A}_3 \end{array}\right] \left[\begin{array}{c:c} \mathbf{B}_1 & \mathbf{B}_2 \\ \hdashline \mathbf{B}_3 & \mathbf{B}_4 \end{array}\right] = \left[\begin{array}{c:c} \mathbf{I} & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{I} \end{array}\right]$

gives $\left[\begin{array}{c:c} \mathbf{B}_1 & \mathbf{B}_2 \\ \hdashline \mathbf{B}_3 & \mathbf{B}_4 \end{array}\right] = \left[\begin{array}{c:c} \mathbf{A}_1^{-1} & -\mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{A}_3^{-1} \\ \hdashline \mathbf{0} & \mathbf{A}_3^{-1} \end{array}\right]$

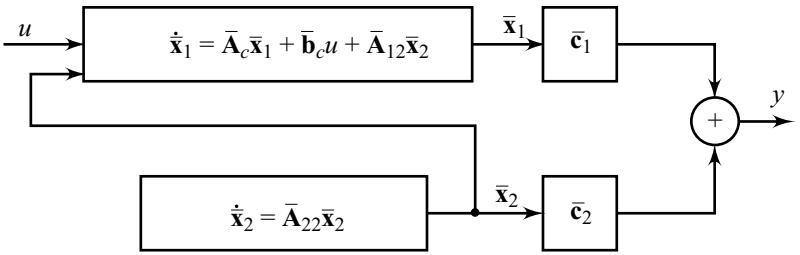


Fig. 5.20 The controllability canonical form of a state variable model

$$
\begin{aligned}
G(s) &= \begin{bmatrix}\bar{\mathbf{c}}_1 & \bar{\mathbf{c}}_2\end{bmatrix}\begin{bmatrix}s\mathbf{I}-\bar{\mathbf{A}}_c & -\bar{\mathbf{A}}_{12}\\ \mathbf{0} & s\mathbf{I}-\bar{\mathbf{A}}_{22}\end{bmatrix}^{-1}\begin{bmatrix}\bar{\mathbf{b}}_c\\ \mathbf{0}\end{bmatrix}\\
&= \begin{bmatrix}\bar{\mathbf{c}}_1 & \bar{\mathbf{c}}_2\end{bmatrix}\begin{bmatrix}(s\mathbf{I}-\bar{\mathbf{A}}_c)^{-1} & (s\mathbf{I}-\bar{\mathbf{A}}_c)^{-1}\bar{\mathbf{A}}_{12}(s\mathbf{I}-\bar{\mathbf{A}}_{22})^{-1}\\ \mathbf{0} & (s\mathbf{I}-\bar{\mathbf{A}}_{22})^{-1}\end{bmatrix}\\
&= \bar{\mathbf{c}}_1(s\mathbf{I}-\bar{\mathbf{A}}_c)^{-1}\bar{\mathbf{b}}_c
\end{aligned}
$$

Therefore, the input-output relationship for the system is dependent only on the controllable part of the system. We will refer to the eigenvalues of $\bar{\mathbf{A}}_c$ as *controllable poles* and the eigenvalues of $\bar{\mathbf{A}}_{22}$ as *uncontrollable poles*.

Only the controllable poles appear in the transfer function model; the uncontrollable poles are canceled by the zeros.

Theorem 5.6 Consider the nth-order system

$$
\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u\\
y &= \mathbf{c}\mathbf{x}
\end{aligned}
\tag{5.124a}
$$

Assume that

$$
\rho(\mathbf{V}) = \rho\begin{bmatrix}\mathbf{c}\\ \mathbf{cA}\\ \vdots\\ \mathbf{cA}^{n-1}\end{bmatrix} = l < n
$$

Consider the equivalence transformation

$$
\bar{\mathbf{x}} = \mathbf{Q}\mathbf{x} = \begin{bmatrix}\mathbf{Q}_1\\ \mathbf{Q}_2\end{bmatrix}\mathbf{x} \tag{5.124b}
$$

where $\mathbf{Q}_1$ is composed of l linearly independent rows of $\mathbf{V}$, $(n - l)$ rows of $\mathbf{Q}_2$ are chosen arbitrarily so that matrix $\mathbf{Q}$ is nonsingular.

The equivalence transformation (5.124b) transforms the system (5.124a) to the following form:

$$
\begin{bmatrix}\dot{\bar{\mathbf{x}}}_1\\ \dot{\bar{\mathbf{x}}}_2\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{A}}_0 & \mathbf{0}\\ \bar{\mathbf{A}}_{21} & \bar{\mathbf{A}}_{22}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}_1\\ \bar{\mathbf{x}}_2\end{bmatrix} + \begin{bmatrix}\bar{\mathbf{b}}_1\\ \bar{\mathbf{b}}_2\end{bmatrix}u = \bar{\mathbf{A}}\,\bar{\mathbf{x}} + \bar{\mathbf{b}}u \tag{5.124c}
$$

$$y = [\,\bar{\mathbf{c}}_0 \quad \mathbf{0}\,]\begin{bmatrix}\bar{\mathbf{x}}_1\\ \bar{\mathbf{x}}_2\end{bmatrix} = \bar{\mathbf{c}}\,\bar{\mathbf{x}}$$

where the l-dimensional subsystem

$$\dot{\bar{\mathbf{x}}}_1 = \bar{\mathbf{A}}_0\bar{\mathbf{x}}_1 + \bar{\mathbf{b}}_1 u$$
$$y = \bar{\mathbf{c}}_0\,\bar{\mathbf{x}}_1$$

is observable from y, and the $(n - l)$-dimensional subsystem

$$\dot{\bar{\mathbf{x}}}_2 = \bar{\mathbf{A}}_{22}\,\bar{\mathbf{x}}_2 + \bar{\mathbf{b}}_2\,u + \bar{\mathbf{A}}_{21}\,\bar{\mathbf{x}}_1$$

has no effect upon the output y, and is therefore entirely unobservable, i.e., nothing about $\bar{\mathbf{x}}_2$ can be inferred from output measurement.

This theorem shows that any system which is not completely observable, can be decomposed into the observable and unobservable subsystems shown in Fig. 5.21. The state model (5.124c) is said to be in *observability canonical form*.

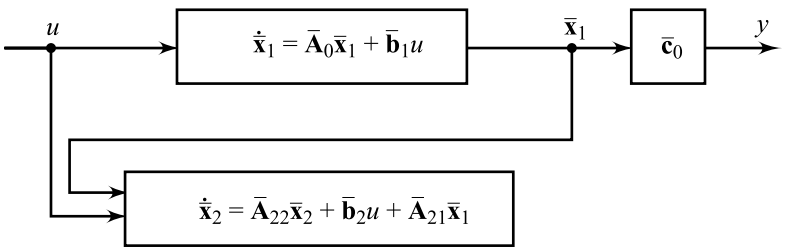


Fig. 5.21 The observability canonical form of a state variable model

Since systems (5.124a) and (5.124c) are equivalent, the set of eigenvalues of matrix $\mathbf{A}$ of system (5.124a) is same as the set of eigenvalues of matrix $\bar{\mathbf{A}}$ of system (5.124c), which is a union of the subsets of eigenvalues of matrices $\bar{\mathbf{A}}_0$ and $\bar{\mathbf{A}}_{22}$. The transfer function of the system (5.124a) may be calculated from (5.124c) as follows:

$$\begin{aligned} G(s) &= [\,\bar{\mathbf{c}}_0 \quad \mathbf{0}]\begin{bmatrix} s\mathbf{I} - \bar{\mathbf{A}}_0 & \mathbf{0} \\ -\bar{\mathbf{A}}_{21} & s\mathbf{I} - \bar{\mathbf{A}}_{22}\end{bmatrix}^{-1}\begin{bmatrix}\bar{\mathbf{b}}_1\\ \bar{\mathbf{b}}_2\end{bmatrix} \\ &= [\,\bar{\mathbf{c}}_0 \quad \mathbf{0}]\begin{bmatrix} (s\mathbf{I} - \bar{\mathbf{A}}_0)^{-1} & \mathbf{0} \\ (s\mathbf{I} - \bar{\mathbf{A}}_{22})^{-1}\,\bar{\mathbf{A}}_{21}(s\mathbf{I} - \bar{\mathbf{A}}_0)^{-1} & (s\mathbf{I} - \bar{\mathbf{A}}_{22})^{-1}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{b}}_1\\ \bar{\mathbf{b}}_2\end{bmatrix} \\ &= \bar{\mathbf{c}}_0(s\,\mathbf{I} - \bar{\mathbf{A}}_0)^{-1}\,\bar{\mathbf{b}}_1 \end{aligned} \tag{5.125}$$

which shows that the unobservable part of the system does not affect the input-output relationship. We will refer to the eigenvalues of $\bar{\mathbf{A}}_0$ as *observable poles* and the eigenvalues of $\bar{\mathbf{A}}_{22}$ as *unobservable poles*.

We now examine the use of state variable and transfer function models of a system to study its dynamic properties.

We know that a system is asymptotically stable if all the eigenvalues of the characteristic matrix $\mathbf{A}$ of its state variable model, are in the left half of the complex plane. Also, we know that a system is

(Bounded-Input Bounded-Output) BIBO stable if all the poles of its transfer function model are in the left half of the complex plane. Since, in general, the poles of the transfer function model of a system are a subset of the eigenvalues of the characteristic matrix **A** of the system, *asymptotic stability always implies BIBO stability*.

The reverse, however, may not always be true because the eigenvalues of the uncontrollable and/or unobservable part of the system are hidden from the BIBO stability analysis. These may lead to instability of a BIBO stable system. When a state variable model is both controllable and observable, all the eigenvalues of characteristic matrix **A** appear as poles in the corresponding transfer function. Therefore, *BIBO stability implies asymptotic stability only for completely controllable and completely observable system*.

To conclude, we may say that the transfer function model of a system represents its complete dynamics only if the system is both controllable and observable.

5.10 MULTIVARIABLE SYSTEMS

Many of the analysis results developed in earlier sections of this chapter for SISO systems have obvious extensions for MIMO systems.

Consider a general MIMO system shown in the block diagram of Fig. 5.22. The input variables are represented by u_1, u_2, ..., u_p, and the output variables by y_1, y_2, ..., y_q. The state, as in the case of SISO systems, is represented by variables x_1, x_2, ..., x_n.

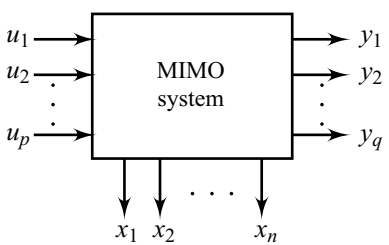


Fig. 5.22 A general MIMO system

The state variable model for a MIMO system takes the following form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t);\ \mathbf{x}(t_0) \triangleq \mathbf{x}^0 \tag{5.126a}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{5.126b}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix};\ \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & & \vdots \\ c_{q1} & c_{q2} & \cdots & c_{qn} \end{bmatrix},\ \mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1p} \\ d_{21} & d_{22} & \cdots & d_{2p} \\ \vdots & \vdots & & \vdots \\ d_{q1} & d_{q2} & \cdots & d_{qp} \end{bmatrix}$$

A, **B**, **C** and **D** are, respectively, $n \times n$, $n \times p$, $q \times n$, and $q \times p$ constant matrices; **x** is $n \times 1$ state vector, **u** is $p \times 1$ input vector, and **y** is $q \times 1$ output vector.

The solution of the state equation (5.126a) is given by (refer to Eqn. (5.100))

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\,\mathbf{B}\mathbf{u}(\tau)\,d\tau \tag{5.127a}$$

The output

$$\mathbf{y}(t) = \mathbf{C}\left[e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau\right] + \mathbf{D}\mathbf{u}(\tau) \tag{5.127b}$$

In the transform domain, the input-output behavior of the system (5.126) is determined entirely by the matrix (refer to Eqn. (5.28))

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{D} \tag{5.128a}$$

This matrix is called the *transfer function matrix* of system (5.126), and it has the property that the input $\mathbf{U}(s)$ and output $\mathbf{Y}(s)$ of Eqns (5.126) are related by

$$\underset{(q\times 1)}{\mathbf{Y}(s)} = \underset{(q\times p)}{\mathbf{G}(s)}\ \underset{(p\times 1)}{\mathbf{U}(s)} \tag{5.128b}$$

whenever $\mathbf{x}^0 = \mathbf{0}$.

In an expanded form, Eqn. (5.128b) can be written as

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \\ \vdots \\ Y_q(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) & \cdots & G_{1p}(s) \\ G_{21}(s) & G_{22}(s) & \cdots & G_{2p}(s) \\ \vdots & \vdots & & \vdots \\ G_{q1}(s) & G_{q2}(s) & \cdots & G_{qp}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \\ \vdots \\ U_p(s) \end{bmatrix}$$

The (i, j)th element $G_{ij}(s)$ of $\mathbf{G}(s)$ is the transfer function relating the ith output to the jth input.

Example 5.21

The scheme of Fig. 5.23 describes a simple concentration control process. Two concentrated solutions of some chemical with constant concentrations C_1 and C_2 are fed with flow rates $Q_1(t) = \bar{Q}_1 + q_1(t)$, and $Q_2(t) = \bar{Q}_2 + q_2(t)$, respectively, and are continuously mixed in the tank. The outflow from the mixing tank is at a rate $Q(t) = \bar{Q} + q(t)$ with concentration $C(t) = \bar{C} + c(t)$. Let it be assumed that stirring causes perfect mixing so that the concentration of the solution in the tank is uniform throughout, and equals that of the outflow. We shall also assume that the density remains constant.

Let $V(t) = \bar{V} + v(t)$ be the volume of the fluid in the tank.

The mass balance equations are

$$\frac{d}{dt}[\bar{V} + v(t)] = \bar{Q}_1 + q_1(t) + \bar{Q}_2 + q_2(t) - \bar{Q} - q(t) \tag{5.129a}$$

$$\frac{d}{dt}[\{\bar{C} + c(t)\}\{\bar{V} + v(t)\}] = C_1[\bar{Q}_1 + q_1(t)] + C_2[\bar{Q}_2 + q_2(t)] - [\bar{C} + c(t)][\bar{Q} + q(t)] \tag{5.129b}$$

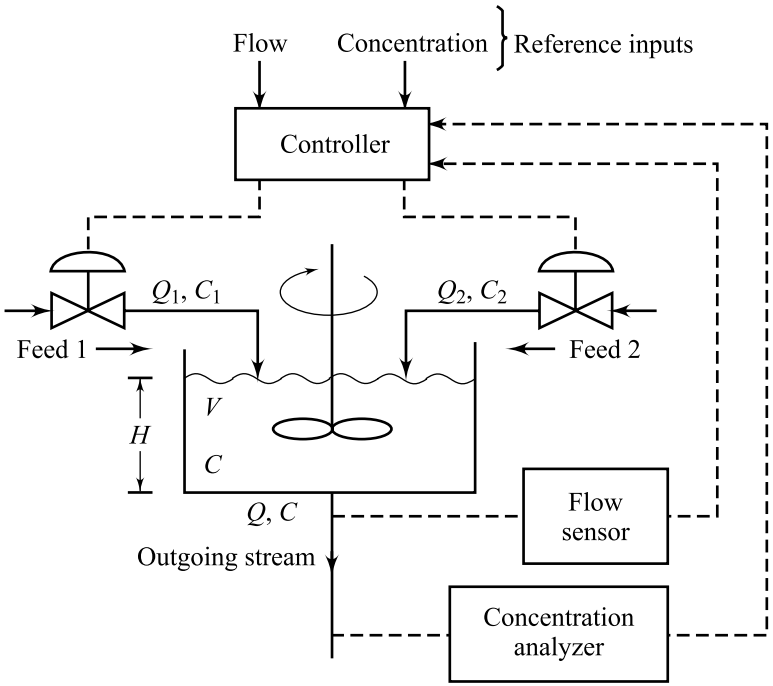


Fig. 5.23 A concentration control process

The flow $Q(t)$ is characterized by the turbulent flow relation

$$Q(t) = k\sqrt{H(t)} = k\sqrt{\frac{V(t)}{A}} \tag{5.130}$$

where $H(t) = \bar{H} + h(t)$ is the head of the liquid in the tank, A is the cross-sectional area of the tank and k is a constant.

The steady-state operation is described by the equations (obtained from Eqns (5.129) and (5.130))

$$0 = \bar{Q}_1 + \bar{Q}_2 - \bar{Q}$$

$$0 = C_1\bar{Q}_1 + C_2\bar{Q}_2 - \bar{C}\bar{Q}$$

$$\bar{Q} = k\sqrt{\frac{\bar{V}}{A}}$$

For small perturbations about the steadystate, Eqn. (5.130) can be linearized using Eqn. (5.11c):

$$Q(t) - \bar{Q} = \frac{k}{\sqrt{A}} \left.\frac{\partial\sqrt{V(t)}}{\partial V(t)}\right|_{V=\bar{V}} (V(t) - \bar{V})$$

or

$$q(t) = \frac{k}{2\bar{V}}\sqrt{\frac{\bar{V}}{A}}\, v(t) = \frac{\bar{Q}}{2\bar{V}}\, v(t)$$

From the foregoing equations, we obtain the following relations describing perturbations about steadystate:

$$\dot{v}(t) = q_1(t) + q_2(t) - \frac{1}{2}\frac{\bar{Q}}{\bar{V}}\, v(t) \tag{5.131a}$$

$$\bar{C}\dot{v}(t) + \bar{V}\dot{c}(t) = C_1\, q_1(t) + C_2\, q_2(t) - \frac{1}{2}\frac{\bar{C}\bar{Q}}{\bar{V}}\, v(t) - \bar{Q}\, c(t) \tag{5.131b}$$

(Second-order terms in perturbation variables have been neglected)

The hold-up time of the tank is

$$\tau = \frac{\bar{V}}{\bar{Q}}$$

Let us define

$$x_1(t) = v(t),\ x_2(t) = c(t),\ u_1(t) = q_1(t),\ u_2(t) = q_2(t),\ y_1(t) = q(t), \text{ and } y_2(t) = c(t)$$

In terms of these variables, we get the following state model from Eqns (5.131):

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -\frac{1}{2\tau} & 0 \\ 0 & -\frac{1}{\tau} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 & 1 \\ \frac{C_1 - \bar{C}}{\bar{V}} & \frac{C_2 - \bar{C}}{\bar{V}} \end{bmatrix} \mathbf{u}(t) \tag{5.132a}$$

$$\mathbf{y}(t) = \begin{bmatrix} \frac{1}{2\tau} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}(t) \tag{5.132b}$$

For the parameters

$\bar{Q}_1 = 10$ liters/sec, $\bar{Q}_2 = 20$ liters/sec, $C_1 = 9$ g-moles/liter, $C_2 = 18$ g-moles/liter, $\bar{V} = 1500$ liters, the state variable model becomes

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{5.133a}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \tag{5.133b}$$

with

$$\mathbf{A} = \begin{bmatrix} -0.01 & 0 \\ 0 & -0.02 \end{bmatrix};\ \mathbf{B} = \begin{bmatrix} 1 & 1 \\ -0.004 & 0.002 \end{bmatrix};\ \mathbf{C} = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$$

In the transform domain, the input-output behavior of the system is given by

$$\mathbf{Y}(s) = \mathbf{G}(s)\, \mathbf{U}(s)$$

where

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$$

For **A**, **B**, and **C** given by Eqns (5.133), we have

$$(s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s + 0.01 & 0 \\ 0 & s + 0.02 \end{bmatrix}$$

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dfrac{1}{s+0.01} & 0 \\ 0 & \dfrac{1}{s+0.02} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -0.004 & 0.002 \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{0.01}{s+0.01} & \dfrac{0.01}{s+0.01} \\ \dfrac{-0.004}{s+0.02} & \dfrac{0.002}{s+0.02} \end{bmatrix} \tag{5.134}$$

Controllability Test

The necessary and sufficient condition for the system (5.126) to be completely controllable, is that the $n \times np$ matrix

$$\mathbf{U} \triangleq [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \cdots \mathbf{A}^{n-1}\mathbf{B}] \tag{5.135}$$

has rank equal to n, i.e., $\rho(\mathbf{U}) = n$.

Observability Test

The necessary and sufficient condition for the system (5.126) to be completely observable, is that the $nq \times n$ matrix

$$\mathbf{V} \triangleq \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{5.136}$$

has rank equal to n, i.e., $\rho(\mathbf{V}) = n$.

Controllability and Observability of State Variable Model in Jordan Canonical Form

The controllability and observability properties can be determined by the inspection of the system equations in Jordan canonical form. A MIMO system with distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ has the following Jordan canonical state model:

$$\dot{\mathbf{x}} = \mathbf{\Lambda x} + \mathbf{Bu} \tag{5.137a}$$

$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du} \tag{5.137b}$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}; \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}; \mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ \vdots & \vdots & & \vdots \\ c_{q1} & c_{q2} & \cdots & c_{qn} \end{bmatrix}$$

The system (5.137) is completely controllable if, and only if, none of the rows of **B** matrix is a zero row, and (5.137) is completely observable if, and only if, none of the columns of **C** matrix is a zero column.

We have been using Jordan canonical structure only for systems with distinct eigenvalues. Refer to [105] for controllability and observability tests using Jordan canonical representation of systems with multiple eigenvalues.

Example 5.22

Consider the mixing-tank system discussed in Example 5.21. Suppose the feeds Q_1 and Q_2 have equal concentrations, i.e., $C_1 = C_2 = C_0$ (Fig. 5.23). Then the steady-state concentration in the tank is also C_0, and from Eqn. (5.132a) we have

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -\frac{1}{2\tau} & 0 \\ 0 & -\frac{1}{\tau} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{u}(t)$$

This state variable model is in Jordan canonical form. Since one row of the **B** matrix is a zero row, the system is not completely controllable. As is obvious from the Jordan canonical model, the input $\mathbf{u}(t)$ affects only the state variable $x_1(t)$, the incremental volume. The variable $x_2(t)$, the incremental concentration, has no connection with the input $\mathbf{u}(t)$.

If $C_1 \neq C_2$, the system is completely controllable.

REVIEW EXAMPLES

Review Example 5.1

A feedback system has a closed-loop transfer function

$$\frac{Y(s)}{R(s)} = \frac{10(s+4)}{s(s+1)(s+3)}$$

Construct the following three different state models for this system:

(a) One where the system matrix **A** is a diagonal matrix.
(b) One where **A** is in first companion form.
(c) One where **A** is in second companion form.

Solution

(a) The given transfer function can be expressed as

$$\frac{Y(s)}{R(s)} = \frac{10(s+4)}{s(s+1)(s+3)} = \frac{40/3}{s} + \frac{-15}{s+1} + \frac{5/3}{s+3}$$

Therefore,

$$Y(s) = \frac{40/3}{s} R(s) + \frac{-15}{s+1} R(s) + \frac{5/3}{s+3} R(s)$$

Let $X_1(s) = \dfrac{40/3}{s} R(s)$; this gives $\dot{x}_1 = \dfrac{40}{3} r$

$X_2(s) = \dfrac{-15}{s+1} R(s)$; this gives $\dot{x}_2 + x_2 = -15r$

$X_3(s) = \dfrac{5/3}{s+3} R(s)$; this gives $\dot{x}_3 + 3x_3 = \dfrac{5}{3} r$

In terms of x_1, x_2 and x_3, the output $y(t)$ is given by

$$y(t) = x_1(t) + x_2(t) + x_3(t)$$

A state variable formulation, for the given transfer function, is defined by the following matrices:

$$\mathbf{\Lambda} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -3 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 40/3 \\ -15 \\ 5/3 \end{bmatrix}; \mathbf{c} = [1 \quad 1 \quad 1]; d = 0$$

Note that the coefficient matrix **A** is diagonal, and the state model is in Jordan canonical form.

We now construct two state models for the given transfer function in companion form. To do this, we express the transfer function as

$$\frac{Y(s)}{R(s)} = \frac{10(s+4)}{s(s+1)(s+3)} = \frac{10s+40}{s^3+4s^2+3s} = \frac{\beta_0 s^3 + \beta_1 s^2 + \beta_2 s + \beta_3}{s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3};$$

$$\beta_0 = \beta_1 = 0, \beta_2 = 10, \beta_3 = 40, \alpha_1 = 4, \alpha_2 = 3, \alpha_3 = 0$$

(b) With reference to Eqns (5.54), we obtain the following state model in the first companion form:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -3 & -4 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{c} = [40 \quad 10 \quad 0]; d = 0$$

(c) With reference to Eqns (5.56), the state model in second companion form becomes

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -3 \\ 0 & 1 & -4 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 40 \\ 10 \\ 0 \end{bmatrix}, \mathbf{c} = [0 \quad 0 \quad 1]; d = 0$$

Review Example 5.2

A linear time-invariant system is characterized by the homogeneous state equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

(a) Compute the solution of the homogeneous equation assuming the initial state vector

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Employ both the Laplace transform method and the canonical transformation method.

(b) Consider now that the system has a forcing function and is represented by the following nonhomogeneous state equation:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

where u is a unit-step input.

Compute the solution of this equation assuming initial conditions of part (a).

Solution

(a) Since

$$(s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s & -1 \\ 0 & s+2 \end{bmatrix}$$

we obtain

$$(s\mathbf{I} - \mathbf{A})^{-1} = \begin{bmatrix} \dfrac{1}{s} & \dfrac{1}{s(s+2)} \\ 0 & \dfrac{1}{s+2} \end{bmatrix}$$

Hence

$$e^{\mathbf{A}t} = \mathscr{L}^{-1}\left[(s\mathbf{I} - \mathbf{A})^{-1}\right] = \begin{bmatrix} 1 & \frac{1}{2}(1 - e^{-2t}) \\ 0 & e^{-2t} \end{bmatrix}$$

To obtain the state transition matrix $e^{\mathbf{A}t}$ by the canonical transformation method, we compute the eigenvalues and eigenvectors of matrix **A**. The roots of the characteristic equation

$$|\lambda\mathbf{I} - \mathbf{A}| = 0$$

are $\lambda_1 = 0$, and $\lambda_2 = -2$. These are the eigenvalues of matrix **A**. Eigenvectors corresponding to the distinct eigenvalues λ_i, may be obtained from the nonzero columns of $adj(\lambda_i\mathbf{I} - \mathbf{A})$.

For the given **A** matrix

$$adj(\lambda_i\mathbf{I} - \mathbf{A}) = \begin{bmatrix} \lambda_i + 2 & 1 \\ 0 & \lambda_i \end{bmatrix}$$

For $\lambda_1 = 0$, $\quad adj(\lambda_1\mathbf{I} - \mathbf{A}) = \begin{bmatrix} 2 & 1 \\ 0 & 0 \end{bmatrix}$

The eigenvector $\mathbf{v}_1$ corresponding to the eigenvalue λ_1 is, therefore, given by

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

For $\lambda_2 = -2$, $\quad adj\,(\lambda_2\mathbf{I} - \mathbf{A}) = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix}$

The eigenvector $\mathbf{v}_2$ corresponding to the eigenvalue λ_2 is given by

$$\mathbf{v}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

The transformation matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix}$$

gives

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{\Lambda} = \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix}$$

The state transition matrix (refer to Eqn. (5.93))

$$e^{\mathbf{A}t} = \mathbf{P}e^{\mathbf{\Lambda}t}\,\mathbf{P}^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix}\begin{bmatrix} e^{0} & 0 \\ 0 & e^{-2t} \end{bmatrix}\begin{bmatrix} 1 & \frac{1}{2} \\ 0 & -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2}(1-e^{-2t}) \\ 0 & e^{-2t} \end{bmatrix}$$

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(b) $$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{b}u(\tau)dt$$

Now

$$\int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{b}u(\tau)d\tau = \begin{bmatrix} \frac{1}{2}\int_0^t [1-e^{-2(t-\tau)}]d\tau \\ \int_0^t e^{-2(t-\tau)}d\tau \end{bmatrix} = \begin{bmatrix} -\frac{1}{4}+\frac{1}{2}t+\frac{1}{4}e^{-2t} \\ \frac{1}{2}(1-e^{-2t}) \end{bmatrix}$$

Therefore,

$$x_1(t) = -\tfrac{1}{4} + \tfrac{1}{2}t + \tfrac{1}{4}e^{-2t}$$
$$x_2(t) = \tfrac{1}{2}\,(1 - e^{-2t})$$

Review Example 5.3

Given

$$\underset{n\times n}{\mathbf{\Lambda}} = \begin{bmatrix} \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & \lambda_1 \end{bmatrix}$$

Compute $e^{\mathbf{\Lambda}t}$ using the Cayley–Hamilton technique.

Solution Equations (5.98) outline the procedure of evaluation of matrix exponential using the Cayley–Hamilton technique.

The matrix $\mathbf{\Lambda}$ has n eigenvalues at $\lambda = \lambda_1$. To evaluate $f(\mathbf{\Lambda}) = e^{\mathbf{\Lambda}t}$, we define (refer to Eqn. (5.98b)) the polynomial $g(\lambda)$ as

$$g(\lambda) = \beta_0 + \beta_1\lambda + \cdots + \beta_{n-1}\lambda^{n-1}$$

This polynomial may be rearranged as

$$g(\lambda) = b_0 + b_1(\lambda - \lambda_1) + \cdots + b_{n-1}(\lambda - \lambda_1)^{n-1}$$

The coefficients $b_0, b_1, \ldots, b_{n-1}$ are given by the following equations (refer to Eqns (5.98c)):

$$f(\lambda_1) = g(\lambda_1)$$

$$\left.\frac{d}{d\lambda} f(\lambda)\right|_{\lambda=\lambda_1} = \left.\frac{d}{d\lambda} g(\lambda)\right|_{\lambda=\lambda_1}$$

$$\vdots$$

$$\left.\frac{d^{n-1}}{d\lambda^{n-1}} f(\lambda)\right|_{\lambda=\lambda_1} = \left.\frac{d^{n-1}}{d\lambda^{n-1}} g(\lambda)\right|_{\lambda=\lambda_1}$$

Solving, we get

$$b_0 = e^{\lambda_1 t}$$

$$b_1 = \frac{t}{1!}e^{\lambda_1 t}$$

$$b_2 = \frac{t^2}{2!}e^{\lambda_1 t}$$

$$\vdots$$

$$b_{n-1} = \frac{t^{n-1}}{(n-1)!}e^{\lambda_1 t}$$

Therefore,

$$e^{\mathbf{\Lambda}t} = b_0\mathbf{I} + b_1(\mathbf{\Lambda} - \lambda_1\mathbf{I}) + \cdots + b_{n-1}(\mathbf{\Lambda} - \lambda_1\mathbf{I})^{n-1}$$

$$(\mathbf{\Lambda} - \lambda_1\mathbf{I}) = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$(\mathbf{\Lambda} - \lambda_1\mathbf{I})(\mathbf{\Lambda} - \lambda_1\mathbf{I}) = \begin{bmatrix} 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\vdots$$

$$e^{\Lambda t} = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_{n-1} \\ 0 & b_0 & b_1 & \cdots & b_{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & b_0 \end{bmatrix}$$

$$= \begin{bmatrix} e^{\lambda_1 t} & te^{\lambda_1 t} & t^2 e^{\lambda_1 t}/2! & \cdots & t^{n-1} e^{\lambda_1 t}/(n-1)! \\ 0 & e^{\lambda_1 t} & te^{\lambda_1 t} & \cdots & t^{n-2} e^{\lambda_1 t}/(n-2)! \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & e^{\lambda_1 t} \end{bmatrix}$$

Review Example 5.4

The motion of a satellite in the equatorial (r, θ) plane is given by [122] the state equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3\omega^2 & 0 & 0 & 2\omega \\ 0 & 0 & 0 & 1 \\ 0 & -2\omega & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \mathbf{Ax} + \mathbf{Bu}$$

where ω is the angular frequency of the satellite in circular, equatorial orbit; $x_1(t)$ and $x_3(t)$ are, respectively, the deviations in position variables $r(t)$ and $\theta(t)$ of the satellite; and $x_2(t)$ and $x_4(t)$ are, respectively, the deviations in velocity variables $\dot{r}(t)$ and $\dot{\theta}(t)$. The inputs $u_1(t)$ and $u_2(t)$ are the thrusts u_r and u_θ in the radial and tangential directions, respectively, applied by small rocket engines or gas jets ($\mathbf{u} = \mathbf{0}$ when $\mathbf{x} = \mathbf{0}$).

(a) Prove that the system is completely controllable.
(b) Suppose that the tangential thruster becomes inoperable. Determine the controllability of the system with the radial thruster alone.
(c) Suppose that the radial thruster becomes inoperable. Determine the controllability of the system with the tangential thruster alone.
(d) Prove that the system is completely observable from radial ($x_1 = r$) and tangential ($x_3 = \theta$) position measurements.
(e) Suppose that the tangential measuring device becomes inoperable. Determine the observability of the system from radial position measurement alone.
(f) Suppose that the radial measurements are lost. Determine the observability of the system from tangential position measurement alone.

Solution

(a) The controllability matrix

$$\mathbf{U} = [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \quad \mathbf{A}^3\mathbf{B}]$$

Consider the matrix:

$$\mathbf{U}_1 = [\mathbf{B} \quad \mathbf{AB}] = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 2\omega \\ 0 & 0 & 0 & 1 \\ 0 & 1 & -2\omega & 0 \end{bmatrix}$$

$$|\mathbf{U}_1| = -1$$

Therefore, $\rho(\mathbf{U}_1) = \rho(\mathbf{U}) = 4$; the system is completely controllable.

(b) With $u_2 = 0$, the **B** matrix becomes

$$\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The controllability matrix

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab} \quad \mathbf{A}^2\mathbf{b} \quad \mathbf{A}^3\mathbf{b}] = \begin{bmatrix} 0 & 1 & 0 & -\omega^2 \\ 1 & 0 & -\omega^2 & 0 \\ 0 & 0 & -2\omega & 0 \\ 0 & -2\omega & 0 & 2\omega^3 \end{bmatrix}$$

$$|\mathbf{U}| = -\begin{vmatrix} 1 & 0 & -\omega^2 \\ 0 & -2\omega & 0 \\ -2\omega & 0 & 2\omega^3 \end{vmatrix} = -[-2\omega(2\omega^3 - 2\omega^3)] = 0$$

Therefore, $\rho(\mathbf{U}) < 4$, and the system is not completely controllable with u_1 alone.

(c) With $u_1 = 0$, the **B** matrix becomes

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The controllability matrix

$$\mathbf{U} = \begin{bmatrix} 0 & 0 & 2\omega & 0 \\ 0 & 2\omega & 0 & -2\omega^3 \\ 0 & 1 & 0 & -4\omega^2 \\ 1 & 0 & -4\omega^2 & 0 \end{bmatrix}$$

$$|\mathbf{U}| = 2\omega \begin{vmatrix} 0 & 2\omega & -2\omega^3 \\ 0 & 1 & -4\omega^2 \\ 1 & 0 & 0 \end{vmatrix} = -12\omega^4 \neq 0$$

Therefore, $\rho(\mathbf{U}) = 4$, and the system is completely controllable with u_2 alone.

(d) The observability matrix

$$\mathbf{V} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \mathbf{CA}^3 \end{bmatrix}$$

Taking radial and tangential position measurements as the outputs, we have

$$y_1 = x_1; y_2 = x_3$$

or

$$\mathbf{y} = \mathbf{Cx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}$$

Consider the matrix

$$\mathbf{V}_1 = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$|\mathbf{V}_1| \neq 0$$

Therefore, $\rho(\mathbf{V}_1) = \rho(\mathbf{V}) = 4$, and the system is completely observable.

(e) With $x_3 = 0$, the **C** matrix becomes

$$\mathbf{c} = [1 \quad 0 \quad 0 \quad 0]$$

The observability matrix

$$\mathbf{V} = \begin{bmatrix} \mathbf{c} \\ \mathbf{cA} \\ \mathbf{cA}^2 \\ \mathbf{cA}^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3\omega^2 & 0 & 0 & 2\omega \\ 0 & -\omega^2 & 0 & 0 \end{bmatrix}$$

$$|\mathbf{V}| = 0$$

Therefore, $\rho(\mathbf{V}) < 4$, and the system is not completely observable from $y_1 = x_1$ alone.

(f) With $x_1 = 0$, the **C** matrix becomes

$$\mathbf{c} = [0 \quad 0 \quad 1 \quad 0]$$

The observability matrix

$$\mathbf{V} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -2\omega & 0 & 0 \\ -6\omega^3 & 0 & 0 & -4\omega^2 \end{bmatrix}$$

$$|\mathbf{V}| = -12\omega^4 \neq 0$$

Therefore, $\rho(\mathbf{V}) = 4$, and the system is completely observable from $y_2 = x_3$ alone.

PROBLEMS

5.1 Figure P5.1 shows a control scheme for controlling the azimuth angle of a rotating antenna. The plant consists of an armature-controlled dc motor with dc generator used as an amplifier. The parameters of the plant are given below.

Motor torque constant, $K_T = 1.2$ newton-m/amp
Motor back emf constant, $K_b = 1.2$ V/(rad/sec)
Generator gain constant, $K_g = 100$ V/amp
Motor to load gear ratio, $n = (\dot{\theta}_L / \dot{\theta}_M) = 1/2$

$R_f = 21\ \Omega$, $L_f = 5$H, $R_g = 9\ \Omega$, $L_g = 0.06$ H, $R_a = 10\ \Omega$, $L_a = 0.04$ H,
$J = 1.6$ newton-m/(rad/sec^2), $B = 0.04$ newton-m/(rad/sec), motor inertia and friction are negligible.

Taking physically meaningful and measurable variables as state variables, derive a state model for the system.

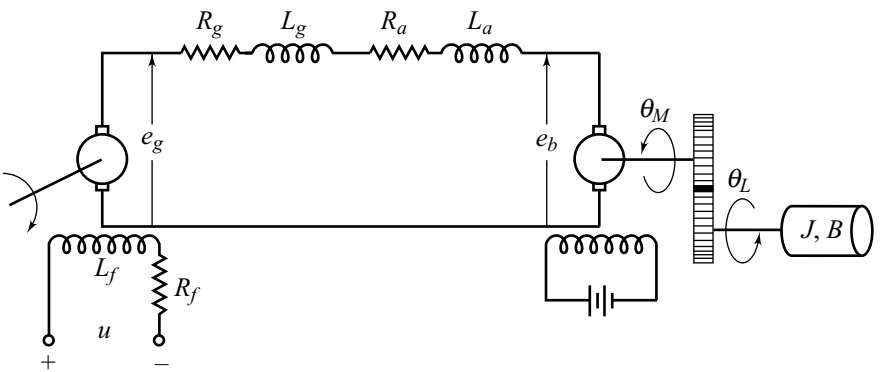


Fig. P5.1

5.2 Figure P5.2 shows a position control system with state variable feedback. The plant consists of a field-controlled dc motor with a dc amplifier. The parameters of the plant are given below.

Amplifier gain, $K_A = 50$ volt/volt
Motor field resistance, $R_f = 99\ \Omega$
Motor field inductance, $L_f = 20$ H
Motor torque constant, $K_T = 10$ newton-m/amp
Moment of inertia of load, $J = 0.5$ newton-m/(rad/sec^2)
Coefficient of viscous friction of load, $B = 0.5$ newton-m/(rad/sec)
Motor inertia and friction are negligible.

Taking $x_1 = \theta$, $x_2 = \dot{\theta}$, and $x_3 = i_f$ as the state variables, $u = e_f$ as the input, and $y = \theta$ as the output, derive a state variable model for the plant.

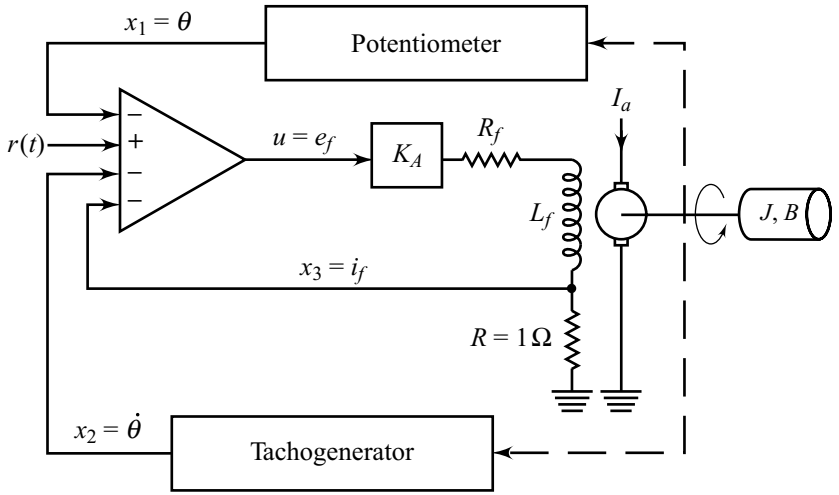


Fig. P5.2

5.3 Figure P5.3 shows the block diagram of a motor-driven, single-link robot manipulator with position and velocity feedback. The drive motor is an armature-controlled dc motor; e_a is armature voltage, i_a is armature current, θ_M is the motor shaft position and $\dot{\theta}_M$ is motor shaft velocity. θ_L is the position of the robot arm.

Taking θ_M, $\dot{\theta}_M$ and i_a as state variables, derive a state model for the feedback system.

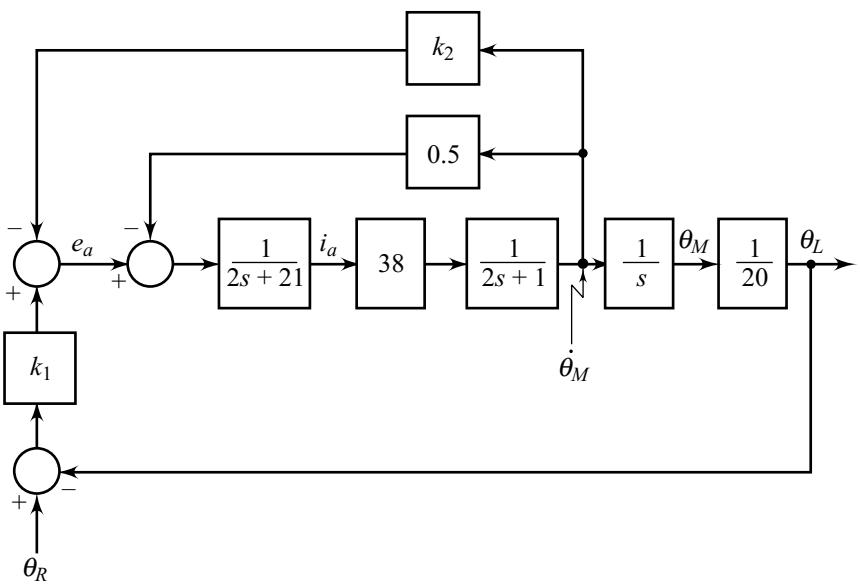


Fig. P5.3

5.4 Figure P5.4 shows the block diagram of a speed control system with state variable feedback. The drive motor is an armature-controlled dc motor with armature resistance R_a, armature inductance

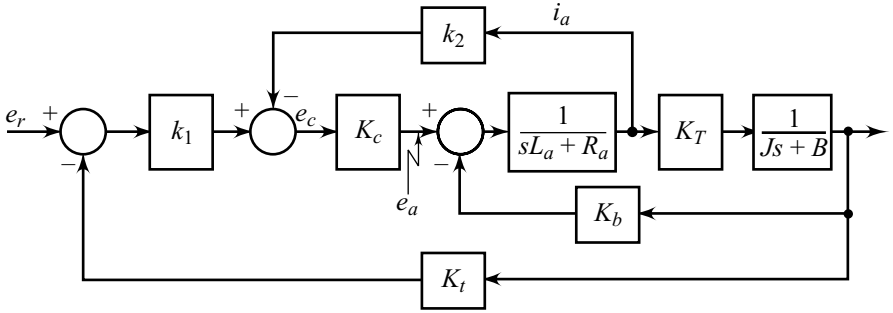


Fig. P5.4

L_a, motor torque constant K_T, inertia referred to motor shaft J, viscous friction coefficient referred to motor shaft B, back emf constant K_b, and tachogenerator constant K_t. The applied armature voltage is controlled by a three-phase full-converter. We have assumed a linear relationship between the control voltage e_c and the armature voltage e_a; e_r is the reference voltage corresponding to the desired speed.

Taking $x_1 = \omega$ (speed) and $x_2 = i_a$ (armature current) as the state variables, $u = e_r$ as the input, and $y = \omega$ as the output, derive a state variable model for the feedback system.

5.5 Consider the system

$$\dot{\mathbf{x}} = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = [1 \quad 0]\, \mathbf{x}$$

A similarity transformation is defined by

$$\mathbf{x} = \mathbf{P}\bar{\mathbf{x}} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \bar{\mathbf{x}}$$

(a) Express the state model in terms of the states $\bar{\mathbf{x}}(t)$.

(b) Draw state diagrams in signal-flow graph form for the state models in $\mathbf{x}(t)$ and $\bar{\mathbf{x}}(t)$.

(c) Show by Mason's gain formula that the transfer functions for the two state diagrams in (b) are equal.

5.6 Consider a double-integrator plant described by the differential equation

$$\frac{d^2\theta(t)}{dt^2} = u(t)$$

(a) Develop a state equation for this system with u as the input, and θ and $\dot{\theta}$ as the state variables x_1 and x_2, respectively.

(b) A similarity transformation is defined as

$$\mathbf{x} = \mathbf{P}\,\bar{\mathbf{x}} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \bar{\mathbf{x}}$$

Express the state equation in terms of the states $\bar{\mathbf{x}}(t)$.

(c) Show that the eigenvalues of the system matrices of the two state equations in (a) and (b), are equal.

5.7 A system is described by the state equation

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u;\ \mathbf{x}(0) = \mathbf{x}^0$$

Using the Laplace transform technique, transform the state equation into a set of linear algebraic equations in the form

$$\mathbf{X}(s) = \mathbf{G}(s)\mathbf{x}^0 + \mathbf{H}(s)U(s)$$

5.8 Give a block diagram for the programming of the system of Problem 5.7 on an analog computer.

5.9 The state diagram of a linear system is shown in Fig. P5.9. Assign the state variables, and write the dynamic equations of the system.

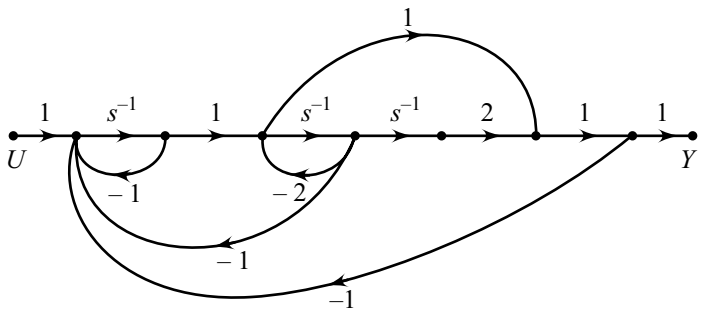


Fig. P5.9

5.10 Construct a state model for the system of Fig. P5.10.

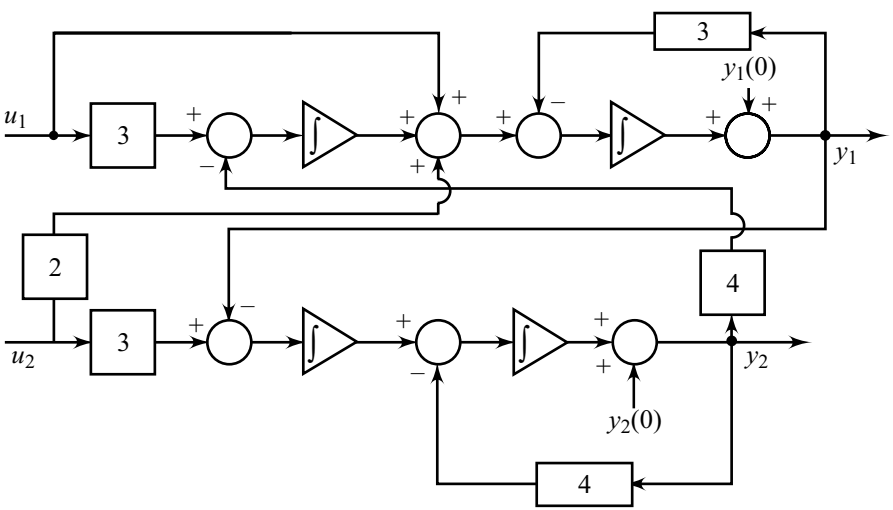


Fig. P5.10

5.11 Derive transfer functions corresponding to the following state models:

(a) $\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u; \; y = [1 \quad 0] \, \mathbf{x}$ (b) $\dot{\mathbf{x}} = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u; \; y = [1 \quad 0] \, \mathbf{x}$

5.12 Derive the transfer function matrix corresponding to the following state model, using resolvent algorithm.

$$\dot{\mathbf{x}} = \begin{bmatrix} 2 & -1 & 0 \\ 1 & 1 & 2 \\ -1 & 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{u}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \mathbf{x}$$

5.13 Figure P5.13 shows the block diagram of a control system with state variable feedback and integral control. The plant model is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -3 & 2 \\ 4 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$y = [0 \quad 1] \, \mathbf{x}$$

(a) Derive a state model of the feedback system.

(b) Derive the transfer function $Y(s)/R(s)$.

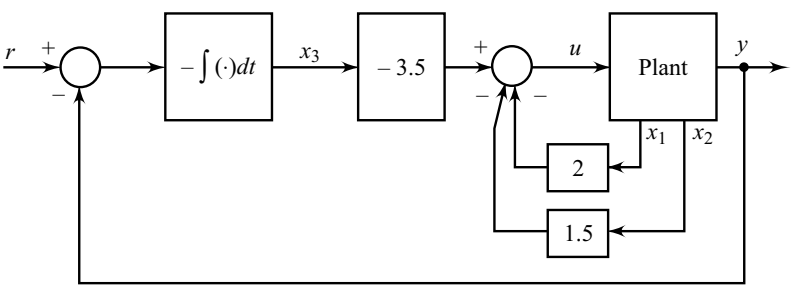


Fig. P5.13

5.14 Construct state models for the systems of Fig. P5.14a and Fig. P5.14b, taking outputs of simple lag blocks as state variables.

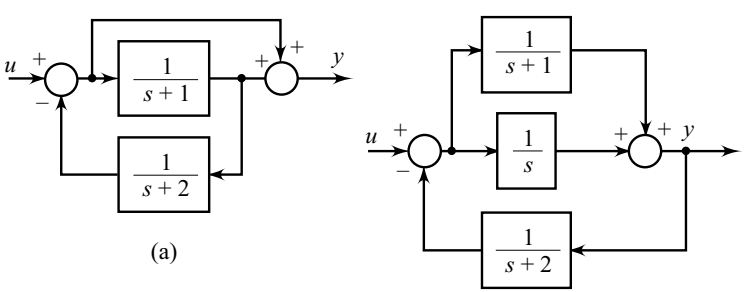


Fig. P5.14

5.15 Derive a state model for the two-input, two-output feedback control system shown in Fig. P5.15. Take outputs of simple lags as state variables.

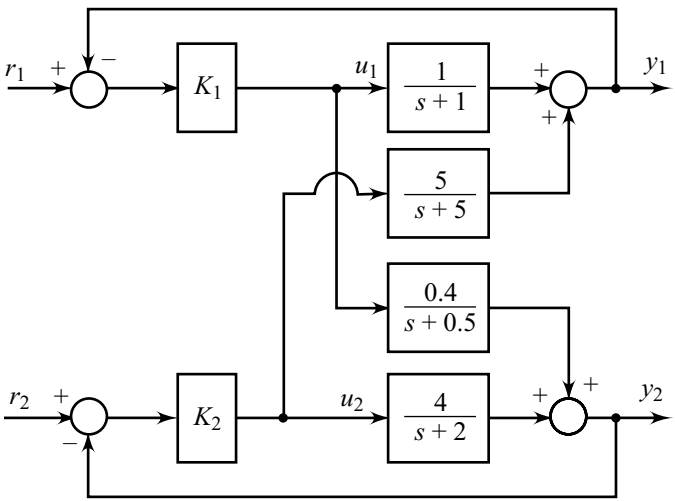


Fig. P5.15

5.16 Construct state models for the following transfer functions. Obtain different canonical form for each system.

(i) $\dfrac{s+3}{s^2+3s+2}$ (ii) $\dfrac{5}{(s+1)^2(s+2)}$ (iii) $\dfrac{s^3+8s^2+17s+8}{(s+1)(s+2)(s+3)}$

Give block diagrams for the analog computer simulation of these transfer functions.

5.17 Construct state models for the following differential equations. Obtain a different canonical form for each system.

(i) $\dddot{y}+3\ddot{y}+2\dot{y} = \dot{u} + u$ (ii) $\dddot{y}+6\ddot{y}+11\dot{y}+6y = u$

(iii) $\dddot{y}+6\ddot{y}+11\dot{y}+6y = \dddot{u}+8\ddot{u}+17\dot{u}+8u$

5.18 Derive two state models for the system with transfer function

$$\frac{Y(s)}{U(s)} = \frac{50(1+s/5)}{s(1+s/2)(1+s/50)}$$

(a) One for which the system matrix is a companion matrix.

(b) One for which the system matrix is diagonal.

5.19 (a) Obtain state variable model in Jordan canonical form for the system with transfer function

$$\frac{Y(s)}{U(s)} = \frac{2s^2+6s+5}{(s+1)^2(s+2)}$$

(b) Find the response $y(t)$ to a unit-step input using the state variable model in (*a*).

(c) Give a block diagram for analog computer simulation of the transfer function.

5.20 Find the eigenvalues and eigenvectors for the following matrices:

(i) $\begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$ (ii) $\begin{bmatrix} -3 & 2 \\ -1 & 0 \end{bmatrix}$ (iii) $\begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 2 \\ -12 & -7 & -6 \end{bmatrix}$

5.21 (a) If $\lambda_1, \lambda_2, \ldots, \lambda_n$ are distinct eigenvalues of

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_n & -\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_1 \end{bmatrix}$$

prove that the matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \cdots & \lambda_n^2 \\ \vdots & \vdots & & \vdots \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \cdots & \lambda_n^{n-1} \end{bmatrix}$$

transforms **A** into Jordan canonical form.

(b) Using the result in (a), find the eigenvalues and eigenvectors of the following matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -24 & -26 & -9 \end{bmatrix}$$

5.22 Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -4 & -3 \end{bmatrix}$$

(a) Suggest a transformation matrix **P** such that $\mathbf{\Lambda} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$ is in Jordan canonical form.

(b) Matrix $\mathbf{\Lambda}$ in (a) has complex elements. Real arithmetic is often preferable, and can be achieved by further transformation. Suggest a transformation matrix **Q** such that $\mathbf{Q}^{-1}\mathbf{\Lambda}\mathbf{Q}$ has all real elements.

5.23 Given the system

$$\dot{\mathbf{x}} = \begin{bmatrix} -4 & 3 \\ -6 & 5 \end{bmatrix} \mathbf{x} = \mathbf{A}\mathbf{x}$$

Determine eigenvalues and eigenvectors of matrix **A**, and use these results to find the state transition matrix.

5.24 Using Laplace transform method, find the matrix exponential $e^{\mathbf{A}t}$ for

(a) $\mathbf{A} = \begin{bmatrix} 0 & -3 \\ 1 & -4 \end{bmatrix}$ (b) $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix}$

5.25 Using the Cayley–Hamilton technique, find $e^{\mathbf{A}t}$ for

(a) $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix}$ (b) $\mathbf{A} = \begin{bmatrix} 0 & 2 \\ -2 & -4 \end{bmatrix}$

5.26 Given the system

$$\dot{\mathbf{x}} = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u$$

(a) Obtain a state diagram in signal-flow graph form.

(b) From the signal-flow graph, determine the state equation in the form

$$\mathbf{X}(s) = \mathbf{G}(s)\mathbf{x}(0) + \mathbf{H}(s)U(s)$$

(c) Using inverse Laplace transformation, obtain the

(i) zero-input response to initial condition

$$\mathbf{x}(0) = [x_1^0 \quad x_2^0]^T;$$

(ii) zero-state response to unit-step input.

5.27 A linear time-invariant system is described by the following state model:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} u$$

$$y = [1 \quad 0 \quad 0]\, \mathbf{x}$$

Diagonalize the coefficient matrix of the state model using a similarity transformation, and from there obtain the explicit solutions for the state vector and output when the control force u is a unit-step function and the initial state vector is

$$\mathbf{x}(0) = [0 \quad 0 \quad 2]^T$$

5.28 Consider the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u;\ \mathbf{x}(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$y = [1 \quad 0]\, \mathbf{x}$$

(a) Determine the stability of the system.

(b) Find the output response of the system to unit-step input.

5.29 Find the response of the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{u};\ \mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \mathbf{x}$$

to the following input:

$$\mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} \mu(t) \\ e^{-3t}\mu(t) \end{bmatrix};\ \mu(t) \text{ is unit-step function.}$$

5.30 Figure P5.30 shows the block diagram of a control system with state variable feedback and feedforward control. The plant model is

$$\dot{\mathbf{x}} = \begin{bmatrix} -3 & 2 \\ 4 & -5 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$y = [0 \quad 1]\mathbf{x}$$

(a) Derive a state model for the feedback system.

(b) Find the output $y(t)$ of the feedback system to a unit-step input $r(t)$; the initial state is assumed to be zero.

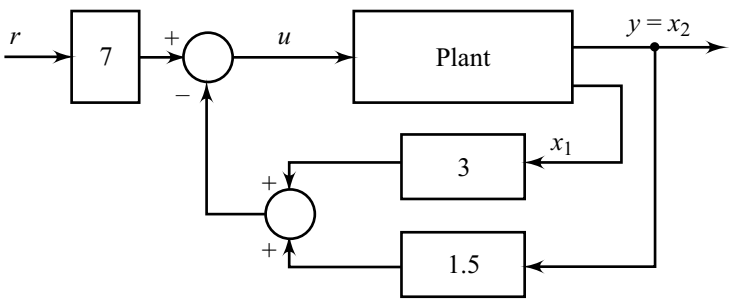


Fig. P5.30

5.31 Consider the state equation

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \mathbf{x}$$

Find a set of states $x_1(1)$ and $x_2(1)$ such that $x_1(2) = 2$.

5.32 Consider the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 2 \\ -12 & -7 & -6 \end{bmatrix} \mathbf{x}$$

(a) Find the modes of the system.

(b) Find the initial condition vector $\mathbf{x}(0)$ which will only excite the mode corresponding to the eigenvalue with the most negative real part.

5.33 Consider the system

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \mathbf{x}(t)$$

$$y(t) = [1 \quad 2]\,\mathbf{x}(t)$$

(a) Show that the system modes are e^{-t} and e^{2t}.

(b) Find a set of initial conditions such that the mode e^{2t} is suppressed in $y(t)$.

5.34 The following facts are known about the linear system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t).$$

If $\mathbf{x}(0) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, then $\mathbf{x}(t) = \begin{bmatrix} e^{-2t} \\ -2e^{-2t} \end{bmatrix}$

If $\mathbf{x}(0) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, then $\mathbf{x}(t) = \begin{bmatrix} e^{-t} \\ -e^{-t} \end{bmatrix}$

Find $e^{\mathbf{A}t}$ and hence $\mathbf{A}$.

5.35 Show that the pair $\{\mathbf{A}, \mathbf{c}\}$ is completely observable for all values of α_i's.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -\alpha_n \\ 1 & 0 & \cdots & 0 & -\alpha_{n-1} \\ 0 & 1 & \cdots & 0 & -\alpha_{n-2} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_1 \end{bmatrix}$$

$$\mathbf{c} = [0 \quad 0 \quad \cdots \quad 0 \quad 1]$$

5.36 Show that the pair $\{\mathbf{A}, \mathbf{b}\}$ is completely controllable for all values of α_i's.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_n & -\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_1 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

5.37 Given the system

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}; \mathbf{y} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x}$$

What can we say about controllability and observability—without making any further calculations?

5.38 Determine the controllability and observability properties of the following systems:

(i) $\mathbf{A} = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \mathbf{c} = [1 \quad -1]$

(ii) $\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}; \mathbf{c} = [0 \quad 1]$

(iii) $\mathbf{A} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}; \mathbf{C} = \begin{bmatrix} 1 & 1 & 2 \\ 3 & 1 & 5 \end{bmatrix}$

(iv) $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{c} = [10 \quad 0 \quad 0]$

(v) $\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -3 \\ 0 & 1 & -4 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 40 \\ 10 \\ 0 \end{bmatrix}; \mathbf{c} = [0 \quad 0 \quad 1]$

5.39 The following models realize the transfer function $G(s) = \dfrac{1}{s+1}$.

(i) $\mathbf{A} = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; $\mathbf{c} = [0 \quad 1]$

(ii) $\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 0 & -3 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

(iii) $\mathbf{A} = \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [0 \quad 1]$

Investigate the controllability and observability properties of these models.

Find a state variable model, for the given transfer function, which is both controllable and observable.

5.40 Consider the systems

(i) $\mathbf{A} = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; $\mathbf{c} = [0 \quad 1]$

(ii) $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [4 \quad 5 \quad 1]$

Determine the transfer function in each case. What can we say about controllability and observability properties—without making any further calculations?

5.41 Consider the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -2 & 1 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix} u; \; y = [1 \quad 0 \quad 0]\, \mathbf{x}$$

(a) Find the eigenvalues of **A** and from there determine the stability of the system.

(b) Find the transfer function model and from there determine the stability of the system.

(c) Are the two results the same? If not, why?

5.42 Given a transfer function

$$G(s) = \frac{10}{s(s+1)} = \frac{Y(s)}{U(s)}$$

Construct the following three different state models for this system:

(a) One which is both controllable and observable.

(b) One which is controllable but not observable.

(c) One which is observable but not controllable.

5.43 Prove that the transfer function

$$G(s) = Y(s)/U(s)$$

of the system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t)$$
$$y(t) = \mathbf{c}\mathbf{x}(t) + du(t)$$

is invariant under state transformation $\mathbf{x}(t) = \mathbf{P}\,\overline{\mathbf{x}}(t)$; **P** is a constant nonsingular matrix.

Chapter 6

State Variable Analysis of Digital Control Systems

6.1 INTRODUCTION

In the previous chapter of this book, we treated in considerable detail, the analysis of linear continuous-time systems using state variable methods. In this chapter, we give a condensed review of the same methods for linear discrete-time systems. Since the theory of linear discrete-time systems—very closely—parallels the theory of linear continuous-time systems, many of the results are similar. For this reason, the comments in this chapter are brief, except in those cases where the results for discrete-time systems deviate markedly from the continuous-time situation. For the same reason, many proofs are omitted.

We will be mostly concerned with Single-Input, Single-Output (SISO) system configurations of the type shown in the block diagram of Fig. 6.1. The plant in the figure, is a physical process characterized by continuous-time input and output variables. A digital computer is used to control the continuous-time plant. The interface system that takes care of the communication between the digital computer and the continuous-time plant consists of analog-to-digital (A/D) converter and digital-to-analog (D/A) converter. In order to analyze such a system, it is often convenient to represent the continuous-time plant, together with the D/A converter and the A/D converter, by an equivalent discrete-time system.

The discrete-time systems we will come across can, therefore, be classified into two types.

(i) Inherently discrete-time systems (digital processors), where it makes sense to consider the system at discrete instants of time only, and what happens in between is irrelevant.

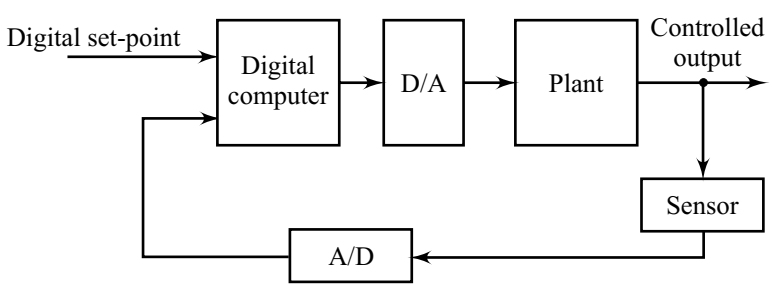


Fig. 6.1 Basic structure of digital control systems

(ii) Discrete-time systems that result from considering continuous-time systems at discrete instants of time only.

6.2 STATE DESCRIPTIONS OF DIGITAL PROCESSORS

A discrete-time system is a transformation, or operator, that maps a given input sequence $u(k)$ into an output sequence $y(k)$. Classes of discrete-time systems are defined by placing constraints on the transformation. As they are relatively easy to characterize mathematically, and as they can be designed to perform useful signal processing functions, the class of linear time-invariant systems will be studied here.

In the control structure of Fig. 6.1, the digital computer transforms an input sequence into a form which is, in some sense, more desirable. Therefore, the discrete-time systems we consider here, are in fact *computer programs*. Needless to say, digital computers can do many things other than control dynamic systems; it is our purpose to examine their characteristics when doing this elementary control task.

State variable model of a SISO discrete-time system consists of a set of first-order difference equations relating state variables $x_1(k)$, $x_2(k)$, ..., $x_n(k)$ of the discrete-time system to the input $u(k)$; the output $y(k)$ is algebraically related to the state variables and the input. Assuming that the input is switched on to the system at $k = 0$ ($u(k) = 0$ for $k < 0$), then the initial state is given by

$$\mathbf{x}(0) \triangleq \mathbf{x}^0; \text{ a specified } n \times 1 \text{ vector}$$

The dynamics of a linear time-invariant system is described by equations of the form

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k); \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{6.1a}$$

$$y(k) = \mathbf{c}\mathbf{x}(k) + du(k) \tag{6.1b}$$

where $\mathbf{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} = n \times 1$ state vector of nth-order system

$u(k)$ = system input

$y(k)$ = defined output

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{bmatrix} = n \times n \text{ constant matrix}$$

$$\mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} = n \times 1 \text{ constant column matrix}$$

$\mathbf{c} = [c_1 \quad c_2 \cdots c_n] = 1 \times n$ constant row matrix

d = scalar, representing direct coupling between input and output

Equation (6.1a) is called the *state equation* of the system, Eqn. (6.1b) is called the *output equation*; the two equations together give the *state variable model* of the system.

6.2.1 Conversion of State Variable Models to Transfer Functions

In the study of linear time-invariant discrete-time equations, we may also apply the z-transform techniques. Taking the z-transform of Eqns (6.1), we obtain:

$$z\mathbf{X}(z) - z\mathbf{x}^0 = \mathbf{F}\mathbf{X}(z) + \mathbf{g}U(z)$$
$$Y(z) = \mathbf{c}\mathbf{X}(z) + dU(z)$$

where $\mathbf{X}(z) \triangleq \mathscr{Z}[\mathbf{x}(k)]$; $U(z) \triangleq \mathscr{Z}[u(k)]$; $Y(z) \triangleq \mathscr{Z}[y(k)]$

Manipulation of these equations, gives

$$(z\mathbf{I} - \mathbf{F})\,\mathbf{X}(z) = z\mathbf{x}^0 + \mathbf{g}U(z);\ \mathbf{I} \text{ is } n \times n \text{ identity matrix}$$

or

$$\mathbf{X}(z) = (z\mathbf{I} - \mathbf{F})^{-1} z\mathbf{x}^0 + (z\mathbf{I} - \mathbf{F})^{-1}\mathbf{g}U(z) \tag{6.2a}$$

$$Y(z) = \mathbf{c}(z\mathbf{I} - \mathbf{F})^{-1} z\mathbf{x}^0 + [\mathbf{c}(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{g} + d]\,U(z) \tag{6.2b}$$

Equations (6.2) are algebraic equations. If $\mathbf{x}^0$ and $U(z)$ are known, $\mathbf{X}(z)$ can be computed from these equations.

In the case of zero initial state (i.e., $\mathbf{x}^0 = \mathbf{0}$), the input-output behavior of the system (6.1) is determined entirely by the transfer function

$$\frac{Y(z)}{U(z)} = G(z) = \mathbf{c}(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{g} + d \tag{6.3a}$$

$$= \mathbf{c}\frac{(z\mathbf{I} - \mathbf{F})^{+}\mathbf{g}}{|\,z\mathbf{I} - \mathbf{F}\,|} + d \tag{6.3b}$$

where $(z\mathbf{I} - \mathbf{F})^{+}$ = adjoint of the matrix $(z\mathbf{I} - \mathbf{F})$

$|z\mathbf{I} - \mathbf{F}|$ = determinant of the matrix $(z\mathbf{I} - \mathbf{F})$

$|\lambda\mathbf{I} - \mathbf{F}|$ is the *characteristic polynomial* of matrix $\mathbf{F}$. The roots of this polynomial are the *characteristic roots* or *eigenvalues* of matrix $\mathbf{F}$.

From Eqn. (6.3b), we observe that the characteristic polynomial of matrix $\mathbf{F}$ of the system (6.1), is same as the *denominator* polynomial of the corresponding transfer function $G(z)$. If there are no cancellations between the numerator and denominator polynomials of $G(z)$ in Eqn. (6.3b), the eigenvalues of matrix $\mathbf{F}$ are same as the poles of $G(z)$.

In a later section, we shall see that for a completely controllable and observable state variable model, the eigenvalues of matrix $\mathbf{F}$ are same as the poles of the corresponding transfer function.

6.2.2 Conversion of Transfer Functions to Canonical State Variable Models

In Chapters 2–4, we have seen that transform-domain design techniques yield digital control algorithms in the form of transfer functions of the form

$$D(z) = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_{n-1} z + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_{n-1} z + \alpha_n} \tag{6.4}$$

where the coefficients α_i and β_i are real constant scalars. Equation (6.4) represents an nth-order digital controller. Several different structures for realization of this controller—using delay elements, adders, and multipliers—were presented in Section 3.4. Each of these realizations is a dynamic system with n first-order dynamic elements—the unit delayers. We know that output of a first-order dynamic element represents the *state* of that element. Therefore, each realization of Eqn. (6.4) is, in fact, a *state diagram*; by labeling the unit-delayer outputs as state variables, we can obtain the state variable model.

In the following discussion, we shall use two of the structures presented in Section 3.4 for obtaining canonical state variable models corresponding to the general transfer function

$$G(z) = \frac{Y(z)}{U(z)} = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_{n-1} z + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_{n-1} z + \alpha_n} \tag{6.5}$$

Revisiting Section 3.4 at this stage will be helpful in our discussion.

First Companion Form

A direct realization structure for the system described by Eqn. (6.5) is shown in Fig. 6.2. Notice that n delay elements have been used in this realization. The coefficients α_1, α_2, ..., α_n appear as feedback elements, and the coefficients β_0, β_1, ..., β_n appear as feedforward elements. To get one state variable model, we identify the output of each unit delayer with a state variable—starting at the right and proceeding to the left. The corresponding difference equations are

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= x_3(k) \\ &\vdots \\ x_{n-1}(k+1) &= x_n(k) \\ x_n(k+1) &= -\alpha_n x_1(k) - \alpha_{n-1} x_2(k) - \cdots - \alpha_1 x_n(k) + u(k) \end{aligned} \tag{6.6a}$$

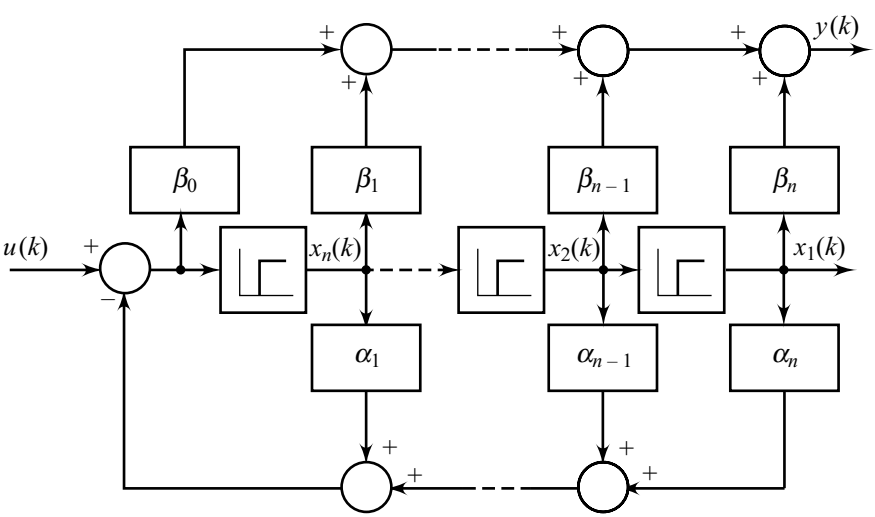


Fig. 6.2 A direct realization structure for system given by Eqn. (6.5)

Careful examination of Fig. 6.2 reveals that there are two paths from the output of each unit delayer to the system output: one path upward through the box labeled β_i, and a second path down through the box labeled α_i and thence through the box labeled β_0. As a consequence

$$y(k) = (\beta_n - \alpha_n\beta_0)\, x_1(k) + (\beta_{n-1} - \alpha_{n-1}\,\beta_0)\, x_2(k) + \cdots + (\beta_1 - \alpha_1\,\beta_0)\, x_n(k) + \beta_0\, u(k) \tag{6.6b}$$

The state and output equations (6.6), organized in vector-matrix form, are given below.

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{c}\mathbf{x}(k) + du(k) \end{aligned} \tag{6.7}$$

with

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_n & -\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_1 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{c} = [\beta_n - \alpha_n\beta_0,\ \beta_{n-1} - \alpha_{n-1}\beta_0,\ \ldots,\ \beta_1 - \alpha_1\beta_0];\ d = \beta_0$$

The matrix **F** in Eqns (6.7) has a very special structure—the coefficients of the denominator of the transfer function preceded by minus signs form a string along the bottom row of the matrix. The rest of the matrix is zero except for the 'superdiagonal' terms which are all unity. A matrix with this structure is said to be in *companion form*. We call the state variable model (6.7) the *first companion form*[1] state model for the transfer function (6.5); another companion form follows.

Second Companion Form

In the first companion form, the coefficients of the denominator of the transfer function appear in one of the rows of the **F** matrix. There is another companion form in which the coefficients appear in a column of the **F** matrix. This can be obtained from another direct realization structure shown in Fig. 6.3. We

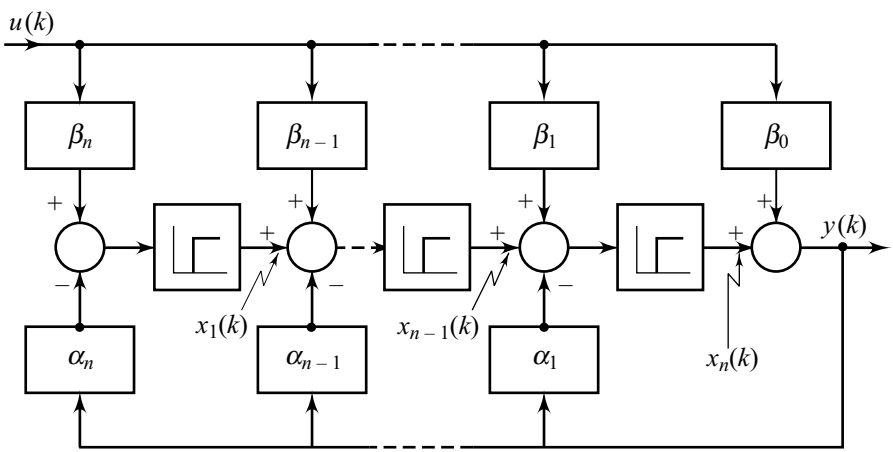


Fig. 6.3 An alternative direct realization structure for the system given by Eqn. (6.5)

[1] The pair (**F**, **g**) of Eqns (6.7) is completely controllable for all values of α_i's (Refer to Problem 5.36).

identify the output of each unit delayer with a state variable—starting at the left and proceeding to the right. The corresponding difference equations are

$$\begin{aligned} x_n(k+1) &= x_{n-1}(k) - \alpha_1(x_n(k) + \beta_0 u(k)) + \beta_1 u(k) \\ x_{n-1}(k+1) &= x_{n-2}(k) - \alpha_2(x_n(k) + \beta_0 u(k)) + \beta_2 u(k) \\ &\vdots \\ x_2(k+1) &= x_1(k) - \alpha_{n-1}(x_n(k) + \beta_0 u(k)) + \beta_{n-1} u(k) \\ x_1(k+1) &= -\alpha_n(x_n(k) + \beta_0 u(k)) + \beta_n u(k) \\ y(k) &= x_n(k) + \beta_0 u(k) \end{aligned}$$

The state and output equations, organized in vector-matrix form, are given below.

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{c}\mathbf{x}(k) + du(k) \end{aligned} \tag{6.8}$$

with

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -\alpha_n \\ 1 & 0 & \cdots & 0 & -\alpha_{n-1} \\ 0 & 1 & \cdots & 0 & -\alpha_{n-2} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_1 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} \beta_n - \alpha_n\beta_0 \\ \beta_{n-1} - \alpha_{n-1}\beta_0 \\ \vdots \\ \beta_1 - \alpha_1\beta_0 \end{bmatrix}$$

$$\mathbf{c} = [0 \quad 0 \cdots 0 \quad 1]; d = \beta_0$$

Comparing the **F**, **g** and **c** matrices of the *second companion form*[2] with that of the first, we observe that **F**, **g**, and **c** matrices of one companion form correspond to the transpose of **F**, **c**, and **g** matrices, respectively, of the other.

Both the companion forms of state variable models play an important role in pole-placement design through state feedback. This will be discussed in Chapter 7.

Jordan Canonical Form

In the two canonical forms (6.7) and (6.8), the coefficients of the denominator of the transfer function appear in one of the rows or columns of matrix **F**. In another of the canonical forms, the poles of the transfer function form a string along the main diagonal of the matrix. The canonical form follows directly from the parallel realization structure of transfer function. We shall first discuss the case where all poles are distinct. Then we shall consider the case where multiple poles are involved.

Case I: The transfer function involves distinct poles only

Assume that $z = \lambda_i$ (i = 1, 2, ..., n) are the distinct poles of the given transfer function (6.5). Partial-fraction expansion of the transfer function gives

$$\begin{aligned} \frac{Y(z)}{U(z)} &= G(z) = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_{n-1} z + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_{n-1} z + \alpha_n} \\ &= \beta_0 + \frac{\beta_1' z^{n-1} + \beta_2' z^{n-2} + \cdots + \beta_n'}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n} \end{aligned}$$

[2] The pair {**F**, **c**} of Eqns (6.8) is completely observable for all values of α_i's (Refer to Problem 5.35).

$$= \beta_0 + \frac{\beta_1' z^{n-1} + \beta_2' z^{n-2} + \cdots + \beta_n'}{(z-\lambda_1)(z-\lambda_2)\cdots(z-\lambda_n)} = \beta_0 + G'(z)$$

$$= \beta_0 + \frac{r_1}{z-\lambda_1} + \frac{r_2}{z-\lambda_2} + \cdots + \frac{r_n}{z-\lambda_n} \tag{6.9}$$

The coefficients $r_i\,(i = 1, 2, ..., n)$ are the residues of the transfer function $G'(z)$ at the corresponding poles at $z = \lambda_i (i = 1, 2, ..., n)$. A parallel realization structure of the transfer function (6.9) is shown in Fig. 6.4.

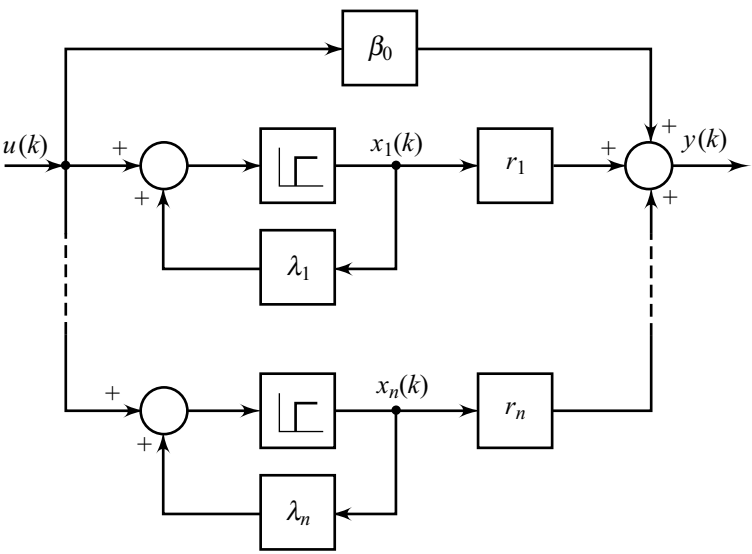


Fig. 6.4 A parallel structure realization for the system given by Eqn. (6.9)

Identifying the outputs of the delayers with the state variables results in the following state and output equations:

$$\mathbf{x}(k+1) = \mathbf{\Lambda x}(k) + \mathbf{g}u(k) \tag{6.10}$$

$$y(k) = \mathbf{c x}(k) + du(k)$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}; \mathbf{g} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$\mathbf{c} = [r_1 \quad r_2 \cdots r_n]; d = \beta_0$$

It is observed that for this canonical state variable model, the matrix $\mathbf{\Lambda}$ is a diagonal matrix with the poles of $G(z)$ as its diagonal elements.

Case II: The transfer function involves multiple poles

When the transfer function $G(z)$ involves multiple poles, the partial fraction expansion will not be as simple as (6.9). In the discussion that follows, we assume that $G(z)$ involves a multiple pole of order m at

$z = \lambda_1$, and that all other poles are distinct. Performing the partial fraction expansion for this case, we get

$$\begin{aligned}\frac{Y(z)}{U(z)} &= G(z) = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_{n-1} z + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_{n-1} z + \alpha_n} \\ &= \beta_0 + \frac{\beta_1' z^{n-1} + \beta_2' z^{n-2} + \cdots + \beta_n'}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n} \\ &= \beta_0 + \frac{\beta_1' z^{n-1} + \beta_2' z^{n-2} + \cdots + \beta_n'}{(z-\lambda_1)^m (z-\lambda_{m+1}) \cdots (z-\lambda_n)} \\ &= \beta_0 + H_1(z) + H_{m+1}(z) + \cdots + H_n(z) \end{aligned} \tag{6.11a}$$

where
$$H_{m+1}(z) = \frac{r_{m+1}}{z-\lambda_{m+1}}, \ldots, H_n(z) = \frac{r_n}{z-\lambda_n}, \tag{6.11b}$$

and
$$H_1(z) = \frac{r_{11}}{(z-\lambda_1)^m} + \frac{r_{12}}{(z-\lambda_1)^{m-1}} + \cdots + \frac{r_{1m}}{(z-\lambda_1)} \tag{6.11c}$$

A realization of $H_1(z)$ is shown in Fig. 6.5. Other terms of Eqn. (6.11a) may be realized as per Fig. 6.4.

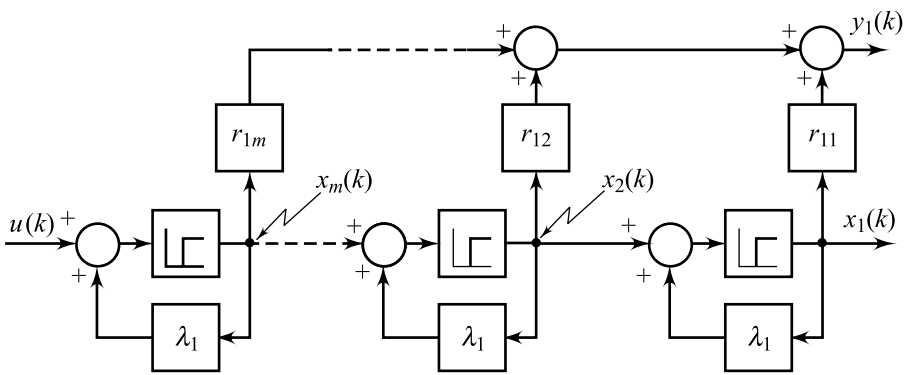


Fig. 6.5 A realization of $H_1(z)$ given by Eqn. (6.11c)

Identifying the outputs of the delayers with the state variables results in the following state and output equations:

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{\Lambda x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{cx}(k) + du(k)\end{aligned} \tag{6.12}$$

with

$$\mathbf{\Lambda} = \left[\begin{array}{ccccc:ccc} \lambda_1 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_1 & 0 & \cdots & 0 \\ \hdashline 0 & 0 & 0 & \cdots & 0 & \lambda_{m+1} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & \lambda_n \end{array}\right] \quad \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \quad \begin{matrix} m \times m \text{ Jordan block} \\ \\ \\ \\ \\ \\ \\ \end{matrix}$$

$$\mathbf{g} = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 1 \\ \hdashline 1 \\ \vdots \\ 1 \end{array}\right]$$

$$\mathbf{c} = \left[r_{11} \quad r_{12} \quad \cdots \quad r_{1m} \,\vdots\, r_{m+1} \quad \cdots \quad r_n \right]; \; d = \beta_0$$

Note that the $\mathbf{\Lambda}$ matrix in Eqns (6.12) is block diagonal:

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_{m+1} & \cdots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{\Lambda}_n \end{bmatrix}$$

with

$$\mathbf{\Lambda}_1 = \begin{bmatrix} \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_1 \end{bmatrix} = \begin{array}{l} m \times m \text{ submatrix corresponding} \\ \text{to eigenvalue } \lambda_1 \text{ of multiplicity } m \end{array}$$

$\mathbf{\Lambda}_{m+1} = \lambda_{m+1} = 1 \times 1$ submatrix corresponding to simple eigenvalue λ_{m+1}

$\vdots$

$\mathbf{\Lambda}_n = \lambda_n = 1 \times 1$ submatrix corresponding to simple eigenvalue λ_n

The matrix $\mathbf{\Lambda}_1$ has two diagonals: the principal diagonal has the corresponding pole λ_1 and the super diagonal has all 1s. This structure is said to be in *Jordan form*; for this reason the model (6.12) is identified as *Jordan canonical form* state model.

The state variable model (6.10) derived for the case of distinct poles, is a special case of Jordan canonical form wherein each Jordan block is of 1×1 dimension.

6.3 STATE DESCRIPTION OF SAMPLED CONTINUOUS-TIME PLANTS

Systems that consist of an interconnection of a discrete-time system and a continuous-time system are frequently encountered. An example of particular interest occurs when a digital computer is used to control a continuous-time plant. Whenever such interconnections exist, there must be some type of *interface system* that takes care of the communication between the discrete-time and continuous-time systems. In the system of Fig. 6.1, the interface function is performed by D/A and A/D converters.

Simple models of the interface actions of D/A and A/D converters have been developed in Chapter 2. A brief review is in order here.

A simple model of A/D converter is shown in Fig. 6.6. A continuous-time function $f(t)$, $t \geq 0$, is the input, and the sequence of real numbers $f(k)$, $k = 0, 1, 2, \ldots$, is the output; the following relation holds between input and output:

$$f(k) = f(t = kT); \; T \text{ is the time interval between samples} \tag{6.13a}$$

A simple model of D/A converter is shown in Fig. 6.7. A sequence of numbers $f(k)$, $k = 0, 1, 2, \ldots$, is the input, and the continuous-time function $f^+(t)$, $t \geq 0$, is the output; the following relation holds between input and output:

$$f^+(t) = f(k); \; kT \leq t < (k+1)T \tag{6.13b}$$

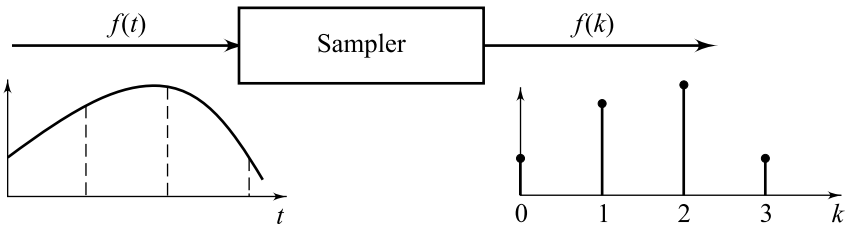


Fig. 6.6 A model of an A/D converter

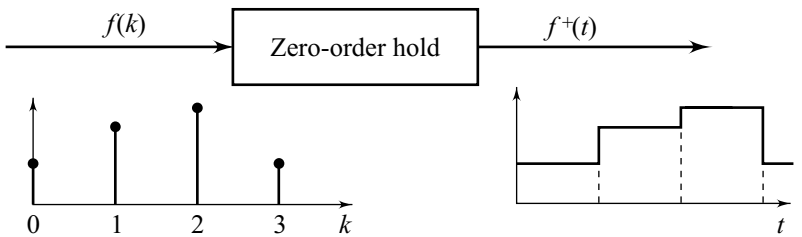


Fig. 6.7 A model of a D/A converter

Figure 6.8 illustrates a typical example of an interconnection of discrete-time and continuous-time systems. In order to analyze such a system, it is often convenient to represent the continuous-time system together with the zero-order hold (ZOH) and the sampler, by an *equivalent discrete-time system*.

We assume that the continuous-time system of Fig. 6.8 is a linear system with state variable model

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u^{+}(t) \tag{6.14a}$$

$$y(t) = \mathbf{c}\mathbf{x}(t) + du^{+}(t) \tag{6.14b}$$

where $\mathbf{x}(t)$ is $n \times 1$ state vector, $u^{+}(t)$ is scalar input, $y(t)$ is scalar output; $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$, and d are, respectively, $n \times n$, $n \times 1$, $1 \times n$, and 1×1 real constant matrices.

The solution of Eqn. (6.14a) with t_0 as initial time is

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\,\mathbf{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\,\mathbf{b}u^{+}(\tau)\,d\tau \tag{6.15}$$

Since we use a ZOH (refer to Eqn. (6.13b)),

$$u^{+}(t) = u(kT);\ kT \le t < (k+1)T;\ k = 0, 1, 2, \ldots$$

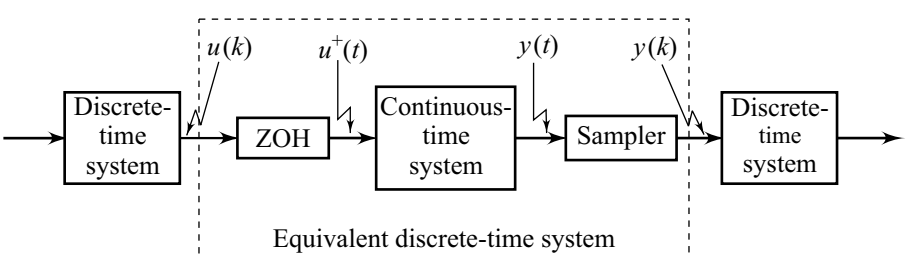


Fig. 6.8 Interconnection of discrete-time and continuous-time systems

Then from Eqn. (6.15), we can write

$$\mathbf{x}(t) = e^{\mathbf{A}(t-kT)}\,\mathbf{x}(kT) + \left[\int_{kT}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{b}\,d\tau\right] u(kT);\ kT \le t < (k+1)T \tag{6.16}$$

In response to the input $u(kT)$, the state settles to the value $\mathbf{x}((k+1)T)$ prior to the application of the input $u((k+1)T)$, where

$$\begin{aligned}\mathbf{x}((k+1)T) &= e^{\mathbf{A}T}\,\mathbf{x}(kT) + \left[\int_{kT}^{(k+1)T} e^{\mathbf{A}[(k+1)T-\tau]}\mathbf{b}\,d\tau\right] u(kT)\\ &= \mathbf{F}\mathbf{x}(kT) + \mathbf{g}u(kT)\end{aligned} \tag{6.17}$$

Letting $\sigma = (\tau - kT)$ in Eqn. (6.17), we have

$$\mathbf{g} = \int_0^T e^{\mathbf{A}(T-\sigma)}\,\mathbf{b}d\sigma$$

With $\theta = T - \sigma$, we get

$$\mathbf{g} = \int_0^T e^{\mathbf{A}\theta}\,\mathbf{b}d\theta$$

If we are interested in the value of $\mathbf{x}(t)$ (or $y(t)$) between sampling instants, we first solve for $\mathbf{x}(kT)$ for any k using Eqn. (6.17), and then use Eqn. (6.16) to determine $\mathbf{x}(t)$ for $kT \le t < (k+1)T$.

Since we have a sampler in configuration of Fig. 6.8 (refer to Eqn. (6.13a)), we have from Eqn. (6.14b),

$$y(kT) = \mathbf{c}\mathbf{x}(kT) + du(kT)$$

State description of the equivalent discrete-time system of Fig. 6.8 is, therefore, of the form

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \tag{6.18a}$$

$$y(k) = \mathbf{c}\mathbf{x}(k) + du(k) \tag{6.18b}$$

where

$$\mathbf{F} = e^{\mathbf{A}T} \tag{6.18c}$$

$$\mathbf{g} = \int_0^T e^{\mathbf{A}\theta}\,\mathbf{b}d\theta \tag{6.18d}$$

Machine Computation of e^{AT}

There are several methods available for computing $e^{\mathbf{A}T}$. Some of these methods have been discussed in the earlier chapter. Standard computer programs based on these methods are available.

In the following, we present an alternative technique of computing $e^{\mathbf{A}T}$. The virtues of this technique are its simplicity and the ease of programming.

The infinite series expansion for $\mathbf{F} = e^{\mathbf{A}T}$ is

$$\begin{aligned}\mathbf{F} = e^{\mathbf{A}T} &= \mathbf{I} + \mathbf{A}T + \frac{1}{2!}\,\mathbf{A}^2T^2 + \frac{1}{3!}\,\mathbf{A}^3T^3 + \cdots\\ &= \sum_{i=0}^{\infty} \frac{\mathbf{A}^iT^i}{i!};\ \mathbf{A}^0 = \mathbf{I}\end{aligned} \tag{6.19}$$

For a finite T, this series is uniformly convergent (Section 5.7). It is, therefore, possible to evaluate $\mathbf{F}$ within prescribed accuracy. If the series is truncated at $i = N$, then we may write the finite series sum as

$$\bar{\mathbf{F}} = \sum_{i=0}^{N} \frac{\mathbf{A}^i T^i}{i!} \tag{6.20}$$

which represents the infinite series approximation. The larger the N, the better is the approximation. We evaluate $\bar{\mathbf{F}}$ by a series in the form

$$\bar{\mathbf{F}} = \mathbf{I} + \mathbf{A}T\left(\mathbf{I} + \frac{\mathbf{A}T}{2}\left\{\mathbf{I} + \frac{\mathbf{A}T}{3}\left[\mathbf{I} + \cdots + \frac{\mathbf{A}T}{N-1}\left(\mathbf{I} + \frac{\mathbf{A}T}{N}\right)\cdots\right]\right\}\right) \tag{6.21}$$

which has better numerical properties than the direct series of powers. Starting with the innermost factor, this nested product expansion lends itself easily to digital programming. The empirical relation giving the number of terms, N, is

$$N = \min\{3 \parallel \mathbf{A}T \parallel + 6, 100\} \tag{6.22}$$

where $\parallel \mathbf{A}T \parallel$ is a norm of the matrix $\mathbf{A}T$. There are several different forms of matrix norms commonly used. Any one of them may be used in Eqn. (6.22). Two forms of matrix norms are defined in Section 5.2.

The relation (6.22) assumes that no more than 100 terms are included. The series $e^{\mathbf{A}T}$ will be accurate to, at least, six significant figures.

The integral in Eqn. (6.18d) can be evaluated term by term, to give

$$\mathbf{g} = \left[\int_0^T \left(\mathbf{I} + \mathbf{A}\theta + \frac{1}{2!}\mathbf{A}^2\theta^2 + \cdots\right) d\theta\right] \mathbf{b} = \sum_{i=0}^{\infty} \frac{\mathbf{A}^i T^{i+1}}{(i+1)!}\mathbf{b} \tag{6.23}$$

$$= \sum_{i=0}^{\infty} \frac{\mathbf{A}^i T^i}{(i+1)!} T\mathbf{b}$$

$$= \left(\mathbf{I} + \frac{\mathbf{A}T}{2!} + \frac{\mathbf{A}^2T^2}{3!} + \cdots\right) T\mathbf{b} = (e^{\mathbf{A}T} - \mathbf{I})\,\mathbf{A}^{-1}\mathbf{b} \tag{6.24}$$

The transition from Eqn. (6.23) to (6.24) is possible only for a nonsingular matrix $\mathbf{A}$. For a singular $\mathbf{A}$, we may evaluate $\mathbf{g}$ from Eqn. (6.23) by the approximation technique described above. Since the series expansion for $\mathbf{g}$ converges faster than that for $\mathbf{F}$, it suffices to determine N for $\mathbf{F}$ from Eqn. (6.22) and apply the same value for $\mathbf{g}$.

Example 6.1

Figure 6.9 shows the block diagram of a digital positioning system. Defining the state variables as

$$x_1(t) = \theta(t), x_2(t) = \dot{\theta}(t),$$

the state variable model of the plant becomes

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u^+(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t) \end{aligned} \tag{6.25}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -5 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \;\; 0]$$

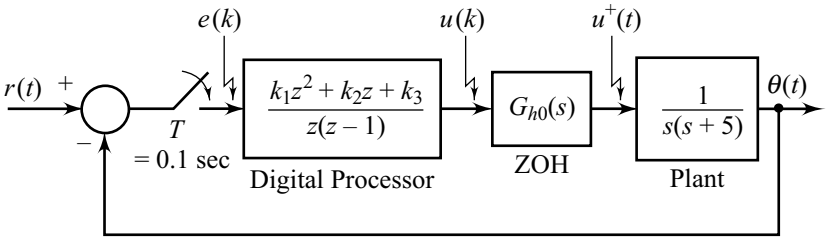


Fig. 6.9 A digital positioning system

Here we apply the Cayley–Hamilton technique to evaluate the state transition matrix $e^{\mathbf{A}t}$.

Eigenvalues of matrix **A** are given by

$$|\lambda\mathbf{I} - \mathbf{A}| = \begin{bmatrix} \lambda & -1 \\ 0 & \lambda + 5 \end{bmatrix} = 0$$

Therefore, $\lambda_1 = 0, \lambda_2 = -5$

Since **A** is of second order, the polynomial $g(\lambda)$ will be of the form (refer to Eqns (5.98)),

$$g(\lambda) = \beta_0 + \beta_1\lambda$$

The coefficients β_0 and β_1 are evaluated from the following equations:

$$1 = \beta_0$$

$$e^{-5t} = \beta_0 - 5\beta_1$$

The result is $\beta_0 = 1$

$$\beta_1 = \frac{1}{5}(1 - e^{-5t})$$

Hence $$e^{\mathbf{A}t} = \beta_0\mathbf{I} + \beta_1\mathbf{A} = \begin{bmatrix} 1 & \frac{1}{5}(1-e^{-5t}) \\ 0 & e^{-5t} \end{bmatrix}$$

The equivalent discrete-time plant with input $u(k)$ and output $\theta(k)$ (refer to Fig. 6.9) is described by the equations

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{c}\mathbf{x}(k) \end{aligned} \tag{6.26}$$

where $$\mathbf{F} = e^{\mathbf{A}T} = \begin{bmatrix} 1 & \frac{1}{5}(1-e^{-5T}) \\ 0 & e^{-5T} \end{bmatrix}$$

$$\mathbf{g} = \int_0^T e^{\mathbf{A}\theta}\,\mathbf{b}\,d\theta = \begin{bmatrix} \int_0^T \frac{1}{5}(1-e^{-5\theta})d\theta \\ \int_0^T e^{-5\theta}d\theta \end{bmatrix} = \begin{bmatrix} \frac{1}{5}(T - \frac{1}{5} + \frac{1}{5}e^{-5T}) \\ \frac{1}{5}(1-e^{-5T}) \end{bmatrix}$$

For $T = 0.1$ sec,

$$\mathbf{F} = \begin{bmatrix} 1 & 0.0787 \\ 0 & 0.6065 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} 0.0043 \\ 0.0787 \end{bmatrix}$$

Consider now the digital processor. The input-output model of the processor is

$$\frac{U(z)}{E(z)} = \frac{k_1 z^2 + k_2 z + k_3}{z(z-1)}$$

Direct digital realization of the processor is shown in Fig. 6.10. Taking outputs of unit delayers as state variables, we get the following state description for the processor dynamics (refer to Eqns (6.8)):

$$\begin{bmatrix} x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_3(k) \\ x_4(k) \end{bmatrix} + \begin{bmatrix} k_3 \\ k_2 + k_1 \end{bmatrix} e(k) \tag{6.27}$$

$$u(k) = x_4(k) + k_1 e(k)$$

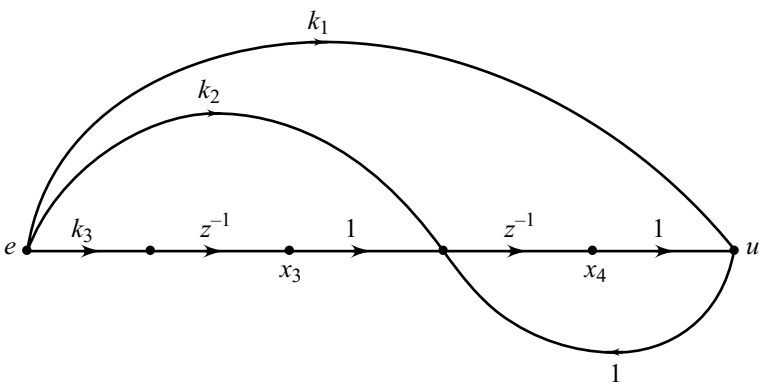


Fig. 6.10 Realization of digital processor of positioning system shown in Fig. 6.9

The processor input is derived from the reference input and the position feedback (Fig. 6.9):

$$e(k) = r(k) - x_1(k) \tag{6.28}$$

From Eqns (6.26)–(6.28), we get the following state variable model for the feedback system of Fig. 6.9.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} 1-0.0043k_1 & 0.0787 & 0 & 0.0043 \\ -0.0787k_1 & 0.6065 & 0 & 0.0787 \\ -k_3 & 0 & 0 & 0 \\ -(k_2+k_1) & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} + \begin{bmatrix} 0.0043k_1 \\ 0.0787k_1 \\ k_3 \\ k_2+k_1 \end{bmatrix} r(k) \tag{6.29}$$

$$y(k) = [1 \quad 0 \quad 0 \quad 0] \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix}$$

6.4 STATE DESCRIPTION OF SYSTEMS WITH DEAD-TIME

Consider a state equation of a single-input system which includes delay in control action:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u^{+}(t - \tau_D) \tag{6.30}$$

where $\mathbf{x}$ is $n \times 1$ state vector, u^{+} is scalar input, τ_D is the dead-time, and $\mathbf{A}$ and $\mathbf{b}$ are, respectively, $n \times n$ and $n \times 1$ real constant matrices.

The solution of Eqn. (6.30) with t_0 as initial time is

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\,\mathbf{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{b}u^{+}(\tau - \tau_D)d\tau$$

If we let $t_0 = kT$ and $t = kT + T$, we obtain

$$\mathbf{x}(kT+T) = e^{\mathbf{A}T}\,\mathbf{x}(kT) + \int_{kT}^{kT+T} e^{\mathbf{A}(kT+T-\tau)}\,\mathbf{b}u^{+}(\tau - \tau_D)\,d\tau$$

With $\sigma = kT + T - \tau$, we get

$$\mathbf{x}(kT+T) = e^{\mathbf{A}T}\,\mathbf{x}(kT) + \int_{0}^{T} e^{\mathbf{A}\sigma}\,\mathbf{b}u^{+}(kT + T - \tau_D - \sigma)\,d\sigma \tag{6.31}$$

If N is the largest integer number of sampling periods in τ_D, we can write

$$\tau_D = NT + \Delta T;\ 0 \le \Delta < 1 \tag{6.32a}$$

Substituting in Eqn. (6.31), we get

$$\mathbf{x}(kT+T) = e^{\mathbf{A}T}\,\mathbf{x}(kT) + \int_{0}^{T} e^{\mathbf{A}\sigma}\,\mathbf{b}u^{+}(kT + T - NT - \Delta T - \sigma)\,d\sigma$$

We introduce a parameter m such that

$$m = 1 - \Delta \tag{6.32b}$$

Then

$$\mathbf{x}(kT+T) = e^{\mathbf{A}T}\,\mathbf{x}(kT) + \int_{0}^{T} e^{\mathbf{A}\sigma}\,\mathbf{b}u^{+}(kT - NT + mT - \sigma)d\sigma \tag{6.33}$$

Since we use a ZOH, u^{+} is piecewise constant. The nature of the integral in Eqn. (6.33), with respect to variable σ, becomes clear from the sketch of the piecewise constant input u^{+} over a segment of time axis near $t = kT - NT$ (Fig. 6.11). The integral runs for σ from 0 to T—which corresponds to t from $kT - NT + mT$ backward to $kT - NT - T + mT$. Over this period, the control first takes on the value $u(kT - NT)$ and then the value $u(kT - NT - T)$. Therefore, we can break the integral in Eqn. (6.33) into two parts as follows:

$$\begin{aligned}\mathbf{x}(kT+T) &= e^{\mathbf{A}T}\,\mathbf{x}(kT) + \left[\int_{0}^{mT} e^{\mathbf{A}\sigma}\mathbf{b}d\sigma\right] u(kT - NT) + \left[\int_{mT}^{T} e^{\mathbf{A}\sigma}\mathbf{b}d\sigma\right] u(kT - NT - T) \\ &= \mathbf{F}\mathbf{x}(kT) + \mathbf{g}_1 u(kT - NT - T) + \mathbf{g}_2 u(kT - NT)\end{aligned} \tag{6.34a}$$

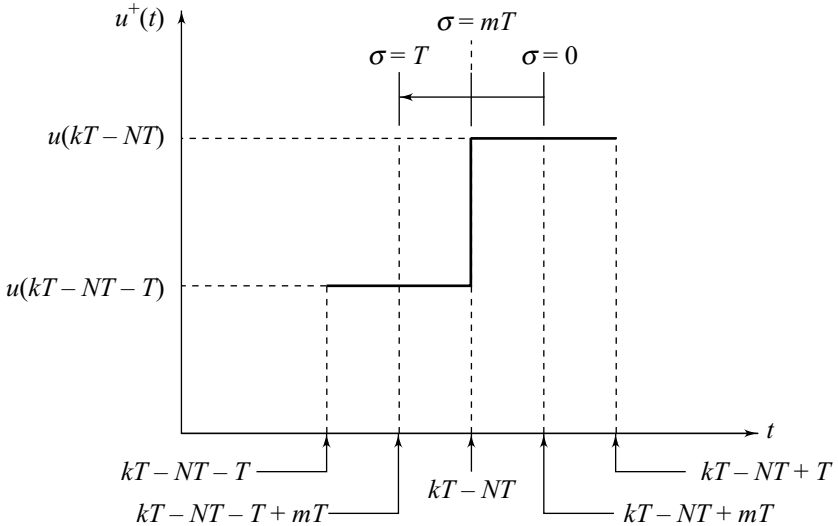


Fig. 6.11

where

$$\mathbf{F} = e^{\mathbf{A}T} \tag{6.34b}$$

$$\mathbf{g}_1 = \int_{mT}^{T} e^{\mathbf{A}\sigma}\,\mathbf{b}\,d\sigma \tag{6.34c}$$

$$\mathbf{g}_2 = \int_{0}^{mT} e^{\mathbf{A}\sigma}\,\mathbf{b}\,d\sigma \tag{6.34d}$$

Setting $\theta = \sigma - mT$ in Eqn. (6.34c), we get

$$\mathbf{g}_1 = \int_{0}^{\Delta T} e^{\mathbf{A}(mT+\theta)}\,\mathbf{b}\,d\theta = e^{\mathbf{A}mT}\int_{0}^{\Delta T} e^{\mathbf{A}\theta}\,\mathbf{b}\,d\theta \tag{6.34e}$$

The matrices/vectors $\mathbf{F}$, $\mathbf{g}_1$ and $\mathbf{g}_2$ can be evaluated by series truncation method discussed in the earlier section.

Equation (6.34a) can be expressed in the standard state variable format. To do this, we consider first the case of $N = 0$. For this case, Eqn. (6.34a) becomes

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}_1 u(k-1) + \mathbf{g}_2 u(k)$$

We must eliminate $u(k-1)$ from the right-hand side, which we do by defining a new state

$$x_{n+1}(k) = u(k-1)$$

The augmented state equation is given by

$$\begin{bmatrix} \mathbf{x}(k+1) \\ x_{n+1}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{g}_1 \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ x_{n+1}(k) \end{bmatrix} + \begin{bmatrix} \mathbf{g}_2 \\ 1 \end{bmatrix} u(k) \tag{6.35}$$

For $N > 0$, Eqn. (6.34a) can be expressed as

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}_1 u(k-N-1) + \mathbf{g}_2 u(k-N)$$

Let us introduce $(N+1)$ new states, defined below as

$$\begin{aligned} x_{n+1}(k) &= u(k-N-1) \\ x_{n+2}(k) &= u(k-N) \\ &\vdots \\ x_{n+N+1}(k) &= u(k-1) \end{aligned}$$

The augmented state equation now becomes

$$\begin{bmatrix} \mathbf{x}(k+1) \\ x_{n+1}(k+1) \\ x_{n+2}(k+1) \\ \vdots \\ x_{n+N}(k+1) \\ x_{n+N+1}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{g}_1 & \mathbf{g}_2 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & 0 & 1 & 0 & \cdots & 0 \\ \mathbf{0} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \mathbf{0} & 0 & 0 & 0 & \cdots & 1 \\ \mathbf{0} & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ x_{n+1}(k) \\ x_{n+2}(k) \\ \vdots \\ x_{n+N}(k) \\ x_{n+N+1}(k) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ 0 \\ \mathbf{0} \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(k) \tag{6.36}$$

Example 6.2

In the following, we reconsider the tank fluid temperature control system discussed in Example 3.3 (refer to Fig. 3.17). The differential equation governing the tank fluid temperature was found to be

$$\dot{x}_1(t) = -x_1(t) + u(t-1.5) \tag{6.37}$$

where

$x_1(t) = \theta(t)$ = tank fluid temperature;
$u(t) = \theta_i(t)$ = temperature of the incoming fluid (control temperature); and
$\tau_D = 1.5$ sec.

Assume that the system is sampled with period $T = 1$ sec. From Eqn. (6.32), we have

$$N = 1, \Delta = 0.5, m = 0.5$$

Equations (6.34b), (6.34d), and (6.34e) give

$$\mathbf{F} = e^{-1} = 0.3679$$

$$\mathbf{g}_2 = \int_0^{0.5} e^{-\sigma}\, d\sigma = 1 - e^{-0.5} = 0.3935$$

$$\mathbf{g}_1 = e^{-0.5} \int_0^{0.5} e^{-\theta}\, d\theta = e^{-0.5} - e^{-1} = 0.2387$$

The discrete-time model of the tank fluid temperature control system becomes (refer to Eqn. (6.34a))

$$x_1(k+1) = 0.3679\, x_1(k) + 0.2387\, u(k-2) + 0.3935\, u(k-1) \tag{6.38}$$

Let us introduce two new states, defined below as

$$\begin{aligned} x_2(k) &= u(k-2) \\ x_3(k) &= u(k-1) \end{aligned}$$

The augmented state equation becomes

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{c}\mathbf{x}(k) \end{aligned} \tag{6.39}$$

with

$$\mathbf{F} = \begin{bmatrix} 0.3679 & 0.2387 & 0.3935 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{c} = [1 \quad 0 \quad 0]$$

From Eqns (6.39), the transfer function model is given as follows:

$$\begin{aligned} G(z) = \frac{Y(z)}{U(z)} &= \mathbf{c}(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{g} \\ &= [1 \ 0 \ 0]\begin{bmatrix} z-0.3679 & -0.2387 & -0.3935 \\ 0 & z & -1 \\ 0 & 0 & z \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \frac{1}{z^2(z-0.3679)}[1 \ 0 \ 0]\begin{bmatrix} z^2 & 0.2387z & 0.2387+0.3935z \\ 0 & z(z-0.3679) & z-0.3679 \\ 0 & 0 & z(z-0.3679) \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \frac{0.2387+0.3935z}{z^2(z-0.3679)} = \frac{0.3935(z+0.6066)}{z^2(z-0.3679)} \end{aligned} \tag{6.40}$$

Note that the same result was obtained earlier in Example 3.3.

6.5 SOLUTION OF STATE DIFFERENCE EQUATIONS

6.5.1 Solution by Recursion

In this section, we investigate the solution of the state equation

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k); \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{6.41}$$

where $\mathbf{x}$ is $n \times 1$ state vector, u is a scalar input, $\mathbf{F}$ is $n \times n$ real constant matrix, and $\mathbf{g}$ is $n \times 1$ real constant vector.

In general, discrete-time equations are easier to solve than differential equations because the former can be solved easily by means of a recursion procedure. The recursion procedure is quite simple and convenient for digital computations.

The solution of Eqn. (6.41) for any positive integer k may be obtained directly by recursion as follows. From $\mathbf{x}(0)$ and $u(0)$, $\mathbf{x}(1)$ can be calculated:

$$\mathbf{x}(1) = \mathbf{F}\mathbf{x}(0) + \mathbf{g}u(0) \tag{6.42a}$$

Then using $\mathbf{x}(1)$ and $u(1)$:

$$\mathbf{x}(2) = \mathbf{F}\mathbf{x}(1) + \mathbf{g}u(1) \tag{6.42b}$$

From $\mathbf{x}(2)$ and $u(2)$:

$$\mathbf{x}(3) = \mathbf{F}\mathbf{x}(2) + \mathbf{g}u(2) \tag{6.42c}$$

$$\vdots$$

From $\mathbf{x}(k-1)$ and $u(k-1)$:

$$\mathbf{x}(k) = \mathbf{F}\mathbf{x}(k-1) + \mathbf{g}u(k-1) \tag{6.42d}$$

6.5.2 Closed-Form Solution

In the following, we obtain the closed-form solution of state equation (6.41).

From Eqns (6.42a)–(6.42b), we obtain

$$\begin{aligned}\mathbf{x}(2) &= \mathbf{F}[\mathbf{F}\mathbf{x}(0) + \mathbf{g}u(0)] + \mathbf{g}u(1)\\ &= \mathbf{F}^2\mathbf{x}(0) + \mathbf{F}\mathbf{g}u(0) + \mathbf{g}u(1)\end{aligned} \tag{6.43}$$

From Eqns (6.43) and (6.42c), we get

$$\begin{aligned}\mathbf{x}(3) &= \mathbf{F}[\mathbf{F}^2\mathbf{x}(0) + \mathbf{F}\mathbf{g}u(0) + \mathbf{g}u(1)] + \mathbf{g}u(2)\\ &= \mathbf{F}^3\mathbf{x}(0) + \mathbf{F}^2\mathbf{g}u(0) + \mathbf{F}\mathbf{g}u(1) + \mathbf{g}u(2)\end{aligned}$$

By repeating this procedure, we obtain

$$\begin{aligned}\mathbf{x}(k) &= \mathbf{F}^k\mathbf{x}(0) + \mathbf{F}^{k-1}\,\mathbf{g}u(0) + \mathbf{F}^{k-2}\,\mathbf{g}u(1) + \cdots + \mathbf{F}^0\,\mathbf{g}u(k-1);\ \mathbf{F}^0 = \mathbf{I}\\ &= \mathbf{F}^k\mathbf{x}(0) + \sum_{i=0}^{k-1} \mathbf{F}^{k-1-i}\,\mathbf{g}u(i)\end{aligned} \tag{6.44}$$

Clearly $\mathbf{x}(k)$ consists of two parts: one representing the contribution of the initial state $\mathbf{x}(0)$, and the other the contribution of the input $u(i)$; $i = 0, 1, 2, ..., (k-1)$.

State Transition Matrix

Notice that it is possible to write the solution of the homogeneous state equations

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{6.45a}$$

as

$$\mathbf{x}(k) = \mathbf{F}^k\mathbf{x}(0) \tag{6.45b}$$

From Eqn. (6.45b) it is observed that the initial state $\mathbf{x}(0)$ at $k = 0$ is driven to the state $\mathbf{x}(k)$ at the sampling instant k. This transition in state is carried out by the matrix $\mathbf{F}^k$. Due to this property, $\mathbf{F}^k$ is known as the *state transition matrix*, and is denoted by $\boldsymbol{\phi}(k)$:

$$\boldsymbol{\phi}(k) = \mathbf{F}^k;\ \boldsymbol{\phi}(0) = \mathbf{I}\ \text{(Identity matrix)} \tag{6.46}$$

In the following, we discuss commonly used methods for evaluating state transition matrix in closed form.

Evaluation Using Inverse z-Transforms Taking the z-transform on both sides of Eqn. (6.45a), yields

$$z\mathbf{X}(z) - z\mathbf{x}(0) = \mathbf{F}\mathbf{X}(z)$$

where $\mathbf{X}(z) \triangleq \mathscr{Z}[\mathbf{x}(k)]$

Solving for $\mathbf{X}(z)$, we get

$$\mathbf{X}(z) = (z\mathbf{I} - \mathbf{F})^{-1} z\mathbf{x}(0)$$

The state vector $\mathbf{x}(k)$ can be obtained by inverse transforming $\mathbf{X}(z)$:

$$\mathbf{x}(k) = \mathscr{Z}^{-1}[(z\mathbf{I} - \mathbf{F})^{-1} z]\, \mathbf{x}(0)$$

Comparing this equation with Eqn. (6.45b), we get

$$\mathbf{F}^k = \boldsymbol{\phi}(k) = \mathscr{Z}^{-1}[(z\mathbf{I} - \mathbf{F})^{-1} z] \tag{6.47}$$

Example 6.3

Consider the matrix $\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -0.16 & -1 \end{bmatrix}$

For this $\mathbf{F}$, $(z\mathbf{I} - \mathbf{F})^{-1} = \begin{bmatrix} z & -1 \\ 0.16 & z+1 \end{bmatrix}^{-1}$

$$= \begin{bmatrix} \dfrac{z+1}{(z+0.2)(z+0.8)} & \dfrac{1}{(z+0.2)(z+0.8)} \\ \dfrac{-0.16}{(z+0.2)(z+0.8)} & \dfrac{z}{(z+0.2)(z+0.8)} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{4/3}{z+0.2} + \dfrac{-1/3}{z+0.8} & \dfrac{5/3}{z+0.2} + \dfrac{-5/3}{z+0.8} \\ \dfrac{-0.8/3}{z+0.2} + \dfrac{0.8/3}{z+0.8} & \dfrac{-1/3}{z+0.2} + \dfrac{4/3}{z+0.8} \end{bmatrix}$$

Therefore $\boldsymbol{\phi}(k) = \mathbf{F}^k = \mathscr{Z}^{-1}[(z\mathbf{I} - \mathbf{F})^{-1} z]$

$$= \mathscr{Z}^{-1} \begin{bmatrix} \frac{4}{3}\dfrac{z}{z+0.2} - \frac{1}{3}\dfrac{z}{z+0.8} & \frac{5}{3}\dfrac{z}{z+0.2} - \frac{5}{3}\dfrac{z}{z+0.8} \\ \frac{-0.8}{3}\dfrac{z}{z+0.2} + \frac{0.8}{3}\dfrac{z}{z+0.8} & \frac{-1}{3}\dfrac{z}{z+0.2} + \frac{4}{3}\dfrac{z}{z+0.8} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{4}{3}(-0.2)^k - \frac{1}{3}(-0.8)^k & \frac{5}{3}(-0.2)^k - \frac{5}{3}(-0.8)^k \\ \frac{-0.8}{3}(-0.2)^k + \frac{0.8}{3}(-0.8)^k & \frac{-1}{3}(-0.2)^k + \frac{4}{3}(-0.8)^k \end{bmatrix}$$

Evaluation Using Similarity Transformation Suppose that $\mathbf{F}$ is an $n \times n$ nondiagonal matrix with distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$. We define the diagonal matrix

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

F and **Λ** are similar matrices; there exists a nonsingular matrix **P** such that (refer to Eqns (5.22))

$$\mathbf{\Lambda} = \mathbf{P}^{-1}\,\mathbf{F}\mathbf{P}$$

Now

$$\mathbf{P}^{-1}\,\mathbf{F}^k\mathbf{P} = \mathbf{P}^{-1}\,[\mathbf{F}\mathbf{F}\cdots\mathbf{F}]\mathbf{P}$$
$$= \mathbf{P}^{-1}\,[(\mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1})\,(\mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1})\cdots(\mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1})]\mathbf{P} = \mathbf{\Lambda}^k$$

Thus the matrices $\mathbf{F}^k$ and $\mathbf{\Lambda}^k$ are similar. Since **Λ** is diagonal, $\mathbf{\Lambda}^k$ is given by

$$\mathbf{\Lambda}^k = \begin{bmatrix} \lambda_1^k & 0 & \cdots & 0 \\ 0 & \lambda_2^k & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n^k \end{bmatrix}$$

The state transition matrix $\mathbf{F}^k$ of matrix **F** with distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ may, therefore, be evaluated using the following relation:

$$\mathbf{F}^k = \mathbf{P}\mathbf{\Lambda}^k\,\mathbf{P}^{-1} = \mathbf{P}\begin{bmatrix} \lambda_1^k & 0 & \cdots & 0 \\ 0 & \lambda_2^k & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n^k \end{bmatrix}\mathbf{P}^{-1} \tag{6.48}$$

where **P** is a transformation matrix that transforms **F** into the diagonal form (For the general case where matrix **F** has multiple eigenvalues, refer to [105]; also refer to Review Example 6.5 given at the end of this chapter).

Example 6.4 *Consider the Matrix*

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

The characteristic equation of the system is

$$|\lambda\mathbf{I} - \mathbf{F}| = \lambda^2 + 3\lambda + 2 = 0$$

which yields $\lambda_1 = -1$ and $\lambda_2 = -2$ as the eigenvalues of **F**.

Since the matrix **F** is in companion form, the eigenvectors[3], and hence the transformation matrix, can easily be obtained (refer to Problem 5.21).

$$\mathbf{P} = \begin{bmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix}$$

gives the diagonalized matrix

$$\mathbf{\Lambda} = \mathbf{P}^{-1}\,\mathbf{F}\mathbf{P} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}$$

[3] Refer to Section 5.6 for methods of determination of eigenvectors for a given general matrix **F**.

From Eqn. (6.48), we may write

$$\mathbf{F}^k = \mathbf{P}\begin{bmatrix}(-1)^k & 0\\ 0 & (-2)^k\end{bmatrix}\mathbf{P}^{-1}$$

$$= \begin{bmatrix}1 & 1\\ -1 & -2\end{bmatrix}\begin{bmatrix}(-1)^k & 0\\ 0 & (-2)^k\end{bmatrix}\begin{bmatrix}2 & 1\\ -1 & -1\end{bmatrix}$$

$$= \begin{bmatrix}2(-1)^k - (-2)^k & (-1)^k - (-2)^k\\ -2(-1)^k + 2(-2)^k & -(-1)^k + 2(-2)^k\end{bmatrix}$$

Evaluation Using Cayley–Hamilton Technique The Cayley–Hamilton technique has already been explained in the earlier chapter. We illustrate the use of this technique for evaluation of $\mathbf{F}^k$ by the following example.

Example 6.5 *Consider the Matrix*

$$\mathbf{F} = \begin{bmatrix}0 & 1\\ -1 & -2\end{bmatrix}$$

Let us evaluate $f(\mathbf{F}) = \mathbf{F}^k$.

Matrix $\mathbf{F}$ has two eigenvalues at $\lambda_1 = \lambda_2 = -1$.

Since $\mathbf{F}$ is of second order, the polynomial $g(\lambda)$ will be of the form (refer to Eqns (5.98)):

$$g(\lambda) = \beta_0 + \beta_1\lambda$$

The coefficients β_0 and β_1 are evaluated from the following equations:

$$f(-1) = (-1)^k = \beta_0 - \beta_1$$

$$\frac{d}{d\lambda}f(\lambda)\bigg|_{\lambda=-1} = k(-1)^{k-1} = \frac{d}{d\lambda}g(\lambda)\bigg|_{\lambda=-1} = \beta_1$$

The result is

$$\beta_0 = (1-k)(-1)^k$$

$$\beta_1 = -k(-1)^k$$

Hence

$$f(\mathbf{F}) = \mathbf{F}^k = \beta_0\mathbf{I} + \beta_1\mathbf{F}$$

$$= (1-k)(-1)^k\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - k(-1)^k\begin{bmatrix}0 & 1\\ -1 & -2\end{bmatrix}$$

$$= (-1)^k\begin{bmatrix}(1-k) & -k\\ k & (1+k)\end{bmatrix}$$

State Transition Equation

The solution of the nonhomogeneous state difference equation (6.41) is given by Eqn. (6.44). In terms of the state transition matrix $\phi(k)$, Eqn. (6.44) can be written in the form

$$\mathbf{x}(k) = \phi(k)\,\mathbf{x}(0) + \sum_{i=0}^{k-1} \phi\,(k-1-i)\,\mathbf{g}u(i) \tag{6.49}$$

This equation is called the *state transition equation*; it describes the change of state relative to the initial conditions $\mathbf{x}(0)$ and the input $u(k)$.

Example 6.6 *Consider the System*

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (-1)^k$$

$$x_1(0) = 1 = x_2(0)$$

$$y(k) = x_1(k)$$

Find $y(k)$ for $k \geq 1$.

Solution For the given state equation, we have

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

For this $\mathbf{F}$, $\phi(k) = \mathbf{F}^k$ was evaluated in Example 6.4:

$$\phi(k) = \mathbf{F}^k = \begin{bmatrix} 2(-1)^k - (-2)^k & (-1)^k - (-2)^k \\ -2(-1)^k + 2(-2)^k & -(-1)^k + 2(-2)^k \end{bmatrix}$$

The state

$$\mathbf{x}(k) = \mathbf{F}^k\,\mathbf{x}(0) + \sum_{i=0}^{k-1} \mathbf{F}^{k-1-i}\,\mathbf{g}u(i)$$

With

$$\mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \text{and} \quad u(k) = (-1)^k,$$

we get

$$y(k) = x_1(k) = 3(-1)^k - 2(-2)^k + \sum_{i=0}^{k-1} [(-1)^{k-1-i} - (-2)^{k-1-i}]\,(-1)^i$$

$$= 3(-1)^k - 2(-2)^k + k(-1)^{k-1} - (-2)^{k-1} \sum_{i=0}^{k-1} \left(\tfrac{1}{2}\right)^i$$

Since[4]

$$\sum_{i=0}^{k-1} \left(\tfrac{1}{2}\right)^i = \frac{1-\left(\frac{1}{2}\right)^k}{1-\frac{1}{2}} = -2\left[\left(\tfrac{1}{2}\right)^k - 1\right], \text{ we have}$$

---

[4] $\displaystyle\sum_{j=0}^{k} a^j = \frac{1-a^{k+1}}{1-a}$; $a \neq 1$.

$$y(k) = 3(-1)^k - 2(-2)^k - k(-1)^k + (-2)^k\ [1-\left(\tfrac{1}{2}\right)^k]$$
$$= 3(-1)^k - 2(-2)^k - k(-1)^k + (-2)^k - (-1)^k = (2-k)\,(-1)^k - (-2)^k$$

6.6 CONTROLLABILITY AND OBSERVABILITY

In this section, we study the controllability and observability properties of linear time-invariant systems described by state variable model of the following form:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \tag{6.50a}$$
$$y(k) = \mathbf{c}\mathbf{x}(k) + du(k) \tag{6.50b}$$

where $\mathbf{F}$, $\mathbf{g}$, $\mathbf{c}$ and d are, respectively, $n \times n$, $n \times 1$, $1 \times n$, and 1×1 matrices. $\mathbf{x}$ is $n \times 1$ state vector, and y and u are, respectively, output and input variables.

6.6.1 Controllability

For the linear system given by Eqns (6.50), if there exists an input $u(k)$; $k \in [0, N-1]$ with N a finite positive integer, which transfers the initial state $\mathbf{x}(0) \triangleq \mathbf{x}^0$ to the state $\mathbf{x}^1$ at $k = N$, the state $\mathbf{x}^0$ is said to be controllable. If all initial states are controllable, the system is said to be *completely controllable* or simply *controllable*. Otherwise, the system is said to be *uncontrollable*.

The following theorem gives a simple controllability test.

Theorem 6.1 The necessary and sufficient condition for the system (6.50) to be completely controllable is that the $n \times n$ controllability matrix,

$$\mathbf{U} \triangleq [\mathbf{g} \quad \mathbf{F}\mathbf{g} \quad \mathbf{F}^2\mathbf{g} \cdots \mathbf{F}^{n-1}\mathbf{g}] \tag{6.51}$$

has rank equal to n, i.e., $\rho(\mathbf{U}) = n$.

Proof Solution of Eqn. (6.50a) is

$$\mathbf{x}(k) = \mathbf{F}^k\mathbf{x}(0) + \sum_{i=0}^{k-1} \mathbf{F}^{k-1-i}\,\mathbf{g}u(i)$$

Letting $\mathbf{x}(0) \triangleq \mathbf{x}^0$ and $\mathbf{x}(n) \triangleq \mathbf{x}^1$, we obtain

$$\mathbf{x}^1 - \mathbf{F}^n\mathbf{x}^0 = \mathbf{F}^{n-1}\mathbf{g}\,u(0) + \mathbf{F}^{n-2}\mathbf{g}u(1) + \cdots + \mathbf{g}u(n-1)$$

or

$$\mathbf{x}^1 - \mathbf{F}^n\mathbf{x}^0 = [\mathbf{g} \quad \mathbf{F}\mathbf{g} \cdots \mathbf{F}^{n-1}\mathbf{g}] \begin{bmatrix} u(n-1) \\ u(n-2) \\ \vdots \\ u(0) \end{bmatrix} \tag{6.52}$$

Since $\mathbf{g}$ is an $n \times 1$ matrix, we find that each of the matrices $\mathbf{g}$, $\mathbf{F}\mathbf{g}$, ..., $\mathbf{F}^{n-1}\mathbf{g}$ is an $n \times 1$ matrix. Therefore,

$$\mathbf{U} = [\mathbf{g} \quad \mathbf{F}\mathbf{g} \cdots \mathbf{F}^{n-1}\mathbf{g}]$$

is an $n \times n$ matrix. If the rank of $\mathbf{U}$ is n, then for arbitrary states $\mathbf{x}^0$ and $\mathbf{x}^1$, there exists a sequence of unconstrained control signals $u(0), u(1), ..., u(n-1)$ that satisfies Eqn. (6.52). Hence, the condition that the rank of the controllability matrix is n gives a sufficient condition for complete controllability.

To prove that the condition $\rho(\mathbf{U}) = n$ is also a necessary condition for complete controllability, we assume that

$$\rho[\mathbf{g} \quad \mathbf{Fg} \cdots \mathbf{F}^{n-1}\mathbf{g}] < n$$

The matrix $\mathbf{U}$ is, therefore, singular and for arbitrary $\mathbf{x}^0$ and $\mathbf{x}^1$, a solution $\{u(0), u(1), ..., u(n-1)\}$ satisfying Eqn. (6.52) does not exist.

Let us attempt a solution of the form $\{u(0), u(1), ..., u(N-1)\}$; $N > n$. This will amount to adding columns $\mathbf{F}^n\mathbf{g}, \mathbf{F}^{n+1}\mathbf{g}, ..., \mathbf{F}^{N-1}\mathbf{g}$ in the $\mathbf{U}$ matrix. But by Cayley–Hamilton theorem, $f(\mathbf{F}) = \mathbf{F}^j; j \geq n$ is a linear combination of $\mathbf{F}^{n-1}, \ldots, \mathbf{F}^1, \mathbf{F}^0$ (refer to Eqn. (5.98d)) and therefore, columns $\mathbf{F}^n\mathbf{g}, \mathbf{F}^{n+1}\mathbf{g}, ..., \mathbf{F}^{N-1}\mathbf{g}$ add no new rank. Thus, if a state cannot be transferred to some other state in n sampling intervals, no matter how long the input sequence $\{u(0), u(1), ..., u(N-1)\}$; $N > n$ is, it still cannot be achieved. Consequently, we find that the *rank condition* given by Eqn. (6.51) is necessary and sufficient condition for complete controllability.

6.6.2 Observability

For the linear system given by Eqns (6.50), if the knowledge of the input $u(k)$; $k \in [0, N-1]$ and the output $y(k)$; $k \in [0, N-1]$ with N a finite positive integer, suffices to determine the state $\mathbf{x}(0) \triangleq \mathbf{x}^0$, the state $\mathbf{x}^0$ is said to be observable. If all initial conditions are observable, the system is said to be *completely observable*, or simply *observable*. Otherwise, the system is said to be *unobservable*.

The following theorem gives a simple observability test.

Theorem 6.2 The necessary and sufficient condition for the system (6.50) to be completely observable is that the $n \times n$ observability matrix

$$\mathbf{V} = \begin{bmatrix} \mathbf{c} \\ \mathbf{cF} \\ \mathbf{cF}^2 \\ \vdots \\ \mathbf{cF}^{n-1} \end{bmatrix} \tag{6.53}$$

has rank equal to n, i.e., $\rho(\mathbf{V}) = n$.

Proof The solution of Eqns (6.50) is

$$y(k) = \mathbf{cF}^k\mathbf{x}(0) + \left[\sum_{i=0}^{k-1} \mathbf{cF}^{k-1-i}\mathbf{g}\, u(i)\right] + du(k)$$

This gives

$$\begin{aligned} y(0) &= \mathbf{cx}(0) + du(0) \\ y(1) &= \mathbf{cFx}(0) + \mathbf{cg}\, u(0) + du(1) \\ &\vdots \\ y(n-1) &= \mathbf{cF}^{n-1}\mathbf{x}(0) + \mathbf{cF}^{n-2}\mathbf{g}\, u(0) + \mathbf{cF}^{n-3}\mathbf{g}\, u(1) + \cdots + \mathbf{cg}\, u(n-2) + du(n-1) \end{aligned}$$

From these equations, we may write

$$\begin{bmatrix} y(0) - du(0) \\ y(1) - \mathbf{cg}\, u(0) - du(1) \\ \vdots \\ y(n-1) - \mathbf{cF}^{n-2}\mathbf{g}\, u(0) - \mathbf{cF}^{n-3}\mathbf{g}\, u(1) - \cdots - \mathbf{cg}\, u(n-2) - du(n-1) \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{cF} \\ \mathbf{cF}^2 \\ \vdots \\ \mathbf{cF}^{n-1} \end{bmatrix} \mathbf{x}^0 = \underset{n\times n}{\mathbf{V}} \times \underset{n\times 1}{\mathbf{x}^0} \tag{6.54}$$

If the rank of **V** is n, then there exists a unique solution $\mathbf{x}^0$ of Eqn. (6.54). Hence, the condition that the rank of the observability matrix is n, gives a sufficient condition for complete observability.

It can easily be proved (refer to proof of Theorem 6.1) that the condition $\rho(\mathbf{V}) = n$ is also a necessary condition for complete observability.

6.6.3 Controllability and Observability of State Variable Model in Jordan Canonical Form

The following result for discrete-time systems easily follows from the corresponding result for continuous-time systems, given in the earlier chapter.

Consider a SISO system with distinct eigenvalues[5] $\lambda_1, \lambda_2, ..., \lambda_n$.

The Jordan canonical state model of the system is of the form

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Lambda x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{cx}(k) + du(k) \end{aligned} \tag{6.55}$$

with $$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}; \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}; \mathbf{c} = [c_1 \quad c_2 \cdots c_n]$$

The system (6.55) is completely controllable if, and only if, none of the elements of the column matrix **g**, is zero. The system (6.55) is completely observable if, and only if, none of the elements of the row matrix **c**, is zero.

6.6.4 Equivalence Between Transfer Function and State Variable Representations

The following result for discrete-time systems easily follows from the corresponding result for continuous-time systems, given in the earlier chapter.

The general state variable model of nth-order linear time-invariant discrete-time system is given by Eqns (6.50):

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{Fx}(k) + \mathbf{g}u(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \\ y(k) &= \mathbf{cx}(k) + du(k) \end{aligned} \tag{6.56}$$

[5] Refer to Gopal [105] for the case of multiple eigenvalues.

The corresponding transfer function model is

$$\frac{Y(z)}{U(z)} = \frac{\mathbf{c}(z\mathbf{I}-\mathbf{F})^{+}\mathbf{g} + d\,|\,z\mathbf{I}-\mathbf{F}\,|}{|\,z\mathbf{I}-\mathbf{F}\,|} \tag{6.57}$$

The uncontrollable and unobservable modes of the state variable model (6.56) do not show up in the corresponding transfer function representation (6.57); the poles of the transfer function are, therefore, a subset of the eigenvalues of matrix **F**, and the asymptotic stability of the system always implies bounded-input, bounded-output (BIBO) stability. The reverse, however, may not be true because the eigenvalues of uncontrollable and/or unobservable parts of the system are hidden from the BIBO stability analysis. When the state variable model (6.56) is both controllable and observable, all the eigenvalues of **F** appear as poles in the transfer function (6.57), and therefore, BIBO stability implies asymptotic stability only for controllable and observable systems.

Conclusion The transfer function model of a system represents its complete dynamics only if the system is both controllable and observable.

6.6.5 Loss of Controllability and Observability due to Sampling

Sampling of a continuous-time system gives a discrete-time system with system matrices that depend on the sampling period. How will that influence the controllability and observability of the sampled system? To get a controllable sampled system, it is necessary that the continuous-time system also be controllable, because the allowable control signals for the sampled system—piecewise constant signals—are a subset of allowable control signals for the continuous-time system. However, it may happen that the controllability is lost for some sampling periods.

The conditions for unobservability are more restricted in the continuous-time case because the output has to be zero over a time interval, while the sampled system output has to be zero only at the sampling instants. This means that the continuous output may oscillate between the sampling times and be zero at the sampling instants. This condition is sometimes called *hidden oscillations*. The sampled system can thus be unobservable—even if the corresponding continuous-time system is observable.

The harmonic oscillator can be used to illustrate the preceding discussion. The transfer function model of the oscillator system is

$$\frac{Y(s)}{U(s)} = \frac{\omega^2}{s^2+\omega^2} \tag{6.58}$$

From this model, we have

$$\ddot{y} + \omega^2 y = \omega^2 u$$

Define

$$x_1 = y;\; x_2 = \frac{1}{\omega}\dot{y}$$

This gives the following state variable representation of the oscillator system:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & \omega\\ -\omega & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ \omega\end{bmatrix}u$$
$$y = [1 \quad 0]\begin{bmatrix}x_1\\ x_2\end{bmatrix} \tag{6.59}$$

The discrete-time state variable representation of the system is obtained as follows. Noting that

$$\mathbf{A} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ \omega \end{bmatrix}$$

we have

$$\mathbf{F} = e^{\mathbf{A}T} = \mathscr{L}^{-1}\,[(s\mathbf{I} - \mathbf{A})^{-1}]|_{t=T} = \mathscr{L}^{-1}\left(\begin{bmatrix} s & -\omega \\ \omega & s \end{bmatrix}^{-1}\right)\Bigg|_{t=T}$$

$$= \mathscr{L}^{-1}\begin{bmatrix} \dfrac{s}{s^2+\omega^2} & \dfrac{\omega}{s^2+\omega^2} \\ \dfrac{-\omega}{s^2+\omega^2} & \dfrac{s}{s^2+\omega^2} \end{bmatrix}\Bigg|_{t=T} = \begin{bmatrix} \cos\omega T & \sin\omega T \\ -\sin\omega T & \cos\omega T \end{bmatrix}$$

and

$$\mathbf{g} = \left[\int_0^T e^{\mathbf{A}\theta}\,d\theta\right]\mathbf{b} = \left[\int_0^T \begin{pmatrix} \cos\omega\theta & \sin\omega\theta \\ -\sin\omega\theta & \cos\omega\theta \end{pmatrix} d\theta\right]\begin{bmatrix} 0 \\ \omega \end{bmatrix} = \begin{bmatrix} 1-\cos\omega T \\ \sin\omega T \end{bmatrix}$$

Hence, the discrete-time state variable representation of the oscillator system becomes

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} \cos\omega T & \sin\omega T \\ -\sin\omega T & \cos\omega T \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1-\cos\omega T \\ \sin\omega T \end{bmatrix} u(k) \tag{6.60}$$

$$y(k) = [1 \quad 0]\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

The determinants of the controllability and observability matrices are

$$|\mathbf{U}| = |[\mathbf{g} \quad \mathbf{Fg}]| = -2\sin\omega T\,(1 - \cos\omega T)$$

$$|\mathbf{V}| = \left|\begin{bmatrix} \mathbf{c} \\ \mathbf{cF} \end{bmatrix}\right| = \sin\omega T$$

Both controllability and observability are lost for $\omega T = n\pi$, $n = 1, 2, \ldots$ (i.e., when the sampling interval is half the period of oscillation of the harmonic oscillator, or an integer multiple of that period), although the corresponding continuous-time system given by Eqns (6.59), is both controllable and observable.

Loss of controllability and/or observability due to sampling occurs only when the continuous-time system has oscillatory modes and the sampling interval is half the period of oscillation of an oscillatory mode, or an integer multiple of that period. This implies that controllability and observability properties of a continuous-time system, are preserved after introduction of sampling if, and only if, for every eigenvalue of the characteristic equation, the relation

$$\text{Re}\,\lambda_i = \text{Re}\,\lambda_j \tag{6.61}$$

implies

$$\text{Im}\,(\lambda_i - \lambda_j) \neq \frac{2n\pi}{T}$$

where T is the sampling period and $n = \pm 1, \pm 2, \ldots$.

We know that controllability and/or observability is lost when the transfer function corresponding to a state model has common poles and zeros. The poles and zeros are functions of sampling interval. This implies that if the choice of sampling interval does not satisfy the condition given by (6.61), pole-zero cancellation will occur in passing from the continuous-time to the discrete-time case; the pole-zero cancellation will not take place if the continuous-time system does not contain complex poles.

It is very unlikely that the sampling interval chosen for a plant control system, would be precisely the one resulting in loss of controllability and/or observability. In fact the rules of thumb, for the choice of sampling interval given in Section 2.13, imply a sampling interval of about one tenth of the period of oscillation of an oscillatory mode, and not just half.

6.7 MULTIVARIABLE SYSTEMS

The state variable model of the multi-input, multi-output (MIMO) system takes the following form (refer to Eqns (2.14)–(2.15)):

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{6.62a}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \tag{6.62b}$$

F, **G**, **C**, and **D** are, respectively, $n \times n$, $n \times p$, $q \times n$ and $q \times p$ constant matrices, **x** is $n \times 1$ state vector, **u** is $p \times 1$ input vector, and **y** is $q \times 1$ output vector.

Many of the analysis results developed in earlier sections of this chapter for SISO systems, have obvious extensions for the system (6.62).

The solution of the state equation (6.62a) is given by (refer to Eqn. (6.44))

$$\mathbf{x}(k) = \mathbf{F}^k\mathbf{x}(0) + \sum_{i=0}^{k-1} \mathbf{F}^{k-1-i}\mathbf{G}\mathbf{u}(i) \tag{6.63a}$$

The output

$$\mathbf{y}(k) = \mathbf{C}\left[\mathbf{F}^k\mathbf{x}(0) + \sum_{i=0}^{k-1} \mathbf{F}^{k-1-i}\mathbf{G}\mathbf{u}(i)\right] + \mathbf{D}\mathbf{u}(k) \tag{6.63b}$$

In the transform domain, the input-output behavior of the system (6.62) is determined entirely by the *transfer function matrix* (refer to Eqns (6.3))

$$\bar{\mathbf{G}}(z) = \mathbf{C}(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{G} + \mathbf{D} \tag{6.64a}$$

The output

$$\underset{q\times 1}{\mathbf{Y}(z)} = \underset{q\times p}{\bar{\mathbf{G}}(z)}\ \underset{p\times 1}{\mathbf{U}(z)} \tag{6.64b}$$

The necessary and sufficient condition for the system (6.62) to be completely controllable is that the $n \times np$ matrix

$$\mathbf{U} \triangleq [\mathbf{G} \quad \mathbf{F}\mathbf{G} \quad \mathbf{F}^2\mathbf{G} \cdots \mathbf{F}^{n-1}\mathbf{G}] \tag{6.65}$$

has rank equal to n.

The necessary and sufficient condition for the system (6.62) to be completely observable is that the $nq \times n$ matrix

$$\mathbf{V} \triangleq \begin{bmatrix} \mathbf{C} \\ \mathbf{CF} \\ \vdots \\ \mathbf{CF}^{n-1} \end{bmatrix} \tag{6.66}$$

has rank equal to n.

A MIMO system with distinct eigenvalues[6] $\lambda_1, \lambda_2, ..., \lambda_n$ has the following Jordan canonical state model.

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Lambda x}(k) + \mathbf{Gu}(k) \\ \mathbf{y}(k) &= \mathbf{Cx}(k) + \mathbf{Du}(k) \end{aligned} \tag{6.67}$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}; \mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1p} \\ g_{21} & g_{22} & \cdots & g_{2p} \\ \vdots & \vdots & & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{np} \end{bmatrix}; \mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ \vdots & \vdots & & \vdots \\ c_{q1} & c_{q2} & \cdots & c_{qn} \end{bmatrix}$$

The system (6.67) is completely controllable if, and only if, none of the rows of **G** matrix is a zero row, and (6.67) is completely observable if, and only if, none of the columns of **C** matrix is a zero column.

Example 6.7

The scheme of Fig. 5.23 (refer to Example 5.21) describes a simple concentration control process. Mathematical model of the plant, given by Eqns (5.133), is reproduced below.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} \end{aligned} \tag{6.68}$$

with $$\mathbf{A} = \begin{bmatrix} -0.01 & 0 \\ 0 & -0.02 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} 1 & 1 \\ -0.004 & -0.002 \end{bmatrix}; \mathbf{C} = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$$

The state, input, and output variables are deviations from steady-state values:

x_1 = incremental volume of fluid in the tank (liters)
x_2 = incremental outgoing concentration (g-moles/liter)
u_1 = incremental feed 1 (liters/sec)
u_2 = incremental feed 2 (liters/sec)
y_1 = incremental outflow (liters/sec)
y_2 = incremental outgoing concentration (g-moles/liter)

Matrix **A** in Eqns (6.68) is in diagonal form; none of the rows of **B** matrix is a zero row, and none of the columns of **C** matrix is a zero column. The state model (6.68) is, therefore, completely controllable and observable.

[6] Refer to Gopal [105] for the case of multiple eigenvalues.

With initial values of x_1 and x_2 equal to zero at $t = 0$, a step of 2 liters/sec in feed 1 results in

$$y(t) = \mathbf{C}\left[\int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau\right]$$

with $$e^{\mathbf{A}t} = \begin{bmatrix} e^{-0.01t} & 0 \\ 0 & e^{-0.02t} \end{bmatrix}$$

and $$\mathbf{u}(\tau) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Solving for $\mathbf{y}(t)$, we get

$$\mathbf{y}(t) = \mathbf{C}\begin{bmatrix} \int_0^t 2e^{-0.01(t-\tau)}d\tau \\ -\int_0^t 0.008e^{-0.02(t-\tau)}d\tau \end{bmatrix} = \mathbf{C}\begin{bmatrix} \frac{2}{0.01}(1-e^{-0.01t}) \\ -0.4\,(1-e^{-0.02t}) \end{bmatrix}$$

Therefore,
$$y_1(t) = 2(1 - e^{-0.01t}) \tag{6.69a}$$
$$y_2(t) = -0.4(1 - e^{-0.02t}) \tag{6.69b}$$

Suppose that the plant (6.68) forms part of a process commanded by a process control computer. As a result, the valve settings change at discrete instants only and remain constant in between. Assuming that these instants are separated by time period $T = 5$ sec, we derive the discrete-time description of the plant.

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}u(k) \tag{6.70a}$$
$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \tag{6.70b}$$
$$\mathbf{F} = e^{\mathbf{A}T} = \begin{bmatrix} e^{-0.01T} & 0 \\ 0 & e^{-0.02T} \end{bmatrix} = \begin{bmatrix} 0.9512 & 0 \\ 0 & 0.9048 \end{bmatrix} \tag{6.70c}$$
$$\mathbf{G} = \int_0^T e^{\mathbf{A}\theta}\mathbf{B}d\theta = \begin{bmatrix} \int_0^T e^{-0.01\theta}d\theta & \int_0^T e^{-0.01\theta}d\theta \\ -0.004\int_0^T e^{-0.02\theta}d\theta & 0.002\int_0^T e^{-0.02\theta}d\theta \end{bmatrix} = \begin{bmatrix} 4.88 & 4.88 \\ -0.019 & 0.0095 \end{bmatrix} \tag{6.70d}$$

Matrix **F** in Eqns (6.70) is in diagonal form; none of the rows of **G** matrix is a zero row, and none of the columns of **C** matrix is a zero column. The state model (6.70) is, therefore, completely controllable and observable.

With initial values of x_1 and x_2 equal to zero at $k = 0$, a step of 2 liters/sec in feed 1 results in

$$\mathbf{y}(k) = \mathbf{C}\left[\sum_{i=0}^{k-1}\mathbf{F}^{k-1-i}\mathbf{G}\mathbf{u}(i)\right]$$

with $\mathbf{F}^k = \begin{bmatrix} (0.9512)^k & 0 \\ 0 & (0.9048)^k \end{bmatrix}$ and $\mathbf{u}(i) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$

Solving for $\mathbf{y}(k)$, we get

$$y_1(k) = 0.01x_1(k) = 0.01 \sum_{i=0}^{k-1} 9.76(0.9512)^{k-1-i}$$

Since (refer to footnote 4)

$$\sum_{i=0}^{k-1}\left(\frac{1}{0.9512}\right)^i = \frac{1-\left(\frac{1}{0.9512}\right)^k}{1-\frac{1}{0.9512}} = \frac{0.9512}{-0.0488}\,[1-(0.9512)^{-k}]$$

we have

$$y_1(k) = 2[1-(0.9512)^k] \tag{6.71a}$$

$$y_2(k) = x_2(k) = -0.038 \sum_{i=0}^{k-1} (0.9048)^{k-1-i} = -0.4\,[1-(0.9048)^k] \tag{6.71b}$$

Comparison of $y_1(k)$ and $y_2(k)$ with $y_1(t)$ and $y_2(t)$, shows that the two sets of responses match exactly at the sampling instants.

REVIEW EXAMPLES

Review Example 6.1

Give three different canonical state variable models corresponding to the transfer function

$$G(z) = \frac{4z^3 - 12z^2 + 13z - 7}{(z-1)^2(z-2)}$$

Solution The given transfer function is

$$G(z) = \frac{Y(z)}{U(z)} = \frac{4z^3 - 12z^2 + 13z - 7}{z^3 - 4z^2 + 5z - 2} = \frac{\beta_0 z^3 + \beta_1 z^2 + \beta_2 z + \beta_3}{z^3 + \alpha_1 z^2 + \alpha_2 z + \alpha_3}$$

The controllable canonical state model (first companion form) follows directly from Eqns (6.5) and (6.7):

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & -5 & 4 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = [1 \quad -7 \quad 4] \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + 4u(k)$$

The observable canonical state model (second companion form) follows directly from Eqns (6.5) and (6.8):

$$\begin{bmatrix} \bar{x}_1(k+1) \\ \bar{x}_2(k+1) \\ \bar{x}_3(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & -5 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} \bar{x}_1(k) \\ \bar{x}_2(k) \\ \bar{x}_3(k) \end{bmatrix} + \begin{bmatrix} 1 \\ -7 \\ 4 \end{bmatrix} u(k)$$

$$y(k) = [0 \quad 0 \quad 1] \begin{bmatrix} \bar{x}_1(k) \\ \bar{x}_2(k) \\ \bar{x}_3(k) \end{bmatrix} + 4u(k)$$

The state variable model in Jordan canonical form follows from Eqns (6.11) and (6.12):

$$\frac{Y(z)}{U(z)} = G(z) = \frac{4z^3 - 12z^2 + 13z - 7}{z^3 - 4z^2 + 5z - 2} = 4 + \frac{4z^2 - 7z + 1}{(z-1)^2(z-2)}$$

$$= 4 + \frac{2}{(z-1)^2} + \frac{1}{(z-1)} + \frac{3}{(z-2)}$$

$$\begin{bmatrix} \hat{x}_1(k+1) \\ \hat{x}_2(k+1) \\ \hat{x}_3(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \hat{x}_3(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = [2 \quad 1 \quad 3] \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \hat{x}_3(k) \end{bmatrix} + 4u(k)$$

Review Example 6.2

Prove that a discrete-time system obtained by zero-order-hold sampling of an asymptotically stable continuous-time system is also asymptotically stable.

Solution Consider an asymptotically stable continuous-time system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \tag{6.72}$$

We assume for simplicity, that the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ of matrix $\mathbf{A}$ are all distinct. Let $\mathbf{P}$ be a transformation matrix such that

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

This gives (refer to Eqn. (5.93))

$$\mathbf{P}^{-1}e^{\mathbf{A}t}\mathbf{P} = e^{\mathbf{\Lambda}t}$$
$$= \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix}$$

The zero-order-hold sampling of the continuous-time system (6.72) results in a discrete-time system

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k)$$

where $\mathbf{F} = e^{\mathbf{A}T}$; T = sampling interval

The characteristic polynomial of the system is

$$|z\mathbf{I} - \mathbf{F}| = |z\mathbf{I} - e^{\mathbf{A}T}| = |\mathbf{P}^{-1}|\,|z\mathbf{I} - e^{\mathbf{A}T}|\,|\mathbf{P}|$$
$$= |z\mathbf{P}^{-1}\mathbf{P} - \mathbf{P}^{-1}e^{\mathbf{A}T}\,\mathbf{P}| = |z\mathbf{I} - e^{\mathbf{\Lambda}T}|$$
$$= (z - e^{\lambda_1 T})\,(z - e^{\lambda_2 T}) \cdots (z - e^{\lambda_n T})$$

Notice that the eigenvalues of $\mathbf{F}$ are given by $z_i = e^{\lambda_i T}$; $i = 1, 2, ..., n$. We see the equivalence of $\mathrm{Re}\,\lambda_i < 0$ and $|z_i| < 1$. Thus, the discrete-time system, obtained by zero-order-hold sampling of an asymptotically stable continuous-time system, is also asymptotically stable.

The proof for the case where matrix $\mathbf{A}$ has multiple eigenvalues follows on identical lines.

Review Example 6.3

Consider a unity-feedback system with the plant

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$
$$y = \mathbf{c}\mathbf{x}$$

where $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ K \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

(a) Find the range of values of K for which the closed-loop system is stable.

(b) Introduce now a sampler and zero-order hold in the forward path of the closed-loop system. Show that sampling has a destabilizing effect on the stability of the closed-loop system. To establish this result, you may find the range of values of K for which the closed-loop digital system is stable when (i) $T = 0.4$ sec, and (ii) $T = 3$ sec; and then compare with that obtained in (a) above.

Solution Consider the feedback system of Fig. 6.12a. Substituting

$$u = r - y = r - x_1$$

in the plant model, we get the following state variable description of the closed-loop system:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -K & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ K \end{bmatrix} r$$

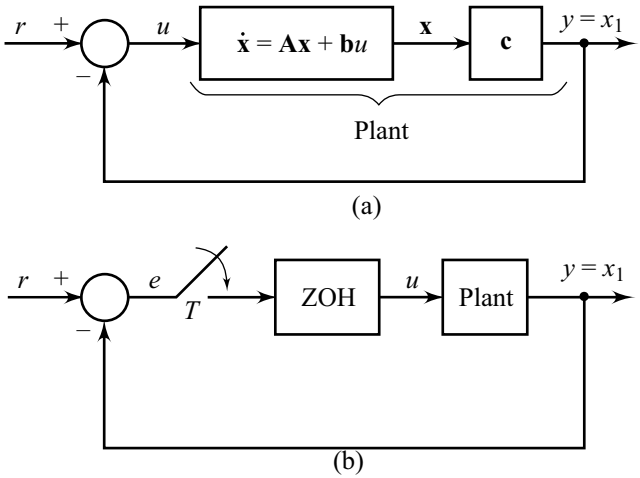


Fig. 6.12

The characteristic equation of the closed-loop system is

$$\lambda^2 + 2\lambda + K = 0$$

The closed-loop system is stable for all values of $K > 0$.

Figure 6.12b shows a block diagram of the closed-loop digital system. The discrete-time description of the plant is obtained as follows:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}\,u(k)$$
$$y(k) = \mathbf{c}\mathbf{x}(k)$$

where
$$\mathbf{F} = e^{\mathbf{A}T}; \text{ and } \mathbf{g} = \int_0^T e^{\mathbf{A}\theta}\,\mathbf{b}\,d\theta$$

Now
$$e^{\mathbf{A}t} = \mathscr{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}] = \mathscr{L}^{-1}\left(\begin{bmatrix} s & -1 \\ 0 & s+2 \end{bmatrix}^{-1}\right) = \mathscr{L}^{-1}\begin{bmatrix} \frac{1}{s} & \frac{1}{s(s+2)} \\ 0 & \frac{1}{s+2} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \frac{1}{2}(1-e^{-2t}) \\ 0 & e^{-2t} \end{bmatrix}$$

Therefore,
$$\mathbf{F} = e^{\mathbf{A}T} = \begin{bmatrix} 1 & \frac{1}{2}(1-e^{-2T}) \\ 0 & e^{-2T} \end{bmatrix}$$

$$\mathbf{g} = \int_0^T \begin{bmatrix} \frac{K}{2}(1-e^{-2\theta}) \\ Ke^{-2\theta} \end{bmatrix} d\theta = \frac{1}{2}K \begin{bmatrix} T - \frac{1}{2} + \frac{1}{2}e^{-2T} \\ 1 - e^{-2T} \end{bmatrix}$$

The state variable description of the closed-loop digital system, becomes

$$\mathbf{x}(k+1) = \begin{bmatrix} 1-\frac{1}{2}K(T-\frac{1}{2}+\frac{1}{2}e^{-2T}) & \frac{1}{2}(1-e^{-2T}) \\ -\frac{1}{2}K(1-e^{-2T}) & e^{-2T} \end{bmatrix} \mathbf{x}(k) + \frac{1}{2}K \begin{bmatrix} T-\frac{1}{2}+\frac{1}{2}e^{-2T} \\ 1-e^{-2T} \end{bmatrix} r(k)$$

The characteristic equation is given by

$$\lambda^2 + [-(1+e^{-2T}) + \tfrac{1}{2}K(T-\tfrac{1}{2}+\tfrac{1}{2}e^{-2T})]\lambda + e^{-2T} + \tfrac{1}{2}K(\tfrac{1}{2} - \tfrac{1}{2}e^{-2T} - Te^{-2T}) = 0$$

Case I: $T = 0.4$ sec

For this value of sampling period, the characteristic polynomial becomes

$$\Delta(\lambda) = \lambda^2 + (0.062K - 1.449)\lambda + 0.449 + 0.048K$$

Applying the Jury stability test (refer to Eqns (2.73)–(2.75)), we find that the system is stable if the following conditions are satisfied:

$$\Delta(1) = 1 + 0.062K - 1.449 + 0.449 + 0.048K > 0$$

$$\Delta(-1) = 1 - 0.062K + 1.449 + 0.449 + 0.048K > 0$$

$$|0.449 + 0.048K| < 1$$

These conditions are satisfied for $0 < K < 11.479$.

Case II: $T = 3$ sec

For this value of the sampling period, the characteristic polynomial becomes

$$\Delta(\lambda) = \lambda^2 + (1.2506K - 1.0025)\lambda + 0.0025 + 0.2457K$$

The system is found to be stable for $0 < K < 1.995$.

Thus, the system which is stable for all $K > 0$ when $T = 0$ (continuous-time system), becomes unstable for $K > 11.479$ when $T = 0.4$ sec. When T is increased to 3 sec, it becomes unstable for $K > 1.995$. It means that increasing the sampling period (or decreasing the sampling rate) reduces the margin of stability.

Review Example 6.4

A closed-loop computer control system is shown in Fig. 6.13. The digital controller is described by the difference equation

$$e_2(k+1) + ae_2(k) = be_1(k)$$

The state variable model of the plant is given below.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

$$y = \mathbf{c}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \quad 0]$$

Obtain discrete-time state description for the closed-loop system.

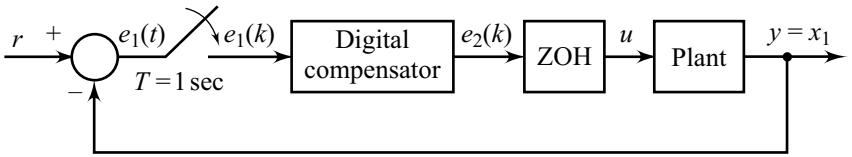


Fig. 6.13

Solution

Given $$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}$$

$$e^{\mathbf{A}t} = \mathscr{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}] = \mathscr{L}^{-1}\begin{bmatrix} \frac{1}{s} & \frac{1}{s(s+1)} \\ 0 & \frac{1}{s+1} \end{bmatrix} = \begin{bmatrix} 1 & 1-e^{-t} \\ 0 & e^{-t} \end{bmatrix}$$

The discretized state equation of the plant is

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \tag{6.73a}$$

where $$\mathbf{F} = e^{\mathbf{A}T} = \begin{bmatrix} 1 & 1-e^{-T} \\ 0 & e^{-T} \end{bmatrix}$$

$$\mathbf{g} = \int_0^T e^{\mathbf{A}\theta}\,\mathbf{b}\,d\theta = \begin{bmatrix} \int_0^T (1-e^{-\theta})\,d\theta \\ \int_0^T e^{-\theta}\,d\theta \end{bmatrix} = \begin{bmatrix} T-1+e^{-T} \\ 1-e^{-T} \end{bmatrix} \tag{6.73b}$$

For $T = 1$ sec, we have

$$\mathbf{F} = \begin{bmatrix} 1 & 0.632 \\ 0 & 0.368 \end{bmatrix};\ \mathbf{g} = \begin{bmatrix} 0.368 \\ 0.632 \end{bmatrix} \tag{6.73c}$$

Consider now, the feedback system of Fig. 6.13 with the plant described by the equation (refer to Eqns 6.73)):

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.632 \\ 0 & 0.368 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.368 \\ 0.632 \end{bmatrix} e_2(k) \tag{6.74}$$

$e_2(k)$ may be taken as the third state variable $x_3(k)$ whose dynamics are given by

$$x_3(k+1) = -a\,x_3(k) + b\,e_1(k) = -a\,x_3(k) + b\,(r(k) - x_1(k)) = -b\,x_1(k) - a\,x_3(k) + b\,r(k) \tag{6.75}$$

From Eqns (6.74)–(6.75), we get the following state variable model for the closed-loop digital system of Fig. 6.13:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.632 & 0.368 \\ 0 & 0.368 & 0.632 \\ -b & 0 & -a \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix} r(k)$$

$$y(k) = [1 \quad 0 \quad 0] \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}$$

Review Example 6.5

Given

$$\underset{n \times n}{\mathbf{\Lambda}} = \begin{bmatrix} \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & \lambda_1 \end{bmatrix}$$

Compute $\mathbf{\Lambda}^k$ using the Cayley–Hamilton technique.

Solution Equations (5.98) outline the procedure of evaluation of functions of a matrix using the Cayley–Hamilton technique.

The matrix $\mathbf{\Lambda}$ has n eigenvalues at $\lambda = \lambda_1$. To evaluate $f(\mathbf{\Lambda}) = \mathbf{\Lambda}^k$, we define (refer to Eqn. (5.98b)) the polynomial $g(\lambda)$ as

$$g(\lambda) = \beta_0 + \beta_1 \lambda + \cdots + \beta_{n-1} \lambda^{n-1}$$

This polynomial may be rearranged as

$$g(\lambda) = b_0 + b_1(\lambda - \lambda_1) + \cdots + b_{n-1} (\lambda - \lambda_1)^{n-1}$$

The coefficients $b_0, b_1, ..., b_{n-1}$ are given by the following equations (refer to Eqns (5.98c)):

$$f(\lambda_1) = g(\lambda_1)$$

$$\frac{d}{d\lambda} f(\lambda)\bigg|_{\lambda=\lambda_1} = \frac{d}{d\lambda} g(\lambda)\bigg|_{\lambda=\lambda_1}$$

$$\vdots$$

$$\frac{d^{n-1}}{d\lambda^{n-1}} f(\lambda)\bigg|_{\lambda=\lambda_1} = \frac{d^{n-1}}{d\lambda^{n-1}} g(\lambda)\bigg|_{\lambda=\lambda_1}$$

Solving, we get

$$b_0 = \lambda_1^k$$

$$b_1 = \frac{k}{1!} \lambda_1^{k-1}$$

$$b_2 = \frac{k(k-1)}{2!} \lambda_1^{k-2}$$

$$\vdots$$

$$b_{n-1} = \frac{k(k-1)(k-2)\cdots(k-n+2)}{(n-1)!}\lambda_1^{k-n+1}$$
$$= \frac{k(k-1)(k-2)\cdots(k-n+2)(k-n+1)(k-n)\cdots 1}{(k-n+1)(k-n)\cdots 1}\left[\frac{1}{(n-1)!}\right]\lambda_1^{k-n+1}$$
$$= \frac{k!}{(k-n+1)!(n-1)!}\lambda_1^{k-n+1}$$

Therefore (refer to Review Example 5.3),

$$\mathbf{\Lambda}^k = b_0\mathbf{I} + b_1(\mathbf{\Lambda} - \lambda_1\mathbf{I}) + \cdots + b_{n-1}(\mathbf{\Lambda} - \lambda_1\mathbf{I})^{n-1}$$
$$= \begin{bmatrix} \lambda_1^k & \frac{k}{1!}\lambda_1^{k-1} & \frac{k(k-1)}{2!}\lambda_1^{k-2} & \cdots & \frac{k!\lambda_1^{k-n+1}}{(k-n+1)!(n-1)!} \\ 0 & \lambda_1^k & \frac{k}{1!}\lambda_1^{k-1} & \cdots & \bullet \\ 0 & 0 & \lambda_1^k & \cdots & \bullet \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_1^k \end{bmatrix}$$

PROBLEMS

6.1 A system is described by the state equation

$$\mathbf{x}(k+1) = \begin{bmatrix} -3 & 1 & 0 \\ -4 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} -3 \\ -7 \\ 0 \end{bmatrix}u(k);\ \mathbf{x}(0) = \mathbf{x}^0$$

Using the z-transform technique, transform the state equation into a set of linear algebraic equations in the form

$$\mathbf{X}(z) = \mathbf{G}(z)\mathbf{x}^0 + \mathbf{H}(z)U(z)$$

6.2 Give a block diagram for digital realization of the state equation of Problem 6.1.

6.3 Obtain the transfer function description for the following system:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 2 & -5 \\ \frac{1}{2} & -1 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}u(k);\ y(k) = 2x_1(k)$$

6.4 A second-order multivariable system is described by the following equations:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 2 & -5 \\ \frac{1}{2} & -1 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 & -2 & 0 \\ 0 & 1 & 3 \end{bmatrix}\begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \end{bmatrix}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 & 4 & 0 \\ 0 & 0 & -2 \end{bmatrix}\begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \end{bmatrix}$$

Convert the state variable model into a transfer function matrix.

6.5 The state diagram of a linear system is shown in Fig. P6.5. Assign the state variables and write the dynamic equations of the system.

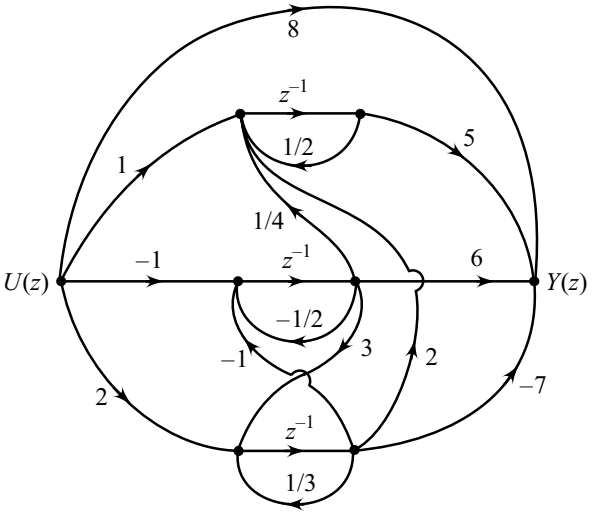


Fig. P6.5

6.6 Set up a state variable model for the system of Fig. P6.6.

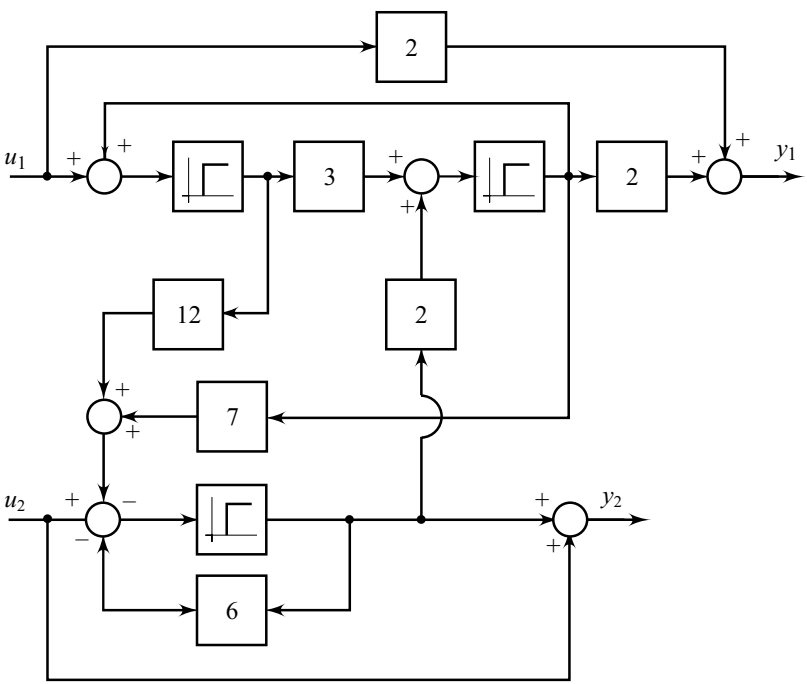


Fig. P6.6

6.7 Obtain the companion form realizations for the following transfer functions. Obtain different companion form for each system.

(i) $$\frac{Y(z)}{R(z)} = \frac{3z^2 - z - 3}{z^2 + \frac{1}{3}z - \frac{2}{3}}$$

(ii) $$\frac{Y(z)}{R(z)} = \frac{-2z^3 + 2z^2 - z + 2}{z^3 + z^2 - z - \frac{3}{4}}$$

6.8 Obtain the Jordan canonical form realizations for the following transfer functions.

(i) $$\frac{Y(z)}{R(z)} = \frac{z^3 + 8z^2 + 17z + 8}{(z+1)(z+2)(z+3)}$$

(ii) $$\frac{Y(z)}{R(z)} = \frac{3z^3 - 4z + 6}{(z - \frac{1}{3})^3}$$

6.9 Find state variable models for the following difference equations. Obtain different canonical form for each system.

(i) $y(k+3) + 5\,y(k+2) + 7\,y(k+1) + 3\,y(k) = 0$

(ii) $y(k+2) + 3\,y(k+1) + 2\,y(k) = 5\,r(k+1) + 3\,r(k)$

(iii) $y(k+3) + 5\,y(k+2) + 7\,y(k+1) + 3\,y(k) = r(k+1) + 2\,r(k)$

6.10 Given

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -3 & 4 \end{bmatrix}$$

Determine $\phi(k) = \mathbf{F}^k$ using

(a) the z-transform technique;

(b) similarity transformation; and

(c) Cayley–Hamilton technique.

6.11 Consider the system

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k);\ \mathbf{x}(0) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$y(k) = \mathbf{c}\mathbf{x}(k)$$

with $$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -0.16 & -1 \end{bmatrix};\ \mathbf{g} = \begin{bmatrix} 1 \\ 1 \end{bmatrix};\ \mathbf{c} = [1 \quad 0]$$

Find the closed-form solution for $y(k)$ when $u(k)$ is unit-step sequence.

6.12 Consider the system

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{d}u(k)$$

with $$\mathbf{F} = \begin{bmatrix} \frac{3}{2} & -1 \\ 1 & -1 \end{bmatrix};\ \mathbf{g} = \begin{bmatrix} 3 \\ 2 \end{bmatrix};\ \mathbf{x}(0) = \begin{bmatrix} -5 \\ 1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} -3 & 4 \\ -1 & 1 \end{bmatrix};\ \mathbf{d} = \begin{bmatrix} -2 \\ 0 \end{bmatrix};\ u(k) = \left(\tfrac{1}{2}\right)^k,\ k \geq 0$$

Find the response $y(k)$, $k \geq 0$.

6.13 Consider the system

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k)$$

with

$$\mathbf{F} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

(a) Find the modes of the free response.

(b) Find $\mathbf{x}(k)$ for

$$\mathbf{x}(0) = [0 \quad 1 \quad 1]^T$$

6.14 Consider the continuous-time system

$$G_a(s) = \frac{Y(s)}{R(s)} = \frac{1}{s(s+2)}$$

Insert sample-and-hold devices and determine the vector difference state model for digital simulation of the continuous-time system when the computation interval is $T = 1$ sec. Use the following methods to obtain the simulation model.

(a) Obtain $G(z)$ by taking the z transform of $G_a(s)$ when it is preceded by a sampler-and-hold; convert $G(z)$ into a vector difference state model.

(b) Obtain a continuous-time state model for the given $G_a(s)$; insert sample-and-hold and discretize the model.

6.15 Consider a continuous-time system

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -2 & 2 \\ 1 & -3 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} -1 \\ 5 \end{bmatrix} u(t)$$

$$y(t) = [2 \quad -4]\, \mathbf{x}(t) + 6\, u(t)$$

Insert sample-and-hold devices and determine the vector difference state model for digital simulation of the continuous-time system when the computation interval $T = 0.2$.

6.16 The plant of a single-input, single-output digital control system is shown in the block diagram of Fig. P6.16, where $u(t)$ is the control input and $w(t)$ is a unit-step load disturbance. Obtain the state difference equations of the plant. Sampling period $T = 0.1$ second.

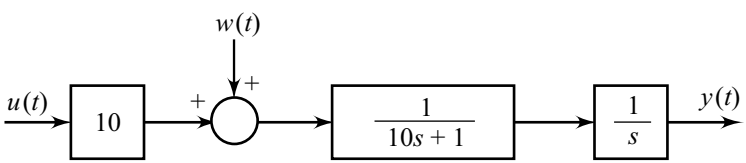


Fig. P6.16

6.17 The mathematical model of the plant of a two-input, two-output temperature control system is given below.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}$$

$$\mathbf{A} = \begin{bmatrix} -0.1 & 0 \\ 0.1 & -0.1 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}; \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

For the computer control of this system, obtain the discrete-time model of the plant. Sampling period $T = 3$ seconds.

6.18 Consider the closed-loop control system shown in Fig. P6.18.

(a) Obtain the z-transform of the feedforward transfer function.

(b) Obtain the closed-loop transfer function, and convert it into a state variable model for digital simulation.

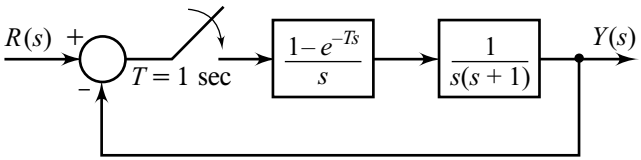


Fig. P6.18

6.19 The mathematical model of the plant of a control system is given below.

$$\frac{Y(s)}{U(s)} = G_a(s) = \frac{e^{-0.4s}}{s+1}$$

For digital simulation of the plant, obtain a vector difference state model with $T = 1$ sec as the sampling period. Use the following methods to obtain the plant model:

(a) Sample $G_a(s)$ with a zero-order hold and convert the resulting discrete-time transfer function into a state model.

(b) Convert the given $G_a(s)$ into a state model and sample this model with a zero-order hold.

6.20 Determine zero-order hold sampling of the process

$$\dot{x}(t) = -x(t) + u(t - 2.5)$$

with sampling interval $T = 1$.

6.21 Convert the transfer function

$$\frac{Y(s)}{U(s)} = G_a(s) = \frac{e^{-s\tau_D}}{s^2}\ ;\ 0 \le \tau_D < T$$

into a state model and sample this model with a zero-order hold; T is the sampling interval.

6.22 The plant of a unity-feedback continuous-time control system is described by the equations

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u$$

$$y = x_1$$

(a) Show that the continuous-time closed-loop system is stable.

(b) A sampler and zero-order hold are now introduced in the forward loop. Show that the stable linear continuous-time system becomes unstable upon the introduction of a sampler and a zero-order hold with sampling period $T = 3$ sec.

6.23 The block diagram of a sampled-data system is shown in Fig. P6.23.

(a) Obtain a discrete-time state model for the system.

(b) Obtain the equation for intersample response of the system.

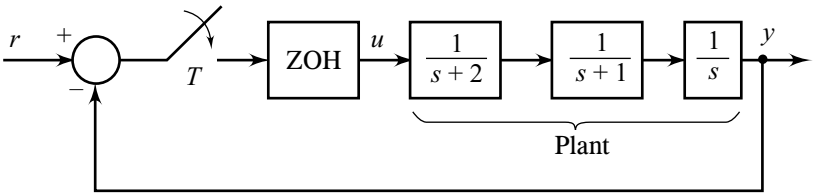


Fig. P6.23

6.24 The block diagram of a sampled-data system is shown in Fig. P6.24. Obtain the discrete-time state model of the system.

Given $$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \quad 0]$$

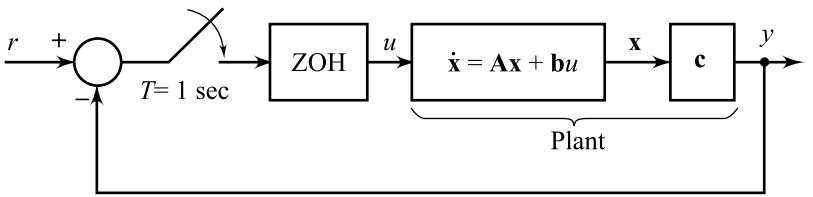


Fig. P6.24

6.25 A closed-loop computer control system is shown in Fig. P6.25. The digital compensator is described by the difference equation

$$e_2(k + 1) + 2e_2(k) = e_1(k)$$

The state model of the plant is, as given in Problem 6.24. Obtain the discrete-time state model for the system.

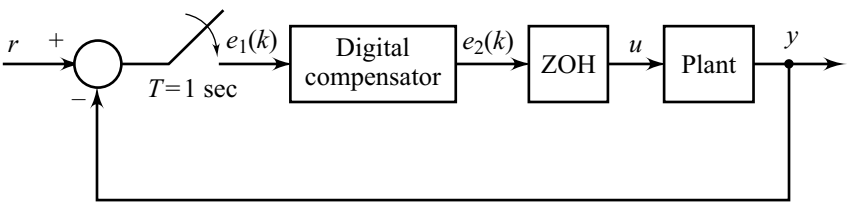


Fig. P6.25

6.26 Consider the closed-loop analog control system shown in Fig. P6.26. For computer control of the process, transform the controller transfer function into a difference equation using backward-difference approximation of the derivative.

Sample the process model with a zero-order hold and obtain the state variable model of the closed-loop computer-controlled system. Take $T = 0.1$ sec as sampling interval.

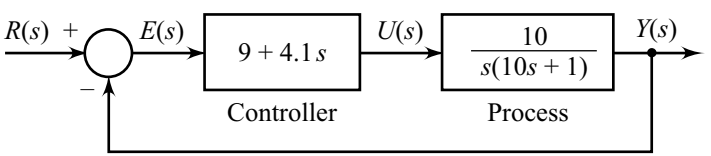


Fig. P6.26

6.27 Investigate the controllability and observability of the following systems:

(a)
$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & -2 \\ 1 & -1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \mathbf{u}(k)$$
$$\mathbf{y}(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}(k)$$

(b)
$$\mathbf{x}(k+1) = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = [1 \quad 1]\, \mathbf{x}(k)$$

6.28 Consider the following continuous-time control system:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$
$$y(t) = x_1(t)$$

Show that the system is completely controllable and completely observable. The control signal $u^+(t)$ is now generated by processing the signal $u(t)$, through a sampler and a zero-order hold. Study the controllability and observability properties of the system under this condition. Determine the values of the sampling period for which the discretized system may exhibit hidden oscillations.

6.29 For the digital system shown in Fig. P6.29, determine what values of T must be avoided so that the system will be assured of complete controllability and observability.

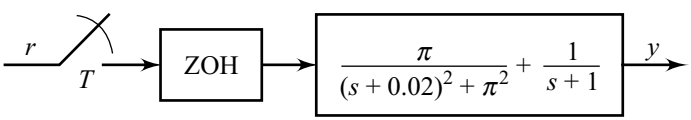


Fig. P6.29

6.30 Consider the state variable model

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}\, r(k)$$
$$y(k) = \mathbf{c}\mathbf{x}(k)$$

with
$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{8} & \frac{3}{4} \end{bmatrix}; \quad \mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \quad \mathbf{c} = [-\tfrac{1}{2} \quad 1]$$

(a) Find the eigenvalues of matrix **F**.

(b) Find the transfer function $G(z) = Y(z)/R(z)$ and determine the poles of the transfer function.

(c) Comment upon the controllability and observability properties of the given system without making any further calculations.

Chapter 7

Pole-Placement Design and State Observers

7.1 INTRODUCTION

The design techniques presented in the preceding chapters are based on either frequency response or the root locus. These transfer function-based methods have been referred to as *classical control design*. The goal of this chapter is to solve the identical problems using different techniques based on state variable formulation. The use of the state-space approach has often been referred to as *modern control design*. However, since the state-space method of description for differential equations is over 100 years old, and was introduced in control design in the late 1950s, it seems somewhat misleading to refer to it as 'modern'. We prefer to refer to the two approaches to design as state variable methods and transform methods.

The transform methods of design are powerful methods of practical design. Most control systems are designed using variations of these methods. An important property of these methods is robustness. The resultant closed-loop system characteristics tend to be insensitive to small inaccuracies in the system model. This property is very important because of the difficulty in finding an accurate linear model of a physical system and also, because many systems have significant nonlinear operations.

The state variable methods appear to be much more dependent on having an accurate system model for the design process. An advantage of these methods is that the system representation provides a complete (internal) description of the system, including possible internal oscillations or instabilities that might be hidden by inappropriate cancellations in the transfer function (input/output) description. The power of state variable techniques is especially apparent when we design controllers for systems with more than one control input or sensed output. However, in this chapter, we will illustrate the state variable design methods using Single-Input, Single-Output (SISO) systems. Methods for Multi-Input, Multi-Output (MIMO) design are discussed in Chapter 8.

In this chapter, we present a design method known as *pole placement* or *pole assignment*. This method is similar to the root-locus design in that, the closed-loop poles may be placed in desired locations. However, pole-placement design allows all closed-loop poles to be placed in desirable locations, whereas the root-locus design procedure allows only the two dominant poles to be placed. There is a cost associated with placing all closed-loop poles, however, because placing all closed-loop poles requires measurement and feedback of all the state variables of the system.

In many applications, all the state variables cannot be measured because of cost considerations, or because of the lack of suitable transducers. In these cases, those state variables that cannot be measured must be estimated from the ones that are measured. Fortunately, we can separate the design into two phases. During the first phase, we design the system as though all states of the system will be measured. The second phase is concerned with the design of the state estimator. In this chapter, we consider both phases of the design process, and the effects that the state estimator has on closed-loop system operation.

Figure 7.1 shows how the state-feedback control law and the state estimator fit together, and how the combination takes the place of, what we have been previously referring to as, dynamic *compensation*. We will see in this chapter that the estimator-based dynamic compensators are very similar to the classical compensators of Chapter 4, in spite of the fact that they are arrived at by entirely different means.

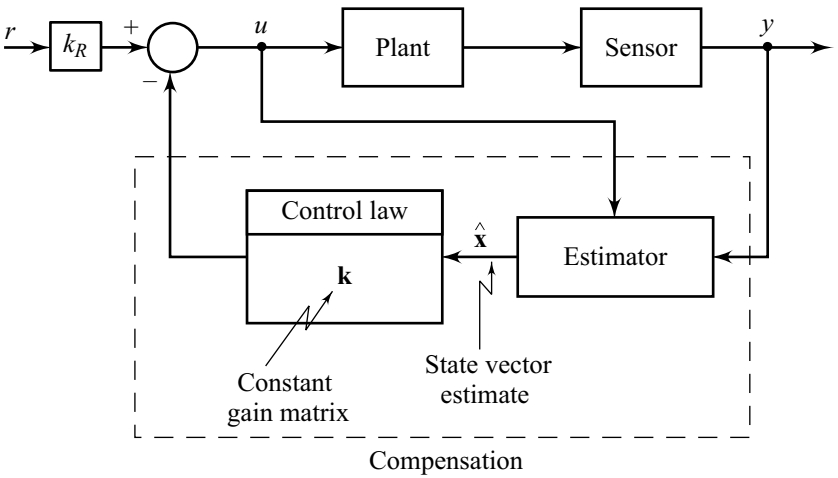


Fig. 7.1 Schematic diagram of a state-feedback control system

7.2 Stability Improvement by State Feedback

An important aspect of feedback system design is the stability of the control system. Whatever we want to achieve with the control system, its stability must be assured. Sometimes the main goal of feedback design is actually to stabilize a system if it is initially unstable, or to improve its stability if transient phenomena do not die out sufficiently fast.

The purpose of this section is to investigate how the stability properties of linear systems can be improved by state feedback.

Consider the single-input linear time-invariant system with *n*th-order state differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \tag{7.1}$$

If we suppose that all the n state variables $x_1, x_2, \ldots, x_n$ can be accurately measured at all times, it is possible to implement a linear control law of the form

$$u(t) = -k_1x_1(t) - k_2x_2(t) - \cdots - k_nx_n(t) = -\mathbf{k}\mathbf{x}(t) \tag{7.2}$$

where $\mathbf{k} = [k_1\ k_2\ \cdots\ k_n]$

is a constant state-feedback gain matrix. With this state-feedback control law, the closed-loop system is described by the state differential equation

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{bk})\,\mathbf{x}(t) \tag{7.3}$$

and the characteristic equation of the closed-loop system is

$$|\,s\mathbf{I} - (\mathbf{A} - \mathbf{bk})\,| = 0 \tag{7.4}$$

When evaluated, this yields an nth-order polynomial in s containing the n gains $k_1, k_2, ..., k_n$. The control-law design then consists of picking the gains so that the roots of Eqn. (7.4) are in desirable locations. In the next section, we find that under a mildly restrictive condition (namely, the system (7.1) must be completely controllable), all the eigenvalues of $(\mathbf{A} - \mathbf{bk})$ can be arbitrarily located in the complex plane by choosing $\mathbf{k}$ suitably (with the restriction that complex eigenvalues occur in complex-conjugate pairs).

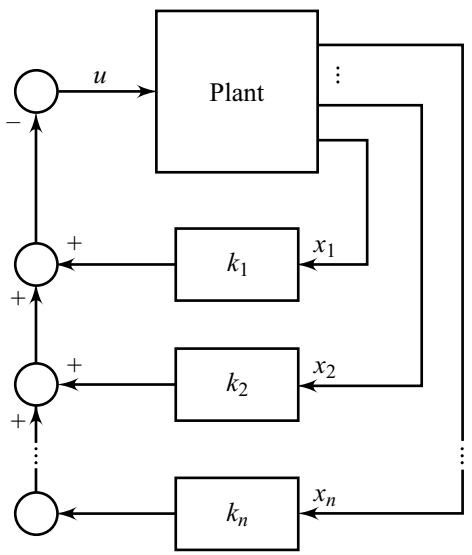


Fig. 7.2 Control configuration for a state regulator

If all the eigenvalues of $(\mathbf{A} - \mathbf{bk})$ are placed in the left-half plane, the closed-loop system is, of course, asymptotically stable; $\mathbf{x}(t)$ will decay to zero irrespective of the value of $\mathbf{x}(0)$—the initial perturbation in the state. The system state is thus maintained at zero value in spite of disturbances that act upon the system. Systems with this property are called *regulator systems*. The origin of state space is the *equilibrium state* of the system.

Control configuration for a state regulator is shown in Fig. 7.2. In this structure, there is no command input ($r = 0$). Control systems in which the output must follow the command signals (called *servo systems*) will be considered later.

Selection of desirable locations for the closed-loop poles requires some iteration by the designer. Some of the issues in their selection will be discussed later in this chapter. For now, we will assume that the desired locations are known, say,

$$s = \lambda_1, \lambda_2, \ldots, \lambda_n$$

Then the desired characteristic equation is

$$(s - \lambda_1)(s - \lambda_2) \cdots (s - \lambda_n) = 0 \tag{7.5}$$

The required elements of $\mathbf{k}$ are obtained by matching coefficients in Eqns (7.4) and (7.5), thus forcing the system characteristic equation to be identical with the desired equation. An example should clarify this pole-placement idea.

Example 7.1

Consider the problem of designing an attitude control system for a rigid satellite. Satellites usually require attitude control so that antennas, sensors, and solar panels are properly oriented. For example, antennas

are usually pointed towards a particular location on the earth, while solar panels need to be oriented towards the sun for maximum power generation. To gain an insight into the full three-axis attitude-control system, we often consider one axis at a time. Figure 7.3 depicts this case. The angle θ that describes the satellite orientation, must be measured with respect to an 'inertial' reference, that is, a reference that has no angular acceleration. The control signal comes from the reaction jets that produce a torque $T(t)$ $(= Fd)$ about the mass center.

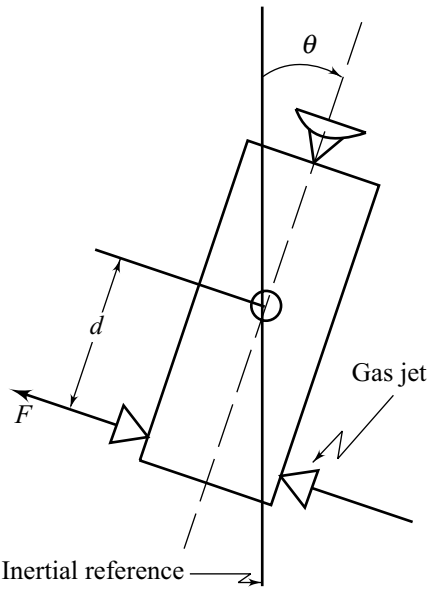


Fig. 7.3 Satellite-attitude control schematic

The satellite is assumed to be in frictionless environment. If $T(t)$ is the system input and $\theta(t)$ is the system output, we have

$$T(t) = J\,\frac{d^2\theta(t)}{dt^2}$$

where J is the moment of inertia of the satellite. Normalizing, we define

$$u = T(t)/J$$

and obtain

$$\ddot{\theta} = u \text{ or } \frac{\theta(s)}{U(s)} = \frac{1}{s^2}$$

This is a reasonably accurate model of a rigid satellite in a frictionless environment, and is useful in examples because of its simplicity.

Choosing $x_1 = \theta$ and $x_2 = \dot{\theta}$ as state variables, we obtain the following state equation for the system.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{7.6}$$

To stabilize the system, the input signal is chosen to be of the form

$$u(t) = -k_1x_1(t) - k_2x_2(t) = -\mathbf{k}\mathbf{x}(t)$$

The state equation for the closed-loop system (Fig. 7.4), then becomes

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{b}\mathbf{k})\mathbf{x} = \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} [k_1 \quad k_2] \right) \mathbf{x} = \begin{bmatrix} 0 & 1 \\ -k_1 & k_2 \end{bmatrix} \mathbf{x}$$

The characteristic equation of the closed-loop system is

$$|s\mathbf{I} - (\mathbf{A} - \mathbf{b}\mathbf{k})| = s^2 + k_2s + k_1 = 0 \tag{7.7}$$

Suppose that the design specifications for this system require $\zeta = 0.707$ with a settling time of 1 sec $\left(\dfrac{4}{\zeta\omega_n} = 1 \text{ or } \zeta\omega_n = 4 \right)$. The closed-loop pole locations needed are at $s = -4 \pm j4$. The desired characteristic equation is

$$(s + 4 + j4)\,(s + 4 - j4) = s^2 + 8s + 32 \tag{7.8}$$

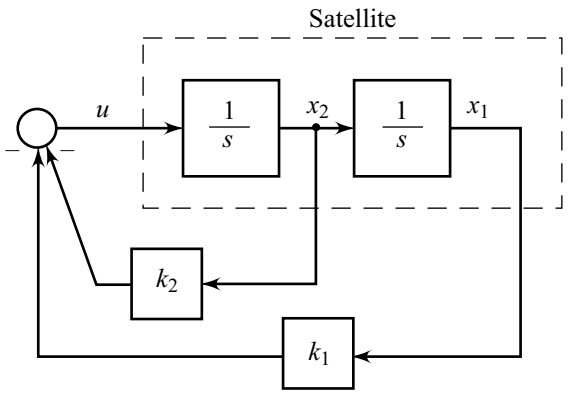


Fig. 7.4 Simulation diagram for the satellite-attitude control system

Equating the coefficients with like powers of s in Eqns (7.7) and (7.8) yields

$$k_1 = 32, k_2 = 8$$

The calculation of the gains using the technique illustrated in this example, becomes rather tedious when the order of the system is larger than three. There are, however, 'canonical' forms of the state variable equations where the algebra for finding the gains is especially simple. One such canonical form—useful in control-law design—is the *controllable canonical form*. Consider a system represented by the transfer function

$$\frac{Y(s)}{U(s)} = \frac{\beta_1 s^{n-1} + \beta_2 s^{n-2} + \cdots + \beta_n}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n}$$

A companion-form realization of this transfer function is given below (refer to Eqns (5.54)):

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u \\ y &= \mathbf{c}\mathbf{x} \end{aligned} \tag{7.9}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_n & -\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_1 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{c} = [\beta_n \quad \beta_{n-1} \cdots \beta_2 \quad \beta_1]$$

The matrix **A** in Eqns (7.9) has a very special structure: the coefficients of the denominator of the transfer function preceded by minus signs, form a string along the bottom row of the matrix. The rest of the matrix is zero except for the superdiagonal terms, which are all unity. It can easily be proved that the pair (**A**,**b**) is completely controllable for all values of α_i's. For this reason, the companion-form realization given by Eqns (7.9) is referred to as the *controllable canonical form*.

One of the advantages of the controllable canonical form is that the controller gains can be obtained from it, just by inspection. The closed-loop system matrix

$$\mathbf{A} - \mathbf{bk} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\alpha_n - k_1 & -\alpha_{n-1} - k_2 & -\alpha_{n-2} - k_3 & \cdots & -\alpha_1 - k_n \end{bmatrix}$$

has the characteristic equation

$$s^n + (\alpha_1 + k_n)s^{n-1} + \cdots + (\alpha_{n-2} + k_3)s^2 + (\alpha_{n-1} + k_2)s + \alpha_n + k_1 = 0$$

and the controller gains can be found by comparing the coefficients of this characteristic equation with Eqn. (7.5).

We now have the basis for a design procedure. Given an arbitrary state variable model and a desired characteristic polynomial, we transform the model to controllable canonical form and solve for the controller gains, by inspection. Since these gains are for the state in the controllable canonical form, we must transform the gains back to the original state. We will develop this pole-placement design procedure in the subsequent sections.

7.3 NECESSARY AND SUFFICIENT CONDITIONS FOR ARBITRARY POLE-PLACEMENT

Consider the linear time-invariant system (7.1) with state-feedback control law (7.2); the resulting closed-loop system is given by Eqn. (7.3). In the following, we shall prove that a necessary and sufficient condition for arbitrary placement of closed-loop eigenvalues in the complex plane (with the restriction that complex eigenvalues occur in complex-conjugate pairs), is that the system (7.1) is completely controllable. We shall first prove the sufficient condition, i.e., if the system (7.1) is completely controllable, all the eigenvalues of (**A** – **bk**) in Eqn. (7.3) can be arbitrarily placed.

In proving the sufficient condition on arbitrary pole-placement, it is convenient to transform the state equation (7.1) into the controllable canonical form (7.9). Let us assume that such a transformation exists and is given by

$$\bar{\mathbf{x}} = \mathbf{P}\mathbf{x} \tag{7.10}$$

$$= \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_n \end{bmatrix} \mathbf{x}$$

$$\mathbf{p}_i = [p_{i1} \quad p_{i2} \cdots p_{in}]; \; i = 1, 2, ..., n$$

Under the transformation (7.10), system (7.1) is transformed to the following controllable canonical model:

$$\dot{\bar{\mathbf{x}}} = \bar{\mathbf{A}}\,\bar{\mathbf{x}} + \bar{\mathbf{b}}\,u \tag{7.11}$$

where

$$\bar{\mathbf{A}} = \mathbf{PAP}^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & & 1 \\ -\alpha_n & -\alpha_{n-1} & -\alpha_{n-2} & \cdots & -\alpha_1 \end{bmatrix}; \ \bar{\mathbf{b}} = \mathbf{Pb} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$|s\mathbf{I} - \bar{\mathbf{A}}| = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1} s + \alpha_n = |s\mathbf{I} - \mathbf{A}|$$

(Characteristic polynomial is invariant under equivalence transformation)

The first equation in the set (7.10) is given by

$$\bar{x}_1 = p_{11} x_1 + p_{12} x_2 + \cdots + p_{1n} x_n = \mathbf{p}_1 \mathbf{x}$$

Taking the derivative on both sides of this equation, we get

$$\dot{\bar{x}}_1 = \mathbf{p}_1 \dot{\mathbf{x}} = \mathbf{p}_1 \mathbf{A}\mathbf{x} + \mathbf{p}_1 \mathbf{b} u$$

But $\dot{\bar{x}}_1$ $(= \bar{x}_2)$ is a function of $\mathbf{x}$ only as per the canonical model (7.11).

Therefore,

$$\mathbf{p}_1 \mathbf{b} = 0 \text{ and } \bar{x}_2 = \mathbf{p}_1 \mathbf{A}\mathbf{x}$$

Taking derivative on both sides once again, we get

$$\mathbf{p}_1 \mathbf{A}\mathbf{b} = 0 \text{ and } \bar{x}_3 = \mathbf{p}_1 \mathbf{A}^2 \mathbf{x}$$

Continuing the process, we obtain

$$\mathbf{p}_1 \mathbf{A}^{n-2} \mathbf{b} = 0 \text{ and } \bar{x}_n = \mathbf{p}_1 \mathbf{A}^{n-1} \mathbf{x}$$

Taking derivative once again, we obtain

$$\mathbf{p}_1 \mathbf{A}^{n-1} \mathbf{b} = 1$$

Thus

$$\bar{\mathbf{x}} = \mathbf{P}\mathbf{x} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_1 \mathbf{A} \\ \vdots \\ \mathbf{p}_1 \mathbf{A}^{n-1} \end{bmatrix} \mathbf{x}$$

where $\mathbf{p}_1$ must satisfy the conditions

$$\mathbf{p}_1 \mathbf{b} = \mathbf{p}_1 \mathbf{A}\mathbf{b} = \cdots = \mathbf{p}_1 \mathbf{A}^{n-2} \mathbf{b} = 0, \ \mathbf{p}_1 \mathbf{A}^{n-1} \mathbf{b} = 1$$

From Eqn. (7.11), we have

$$\mathbf{Pb} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 \mathbf{b} \\ \mathbf{p}_1 \mathbf{A}\mathbf{b} \\ \vdots \\ \mathbf{p}_1 \mathbf{A}^{n-2} \mathbf{b} \\ \mathbf{p}_1 \mathbf{A}^{n-1} \mathbf{b} \end{bmatrix}$$

or

$$\mathbf{p}_1[\mathbf{b} \quad \mathbf{Ab} \cdots \mathbf{A}^{n-2}\mathbf{b} \quad \mathbf{A}^{n-1}\mathbf{b}] = [0 \quad 0 \cdots 0 \quad 1]$$

This gives

$$\mathbf{p}_1 = [0 \quad 0 \cdots 0 \quad 1]\mathbf{U}^{-1}$$

where

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab} \cdots \mathbf{A}^{n-1}\mathbf{b}]$$

is the controllability matrix, which is nonsingular because of the assumption of controllability of the system (7.1).

Therefore, the controllable state model (7.1) can be transformed to the canonical form (7.11) by the transformation.

$$\bar{\mathbf{x}} = \mathbf{P}\mathbf{x} \tag{7.12}$$

where

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_1\mathbf{A} \\ \vdots \\ \mathbf{p}_1\mathbf{A}^{n-1} \end{bmatrix}; \quad \mathbf{p}_1 = [0 \quad 0 \cdots 0 \quad 1]\mathbf{U}^{-1}$$

Under the equivalence transformation (7.12), the state-feedback control law (7.2) becomes

$$u = -\mathbf{k}\mathbf{x} = -\bar{\mathbf{k}}\bar{\mathbf{x}} \tag{7.13}$$

where

$$\bar{\mathbf{k}} = \mathbf{k}\mathbf{P}^{-1} = [\bar{k}_1 \quad \bar{k}_2 \cdots \bar{k}_n]$$

With this control law, system (7.11) becomes

$$\dot{\bar{\mathbf{x}}} = (\bar{\mathbf{A}} - \bar{\mathbf{b}}\bar{\mathbf{k}})\,\bar{\mathbf{x}}$$

$$= \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & & 0 & 1 \\ -\alpha_n - \bar{k}_1 & -\alpha_{n-1} - \bar{k}_2 & -\alpha_{n-2} - \bar{k}_3 & \cdots & -\alpha_2 - \bar{k}_{n-1} & -\alpha_1 - \bar{k}_n \end{bmatrix} \bar{\mathbf{x}} \tag{7.14}$$

$$|s\mathbf{I} - (\bar{\mathbf{A}} - \bar{\mathbf{b}}\bar{\mathbf{k}})| = s^n + (\alpha_1 + \bar{k}_n)s^{n-1} + (\alpha_2 + \bar{k}_{n-1})s^{n-2} + \cdots + (\alpha_{n-1} + \bar{k}_2)\,s + (\alpha_n + \bar{k}_1) \tag{7.15}$$

Since the coefficients $\bar{k}_i$ are arbitrarily chosen real numbers, the coefficients of the characteristic polynomial of $(\mathbf{A} - \mathbf{bk})$ can be given any desired values. Hence, the closed-loop poles can be placed at any desired locations in the complex plane (subject to conjugate pairing: coefficients of a characteristic polynomial will be real only if the complex poles are present in conjugate pairs).

Assume that the desired characteristic polynomial of $(\mathbf{A} - \mathbf{bk})$, and hence $(\bar{\mathbf{A}} - \bar{\mathbf{b}}\bar{\mathbf{k}})$, is

$$s^n + a_1 s^{n-1} + \cdots + a_n$$

From Eqn. (7.15), it is obvious that this requirement is met if $\bar{\mathbf{k}}$ is chosen as

$$\bar{\mathbf{k}} = [a_n - \alpha_n \quad a_{n-1} - \alpha_{n-1} \cdots a_1 - \alpha_1]$$

Transforming the feedback controller (7.13) to the original coordinates, we obtain

$$\mathbf{k} = \bar{\mathbf{k}}\,\mathbf{P} = [a_n - \alpha_n \quad a_{n-1} - \alpha_{n-1} \cdots a_1 - \alpha_1]\mathbf{P} \tag{7.16}$$

This proves that if (7.1) is controllable, the closed-loop poles can be arbitrarily assigned (sufficient condition).

We now derive the necessary condition by proving that if the system (7.1) is not completely controllable, then there are eigenvalues of $(\mathbf{A} - \mathbf{bk})$ that cannot be controlled by state feedback.

It was shown in Section 5.9 that an uncontrollable system can be transformed into controllability canonical form (Eqn. (5.123c))

$$\begin{bmatrix} \dot{\bar{\mathbf{x}}}_1 \\ \dot{\bar{\mathbf{x}}}_2 \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{A}}_c & \bar{\mathbf{A}}_{12} \\ \mathbf{0} & \bar{\mathbf{A}}_{22} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{b}}_c \\ \mathbf{0} \end{bmatrix} u = \bar{\mathbf{A}}\,\bar{\mathbf{x}} + \bar{\mathbf{b}}u$$

where the pair $(\bar{\mathbf{A}}_c, \bar{\mathbf{b}}_c)$ is completely controllable.

The set of eigenvalues of $\bar{\mathbf{A}}$ is the union of the sets of eigenvalues of $\bar{\mathbf{A}}_c$ and $\bar{\mathbf{A}}_{22}$. In view of the form of $\bar{\mathbf{b}}$, it is obvious that the matrix $\bar{\mathbf{A}}_{22}$ is not affected by the introduction of any state feedback of the form $u = -\bar{\mathbf{k}}\,\bar{\mathbf{x}}$. Therefore, the eigenvalues of $\bar{\mathbf{A}}_{22}$ cannot be controlled. This proves the necessary condition.

7.4 STATE REGULATOR DESIGN

Consider the *n*th-order, single-input linear time-invariant system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \tag{7.17}$$

with state-feedback control law

$$u(t) = -\mathbf{k}\mathbf{x}(t) \tag{7.18}$$

The resulting closed-loop system is

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{bk})\,\mathbf{x}(t) \tag{7.19}$$

The eigenvalues of $(\mathbf{A} - \mathbf{bk})$ can be arbitrarily placed in the complex plane (with the restriction that complex eigenvalues occur in complex-conjugate pairs) by choosing $\mathbf{k}$ suitably if, and only if, the system (7.17) is completely controllable.

This important result on pole placement was proved in the previous section. The following design steps for pole placement, emerge from the proof.

Step 1 From the characteristic polynomial of matrix $\mathbf{A}$:

$$|s\mathbf{I} - \mathbf{A}| = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_{n-1} s + \alpha_n \tag{7.20}$$

determine the values of $\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n$.

Step 2 Determine the transformation matrix $\mathbf{P}$ that transforms the system (7.17) into controllable canonical form:

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_1\mathbf{A} \\ \vdots \\ \mathbf{p}_1\mathbf{A}^{n-1} \end{bmatrix}; \quad \begin{aligned} \mathbf{p}_1 &= [0 \quad 0 \cdots 0 \quad 1]\mathbf{U}^{-1} \\ \mathbf{U} &= [\mathbf{b} \quad \mathbf{Ab} \cdots \mathbf{A}^{n-1}\mathbf{b}] \end{aligned} \tag{7.21}$$

Step 3 Using the desired eigenvalues (desired closed-loop poles) λ_1, λ_2, ..., λ_n, write the desired characteristic polynomial:

$$(s - \lambda_1)(s - \lambda_2) \cdots (s - \lambda_n) = s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n \tag{7.22}$$

and determine the values of $a_1, a_2, \cdots, a_{n-1}, a_n$.

Step 4 The required state-feedback gain matrix is determined from the following equation:

$$\mathbf{k} = [a_n - \alpha_n \quad a_{n-1} - \alpha_{n-1} \cdots a_1 - \alpha_1]\,\mathbf{P} \tag{7.23}$$

There are other approaches also, for the determination of the state-feedback gain matrix **k**. In what follows, we shall present a well-known formula, known as the *Ackermann's formula*, which is convenient for computer solution.

From Eqns (7.23) and (7.21), we get

$$\mathbf{k} = [a_n - \alpha_n \quad a_{n-1} - \alpha_{n-1} \cdots a_1 - \alpha_1] \begin{bmatrix} [0 & 0 & \cdots & 0 & 1]\,\mathbf{U}^{-1} \\ [0 & 0 & \cdots & 0 & 1]\,\mathbf{U}^{-1}\mathbf{A} \\ & & \vdots & & \\ [0 & 0 & \cdots & 0 & 1]\,\mathbf{U}^{-1}\mathbf{A}^{n-1} \end{bmatrix}$$

$$= [0 \quad 0 \cdots 0 \quad 1]\mathbf{U}^{-1}\,[(a_1 - \alpha_1)\mathbf{A}^{n-1} + (a_2 - \alpha_2)\mathbf{A}^{n-2} + \cdots + (a_n - \alpha_n)\mathbf{I}] \tag{7.24}$$

The characteristic polynomial of matrix **A** is (Eqn. (7.20))

$$|s\mathbf{I} - \mathbf{A}| = s^n + \alpha_1 s^{n-1} + \alpha_2 s^{n-2} + \cdots + \alpha_{n-1} s + \alpha_n$$

Since the Cayley–Hamilton theorem states that a matrix satisfies its own characteristic equation, we have

$$\mathbf{A}^n + \alpha_1\mathbf{A}^{n-1} + \alpha_2\mathbf{A}^{n-2} + \cdots + \alpha_{n-1}\mathbf{A} + \alpha_n\mathbf{I} = \mathbf{0}$$

Therefore,
$$\mathbf{A}^n = -\alpha_1\mathbf{A}^{n-1} - \alpha_2\mathbf{A}^{n-2} - \cdots - \alpha_{n-1}\mathbf{A} - \alpha_n\mathbf{I} \tag{7.25}$$

From Eqns (7.24) and (7.25), we get

$$\mathbf{k} = [0 \quad 0 \cdots 0 \quad 1]\mathbf{U}^{-1}\,\phi(\mathbf{A}) \tag{7.26a}$$

where

$$\phi(\mathbf{A}) = \mathbf{A}^n + a_1\mathbf{A}^{n-1} + a_2\mathbf{A}^{n-2} + \cdots + a_{n-1}\mathbf{A} + a_n\mathbf{I} \tag{7.26b}$$

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab} \cdots \mathbf{A}^{n-1}\mathbf{b}] \tag{7.26c}$$

Equations (7.26) describe the Ackermann's formula for the determination of the state-feedback gain matrix **k**.

Example 7.2

Recall the inverted pendulum of Example 5.15, shown in Fig. 5.16, in which the object is to apply a force $u(t)$ so that the pendulum remains balanced in the vertical position. We found the linearized equations governing the system to be

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

where

$$\mathbf{x} = [\theta \quad \dot{\theta} \quad z \quad \dot{z}\,]^T$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 16.3106 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1.0637 & 0 & 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ -1.4458 \\ 0 \\ 0.9639 \end{bmatrix}$$

$z(t)$ = horizontal displacement of the pivot on the cart

$\theta(t)$ = rotational angle of the pendulum

It is easy to verify that the characteristic polynomial of matrix **A** is

$$|s\mathbf{I} - \mathbf{A}| = s^4 - 16.3106s^2$$

Since there are poles at 0, 0, 4.039, and – 4.039, the system is quite unstable, as one would expect from physical reasoning.

Suppose we require a feedback control of the form

$$u(t) = -\mathbf{k}\mathbf{x} = -k_1x_1 - k_2x_2 - k_3x_3 - k_4x_4,$$

such that the closed-loop system has the stable pole configuration given by multiple poles at –1. We verified in Example 5.16 that the system under consideration is a controllable system; therefore, such a feedback gain matrix **k** does exist. We will determine the required **k** by using the design equations (7.17)–(7.23).

The controllability matrix

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab} \quad \mathbf{A^2b} \quad \mathbf{A^3b}] = \begin{bmatrix} 0 & -1.4458 & 0 & -23.5816 \\ -1.4458 & 0 & -23.5816 & 0 \\ 0 & 0.9639 & 0 & 1.5379 \\ 0.9639 & 0 & 1.5379 & 0 \end{bmatrix}$$

$$\mathbf{U}^{-1} = \begin{bmatrix} 0 & 0.0750 & 0 & 1.1500 \\ 0.0750 & 0 & 1.1500 & 0 \\ 0 & -0.0470 & 0 & -0.0705 \\ 0.0470 & 0 & -0.0705 & 0 \end{bmatrix}$$

Therefore,

$$\mathbf{p}_1 = [-0.0470 \quad 0 \quad -0.0705 \quad 0]$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_1\mathbf{A} \\ \mathbf{p}_1\mathbf{A}^2 \\ \mathbf{p}_1\mathbf{A}^3 \end{bmatrix} = \begin{bmatrix} -0.0470 & 0 & -0.0705 & 0 \\ 0 & -0.0470 & 0 & -0.0705 \\ -0.6917 & 0 & 0 & 0 \\ 0 & -0.6917 & 0 & 0 \end{bmatrix}$$

$$|s\mathbf{I} - \mathbf{A}| = s^4 + \alpha_1 s^3 + \alpha_2 s^2 + \alpha_3 s + \alpha_4 = s^4 + 0s^3 - 16.3106s^2 + 0\,s + 0$$

$$|s\mathbf{I} - (\mathbf{A} - \mathbf{bk})| = s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4 = (s+1)^4 = s^4 + 4s^3 + 6s^2 + 4s + 1$$

$$\mathbf{k} = [a_4 - \alpha_4 \quad a_3 - \alpha_3 \quad a_2 - \alpha_2 \quad a_1 - \alpha_1]\mathbf{P}$$

$$= [1 \quad 4 \quad 22.3106 \quad 4]\mathbf{P} = [-15.4785 \quad -2.9547 \quad -0.0705 \quad -0.2820]$$

This feedback control law yields a stable closed-loop system so that the entire state vector, when disturbed from the zero state, returns asymptotically to this state. This means that not only is the pendulum balanced ($\theta \to 0$), but that the cart returns to its origin as well ($z \to 0$).

Example 7.3

Let us use Ackermann's formula to the state-regulator design problem of Example 7.1 (satellite-attitude control system). The plant model is given by (Eqn. (7.6))

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

The desired characteristic polynomial is (Eqn. (7.8))

$$s^2 + a_1 s + a_2 = s^2 + 8s + 32$$

To use Ackermann's formula (7.26) to calculate the gain matrix $\mathbf{k}$, we first evaluate $\mathbf{U}^{-1}$ and $\phi(\mathbf{A})$:

$$\mathbf{U} = [\mathbf{b} \quad \mathbf{Ab}] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad \mathbf{U}^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\phi(\mathbf{A}) = \mathbf{A}^2 + a_1\mathbf{A} + a_2\mathbf{I} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + 8\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + 32\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 32 & 8 \\ 0 & 32 \end{bmatrix}$$

Now using Eqn. (7.26a), we obtain

$$\mathbf{k} = [0 \quad 1]\,\mathbf{U}^{-1}\,\phi(\mathbf{A}) = [0 \quad 1]\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 32 & 8 \\ 0 & 32 \end{bmatrix} = [32 \quad 8]$$

The solution is seen to be the same as that obtained in Example 7.1.

Comments

1. Through the pole-placement design procedure described in the present section, it is always possible to stabilize a completely controllable system by state feedback, or to improve its stability by assigning the closed-loop poles to locations in the left-half complex plane. The design procedure, however, gives no guidance as to where, in the left-half plane, the closed-loop poles should be located.

It appears that we can choose the magnitude of the real part of the closed-loop poles to be arbitrarily large, making the system response arbitrarily fast. However, to increase the rate at which the plant responds, the input signal to the plant must become larger, requiring large values of gains. As the magnitudes of the signals in a system increase, the likelihood of the system entering nonlinear regions of operation, increases. For very large signals, this nonlinear operation will occur for almost every physical system. Hence, the linear model that is used in design no longer accurately models the physical system.

Thus, the selection of desired closed-loop poles requires a proper balance of bandwidth, overshoot, sensitivity, control effort, and other design requirements. If the system is of second-order, then the system dynamics (response characteristics) can be precisely correlated to the locations of the desired closed-loop poles. For higher-order systems, the location of the closed-loop poles and the response characteristics are not easily correlated. Hence, in determining the state-feedback gain matrix **k** for a given system, it is desirable to examine, by computer simulations, the response characteristics for several different matrices **k** (based on several different characteristic equations), and choose the one that gives the best overall performance.

2. For the case of single-input systems, the gain matrix **k**, which places the closed-loop poles at the desired locations, is unique.

If the dynamic system under consideration

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

has more than one input, that is, **B** has more than one column, then the gain matrix **K** in the control law

$$\mathbf{u} = -\mathbf{K}\mathbf{x}$$

has more than one row. Since each row of **K** furnishes n gains (n is the order of the system) that can be adjusted, it is clear that in a controllable system there will be more gains available—than are needed—to place all of the closed-loop poles. This is a benefit: the designer has more flexibility in the design; it is possible to specify all the closed-loop poles and still be able to satisfy other requirements. How should these other requirements be specified? The answer to this question may well depend on the circumstances of the particular application. A number of results using the design freedom in multi-input systems to improve robustness of the control system, have appeared in the literature. We will not be able to accommodate these results in this book.

The non-uniqueness in the design of state-feedback control law for multi-input systems, is removed by optimal control theory which is discussed in Chapter 8.

7.5 DESIGN OF STATE OBSERVERS

The pole-placement design procedure introduced in the preceding sections results in control law of the form

$$u(t) = -\mathbf{k}\mathbf{x}(t) \tag{7.27}$$

which requires the ability to directly measure the entire state vector $\mathbf{x}(t)$. Full state-feedback control for many second-order systems, requires feedback of position and rate variables which can easily be measured. However, for most of the higher-order systems, full state measurements are not practical.

Thus, either a new approach that directly accounts for the non-availability of the entire state vector (Chapter 8) is to be devised, or a suitable approximation of the state vector must be determined. The latter approach is much simpler in many situations.

The purpose of this section is to demonstrate the estimation of all the state variables of a system, from the measurements that can be made on the system. If the estimate of the state vector is denoted by $\hat{\mathbf{x}}$, it would be nice if the true state in the control law given by Eqn. (7.27), could be replaced by its estimate

$$u(t) = -\mathbf{k}\hat{\mathbf{x}}(t) \tag{7.28}$$

This indeed is possible, as we shall see in the next section.

A device (or a computer program) that estimates the state variables is called a *state observer*, or simply an *observer*. If the state observer estimates all the state variables of the system, regardless of whether some state variables are available for direct measurement, it is called a *full-order state observer*. However, if accurate measurements of certain states are possible, we may estimate only the remaining states, and the accurately measured signals are then used directly for feedback. The resulting observer is called a *reduced-order state observer*.

7.5.1 Full-Order State Observer

Consider a process described by the state equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \tag{7.29a}$$

where **A** and **b** are, respectively, $n \times n$ and $n \times 1$ real constant matrices. The measurement $y(t)$ is related to the state by the equation

$$y(t) = \mathbf{c}\mathbf{x}(t) \tag{7.29b}$$

where **c** is $1 \times n$ real constant matrix. Without loss of generality, the direct transmission part has been assumed to be zero.

One method of estimating all the state variables that we may consider, is to construct a model of the plant dynamics

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{b}u(t) \tag{7.30}$$

where $\hat{\mathbf{x}}$ is the estimate of the actual state **x**. We know **A**, **b** and $u(t)$, and hence this estimator is satisfactory if we can obtain the correct initial condition $\mathbf{x}(0)$ and set $\hat{\mathbf{x}}(0)$ equal to it. Figure 7.5 depicts this 'open-loop' estimator. However, it is precisely the lack of information on $\mathbf{x}(0)$ that requires the construction of an estimator. If $\dot{\hat{\mathbf{x}}}(0) \neq \mathbf{x}(0)$, the estimated state $\hat{\mathbf{x}}(t)$ obtained from the open-loop scheme of Fig. 7.5 would have a continually growing error or an error that goes to zero too slowly, to be of any use. Furthermore, small errors in our knowledge of the system (**A**, **b**), and the disturbances that enter the system, but not the model, would also cause the estimate to slowly diverge from the true state.

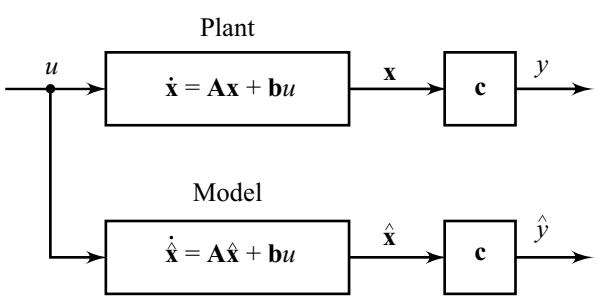


Fig. 7.5 Open-loop estimator

In order to speed up the estimation process and provide a useful state estimate, we feed back the difference between the measured and the estimated outputs—and correct the model continuously with this error signal. This scheme, commonly known as 'Luenberger state observer', is shown in Fig. 7.6, and the equation for it is

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}\,\hat{\mathbf{x}}(t) + \mathbf{b}u(t) + \mathbf{m}(y(t) - \hat{y}(t)) \tag{7.31}$$

where $\mathbf{m}$ is an $n \times 1$ real constant gain matrix.

The state error vector

$$\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t) \tag{7.32}$$

Differentiating both sides, we get

$$\dot{\tilde{\mathbf{x}}}(t) = \dot{\mathbf{x}}(t) - \dot{\hat{\mathbf{x}}}(t)$$

Substituting for $\dot{\mathbf{x}}(t)$ and $\dot{\hat{\mathbf{x}}}(t)$ from Eqns (7.29) and (7.31) respectively, we get

$$\begin{aligned}\dot{\tilde{\mathbf{x}}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) - \mathbf{A}\hat{\mathbf{x}}(t) - \mathbf{b}u(t) - \mathbf{mc}(\mathbf{x}(t) - \hat{\mathbf{x}}(t)) \\ &= (\mathbf{A} - \mathbf{mc})\,\tilde{\mathbf{x}}(t)\end{aligned} \tag{7.33}$$

The characteristic equation of the error is given by

$$|s\mathbf{I} - (\mathbf{A} - \mathbf{mc})| = 0 \tag{7.34a}$$

If $\mathbf{m}$ can (we hope) be chosen so that $(\mathbf{A} - \mathbf{mc})$ has stable and reasonably fast roots, $\tilde{\mathbf{x}}(t)$ will decay to zero irrespective of $\tilde{\mathbf{x}}(0)$. This means that $\hat{\mathbf{x}}(t)$ will converge to $\mathbf{x}(t)$ regardless of the value of $\hat{\mathbf{x}}(0)$, and furthermore, the dynamics of the error can be chosen to be faster than the open-loop dynamics. Note that Eqn. (7.33) is independent of applied control. This is a consequence of assuming $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ to be identical in the plant and the observer. Therefore, the estimation error $\tilde{\mathbf{x}}$ converges to zero and remains there, independent of any known forcing function $u(t)$ on the plant and its effect on the state $\mathbf{x}(t)$. If we

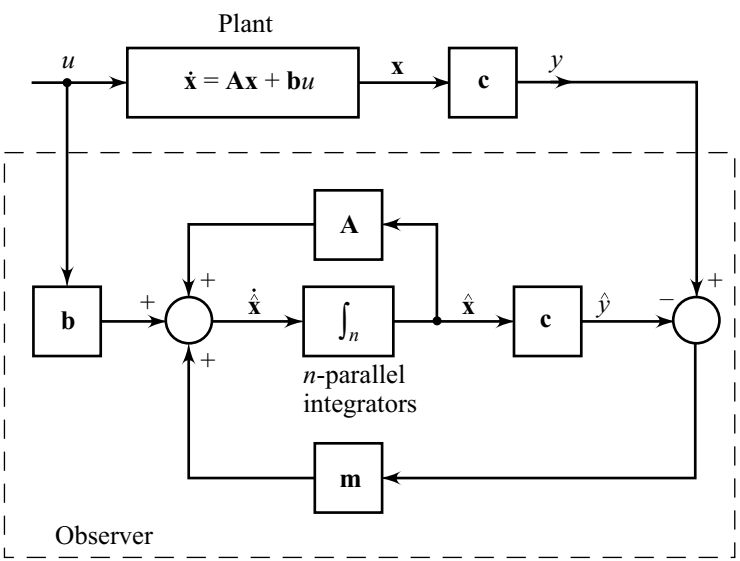


Fig. 7.6 Luenberger state observer

do not have a very accurate model of the plant (**A**, **b**, **c**), the dynamics of the error are no longer governed by Eqn. (7.33). However, **m** can typically be chosen so that the error system is stable and the error is acceptably small, even with small modeling errors and disturbance inputs.

The selection of **m** can be approached in exactly the same fashion as the selection of **k** in the control law design. If we specify the desired location of the observer-error roots as

$$s = \lambda_1, \lambda_2, \ldots, \lambda_n,$$

the desired observer characteristic equation is

$$(s - \lambda_1)(s - \lambda_2) \cdots (s - \lambda_n) = 0 \tag{7.34b}$$

and one can solve for **m** by comparing coefficients in Eqns (7.34a) and (7.34b). However, as we shall see shortly, this can be done only if the system (7.29) is completely observable.

The calculation of the gains using this simple technique becomes rather tedious when the order of the system is larger than three. As in the controller design, there is an *observable canonical form* for which the observer design equations are particularly simple. Consider a system represented by the transfer function

$$\frac{Y(s)}{U(s)} = \frac{\beta_1 s^{n-1} + \beta_2 s^{n-2} + \cdots + \beta_n}{s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n}$$

A companion-form realization of this transfer function is given below (refer to Eqns (5.56)):

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u \\ y &= \mathbf{c}\mathbf{x} \end{aligned} \tag{7.35}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -\alpha_n \\ 1 & 0 & \cdots & 0 & -\alpha_{n-1} \\ 0 & 1 & \cdots & 0 & -\alpha_{n-2} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_1 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} \beta_n \\ \beta_{n-1} \\ \beta_{n-2} \\ \vdots \\ \beta_1 \end{bmatrix}; \mathbf{c} = [0 \quad 0 \cdots 0 \quad 1]$$

It can easily be proved that the pair (**A**,**c**) is completely observable for all values of α_i's. For this reason, the companion form realization given by Eqn. (7.35) is referred to as observable canonical form.

One of the advantages of the observable canonical form is that the observer gains **m** can be obtained from it, just by inspection. The observer-error matrix is

$$(\mathbf{A} - \mathbf{m}\mathbf{c}) = \begin{bmatrix} 0 & 0 & \cdots & 0 & -\alpha_n - m_1 \\ 1 & 0 & \cdots & 0 & -\alpha_{n-1} - m_2 \\ 0 & 1 & \cdots & 0 & -\alpha_{n-2} - m_3 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_1 - m_n \end{bmatrix}$$

which has the characteristic equation

$$s^n + (\alpha_1 + m_n) s^{n-1} + \cdots + (\alpha_{n-2} + m_3)s^2 + (\alpha_{n-1} + m_2) s + \alpha_n + m_1 = 0$$

and the observer gains can be found by comparing the coefficients of this equation with Eqn. (7.34b).

A procedure for observer design, therefore, consists of transforming the given state variable model to observable canonical form, solving for the observer gains, and transforming the gains back to the original state.

We can, however, directly use the equations of the control-law design for computing the observer gain matrix **m**, if we examine the resemblance between the estimation and control problems. In fact, the two problems are mathematically equivalent. This property is called *duality*. The design of a full-order observer requires the determination of the gain matrix **m** such that (**A** – **mc**) has desired eigenvalues λ_i; $i = 1, 2, ..., n$. This is mathematically equivalent to designing a full state-feedback controller for the 'transposed auxiliary system',

$$\dot{\boldsymbol{\zeta}}(t) = \mathbf{A}^T\boldsymbol{\zeta}(t) + \mathbf{c}^T\eta(t) \tag{7.36a}$$

with feedback

$$\eta(t) = -\mathbf{m}^T\boldsymbol{\zeta}(t) \tag{7.36b}$$

so that the closed-loop auxiliary system

$$\dot{\boldsymbol{\zeta}}(t) = (\mathbf{A}^T - \mathbf{c}^T\mathbf{m}^T)\boldsymbol{\zeta}(t) \tag{7.37}$$

has eigenvalues λ_i; $i = 1, 2, ..., n$.

Since

$$det\ \mathbf{W} = det\ \mathbf{W}^T,$$

one obtains

$$det\ [s\mathbf{I} - (\mathbf{A}^T - \mathbf{c}^T\mathbf{m}^T)] = det\ [s\mathbf{I} - (\mathbf{A} - \mathbf{mc})]$$

i.e., the eigenvalues of $(\mathbf{A}^T - \mathbf{c}^T\mathbf{m}^T)$ are same as the eigenvalues of (**A** – **mc**).

Table 7.1 Duality between control and estimation

Control	*Estimation*
A	$\mathbf{A}^T$
b	$\mathbf{c}^T$
k	$\mathbf{m}^T$

By comparing the characteristic equation of the closed-loop system (7.19) and that of the auxiliary system (7.37), we obtain the duality relations given in Table 7.1 between the control and estimation problems. The Ackermann's control-design formula given by Eqns (7.26) becomes the observer-design formula if the substitutions of Table 7.1 are made.

A necessary and sufficient condition for determination of the observer gain matrix **m** for the desired eigenvalues of (**A** – **mc**), is that the auxiliary system (7.36) be completely controllable. The controllability condition for this system is that the rank of

$$[\mathbf{c}^T \quad \mathbf{A}^T\mathbf{c}^T \cdots (\mathbf{A}^T)^{n-1}\ \mathbf{c}^T]$$

is n. This is the condition for complete observability of the original system defined by Eqns (7.29). This means that a necessary and sufficient condition for estimation of the state of the system defined by Eqns (7.29), is that the system be completely observable.

Again by duality, we can say that for the case of single-output systems, the gain matrix **m**, which places the observer poles at desired locations, is unique. In the multi-output case, the same pole configuration

can be achieved by various feedback gain matrices. This non-uniqueness is removed by optimal control theory which is discussed in Chapter 8.

Example 7.4

We will consider the satellite-attitude control system of Example 7.3. The state equation of the plant is

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

with $$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$x_1 = \theta$, the orientation of the satellite; $x_2 = \dot{\theta}$

We assume that the orientation θ can be accurately measured from the antenna signal. Therefore,

$$y = \mathbf{c}\mathbf{x}(t)$$

with $$\mathbf{c} = [1 \quad 0]$$

Let us design a state observer for the system. We choose the observer to be critically damped with a settling time of 0.4 sec $\left(\dfrac{4}{\zeta\omega_n} = 0.4; \zeta\omega_n = 10\right)$. To satisfy these specifications, the observer poles will be placed at $s = -10, -10$.

The transposed auxiliary system is given by

$$\dot{\zeta} = \mathbf{A}^T\zeta + \mathbf{c}^T\eta; \quad \eta = -\mathbf{m}^T\zeta$$

The desired characteristic equation of the closed-loop auxiliary system is

$$s^2 + a_1 s + a_2 = (s + 10)(s + 10) = s^2 + 20s + 100$$

To apply Ackermann's formula given by Eqns (7.26), we compute

$$\mathbf{U}^{-1} = [\mathbf{c}^T \quad \mathbf{A}^T\mathbf{c}^T]^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\phi(\mathbf{A}^T) = (\mathbf{A}^T)^2 + a_1\mathbf{A}^T + a_2\mathbf{I}$$

$$= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + 20\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + 100\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 100 & 0 \\ 20 & 100 \end{bmatrix}$$

The observer gain matrix is given by the equation

$$\mathbf{m}^T = [0 \quad 1]\begin{bmatrix} 100 & 0 \\ 20 & 100 \end{bmatrix} = [20 \quad 100]$$

Therefore, $$\mathbf{m} = \begin{bmatrix} 20 \\ 100 \end{bmatrix}$$

Example 7.5

Consider once again the inverted-pendulum system of Example 7.2. Suppose that the only output available for measurement is $z(t)$, the position of the cart. The linearized equations governing this system are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u; \quad y = \mathbf{c}\mathbf{x}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 16.3106 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1.0637 & 0 & 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ -1.4458 \\ 0 \\ 0.9639 \end{bmatrix}; \mathbf{c} = [0 \quad 0 \quad 1 \quad 0]$$

We verified in Example 5.18 that this system is completely observable. In the following, we design a full-order observer for this system. We choose the observer pole locations as $-2, -2 \pm j1, -3$. The corresponding characteristic equation is

$$s^4 + 9s^3 + 31s^2 + 49s + 30 = 0$$

The transposed auxiliary system is given by

$$\dot{\boldsymbol{\zeta}}(t) = \mathbf{A}^T\boldsymbol{\zeta}(t) + \mathbf{c}^T\eta(t); \; \eta(t) = -\mathbf{m}^T\boldsymbol{\zeta}(t)$$

We will determine the gain matrix $\mathbf{m}$ using the design equations (7.17)–(7.23).

The controllability matrix

$$\mathbf{U} = [\mathbf{c}^T \quad \mathbf{A}^T\mathbf{c}^T \quad (\mathbf{A}^T)^2\mathbf{c}^T \quad (\mathbf{A}^T)^3\mathbf{c}^T] = \begin{bmatrix} 0 & 0 & -1.0637 & 0 \\ 0 & 0 & 0 & -1.0637 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{U}^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.9401 & 0 & 0 & 0 \\ 0 & -0.9401 & 0 & 0 \end{bmatrix}$$

Therefore,

$$\mathbf{p}_1 = [0 \quad -0.9401 \quad 0 \quad 0]$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_1(\mathbf{A}^T) \\ \mathbf{p}_1(\mathbf{A}^T)^2 \\ \mathbf{p}_1(\mathbf{A}^T)^3 \end{bmatrix} = \begin{bmatrix} 0 & -0.9401 & 0 & 0 \\ -0.9401 & 0 & 0 & 0 \\ 0 & -15.3333 & 0 & 1 \\ -15.3333 & 0 & 1 & 0 \end{bmatrix}$$

$$|s\mathbf{I} - \mathbf{A}^T| = s^4 + \alpha_1 s^3 + \alpha_2 s^2 + \alpha_3 s + \alpha_4 = s^4 + 0s^3 - 16.3106s^2 + 0s + 0$$

$$|s\mathbf{I} - (\mathbf{A}^T - \mathbf{c}^T\mathbf{m}^T)| = s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4 = s^4 + 9s^3 + 31s^2 + 49s + 30$$

$$\mathbf{m}^T = [a_4 - \alpha_4 \quad a_3 - \alpha_3 \quad a_2 - \alpha_2 \quad a_1 - \alpha_1]\,\mathbf{P}$$
$$= [30 \quad 49 \quad 47.3106 \quad 9]\,\mathbf{P} = [-184.0641 \quad -753.6317 \quad 9 \quad 47.3106]$$

Therefore,
$$\mathbf{m} = \begin{bmatrix} -184.0641 \\ -753.6317 \\ 9 \\ 47.3106 \end{bmatrix}$$

With this **m**, the observer

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\,\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y$$

will process the cart position $y(t) = z(t)$ and input $u(t)$, to continuously provide an estimate $\hat{\mathbf{x}}(t)$ of the state vector $\mathbf{x}(t)$; and any errors in the estimate will decay at least as fast as e^{-2t}.

7.5.2 Reduced-Order State Observer

The observer developed in the previous subsection reconstructs the entire state vector. However, the measurements usually available are some of the states of the plant. For example, for the satellite-attitude control problem considered in the previous subsection, the measurement is orientation of the satellite, which is $x_1(t)$. The measurement of a state, in general, will be more accurate than any estimate of the state based on the measurement. Hence, it is not logical in most cases to estimate states that we are measuring. One possible exception is the case in which a measurement is very noisy. The state observer for this case may furnish some beneficial noise filtering.

Since we will not usually want to estimate any state that we are measuring, we prefer to design an observer that estimates only those states that are not measured. This type of observer is called a *reduced-order state observer*. We develop design equations for such an observer in this subsection. We consider only the case of one measurement. It is assumed that the state variables are always chosen such that the state measured is $x_1(t)$; we can do this without loss of generality. The output equation then is given by

$$y(t) = x_1(t) = \mathbf{c}\mathbf{x}(t)$$

where $\mathbf{c} = [1 \quad 0 \quad 0 \cdots 0]$

To derive the reduced-order observer, we partition the state vector into two parts: one part is x_1 which is directly measured and the other part is $\mathbf{x}_e$, representing the state variables that need to be estimated:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \mathbf{x}_e(t) \end{bmatrix}$$

If we partition the system matrices accordingly, the complete description of the system is given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{\mathbf{x}}_e \end{bmatrix} = \begin{bmatrix} a_{11} & \mathbf{a}_{1e} \\ \mathbf{a}_{e1} & \mathbf{A}_{ee} \end{bmatrix} \begin{bmatrix} x_1 \\ \mathbf{x}_e \end{bmatrix} + \begin{bmatrix} b_1 \\ \mathbf{b}_e \end{bmatrix} u \tag{7.38a}$$

$$y = [1 \quad \mathbf{0}] \begin{bmatrix} x_1 \\ \mathbf{x}_e \end{bmatrix} \tag{7.38b}$$

The dynamics of the unmeasured state variables are given by

$$\dot{\mathbf{x}}_\mathbf{e} = \mathbf{A}_{ee}\,\mathbf{x}_e + \underbrace{\mathbf{a}_{e1}x_1 + \mathbf{b}_e u}_{\text{known input}} \tag{7.39}$$

where the two rightmost terms are known and can be considered as an input into the $\mathbf{x}_e$ dynamics. Since $x_1 = y$, measured dynamics are given by the scalar equation

$$\dot{x}_1 = \dot{y} = a_{11}\, y + \mathbf{a}_{1e}\, \mathbf{x}_e + b_1 u \tag{7.40}$$

If we collect the known terms of Eqn. (7.40) on one side, we get

$$\underbrace{\dot{y} - a_{11} y - b_1 u}_{\text{known measurement}} = \mathbf{a}_{1e}\mathbf{x}_e \tag{7.41}$$

Note that Eqns (7.39) and (7.41) have the same relationship to the state $\mathbf{x}_e$ that the original equations (7.38) had to the entire state $\mathbf{x}$. Following this line of reasoning, we can establish the following substitutions in the original observer-design equations, to obtain an (reduced-order) observer of $\mathbf{x}_e$:

$$\begin{aligned} \mathbf{x} &\leftarrow \mathbf{x}_e \\ \mathbf{A} &\leftarrow \mathbf{A}_{ee} \\ \mathbf{b}u &\leftarrow \mathbf{a}_{e1} y + \mathbf{b}_e u \\ y &\leftarrow \dot{y} - a_{11} y - b_1 u \\ \mathbf{c} &\leftarrow \mathbf{a}_{1e} \end{aligned} \tag{7.42}$$

Making these substitutions into the equation for full-order observer (Eqn. (7.31)), we obtain the equation of the reduced-order observer:

$$\dot{\hat{\mathbf{x}}}_e = \mathbf{A}_{ee}\, \hat{\mathbf{x}}_e + \underbrace{\mathbf{a}_{e1} y + \mathbf{b}_e u}_{\text{input}} + \mathbf{m}\, \underbrace{(\dot{y} - a_{11} y - b_1 u}_{\text{measurement}} - \mathbf{a}_{1e}\, \hat{\mathbf{x}}_e) \tag{7.43}$$

If we define the estimation error as

$$\tilde{\mathbf{x}}_e = \mathbf{x}_e - \hat{\mathbf{x}}_e \tag{7.44}$$

the dynamics of error are given by subtracting Eqn. (7.43) from Eqn. (7.39):

$$\dot{\tilde{\mathbf{x}}}_e = (\mathbf{A}_{ee} - \mathbf{m}\mathbf{a}_{1e})\, \tilde{\mathbf{x}}_e \tag{7.45}$$

Its characteristic equation is given by

$$|\, s\mathbf{I} - (\mathbf{A}_{ee} - \mathbf{m}\mathbf{a}_{1e})\,| = 0 \tag{7.46}$$

We design the dynamics of this observer by selecting $\mathbf{m}$ so that Eqn. (7.46) matches a desired reduced-order characteristic equation. To carry out the design using state regulator results, we form a 'transposed auxiliary system'

$$\begin{aligned} \dot{\boldsymbol{\zeta}}(t) &= \mathbf{A}_{ee}^T\, \boldsymbol{\zeta}(t) + \mathbf{a}_{1e}^T\, \eta(t) \\ \eta(t) &= -\mathbf{m}^T \boldsymbol{\zeta}(t) \end{aligned} \tag{7.47}$$

Use of Ackermann's formula given by Eqns (7.26) for this auxiliary system gives the gains $\mathbf{m}$ of the reduced-order observer. We should point out that the conditions for the existence of the reduced-order observer are the same as for the full-order observer—namely observability of the pair ($\mathbf{A}$, $\mathbf{c}$).

Let us now look at the implementational aspects of the reduced-order observer given by Eqn. (7.43). This equation can be rewritten as

$$\dot{\hat{\mathbf{x}}}_e = (\mathbf{A}_{ee} - \mathbf{m}\mathbf{a}_{1e})\, \hat{\mathbf{x}}_e + (\mathbf{a}_{e1} - \mathbf{m}a_{11}) y + (\mathbf{b}_e - \mathbf{m}b_1) u + \mathbf{m}\, \dot{y} \tag{7.48}$$

The fact that the reduced-order observer requires the derivative of $y(t)$ as an input, appears to present a practical difficulty. It is known that differentiation amplifies noise, so if y is noisy the use of $\dot{y}$ is unacceptable. To get around this difficulty, we define the new state as

$$\mathbf{x}'_e \triangleq \hat{\mathbf{x}}_e - \mathbf{m}y \tag{7.49a}$$

Then, in terms of this new state, the implementation of the reduced-order observer is given by

$$\dot{\mathbf{x}}'_e = (\mathbf{A}_{ee} - \mathbf{m}\mathbf{a}_{1e})\,\hat{\mathbf{x}}_e + (\mathbf{a}_{e1} - \mathbf{m}a_{11})y + (\mathbf{b}_e - \mathbf{m}b_1)u \tag{7.49b}$$

and $\dot{y}$ no longer appears directly. A block-diagram representation of the reduced-order observer is shown in Fig. 7.7.

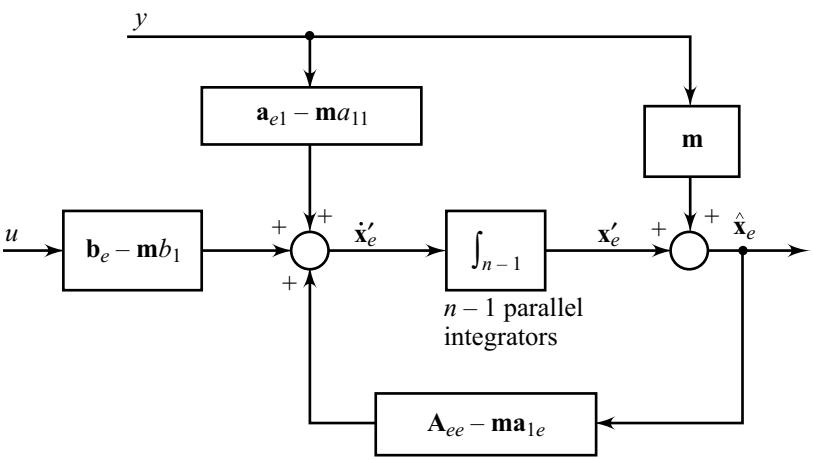


Fig. 7.7 Reduced-order observer structure

Example 7.6

In Example 7.4, a second-order observer for the satellite-attitude control system was designed with the observer poles at $s = -10, -10$. We now design a reduced-order (first-order) observer for the system with observer pole at $s = -10$.

The plant equations are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

The partitioned matrices are

$$\left[\begin{array}{c:c} a_{11} & a_{1e} \\ \hdashline a_{e1} & A_{ee} \end{array}\right] = \left[\begin{array}{c:c} 0 & 1 \\ \hdashline 0 & 0 \end{array}\right] = \left[\begin{array}{c} b_1 \\ \hdashline b_e \end{array}\right] = \left[\begin{array}{c} 0 \\ \hdashline 1 \end{array}\right]$$

From Eqn. (7.46), we find the characteristic equation in terms of m:

$$s - (0 - m) = 0$$

We compare it with the desired equation

$$s + 10 = 0$$

which yields,

$$m = 10$$

The observer equations are (refer to Eqns. (7.49))

$$\dot{x}_2' = -10\,\hat{x}_2 + u$$
$$\hat{x}_2 = x_2' + 10y$$

This completes the design of the reduced-order observer which estimates the angular velocity of the satellite from the measurement of the angular position.

7.6 COMPENSATOR DESIGN BY THE SEPARATION PRINCIPLE

In Sections 7.2–7.4, we studied the design of control laws for systems in which the state variables are all accessible for measurement. We promised to overcome the difficulty of not being able to measure all the state variables by the use of an observer to estimate those state variables that cannot be measured. Then in Section 7.5, we studied the design of observers for systems with known inputs, but not when the state estimate is used for the purpose of control. We are now ready to combine the state-feedback control law with the observer to obtain a compensator for linear systems in which not all the state variables can be measured.

Consider the completely controllable and completely observable system defined by the equations

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u \\ y &= \mathbf{c}\mathbf{x} \end{aligned} \tag{7.50}$$

Suppose we have designed a state-feedback control law

$$u = -\mathbf{k}\mathbf{x} \tag{7.51}$$

using the methods of Section 7.4. And also suppose we have designed a full-order observer

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\,\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}(y - \mathbf{c}\,\hat{\mathbf{x}}) \tag{7.52}$$

using the methods of Section 7.5.

For the state-feedback control based on the observed state $\hat{\mathbf{x}}$,

$$u = -\mathbf{k}\,\hat{\mathbf{x}} \tag{7.53}$$

The control system based on combining the state-feedback control law and state observer, has the configuration shown in Fig. 7.8. Note that the number of state variables in the compensator is equal to the order of the embedded observer and hence is equal to the order of the plant. Thus, the order of the overall closed-loop system, when a full-order observer is used in the compensator, is $2n$ for a plant of

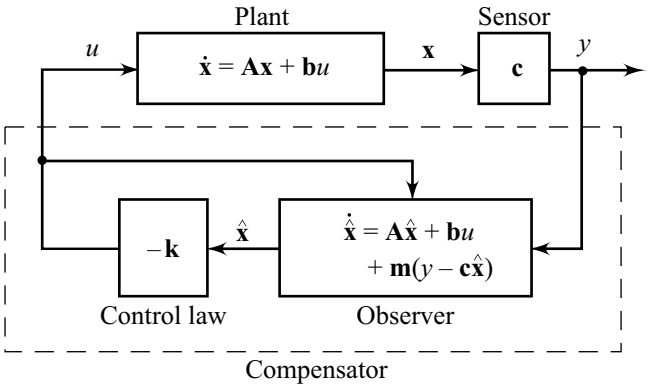


Fig. 7.8 Combined state-feedback control and state estimation

order n. We are interested in the dynamic behavior of the $2n$th-order system comprising the plant and the compensator. With the control law (7.53) used, the plant dynamics become

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{b}\mathbf{k}\hat{\mathbf{x}} = (\mathbf{A} - \mathbf{b}\mathbf{k})\mathbf{x} + \mathbf{b}\mathbf{k}(\mathbf{x} - \hat{\mathbf{x}}) \tag{7.54}$$

The difference between the actual state $\mathbf{x}$ and observed state $\hat{\mathbf{x}}$, has been defined as the error $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$$

Substitution of the error vector into Eqn. (7.54) gives

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{b}\mathbf{k})\mathbf{x} + \mathbf{b}\mathbf{k}\tilde{\mathbf{x}} \tag{7.55}$$

Note that the observer error was given by Eqn. (7.33), repeated here

$$\dot{\tilde{\mathbf{x}}} = (\mathbf{A} - \mathbf{m}\mathbf{c})\tilde{\mathbf{x}} \tag{7.56}$$

Combining Eqns (7.55) and (7.56), we obtain

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\tilde{\mathbf{x}}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{b}\mathbf{k} & \mathbf{b}\mathbf{k} \\ \mathbf{0} & \mathbf{A} - \mathbf{m}\mathbf{c} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \tilde{\mathbf{x}} \end{bmatrix} \tag{7.57}$$

Equation (7.57) describes the dynamics of the $2n$-dimensional system of Fig. 7.8. The characteristic equation for the system is

$$|s\mathbf{I} - (\mathbf{A} - \mathbf{b}\mathbf{k})|\,|s\mathbf{I} - (\mathbf{A} - \mathbf{m}\mathbf{c})| = 0$$

In other words, the poles of the combined system consist of the union of control and observer roots. This means that the design of the control law and the observer can be carried out independently. Yet, when they are used together, the roots are unchanged. This is a special case of the *separation principle*, which holds in much more general contexts and allows for the separate design of control law and estimator in certain stochastic cases.

To compare the state-variable method of design with the transform methods discussed in earlier chapters, we obtain the transfer function model of the compensator used in the control system of Fig. 7.8. The state variable model for this compensator is obtained by including the feedback law $u = -\mathbf{k}\hat{\mathbf{x}}$ (since it is part of the compensator) in the observer equation (7.52).

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= (\mathbf{A} - \mathbf{b}\mathbf{k} - \mathbf{m}\mathbf{c})\hat{\mathbf{x}} + \mathbf{m}y \\ u &= -\mathbf{k}\hat{\mathbf{x}} \end{aligned} \tag{7.58}$$

The formula for conversion of state variable model to the transfer function model is given by Eqn. (5.28). Applying this result to the model given by Eqn. (7.58), we obtain

$$\frac{U(s)}{-Y(s)} = D(s) = \mathbf{k}(s\mathbf{I} - \mathbf{A} + \mathbf{bk} + \mathbf{mc})^{-1}\mathbf{m} \tag{7.59}$$

Figure 7.9 shows the block diagram representation of the system with observer-based controller.

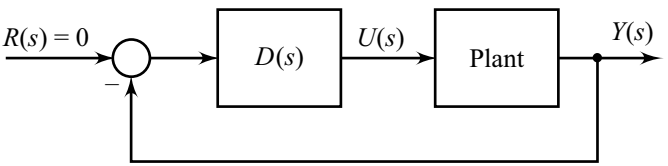


Fig. 7.9 Block diagram representation of a system with observer-based controller

Note that the poles of $D(s)$ in Eqn. (7.59) were neither specified nor used during the state-variable design process. It may even happen that $D(s)$ has one or more poles in the right-half plane; the compensator, in other words, could turn out to be unstable. But the closed-loop system, if so designed, would be stable. There is, however, one problem if the compensator is unstable. The open-loop poles of the system are the poles of the plant and also the poles of the compensator. If the latter are in the right-half plane, then the closed-loop poles may be in the right-half plane when the loop gain becomes too small. Robustness considerations put certain restrictions on the use of unstable compensators to stabilize a system.

Example 7.7

In this example, we study the closed-loop system obtained by implementing the state-feedback control law of Example 7.3 and state-observer design of Examples 7.4 and 7.6, for the attitude control of a satellite. The plant model is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u; \quad y = \mathbf{c}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \quad 0]$$

In Example 7.3, the gain matrix required to place the closed-loop poles at $s = -4 \pm j4$ was calculated to be

$$\mathbf{k} = [32 \quad 8]$$

If both the state variables are available for feedback, the control law becomes

$$u = -\mathbf{k}\mathbf{x} = -[32 \quad 8]\mathbf{x}$$

resulting in the closed-loop system

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{bk})\mathbf{x} = \begin{bmatrix} 0 & 1 \\ -32 & -8 \end{bmatrix} \mathbf{x}$$

Figure 7.10a shows the response of the system to an initial condition $\mathbf{x}(0) = [1 \quad 0]^T$. Assume now that the state-feedback control law is implemented using a full-order observer. In Example 7.4, the observer gain matrix was calculated to be

$$\mathbf{m} = \begin{bmatrix} 20 \\ 100 \end{bmatrix}$$

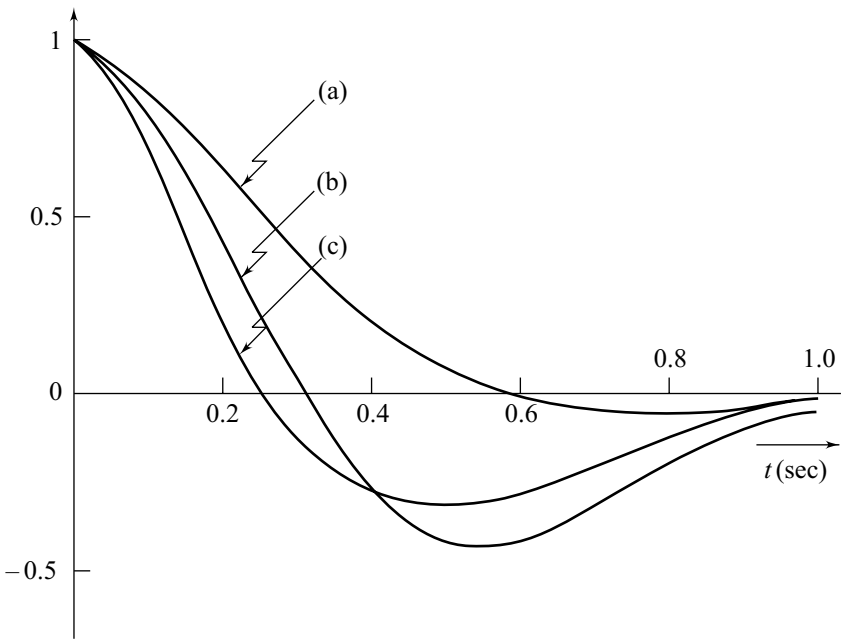


Fig. 7.10 Initial condition response: (a) Without observer (b) With full-order observer (c) With reduced-order observer

The state variable model of the compensator, obtained by cascading the state-feedback control law and the state observer, is obtained as (refer to Eqns (7.58))

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{bk} - \mathbf{mc})\,\hat{\mathbf{x}} + \mathbf{m}y = \begin{bmatrix} -20 & 1 \\ -132 & -8 \end{bmatrix} \hat{\mathbf{x}} + \begin{bmatrix} 20 \\ 100 \end{bmatrix} y$$

$$u = -\mathbf{k}\,\hat{\mathbf{x}} = -[32 \quad 8]\ \hat{\mathbf{x}}$$

The compensator transfer function is (refer to Eqn. (7.59))

$$D(s) = \frac{U(s)}{-Y(s)} = \mathbf{k}(s\mathbf{I} - \mathbf{A} + \mathbf{bk} + \mathbf{mc})^{-1}\mathbf{m} = \frac{1440s + 3200}{s^2 + 28s + 292}$$

The state variable model of the closed-loop system can be constructed as follows:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = u = -32\,\hat{x}_1 - 8\,\hat{x}_2$$

$$\dot{\hat{x}}_1 = -20\,\hat{x}_1 + \hat{x}_2 + 20y = -20\,\hat{x}_1 + \hat{x}_2 + 20x_1$$

$$\dot{\hat{x}}_2 = -132\,\hat{x}_1 - 8\,\hat{x}_2 + 100x_1$$

Therefore,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -32 & -8 \\ 20 & 0 & -20 & 1 \\ 100 & 0 & -132 & -8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}$$

Figure 7.10b shows the response to an initial condition

$$[1 \quad 0 \quad 0 \quad 0]^T$$

Consider now the implementation of state-feedback control law using reduced-order observer. In Example 7.6, the following model was obtained to estimate the state x_2 (state x_1 is directly measured and fed back, and is not estimated using an observer):

$$\hat{x}_2' = x_2' + 10y$$
$$\dot{x}_2' = -10\,\hat{x}_2 + u$$

The control law is given by

$$u = -32x_1 - 8\,\hat{x}_2$$

From these equations, the following transfer function model of the compensator is obtained:

$$\frac{U(s)}{-Y(s)} = \frac{112(s+2.86)}{s+18}$$

The reduced-order compensator is precisely the lead network; this is a pleasant discovery, as it shows that the classical and state variable methods can result in exactly the same type of compensation.

The state variable model of the closed-loop system with the reduced-order compensator is derived below.

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = u = -32x_1 - 8\,\hat{x}_2 = -32x_1 - 8(x_2' + 10x_1) = -112x_1 - 8x_2'$$
$$\dot{x}_2' = -10\,\hat{x}_2 + u = -10\,\hat{x}_2 - 32x_1 - 8\,\hat{x}_2$$
$$= -18(x_2' + 10x_1) - 32x_1 = -18x_2' - 212x_1$$

Therefore,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -112 & 0 & -8 \\ -212 & 0 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_2' \end{bmatrix}$$

Figure 7.10c shows the response to an initial condition

$$[1 \quad 0 \quad 0]^T$$

Comments

1. Underlying the separation principle is a critical assumption, namely, that the observer includes an exact dynamic model of the plant—the process under control. This assumption is almost never valid in reality. In practical systems, the precise dynamic model is rarely known. Even that which is known about

the real process dynamics, is often too complicated to include in the observer. Thus, the observer must, in practice, be configured to use only an approximate model of the plant. This encounter with the real world does not vitiate the separation principle, but means that the effect of an inaccurate plant model must be considered. If the design achieved through use of the separation principle is *robust*, it will be able to tolerate uncertainty of the plant dynamics. Doyle and Stein [124] have proposed a 'design adjustment procedure' to improve robustness with observers.

2. One of the considerations in the design of a gain matrix **k** in the state-feedback control law, is that the resulting control signal u must not be too large; the use of large control effort increases the likelihood of the system entering nonlinear regions of operation. Since the function of the observer is only to process data, there is no limitation on the size of the gain matrix **m** for its realization. (Nowadays, it is all but certain that the entire compensator would be realized by a digital computer. With floating-point numerics, a digital computer would be capable of handling variables of any reasonable dynamic range). Though the realization of observer may impose no limitation on the observer dynamics, it may, nevertheless, be desirable to limit the observer speed of response (bandwidth). Remember that real sensors are noisy, and much of the noise occurs at relatively high frequencies. By limiting the bandwidth of the observer, we can attenuate and smoothen the noise contribution to the compensator output—which is the control signal.

3. The desired closed-loop poles, to be generated by state feedback are chosen to satisfy the performance requirements. The poles of the observer are usually chosen so that the observer response is much faster than the system response. A rule of thumb is to choose an observer response at least two to five times faster than the system response. This is to ensure a faster decay of estimation errors compared with the desired dynamics, thus causing the closed-loop poles generated by state feedback to dominate the total response. If the sensor noise is large enough to be a major concern, one may decide to choose the observer poles to be slower than two times the system poles, which would yield a system with lower bandwidth and more noise-smoothing. However, the total system response in this case will be strongly influenced by the observer poles. Doyle and Stein [124] have shown that the commonly suggested approach of 'speeding-up' observer dynamics will not work in all cases. They have suggested that procedures which drive some observer poles towards stable plant zeros and the rest towards infinity achieve the desired objective.

4. A final comment concerns the reduced-order observer. Due to the presence of a direct transmission term (refer to Fig. 7.7), the reduced-order observer has much higher bandwidth from sensor to control, compared with the full-order observer. Therefore, if sensor noise is a significant factor, the reduced-order observer is less attractive, since the potential savings in complexity is more than offset by the increased sensitivity to noise.

7.7 SERVO DESIGN: INTRODUCTION OF THE REFERENCE INPUT BY FEEDFORWARD CONTROL

In the state regulator design studied in Section 7.4, the characteristic equation of the closed-loop system is chosen so as to give satisfactory transients to disturbances. However, no mention is made of a reference input or of design considerations to yield good transient response with respect to command changes. In

general, these considerations should be taken into account in the design of a control system. This can be done by proper introduction of the reference input into the system equations.

Consider the completely controllable SISO linear time-invariant system with *n*th-order state variable model

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t) \\ y(t) &= \mathbf{c}\mathbf{x}(t) \end{aligned} \tag{7.60}$$

We assume that all the n state variables can be accurately measured at all times. Implementation of appropriately designed control law of the form

$$u(t) = -\mathbf{k}\mathbf{x}(t)$$

results in a state regulator system; any perturbation in the system state will asymptotically decay to the equilibrium state $\mathbf{x} = \mathbf{0}$.

Let us now assume that for the system given by Eqns (7.60), the desired steady-state value of the controlled variable $y(t)$ is a constant reference input r. For this servo system, the desired equilibrium state $\mathbf{x}_s$ is a constant point in state space and is governed by the equations

$$\mathbf{c}\mathbf{x}_s = r \tag{7.61}$$

We can formulate this command-following problem as a 'shifted regulator problem', by shifting the origin of the state space to the equilibrium point $\mathbf{x}_s$. Formulation of the shifted regulator problem is as follows.

Let u_s be the needed input to maintain $\mathbf{x}(t)$ at the equilibrium point $\mathbf{x}_s$, i.e. (refer to Eqns (7.60)),

$$\mathbf{0} = \mathbf{A}\mathbf{x}_s + \mathbf{b}u_s \tag{7.62}$$

Assuming for the present that a u_s exists that satisfies Eqns (7.61)–(7.62), we define shifted input, shifted state, and shifted controlled variable as

$$\begin{aligned} \tilde{u}(t) &= u(t) - u_s \\ \tilde{\mathbf{x}}(t) &= \mathbf{x}(t) - \mathbf{x}_s \\ \tilde{y}(t) &= y(t) - r \end{aligned} \tag{7.63}$$

The shifted variables satisfy the equations

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \mathbf{A}\tilde{\mathbf{x}} + \mathbf{b}\tilde{u} \\ \tilde{y} &= \mathbf{c}\tilde{\mathbf{x}} \end{aligned} \tag{7.64}$$

This system possesses a time-invariant asymptotically stable control law

$$\tilde{u} = -\mathbf{k}\tilde{\mathbf{x}} \tag{7.65}$$

The application of this control law ensures that

$$\tilde{\mathbf{x}} \to \mathbf{0} \; (\mathbf{x}(t) \to \mathbf{x}_s, y(t) \to r)$$

In terms of the original state variables, total control effort

$$u(t) = -\mathbf{k}\mathbf{x}(t) + u_s + \mathbf{k}\mathbf{x}_s \tag{7.66}$$

Manipulation of Eqn. (7.62) gives

$$(\mathbf{A} - \mathbf{b}\mathbf{k})\mathbf{x}_s + \mathbf{b}(u_s + \mathbf{k}\mathbf{x}_s) = \mathbf{0} \quad \text{or} \quad \mathbf{x}_s = -(\mathbf{A} - \mathbf{b}\mathbf{k})^{-1}\mathbf{b}(u_s + \mathbf{k}\mathbf{x}_s)$$

or

$$\mathbf{c}\mathbf{x}_s = r = -\mathbf{c}(\mathbf{A} - \mathbf{b}\mathbf{k})^{-1}\mathbf{b}(u_s + \mathbf{k}\mathbf{x}_s)$$

This equation has a unique solution for $(u_s + \mathbf{k}\mathbf{x}_s)$:

$$(u_s + \mathbf{k}\mathbf{x}_s) = Nr$$

where N is a scalar feedforward gain, given by

$$(N)^{-1} = -\mathbf{c}(\mathbf{A} - \mathbf{b}\mathbf{k})^{-1}\mathbf{b} \tag{7.67}$$

The control law (7.66), therefore, takes the form

$$u(t) = -\mathbf{k}\mathbf{x}(t) + Nr \tag{7.68}$$

The block diagram of Fig. 7.11 shows the configuration of feedback control system with feedforward compensation for nonzero equilibrium state.

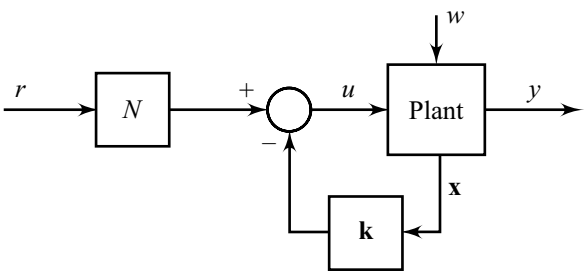


Fig. 7.11 Control configuration of a servo system

Example 7.8

The system considered in this example is the attitude control system for a rigid satellite. The plant equations are (refer to Example 7.1)

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u; \quad y = \mathbf{c}\mathbf{x}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \quad 0]$$

$$x_1(t) = \text{position } \theta(t); x_2(t) = \text{velocity } \dot{\theta}(t)$$

The reference input $r = \theta_r$ is a step function. The desired steady state is

$$\mathbf{x}_s = [\theta_r \quad 0]^T,$$

which is a non-null state.

As the plant has integrating property, the steady-state value u_s of the input must be zero (otherwise the output cannot stay constant). For this case, the shifted regulator problem may be formulated as follows:

$$\tilde{x}_1 = x_1 - \theta_r; \quad \tilde{x}_2 = x_2$$

Shifted state variables satisfy the equation

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{b}u$$

The state-feedback control

$$u = -\mathbf{k}\,\tilde{\mathbf{x}}$$

results in dynamics of $\tilde{\mathbf{x}}$ given by

$$\dot{\tilde{\mathbf{x}}} = (\mathbf{A} - \mathbf{bk})\,\tilde{\mathbf{x}}$$

In Example 7.1, we found that the eigenvalues of $(\mathbf{A} - \mathbf{bk})$ are placed at the desired locations $-4 \pm j4$ when

$$\mathbf{k} = [k_1 \quad k_2] = [32 \quad 8]$$

The control law expressed in terms of the original state variables is given as

$$u = -k_1\tilde{x}_1 - k_2\tilde{x}_2 = -k_1x_1 - k_2x_2 + k_1\theta_r = -\mathbf{kx} + k_1\theta_r$$

As t approaches infinity, $\tilde{\mathbf{x}} \to \mathbf{0}$ ($\mathbf{x} \to [\theta_r \quad 0]^T$), and $u \to 0$.

Figure 7.12 shows a configuration for attitude control of the satellite.

In fact, control configuration of the form shown in Fig. 7.12 may be used for any SISO plant with integrating property.

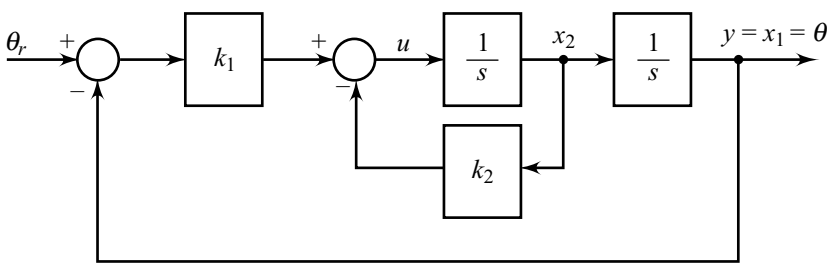


Fig. 7.12 Attitude control of a satellite

7.8 STATE FEEDBACK WITH INTEGRAL CONTROL

Control configuration of Fig. 7.12 produces a generalization of proportional and derivative feedback but it does not include integral control unless special steps are taken in the design process. One way to introduce integral control is to augment the state vector with the desired integral. More specifically, for the system (7.60), we can feedback the state $\mathbf{x}$ as well as the integral of the error in output by augmenting the plant state $\mathbf{x}$ with the extra 'integral state' z, defined by the equation

$$z(t) = \int_0^t (y(t) - r)dt \tag{7.69a}$$

where r is the constant reference input of the system. Since $z(t)$ satisfies the differential equation

$$\dot{z}(t) = y(t) - r = \mathbf{cx}(t) - r \tag{7.69b}$$

it is easily included by augmenting the original system (7.60) as follows:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{c} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ z \end{bmatrix} + \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} u + \begin{bmatrix} \mathbf{0} \\ -1 \end{bmatrix} r \tag{7.70}$$

Since r is constant, in the steady state $\dot{\mathbf{x}} = \mathbf{0}$, $\dot{z} = 0$, provided that the system is stable. This means that the steady-state solutions $\mathbf{x}_s$, z_s and u_s must satisfy the equation

$$\begin{bmatrix} \mathbf{0} \\ -1 \end{bmatrix} r = -\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{c} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_s \\ z_s \end{bmatrix} - \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} u_s$$

Substituting this for the last term in Eqn. (7.70), gives

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{c} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} - \mathbf{x}_s \\ z - z_s \end{bmatrix} + \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} (u - u_s) \tag{7.71}$$

Now define new state variables as follows, representing the deviations from the steady state:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} - \mathbf{x}_s \\ z - z_s \end{bmatrix}; \; \tilde{u} = u - u_s \tag{7.72a}$$

In terms of these variables, Eqn. (7.71) becomes

$$\dot{\tilde{\mathbf{x}}} = \bar{\mathbf{A}}\tilde{\mathbf{x}} + \bar{\mathbf{b}}\tilde{u} \tag{7.72b}$$

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{c} & 0 \end{bmatrix}, \; \bar{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix}$$

The significance of this result is that, by defining the deviations from steady state as state and control variables, the design problem has been reformulated to be the standard regulator problem, with $\tilde{\mathbf{x}} = \mathbf{0}$ as the desired state. We assume that an asymptotically stable solution to this problem exists, and is given by

$$\tilde{u} = -\mathbf{k}\tilde{\mathbf{x}}$$

Partitioning $\mathbf{k}$ appropriately and using Eqns (7.72a) yields

$$\mathbf{k} = [\mathbf{k}_p \quad k_i]$$

$$u - u_s = -[\mathbf{k}_p \quad k_i] \begin{bmatrix} \mathbf{x} - \mathbf{x}_s \\ z - z_s \end{bmatrix} = -\mathbf{k}_p(\mathbf{x} - \mathbf{x}_s) - k_i(z - z_s)$$

The steady-state terms must balance, therefore,

$$u = -\mathbf{k}_p \mathbf{x} - k_i z = -\mathbf{k}_p \mathbf{x} - k_i \int_0^t (y(t) - r)dt \tag{7.73}$$

The control, thus, consists of proportional state feedback and integral control of output error. At steady-state, $\dot{\tilde{\mathbf{x}}} = \mathbf{0}$; therefore,

$$\lim_{t \to \infty} \dot{z}(t) \to 0 \quad \text{or} \quad \lim_{t \to \infty} y(t) \to r$$

Thus, by integrating action, the output y is driven to the no-offset condition.

This will be true even in the presence of constant disturbances acting on the plant. Block diagram of Fig. 7.13 shows the configuration of feedback control system with proportional state feedback and integral control of output error.

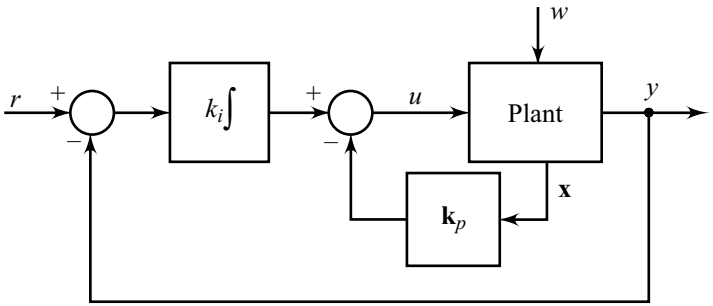


Fig. 7.13 State feedback with integral control

Example 7.9

Suppose the system is given by

$$\frac{Y(s)}{U(s)} = \frac{1}{s+3}$$

with a constant reference command signal. We wish to have integral control with closed-loop poles at $\omega_n = 5$ and $\zeta = 0.5$, which is equivalent to asking for a desired characteristic equation

$$s^2 + 5s + 25 = 0$$

The plant model is

$$\dot{x} = -3x + u; \quad y = x$$

Augmenting the plant state x with the integral state z—defined by the equation

$$z(t) = \int_0^t (y(t) - r)dt$$

we obtain

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -3 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ -1 \end{bmatrix} r$$

In terms of state and control variables representing deviations from the steady state:

$$\tilde{\mathbf{x}} = \begin{bmatrix} x - x_s \\ z - z_s \end{bmatrix}; \ \tilde{u} = u - u_s$$

the state equation becomes

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} -3 & 0 \\ 1 & 0 \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tilde{u}$$

We can find $\mathbf{k}$ from

$$det\left(s\mathbf{I} - \begin{bmatrix} -3 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{k} \right) = s^2 + 5s + 25$$

or $$s^2 + (3 + k_1)s + k_2 = s^2 + 5s + 25$$

Therefore,

$$\mathbf{k} = [2 \quad 25] = [k_p \quad k_i]$$

The control

$$u = -k_p x - k_i z = -2x - 25 \int_0^t (y(t) - r)dt$$

The control configuration is shown in Fig. 7.14, along with a disturbance input w. This system will behave according to the desired closed-loop roots ($\omega_n = 5$, $\zeta = 0.5$) and will exhibit the characteristics of integral control: zero steady-state error to a step r and zero steady-state error to a constant disturbance w.

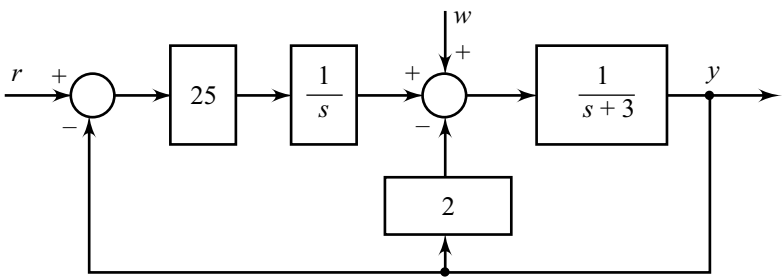


Fig. 7.14 Integral control example

7.9 DIGITAL CONTROL SYSTEMS WITH STATE FEEDBACK

This section covers the key results on the pole-placement design, and state observers for discrete-time systems. Our discussion will be brief because of the strong analogy between the discrete-time and continuous-time cases. Consider the discretized model of the given plant:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{c}\mathbf{x}(k) \end{aligned} \tag{7.74}$$

where $\mathbf{x}$ is the $n \times 1$ state vector, u is the scalar input, y is the scalar output; $\mathbf{F}$, $\mathbf{g}$, and $\mathbf{c}$ are, respectively, $n \times n$, $n \times 1$ and $1 \times n$ real constant matrices; and $k = 0, 1, 2, \ldots$.

We will carry out the design of digital control system for the plant (7.74) in two steps. One step assumes that we have all the elements of the state vector at our disposal for feedback purposes. The next step is to design a state observer which estimates the entire state vector, when provided with the measurements of the system indicated by the output equation in (7.74).

The final step will consist of combining the control law and the observer, where the control law calculations are based on the estimated state variables rather than the actual state.

7.9.1 State Regulator Design

Consider the nth-order, single-input, linear time-invariant system (7.74) with state-feedback control law

$$u(k) = -\mathbf{k}\mathbf{x}(k) \tag{7.75}$$

where

$$\mathbf{k} = [k_1 \quad k_2 \cdots k_n]$$

The resulting closed-loop system is

$$\mathbf{x}(k+1) = (\mathbf{F} - \mathbf{g}\mathbf{k})\mathbf{x}(k) \tag{7.76}$$

If all the eigenvalues of $(\mathbf{F} - \mathbf{g}\mathbf{k})$ are placed inside the unit circle in the complex plane, the state $\mathbf{x}(k)$ will decay to the equilibrium state $\mathbf{x} = \mathbf{0}$ irrespective of the value of $\mathbf{x}(0)$—the initial perturbation in the state.

A necessary and sufficient condition for arbitrary placement of closed-loop eigenvalues (with the restriction that complex eigenvalues occur in conjugate pairs), is that the system (7.74) is completely controllable.

The characteristic equation of the closed-loop system is

$$|z\mathbf{I} - (\mathbf{F} - \mathbf{g}\mathbf{k})| = 0 \tag{7.77a}$$

Assuming that the desired characteristic equation is

$$(z-\lambda_1)(z-\lambda_2)\cdots(z-\lambda_n) = z^n + a_1 z^{n-1} + \cdots + a_{n-1}z + a_n = 0 \tag{7.77b}$$

the required elements of $\mathbf{k}$ are obtained by matching coefficients in Eqns (7.77a) and (7.77b).

The calculation of the gains using this method becomes rather tedious when the order of the system is greater than three. The algebra for finding the gains becomes especially simple when the state variable equations are in controllable canonical form. A design procedure based on the use of controllable canonical state variable model, is given below (refer to Eqns (7.20)–(7.23)).

Step 1 From the characteristic polynomial of matrix $\mathbf{F}$:

$$|z\mathbf{I} - \mathbf{F}| = z^n + \alpha_1 z^{n-1} + \cdots + \alpha_{n-1}\, z + \alpha_n \tag{7.78}$$

determine the values of $\alpha_1, \alpha_2, \ldots, \alpha_n$.

Step 2 Determine the transformation matrix $\mathbf{P}$ that transforms the system (7.74) into controllable canonical form:

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_1\mathbf{F} \\ \vdots \\ \mathbf{p}_1\mathbf{F}^{n-1} \end{bmatrix}; \quad \begin{aligned} \mathbf{p}_1 &= [0 \quad 0 \cdots 0 \quad 1]\,\mathbf{U}^{-1} \\ \mathbf{U} &= [\mathbf{g} \quad \mathbf{F}\mathbf{g} \cdots \mathbf{F}^{n-1}\mathbf{g}] \end{aligned} \tag{7.79}$$

Step 3 Using the desired eigenvalues (desired closed-loop poles) $\lambda_1, \lambda_2, \ldots, \lambda_n$, write the desired characteristic polynomial:

$$(z-\lambda_1)(z-\lambda_2)\cdots(z-\lambda_n) = z^n + a_1\, z^{n-1} + \cdots + a_{n-1}\, z + a_n, \tag{7.80}$$

and determine the values of $a_1, a_2, \cdots, a_{n-1}, a_n$.

Step 4 The required state-feedback gain matrix is determined from the following equation:

$$\mathbf{k} = [a_n - \alpha_n \quad a_{n-1} - \alpha_{n-1} \cdots a_1 - \alpha_1]\mathbf{P} \tag{7.81}$$

The *Ackermann's formula* given below is more convenient for computer solution (refer to Eqns (7.26)).

$$\mathbf{k} = [0 \quad 0 \cdots 0\ 1\ ]\mathbf{U}^{-1}\phi(\mathbf{F}) \tag{7.82}$$

where

$$\phi(\mathbf{F}) = \mathbf{F}^n + a_1\mathbf{F}^{n-1} + \cdots + a_{n-1}\,\mathbf{F} + a_n\mathbf{I}$$

$$\mathbf{U} = [\mathbf{g} \quad \mathbf{Fg} \cdots \mathbf{F}^{n-1}\mathbf{g}]$$

Example 7.10

Consider the problem of attitude control of a rigid satellite. A state variable model of the plant is (refer to Eqn. (7.6))

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{b}u = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

where $x_1 = \theta$ is the attitude angle and u is the system input.

The discrete-time description of the plant (assuming that the input u is applied through a zero-order hold (ZOH)) is given below (refer to Section 6.3).

$$\mathbf{x}(k+1) = \mathbf{Fx}(k) + \mathbf{g}u(k) \tag{7.83}$$

where

$$\mathbf{F} = e^{\mathbf{A}T} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix};\ \mathbf{g} = \int_0^T e^{\mathbf{A}\tau}\,\mathbf{b}d\tau = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}$$

The characteristic equation of the open-loop system is

$$|z\mathbf{I} - \mathbf{F}| = \begin{vmatrix} z-1 & -T \\ 0 & z-1 \end{vmatrix} = (z-1)^2 = 0$$

With the control law

$$u(k) = -\mathbf{kx}(k) = -[k_1 \quad k_2]\,\mathbf{x}(k)$$

the closed-loop system becomes

$$\mathbf{x}(k+1) = (\mathbf{F} - \mathbf{gk})\mathbf{x}(k)$$

The characteristic equation of the closed-loop system is

$$|z\mathbf{I} - (\mathbf{F} - \mathbf{gk})| = z^2 + (Tk_2 + (T^2/2)k_1 - 2)\,z + (T^2/2)k_1 - Tk_2 + 1 = 0 \tag{7.84a}$$

We assume that $T = 0.1$ sec, and the desired characteristic roots of the closed-loop system are $z_{1.2} = 0.875 \angle \pm 17.9^{\circ}$.

Note that these roots correspond to $\zeta = 0.5$, and $\omega_n = 3.6$ (refer to Eqns (4.15)):

$$z_{1,2} = e^{-\zeta\omega_n T} \quad e^{\pm j\omega_n T\sqrt{1-\zeta^2}}$$

The desired characteristic equation is then (approximately)

$$z^2 - 1.6z + 0.70 = 0 \tag{7.84b}$$

Matching coefficients in Eqns (7.84a) and (7.84b), we obtain

$$k_1 = 10,\ k_2 = 3.5$$

7.9.2 Design of State Observers

The control law designed in the last subsection assumed that all states were available for feedback. Since, typically, not all states are measured, the purpose of this subsection is to show how to determine algorithms which will reconstruct all the states, given measurements of a portion of them. If the state is $\mathbf{x}$, then the estimate is $\hat{\mathbf{x}}$ and the idea is to let $u = -\mathbf{k}\hat{\mathbf{x}}$; replacing the true states by their estimates in the control law.

Prediction Observer

An estimation scheme employing a full-order observer is shown in Fig. 7.15, and the equation for it is

$$\hat{\mathbf{x}}(k+1) = \mathbf{F}\hat{\mathbf{x}}(k) + \mathbf{g}u(k) + \mathbf{m}(y(k) - \mathbf{c}\hat{\mathbf{x}}(k)) \tag{7.85}$$

where $\mathbf{m}$ is an $n \times 1$ real constant gain matrix. We will call this a *prediction observer* because the estimate $\hat{\mathbf{x}}(k+1)$ is one sampling period ahead of the measurement $y(k)$.

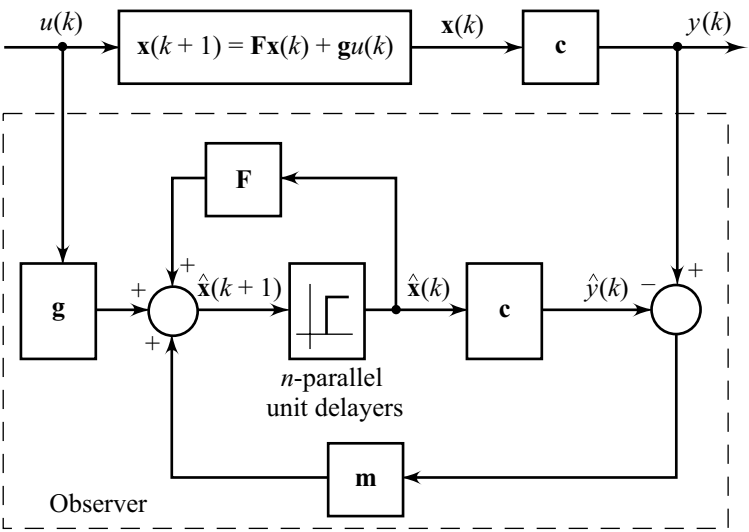


Fig. 7.15 Prediction observer

A difference equation describing the behavior of the error is obtained by subtracting Eqn. (7.85) from Eqn. (7.74):

$$\tilde{\mathbf{x}}(k+1) = (\mathbf{F} - \mathbf{mc})\ \tilde{\mathbf{x}}(k) \tag{7.86}$$

where

$$\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$$

The characteristic equation of the error is given by

$$|z\mathbf{I} - (\mathbf{F} - \mathbf{mc})| = 0 \tag{7.87a}$$

Assuming that the desired characteristic equation is

$$(z - \lambda_1)(z - \lambda_2) \cdots (z - \lambda_n) = 0, \tag{7.87b}$$

the required elements of **m** are obtained by matching coefficients in Eqns (7.87a) and (7.87b). A necessary and sufficient condition for the arbitrary assignment of eigenvalues of (**F** – **mc**), is that the system (7.74) is completely observable.

The problem of designing a full-order observer is mathematically equivalent to designing a full state-feedback controller for the 'transposed auxiliary system'

$$\boldsymbol{\zeta}(k + 1) = \mathbf{F}^T\boldsymbol{\zeta}(k) + \mathbf{c}^T\eta(k) \tag{7.88a}$$

with feedback

$$\eta(k) = -\mathbf{m}^T\boldsymbol{\zeta}(k) \tag{7.88b}$$

so that the closed-loop auxiliary system

$$\boldsymbol{\zeta}(k + 1) = (\mathbf{F}^T - \mathbf{c}^T\mathbf{m}^T)\boldsymbol{\zeta}(k) \tag{7.88c}$$

has eigenvalues λ_i; $i = 1, 2, ..., n$.

This duality principle may be used to design full-order state observers by Ackermann's formula (7.82), or by design procedure given in Eqns (7.78)–(7.81).

Current Observer

The prediction observer given by Eqn. (7.85) arrives at the state estimate $\hat{\mathbf{x}}(k)$ after receiving measurements up through $y(k - 1)$. Hence the control $u(k) = -\mathbf{k}\,\hat{\mathbf{x}}(k)$ does not utilize the information on the current output $y(k)$. For higher-order systems controlled with a slow computer, or any time the sample rates are fast compared to the computation time, this delay between making a measurement and using it in control law may be a blessing. In many systems, however, the computation time required to evaluate Eqn. (7.85) is quite short—compared to the sample period—and the control based on prediction observer may not be as accurate as it could be.

An alternative formulation of the state observer is to use $y(k)$ to obtain the state estimate $\hat{\mathbf{x}}(k)$. This can be done by separating the estimation process into two steps. In the first step we determine $\bar{\mathbf{x}}(k + 1)$, an approximation of $\mathbf{x}(k + 1)$ based on $\hat{\mathbf{x}}(k)$ and $u(k)$, using the model of the plant. In the second step, we use $y(k + 1)$ to improve $\bar{\mathbf{x}}(k + 1)$. The improved $\bar{\mathbf{x}}(k + 1)$ is $\hat{\mathbf{x}}(k + 1)$. The state observer based on this formulation is called the *current observer*. The current observer equations are given by

$$\bar{\mathbf{x}}(k + 1) = \mathbf{F}\hat{\mathbf{x}}(k) + \mathbf{g}u(k) \tag{7.89a}$$

$$\hat{\mathbf{x}}(k + 1) = \bar{\mathbf{x}}(k + 1) + \mathbf{m}[y(k + 1) - \mathbf{c}\bar{\mathbf{x}}(k + 1)] \tag{7.89b}$$

In practice, the current observer cannot be implemented exactly because it is impossible to sample, perform calculations, and output with absolutely no time elapse. However, the errors introduced due to computational delays will be negligible if the computation time is quite short—compared to the sample period.

The error equation for the current observer is similar to the error equation for the prediction observer, given in (7.86). The current-estimate error equation is obtained by subtracting Eqns (7.89) from (7.74).

$$\begin{aligned}\bar{\mathbf{x}}(k+1) &= \mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1)\\ &= \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) - \mathbf{F}\hat{\mathbf{x}}(k) - \mathbf{g}u(k) - \mathbf{mc}[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)]\\ &= \mathbf{F}\tilde{\mathbf{x}}(k) - \mathbf{mcF}\tilde{\mathbf{x}}(k) = (\mathbf{F} - \mathbf{mcF})\tilde{\mathbf{x}}(k)\end{aligned} \tag{7.90}$$

Therefore, the gain matrix **m** is obtained exactly as before, except that **c** is replaced by **cF**.

Reduced-Order Observer

The observers discussed so far, are designed to reconstruct the entire state vector, given measurements of some of the states. To pursue an observer for only the unmeasured states, we partition the state vector into two parts: one part is x_1 which is directly measured, and the other part is $\mathbf{x}_e$, representing the state variables that need to be estimated. If we partition the system matrices accordingly, the complete description of the system (7.74) is given by

$$\begin{bmatrix} x_1(k+1) \\ \mathbf{x}_e(k+1) \end{bmatrix} = \begin{bmatrix} f_{11} & \mathbf{f}_{1e} \\ \mathbf{f}_{e1} & \mathbf{F}_{ee} \end{bmatrix} \begin{bmatrix} x_1(k) \\ \mathbf{x}_e(k) \end{bmatrix} + \begin{bmatrix} g_1 \\ \mathbf{g}_e \end{bmatrix} u(k) \tag{7.91a}$$

$$y(k) = [1 \quad \mathbf{0}] \begin{bmatrix} x_1(k) \\ \mathbf{x}_e(k) \end{bmatrix} \tag{7.91b}$$

The portion describing the dynamics of unmeasured states is

$$\mathbf{x}_e(k+1) = \mathbf{F}_{ee}\,\mathbf{x}_e(k) + \underbrace{\mathbf{f}_{e1}x_1(k) + \mathbf{g}_e u(k)}_{\text{known input}} \tag{7.92}$$

The measured dynamics are given by the scalar equation

$$\underbrace{y(k+1) - f_{11}y(k) - g_1u(k)}_{\text{known measurement}} = \mathbf{f}_{1e}\mathbf{x}_e(k) \tag{7.93}$$

Equations (7.92) and (7.93) have the same relationship to the state $\mathbf{x}_e$ that the original equation (7.74) had to the entire state **x**. Following this reasoning, we arrive at the desired observer by making the following substitutions into the observer equations:

$$\begin{aligned}\mathbf{x} &\leftarrow \mathbf{x}_e\\ \mathbf{F} &\leftarrow \mathbf{F}_{ee}\\ \mathbf{g}u(k) &\leftarrow \mathbf{f}_{e1}\,y(k) + \mathbf{g}_e u(k)\\ y(k) &\leftarrow y(k+1) - f_{11}y(k) - g_1u(k)\\ \mathbf{c} &\leftarrow \mathbf{f}_{1e}\end{aligned} \tag{7.94}$$

Thus, the reduced-order observer equations are

$$\hat{\mathbf{x}}_e(k+1) = \mathbf{F}_{ee}\,\hat{\mathbf{x}}_e(k) + \underbrace{\mathbf{f}_{e1}y(k) + \mathbf{g}_e u(k)}_{\text{input}} + \mathbf{m}\ \underbrace{(y(k+1) - f_{11}y(k) - g_1u(k) - \mathbf{f}_{1e}\hat{\mathbf{x}}_e(k))}_{\text{measurement}} \tag{7.95}$$

Subtracting Eqn. (7.95) from (7.92) yields the error equation

$$\tilde{\mathbf{x}}_e(k+1) = (\mathbf{F}_{ee} - \mathbf{m}\mathbf{f}_{1e})\,\tilde{\mathbf{x}}_e(k) \tag{7.96}$$

where

$$\tilde{\mathbf{x}}_e = \mathbf{x}_e - \hat{\mathbf{x}}_e$$

The characteristic equation is given by

$$|z\mathbf{I} - (\mathbf{F}_{ee} - \mathbf{m}\,\mathbf{f}_{1e})| = 0 \tag{7.97}$$

We design the dynamics of this observer by selecting $\mathbf{m}$ so that Eqn. (7.97) matches a desired reduced-order characteristic equation. The design may be carried out directly or by using duality principle.

7.9.3 Compensator Design by the Separation Principle

If we implement the state-feedback control law using an estimated state vector, the control system can be completed. A schematic of such a scheme, using a prediction observer[1], is shown in Fig. 7.16. Note that by the separation principle, the control law and the state observer can be designed separately, and yet used together.

The portion within the dotted line in Fig. 7.16 corresponds to dynamic compensation. The state variable model of the compensator is obtained by including the state-feedback control (since it is a part of the compensator) in the observer equations, yielding

$$\begin{aligned} \hat{\mathbf{x}}(k+1) &= (\mathbf{F} - \mathbf{gk} - \mathbf{mc})\,\hat{\mathbf{x}}(k) + \mathbf{m}y(k) \\ u(k) &= -\mathbf{k}\,\hat{\mathbf{x}}(k) \end{aligned} \tag{7.98}$$

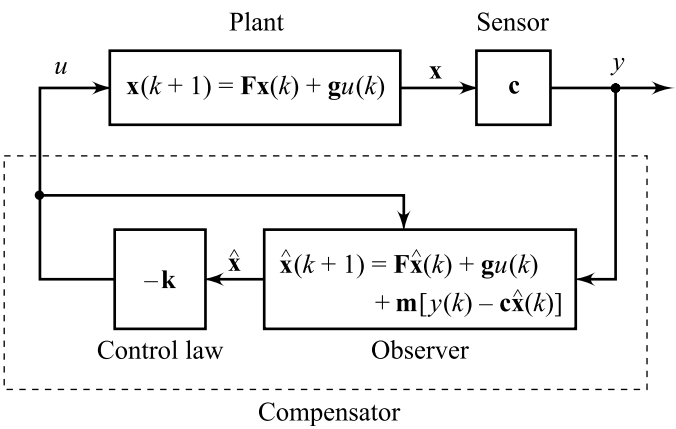


Fig. 7.16 Combined state-feedback control and state estimation

The formula for conversion of a discrete-time state variable model to the transfer function model is given by Eqn. (6.3). Applying this result to the model (7.98), we obtain

$$\frac{U(z)}{-Y(z)} = D(z) = \mathbf{k}(z\mathbf{I} - \mathbf{F} + \mathbf{gk} + \mathbf{mc})^{-1}\mathbf{m} \tag{7.99}$$

[1] We will design the compensator only for the prediction observer case. The other observers give very similar results.

Example 7.11

As an example of complete design, we will add a state observer to the satellite-attitude control considered in Example 7.10. The system equations of motion are (refer to Eqn. (7.83))

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u(k)$$

We assume that the position state x_1 is measured and the velocity state x_2 is to be estimated; the measurement equation is, therefore,

$$y(k) = \mathbf{c}\mathbf{x}(k) = [1 \quad 0]\, \mathbf{x}(k)$$

We will design a first-order observer for the state $x_2(k)$.

The partitioned matrices are

$$\left[\begin{array}{c:c} f_{11} & f_{1e} \\ \hdashline f_{e1} & F_{ee} \end{array}\right] = \left[\begin{array}{c:c} 1 & T \\ \hdashline 0 & 1 \end{array}\right]; \left[\begin{array}{c} g_1 \\ \hdashline g_e \end{array}\right] = \left[\begin{array}{c} T^2/2 \\ \hdashline T \end{array}\right]$$

From Eqn. (7.97), we find the characteristic equation in terms of m:

$$z - (1 - mT) = 0$$

For the observer to be about four times faster than the control, we place the observer pole at

$$z = 0.5\ (\cong (0.835)^4); \text{ therefore, } 1 - mT = 0.5$$

For: $T = 0.1$ sec, $m = 5$. The observer equation is (refer to Eqn. (7.95))

$$\hat{x}_2(k+1) = \hat{x}_2(k) + Tu(k) + m\left(y(k+1) - y(k) - \frac{T^2}{2}\, u(k) - T\,\hat{x}_2(k)\right)$$

$$= 0.5\,\hat{x}_2(k) + 5(y(k+1) - y(k)) + 0.075u(k)$$

Substituting for $u(k)$ from the control law (refer to Example 7.10)

$$u(k) = -10y(k) - 3.5\,\hat{x}_2(k), \tag{7.100a}$$

we obtain

$$\hat{x}_2(k+1) = 0.2375\,\hat{x}_2(k) + 5y(k+1) - 5.75y(k) \tag{7.100b}$$

The two difference equations (7.100a) and (7.100b) complete the design and can be used to control the plant to the desired specifications.

To relate the observer-based state-feedback design to a classical design, one needs to compute the z-transform of Eqns (7.100a) and (7.100b), obtaining

$$\frac{U(z)}{-Y(z)} = \frac{27.5(z - 0.818)}{z - 0.2375}$$

The compensation looks very much like the classical lead compensation that would be used for $1/s^2$ plant.

7.9.4 Servo Design

Let us assume that for the system given by Eqns (7.74), the desired steady-state value for the controlled variable $y(k)$ is a constant reference input r. For this servo system, the desired equilibrium state $\mathbf{x}_s$ is a

constant point in state space, and is governed by the equation

$$\mathbf{c}\mathbf{x}_s = r \tag{7.101a}$$

We formulate this command-following problem as a 'shifted regulator problem' by shifting the origin of the state space to the equilibrium point $\mathbf{x}_s$. Let u_s be the needed input to maintain $\mathbf{x}(k)$ at the equilibrium point $\mathbf{x}_s$, i.e. (refer to Eqns (7.74)),

$$\mathbf{x}_s = \mathbf{F}\mathbf{x}_s + \mathbf{g}u_s \tag{7.101b}$$

Assuming for the present, that a u_s exists that satisfies Eqns (7.101a)–(7.101b), we define shifted input, shifted state, and shifted controlled variable as

$$\begin{aligned} \tilde{u}(k) &= u(k) - u_s \\ \tilde{\mathbf{x}}(k) &= \mathbf{x}(k) - \mathbf{x}_s \\ \tilde{y}(k) &= y(k) - r \end{aligned} \tag{7.102}$$

The shifted variables satisfy the equations

$$\begin{aligned} \tilde{\mathbf{x}}(k+1) &= \mathbf{F}\,\tilde{\mathbf{x}}(k) + \mathbf{g}\tilde{\mathbf{u}}(k) \\ \tilde{y}(k) &= \mathbf{c}\,\tilde{\mathbf{x}}(k) \end{aligned} \tag{7.103}$$

This system possesses a time-invariant asymptotically stable control law (assuming $\{\mathbf{F}, \mathbf{g}\}$ is controllable)

$$\tilde{\mathbf{u}} = -\mathbf{k}\tilde{\mathbf{x}}$$

The application of this control law ensures that

$$\tilde{\mathbf{x}}(k) \to \mathbf{0} \;\; (\mathbf{x}(k) \to \mathbf{x}_s;\, y(k) \to r)$$

In terms of the original state variables, total control effort

$$u(k) = -\mathbf{k}\mathbf{x}(k) + u_s + \mathbf{k}\mathbf{x}_s \tag{7.104}$$

Manipulation of Eqn. (7.101b) gives

$$(\mathbf{F} - \mathbf{g}\mathbf{k} - \mathbf{I})\mathbf{x}_s + \mathbf{g}(u_s + \mathbf{k}\mathbf{x}_s) = \mathbf{0}$$

or

$$\mathbf{x}_s = -(\mathbf{F} - \mathbf{g}\mathbf{k} - \mathbf{I})^{-1}\mathbf{g}(u_s + \mathbf{k}\mathbf{x}_s)$$

or

$$\mathbf{c}\mathbf{x}_s = r = -\mathbf{c}(\mathbf{F} - \mathbf{g}\mathbf{k} - \mathbf{I})^{-1}\mathbf{g}(u_s + \mathbf{k}\mathbf{x}_s)$$

This equation has a unique solution for $(u_s + \mathbf{k}\mathbf{x}_s)$:

$$(u_s + \mathbf{k}\mathbf{x}_s) = Nr$$

where N is a scalar feedforward gain, given by

$$(N)^{-1} = -\mathbf{c}(\mathbf{F} - \mathbf{g}\mathbf{k} - \mathbf{I})^{-1}\mathbf{g} \tag{7.105}$$

The control law (7.104), therefore, takes the form

$$u(k) = -\mathbf{k}\mathbf{x}(k) + Nr \tag{7.106}$$

7.9.5 State Feedback with Integral Control

In the following, we study a control scheme for the system (7.74) where we feedback the state $\mathbf{x}$ as well as the integral of the error in the output.

One way to introduce an integrator is to augment the plant state vector $\mathbf{x}$ with the 'integral state' v that integrates the difference between the output $y(k)$ and the constant reference input r. The 'integral state' v is defined by

$$v(k) = v(k-1) + y(k) - r \tag{7.107a}$$

This equation can be rewritten as follows:

$$\begin{aligned} v(k+1) &= v(k) + y(k+1) - r = v(k) + \mathbf{c}[\mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k)] - r \\ &= \mathbf{cF}\mathbf{x}(k) + v(k) + \mathbf{cg}u(k) - r \end{aligned} \tag{7.107b}$$

From Eqns (7.74) and (7.107b), we obtain

$$\begin{bmatrix} \mathbf{x}(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{cF} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} \mathbf{g} \\ \mathbf{cg} \end{bmatrix} u(k) + \begin{bmatrix} \mathbf{0} \\ -1 \end{bmatrix} r \tag{7.108}$$

Since r is constant, in the steady state $\mathbf{x}(k+1) = \mathbf{x}(k)$ and $v(k+1) = v(k)$ provided that the system is stable. This means that the steady-state solutions $\mathbf{x}_s$, v_s and u_s must satisfy the equation

$$\begin{bmatrix} \mathbf{0} \\ -1 \end{bmatrix} r = \begin{bmatrix} \mathbf{x}_s \\ v_s \end{bmatrix} - \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{cF} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_s \\ v_s \end{bmatrix} - \begin{bmatrix} \mathbf{g} \\ \mathbf{cg} \end{bmatrix} u_s$$

Substituting this for the last term in Eqn. (7.108) gives

$$\tilde{\mathbf{x}}(k+1) = \bar{\mathbf{F}}\,\tilde{\mathbf{x}}(k) + \bar{\mathbf{g}}\,\tilde{u}(k) \tag{7.109}$$

where

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} - \mathbf{x}_s \\ v - v_s \end{bmatrix}; \quad \tilde{u} = u - u_s$$

$$\bar{\mathbf{F}} = \begin{bmatrix} \mathbf{F} & 0 \\ \mathbf{cF} & 1 \end{bmatrix}; \; \bar{\mathbf{g}} = \begin{bmatrix} \mathbf{g} \\ \mathbf{cg} \end{bmatrix}$$

The significance of this result is that by defining the deviations from steady state as state and control variables, the design problem has been reformulated to be the standard regulator problem, with $\tilde{\mathbf{x}} = \mathbf{0}$ as the desired state. We assume that an asymptotically stable solution to this problem exists and is given by

$$\tilde{u}(k) = -\mathbf{k}\,\tilde{\mathbf{x}}(k)$$

Partitioning $\mathbf{k}$ appropriately and using Eqn. (7.109) yields

$$\mathbf{k} = [\mathbf{k}_p \quad k_i]$$

$$u - u_s = -[\mathbf{k}_p \quad k_i] \begin{bmatrix} \mathbf{x} - \mathbf{x}_s \\ v - v_s \end{bmatrix} = -\mathbf{k}_p(\mathbf{x} - \mathbf{x}_s) - k_i(v - v_s)$$

The steady-state terms must balance, therefore,

$$u(k) = -\mathbf{k}_p\mathbf{x}(k) - k_i v(k) \tag{7.110}$$

At steady state, $\tilde{\mathbf{x}}(k+1) - \tilde{\mathbf{x}}(k) = \mathbf{0}$; therefore,

$$v(k+1) - v(k) = 0 = y(k) - r, \text{ i.e., } y(k) \to r$$

The block diagram of Fig. 7.17 shows the control configuration.

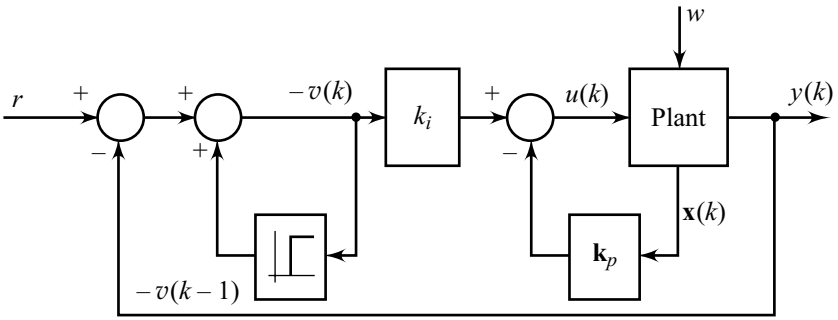


Fig. 7.17 State feedback with integral control

Example 7.12

Consider the problem of digital control of a plant described by the transfer function

$$G(s) = \frac{1}{s+3}$$

Discretization of the plant model gives

$$G_{h0}G(z) = \frac{Y(z)}{U(z)} = \mathscr{Z}\left[\left(\frac{1-e^{-sT}}{s}\right)\left(\frac{1}{s+3}\right)\right]$$

$$= (1-z^{-1})\,\mathscr{Z}\left[\frac{1}{s(s+3)}\right] = \frac{1}{3}\left(\frac{1-e^{-3T}}{z-e^{-3T}}\right)$$

For a sampling interval $T = 0.1$ sec,

$$G_{h0}G(z) = \frac{0.0864}{z-0.741}$$

The difference equation model of the plant is

$$y(k+1) = 0.741y(k) + 0.0864u(k)$$

The plant has a constant reference command signal. We wish to design a PI control algorithm that results in system response characteristics: $\zeta = 0.5$, $\omega_n = 5$. This is equivalent to asking for the closed-loop poles at

$$z_{1,2} = e^{-\zeta\omega_n T}e^{\pm j\omega_n T\sqrt{1-\zeta^2}} = 0.7788 \angle \pm 24.82^{\circ} = 0.7068 \pm j0.3269$$

The desired characteristic equation is, therefore,

$$(z - 0.7068 - j0.3269)(z - 0.7068 + j0.3269) = z^2 - 1.4136z + 0.6065 = z^2 + a_1z + a_2 = 0$$

Augmenting the plant state $y(k)$ with the 'integral state' $v(k)$ defined by

$$v(k) = v(k-1) + y(k) - r,$$

we obtain

$$\begin{bmatrix} y(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} 0.741 & 0 \\ 0.741 & 1 \end{bmatrix}\begin{bmatrix} y(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} 0.0864 \\ 0.0864 \end{bmatrix}u(k) + \begin{bmatrix} 0 \\ -1 \end{bmatrix}r$$

In terms of state variables representing deviations from the steady state:

$$\tilde{\mathbf{x}} = \begin{bmatrix} y - y_s \\ v - v_s \end{bmatrix},$$

the state equation becomes

$$\tilde{\mathbf{x}}(k+1) = \bar{\mathbf{F}}\,\tilde{\mathbf{x}}(k) + \bar{\mathbf{g}}\,\tilde{u}\,(k)$$

where

$$\bar{\mathbf{F}} = \begin{bmatrix} 0.741 & 0 \\ 0.741 & 1 \end{bmatrix};\ \bar{\mathbf{g}} = \begin{bmatrix} 0.0864 \\ 0.0864 \end{bmatrix}$$

By Ackermann's formula (7.82),

$$\mathbf{k} = [0 \quad 1]\mathbf{U}^{-1}\boldsymbol{\phi}(\bar{\mathbf{F}})$$

where

$$\boldsymbol{\phi}(\bar{\mathbf{F}}) = \bar{\mathbf{F}}^{\mathbf{2}} + a_1\bar{\mathbf{F}} + a_2\mathbf{I} = \begin{bmatrix} 0.108 & 0 \\ 0.2425 & 0.1929 \end{bmatrix}$$

$$\mathbf{U}^{-1} = [\bar{\mathbf{g}} \quad \bar{\mathbf{F}}\,\bar{\mathbf{g}}]^{-1} = \begin{bmatrix} 0.0864 & 0.064 \\ 0.0864 & 0.15 \end{bmatrix}^{-1} = \frac{1}{7.43\times10^{-3}} \begin{bmatrix} 0.15 & -0.064 \\ -0.0864 & 0.0864 \end{bmatrix}$$

This gives

$$\mathbf{k} = [1.564 \quad 2.243]$$

The control algorithm is given by

$$u(k) = -1.564y(k) - 2.243v(k)$$

7.10 DEADBEAT CONTROL BY STATE FEEDBACK AND DEADBEAT OBSERVERS

A completely controllable and observable SISO system of order n is considered.

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \\ y(k) &= \mathbf{c}\mathbf{x}(k) \end{aligned} \tag{7.111}$$

With the state-feedback control law

$$u(k) = -\mathbf{k}\mathbf{x}(k) \tag{7.112a}$$

the closed-loop system becomes

$$\mathbf{x}(k+1) = (\mathbf{F} - \mathbf{g}\mathbf{k})\mathbf{x}(k) \tag{7.112b}$$

with the characteristic equation

$$|z\mathbf{I} - (\mathbf{F} - \mathbf{g}\mathbf{k})| = 0 \tag{7.112c}$$

The control-law design consists of picking the gains $\mathbf{k}$ so that Eqn. (7.112c) matches the desired characteristic equation

$$z^n + a_1 z^{n-1} + \cdots + a_{n-1}z + a_n = 0$$

A case of special interest occurs when $a_1 = a_2 = \cdots = a_{n-1} = a_n = 0$, that is, desired characteristic equation is

$$z^n = 0 \tag{7.113}$$

By the Cayley–Hamilton theorem (a matrix satisfies its own characteristic equation),

$$(\mathbf{F} - \mathbf{gk})^n = \mathbf{0}$$

This result implies that the force-free response of closed-loop system (7.112b),

$$\mathbf{x}(k) = (\mathbf{F} - \mathbf{gk})^k \mathbf{x}(0) = \mathbf{0} \text{ for } k \geq n$$

In other words, any initial state $\mathbf{x}(0)$ is driven to the equilibrium state $\mathbf{x} = \mathbf{0}$ in (at most) n steps. The feedback control law that assigns all the closed-loop poles to origin is, therefore, a *deadbeat control law*. A state observer defined by the equation

$$\hat{\mathbf{x}}(k+1) = \mathbf{F}\,\hat{\mathbf{x}}(k) + \mathbf{g}u(k) + \mathbf{m}[y(k) - \mathbf{c}\,\hat{\mathbf{x}}(k)] \tag{7.114}$$

gives an estimate $\hat{\mathbf{x}}(k)$ of the state $\mathbf{x}(k)$; the observer design procedure consists of picking the gains $\mathbf{m}$ so that the error system

$$\tilde{\mathbf{x}}(k+1) = (\mathbf{F} - \mathbf{mc})\,\tilde{\mathbf{x}}(k) \tag{7.115}$$

has the desired characteristic equation. A case of special interest occurs when all the observer poles (i.e., eigenvalues of $(\mathbf{F} - \mathbf{mc})$) are zero. In analogy with the deadbeat control law, we refer to observers with this property as *deadbeat observers*.

Comments

The concept of deadbeat performance is unique to discrete-time systems. By deadbeat control, any nonzero error vector will be driven to zero in (at most) n sampling periods if the magnitude of the scalar control $u(k)$ is unbounded. The settling time depends on the sampling period T. If T is chosen very small, the settling time will also be very small, which implies that the control signal must have an extremely large magnitude. The designer must choose the sampling period for which an extremely large control magnitude is not required in normal operation of the system. Thus, in deadbeat control, the sampling period is the only design parameter.

Example 7.13

The system considered in this example is the attitude control system for a rigid satellite.

The plant equations are (refer to Example 7.10)

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k)$$

where

$$\mathbf{F} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}$$

$x_1(k)$ = position state θ; $x_2(k)$ = velocity state ω

The reference input $r = \theta_r$, a step function. The desired steady state

$$\mathbf{x}_s = [\theta_r \quad 0]^T$$

which is a non-null state.

As the plant has integrating property, the steady-state value u_s of the input must be zero (otherwise the output cannot stay constant). For this case, the shifted regulator problem may be formulated as follows:

$$\tilde{x}_1 = x_1 - \theta_r; \quad \tilde{x}_2 = x_2$$

Shifted state variables satisfy the equations

$$\tilde{\mathbf{x}}(k+1) = \mathbf{F}\,\tilde{\mathbf{x}}(k) + \mathbf{g}u(k)$$

The state-feedback control

$$u(k) = -\mathbf{k}\,\tilde{\mathbf{x}}(k)$$

results in the dynamics of $\tilde{\mathbf{x}}$ given by

$$\tilde{\mathbf{x}}(k+1) = (\mathbf{F} - \mathbf{gk})\,\tilde{\mathbf{x}}(k)$$

We now determine the gain matrix $\mathbf{k}$ such that the response to an arbitrary initial condition is deadbeat. The desired characteristic equation is

$$z^2 = 0$$

Using Ackermann's formula (7.82), we obtain

$$\mathbf{k} = [0 \quad 1]\mathbf{U}^{-1}\,\boldsymbol{\phi}(\mathbf{F})$$

where

$$\boldsymbol{\phi}(\mathbf{F}) = \mathbf{F}^2 = \begin{bmatrix} 1 & 2T \\ 0 & 1 \end{bmatrix}; \quad \mathbf{U}^{-1} = [\mathbf{g} \quad \mathbf{Fg}]^{-1} = \begin{bmatrix} -\dfrac{1}{T^2} & \dfrac{3}{2T} \\ \dfrac{1}{T^2} & -\dfrac{1}{2T} \end{bmatrix}$$

This gives

$$\mathbf{k} = \begin{bmatrix} \dfrac{1}{T^2} & \dfrac{3}{2T} \end{bmatrix}$$

For $T = 0.1$ sec, $\mathbf{k} = [100 \quad 15]$

The control law expressed in terms of original state variables is given as

$$u(k) = -k_1\,\tilde{x}_1(k) - k_2\,\tilde{x}_2(k) = -100(x_1(k) - \theta_r) - 15x_2(k)$$

Example 7.14

Reconsider the problem of attitude control of a satellite. For implementation of the design of the previous example, we require the states $x_1(k)$ and $x_2(k)$ to be measurable. Assuming that the output $y(k) = x_1(k)$ is the only state variable that can be measured, we design a state observer for the system. It is desired that the error vector exhibits deadbeat response. The measurement equation is

$$y(k) = \mathbf{cx}(k) = [1 \quad 0]\mathbf{x}(k)$$

The prediction observer for the system is given as

$$\hat{\mathbf{x}}(k+1) = \mathbf{F}\,\hat{\mathbf{x}}(k) + \mathbf{g}u(k) + \mathbf{m}(y(k) - \mathbf{c}\,\hat{\mathbf{x}}(k))$$

The gains **m** may be calculated by solving the state regulator design problem for the 'transposed auxiliary system'

$$\boldsymbol{\zeta}(k+1) = \mathbf{F}^T\boldsymbol{\zeta}(k) + \mathbf{c}^T\eta(k)$$
$$\eta(k) = -\mathbf{m}^T\boldsymbol{\zeta}(k)$$

The desired characteristic equation is

$$z^2 = 0$$

Using Ackermann's formula, we obtain

$$\mathbf{m}^T = [0 \quad 1]\mathbf{U}^{-1}\boldsymbol{\phi}(\mathbf{F}^T)$$

where

$$\boldsymbol{\phi}(\mathbf{F}^T) = (\mathbf{F}^T)^2 = \begin{bmatrix} 1 & 0 \\ 2T & 1 \end{bmatrix}; \quad \mathbf{U}^{-1} = [\mathbf{c}^T \quad \mathbf{F}^T\mathbf{c}^T]^{-1} = \begin{bmatrix} 1 & -1/T \\ 0 & 1/T \end{bmatrix}$$

This gives

$$\mathbf{m}^T = [2 \quad 1/T]$$

For $T = 0.1$ sec, $\mathbf{m} = \begin{bmatrix} 2 \\ 10 \end{bmatrix}$

REVIEW EXAMPLES

Review Example 7.1

DC motors are widely used in speed-control drives. In most applications, the armature voltage of the motor is controlled in a closed-loop feedback system. Figure 7.18a shows a plant model of a speed control system.

The state variables of the plant can be chosen as the motor shaft velocity $\omega(t)$, and the armature current $i_a(t)$. If both the state variables are used in feedback, then two voltages proportional, respectively, to these two state variables must be generated. The generation of the voltage proportional to $\omega(t)$ can be achieved by use of a tachogenerator. A voltage proportional to $i_a(t)$ can be generated by inserting a sampling resistor R_s in the armature circuit, as shown in Fig. 7.18a. It may, however, be noted that if R_s is very small, the voltage across R_s may consist largely of noise; and if R_s is large, the voltage is more accurate, but a considerable amount of power is wasted in R_s and the efficiency of the system is reduced.

In modern speed-control drives, thyristor rectifier is used as a power amplifier [5]. The thyristor rectifier is supplied by an external single-phase or three-phase ac power, and it amplifies its input voltage u, to produce an output voltage e_a, which is supplied to the armature of the dc motor. The state-feedback control, requiring the feedback of both the motor-shaft velocity and the armature current can, in fact, be effectively used to provide current-limiting protective feature to prevent damage to the thyristors.

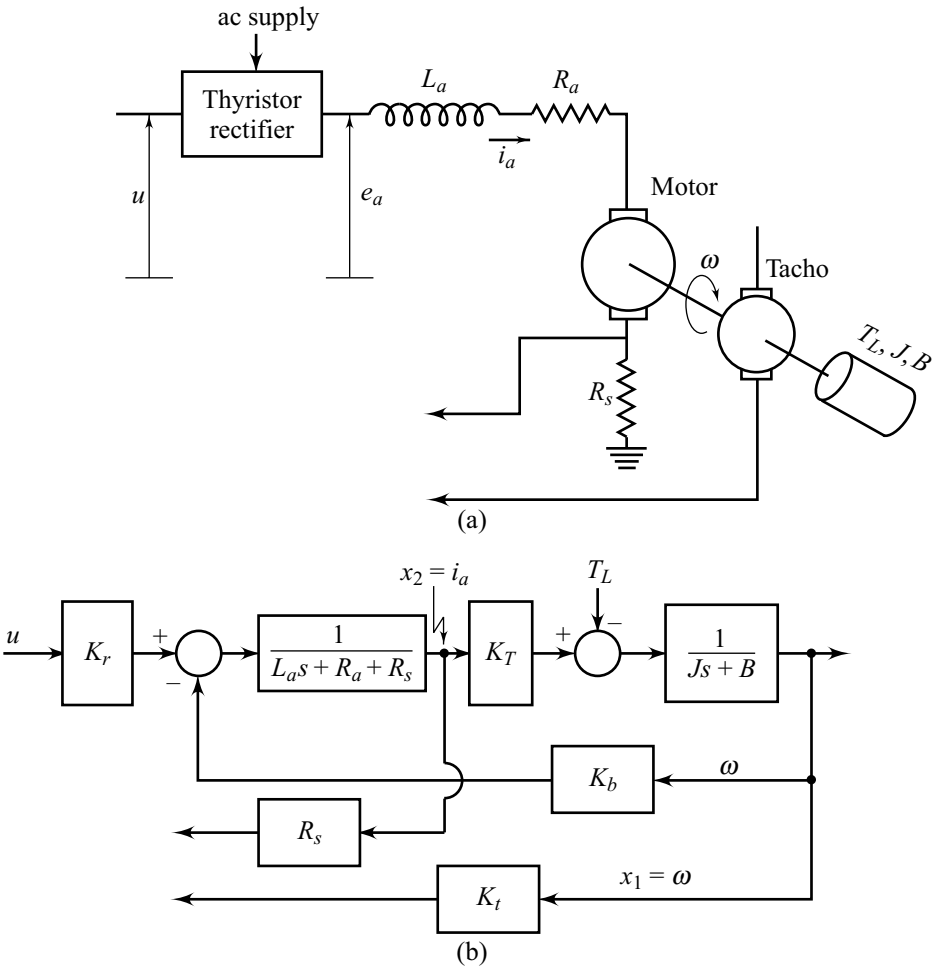


Fig. 7.18 Plant model of a speed-control system

The voltage u is fed to the driver of the thyristor rectifier. The driver produces time-gate pulses that control the conduction of the thyristors in the rectifier module. The rectified output voltage e_a depends on the firing angle of the pulses relative to the ac supply waveform. A linear relationship between the input voltage u and the output voltage e_a can be obtained when a proper firing control scheme is used. The time constants associated with the rectifier are negligibly small. Neglecting the dynamics of the rectifier, we get

$$e_a(t) = K_r\, u(t)$$

where K_r is the gain of the rectifier.

Figure 7.18b shows the functional block diagram of the plant with

B = viscous-friction coefficient of motor and load;
J = moment of inertia of motor and load;

K_T = motor torque constant;
K_b = motor back-emf constant;
T_L = constant load torque;
L_a = armature inductance; and
$R_a + R_s$ = armature resistance.

As seen from Fig. 7.18b, the plant is a type-0 system. A control law of the form

$$u(t) = -\mathbf{kx} + Nr$$

can shape the dynamics of the state variables $x_1(t) = \omega(t)$ and $x_2(t) = i_a(t)$ with zero steady-state error in $\omega(t)$ to constant reference input r. The closed-loop system will, however, be a type-0 system resulting in steady-state errors to constant disturbances. We assume that steady-state performance specifications require a Type-1 system. Hence we employ state feedback with integral control. A block diagram of the control configuration is shown in Fig. 7.19.

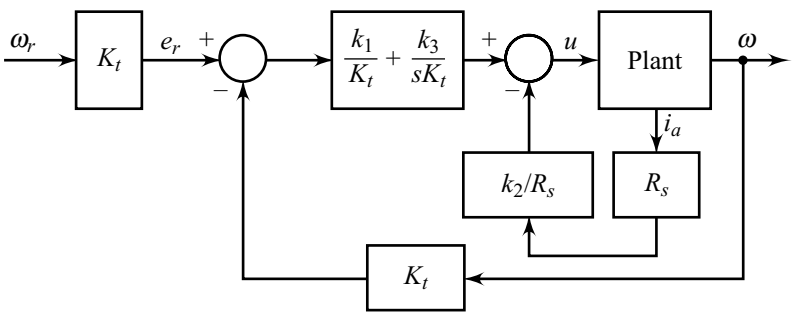


Fig. 7.19 Control configuration for a speed-control system

The state equations of the plant are

$$J\dot{x}_1 + Bx_1 = K_T x_2 - T_L$$
$$L_a \dot{x}_2 + (R_a + R_s)x_2 = K_r u - K_b x_1$$

or

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{b}u + \gamma T_L$$

where

$$\mathbf{A} = \begin{bmatrix} -\frac{B}{J} & \frac{K_T}{J} \\ -\frac{K_b}{L_a} & -\frac{(R_a + R_s)}{L_a} \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ \frac{K_r}{L_a} \end{bmatrix}; \gamma = \begin{bmatrix} -\frac{1}{J} \\ 0 \end{bmatrix}$$

Let the parameter values be such that these matrices become

$$\mathbf{A} = \begin{bmatrix} -0.5 & 10 \\ -0.1 & -10 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 100 \end{bmatrix}; \gamma = \begin{bmatrix} -10 \\ 0 \end{bmatrix}$$

We define an additional state variable x_3 as

$$x_3 = \int_0^t (\omega - r)\, dt,$$

i.e., $$\dot{x}_3 = \omega - r = x_1 - r$$

Augmenting this state variable with the plant equations, we obtain

$$\dot{\bar{\mathbf{x}}} = \bar{\mathbf{A}}\,\bar{\mathbf{x}} + \bar{\mathbf{b}}\,u + \Gamma\mathbf{w}$$

where

$$\bar{\mathbf{x}} = [x_1 \quad x_2 \quad x_3]^T;\ \mathbf{w} = [T_L \quad r]^T$$

$$\bar{\mathbf{A}} = \begin{bmatrix} -0.5 & 10 & 0 \\ -0.1 & -10 & 0 \\ 1 & 0 & 0 \end{bmatrix};\ \bar{\mathbf{b}} = \begin{bmatrix} 0 \\ 100 \\ 0 \end{bmatrix};\ \Gamma = \begin{bmatrix} -10 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}$$

The controllability matrix

$$\mathbf{U} = [\bar{\mathbf{b}} \quad \bar{\mathbf{A}}\,\bar{\mathbf{b}} \quad \bar{\mathbf{A}}^2\bar{\mathbf{b}}]$$

$$= \begin{bmatrix} 0 & 1{,}000 & -10{,}500 \\ 100 & -1{,}000 & 9{,}900 \\ 0 & 0 & 1{,}000 \end{bmatrix}$$

The determinant of $\mathbf{U}$ is nonzero. The pair $(\bar{\mathbf{A}}, \bar{\mathbf{b}})$ is, therefore, completely controllable and the conditions for pole placement by state feedback and integral control, are satisfied.

The characteristic polynomial of the closed-loop system is given by

$$|\,s\mathbf{I} - (\bar{\mathbf{A}} - \bar{\mathbf{b}}\,\bar{\mathbf{k}})\,| = \begin{bmatrix} s+0.5 & -10 & 0 \\ 0.1+100k_1 & s+10+100\,k_2 & 100k_3 \\ -1 & 0 & s \end{bmatrix}$$

$$= s^3 + (10.5 + 100k_2)\,s^2 + (6 + 50k_2 + 1{,}000k_1)s + 1{,}000k_3 \qquad (7.116a)$$

Let the desired characteristic polynomial be

$$s^3 + 87.5\,s^2 + 5{,}374.5\,s + 124{,}969 = (s + 35.4)(s + 26.05 + j53.4)(s + 26.05 - j53.4) \qquad (7.116b)$$

The quadratic term has a natural frequency $\omega_n = 59.39$ rad/sec, and a damping ratio $\zeta = 0.44$.

Matching the corresponding coefficients of Eqns (7.116a) and (7.116b), we obtain

$$k_1 = 5.33,\ k_2 = 0.77,\ k_3 = 124.97$$

With these values of the feedback gains, the state variable model of the closed-loop system becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -0.5 & 10 & 0 \\ -533.1 & -87 & -12497 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} -10 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} T_L \\ r \end{bmatrix}$$

At steady state, $\dot{\bar{\mathbf{x}}} = \mathbf{0}$ and, therefore, the motor velocity $x_1 = \omega(t)$ will approach the constant reference set point r as t approaches infinity, independent of the disturbance torque T_L.

Review Example 7.2

One of the most common uses of feedback control is to position an inertia load using an electric motor. The inertia load may consist of a very large, massive object such as a radar antenna or a small object such as a precision instrument. Armature-controlled dc motors are used in many applications for positioning the load.

We consider here a motor-driven inertia system described by the following equations (refer to Eqns (5.14)).

$$u(t) = R_a\, i_a(t) + K_b\, \omega(t) = R_a\, i_a(t) + K_b \frac{d\theta(t)}{dt}$$

$$K_T\, i_a(t) = J\, \frac{d\omega(t)}{dt} = J \frac{d^2\theta(t)}{dt^2}$$

where

u = applied armature voltage;
R_a = armature resistance;
i_a = armature current;
θ = angular position of the motor shaft;
ω = angular velocity of the motor shaft;
K_b = back emf constant;
K_T = motor torque constant; and
J = moment of inertia referred to the motor shaft.

Taking $x_1 = \theta$, and $x_2 = \dot{\theta} = \omega$ as the state variables, we obtain the following state variable equations for the system.

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{K_T K_b}{JR_a} x_2 + \frac{K_T}{JR_a}\, u = -\alpha\, x_2 + \beta\, u$$

Assume that the physical parameters of the motor and the load yield $\alpha = 1$, $\beta = 1$. Then

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix};\ \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The discrete-time description of this system, with sampling period $T = 0.1$ sec, is given by the following equations (refer to Section 6.3).

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k) \tag{7.117}$$

where

$$\mathbf{F} = e^{\mathbf{A}T} = \begin{bmatrix} 1 & 0.0952 \\ 0 & 0.905 \end{bmatrix};\ \mathbf{g} = \int_0^T e^{\mathbf{A}\tau}\, \mathbf{b}\, d\tau = \begin{bmatrix} 0.00484 \\ 0.0952 \end{bmatrix}$$

In this model, $x_1(k)$ is the shaft position and $x_2(k)$ is the shaft velocity. We assume that $x_1(k)$ and $x_2(k)$ can easily be measured using shaft encoders.

We choose the control configuration of Fig. 7.20 for digital positioning of the load; θ_r is a constant reference command. In terms of the error variables

$$\tilde{x}_1(k) = x_1(k) - \theta_r;\ \tilde{x}_2(k) = x_2(k) \tag{7.118}$$

the control signal

$$u(k) = -k_1 \tilde{x}_1(k) - k_2 \tilde{x}_2(k) = -\mathbf{k}\,\tilde{\mathbf{x}}(k) \tag{7.119}$$

where the gain matrix

$$\mathbf{k} = [k_1 \quad k_2]$$

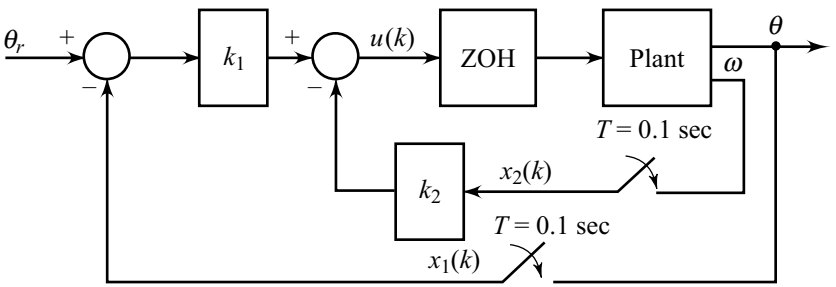


Fig. 7.20 Control configuration for a digital positioning system

The dynamics of the error-vector $\tilde{\mathbf{x}}(k)$ are given by the equations

$$\begin{aligned}\tilde{x}_1(k+1) &= x_1(k+1) - \theta_r = x_1(k) + 0.0952\, x_2(k) + 0.00484\, u(k) - \theta_r \\ &= \tilde{x}_1(k) + 0.0952\ \tilde{x}_2(k) + 0.00484\, u(k) \\ \tilde{x}_2(k+1) &= 0.905\ \tilde{x}_2(k) + 0.0952\, u(k)\end{aligned}$$

or

$$\tilde{\mathbf{x}}(k+1) = \mathbf{F}\,\tilde{\mathbf{x}}(k) + \mathbf{g}u(k)$$

where $\mathbf{F}$ and $\mathbf{g}$ are given by Eqn. (7.117).

Substituting for $u(k)$ from Eqn. (7.119), we obtain the following closed-loop model of the error dynamics:

$$\begin{aligned}\tilde{\mathbf{x}}(k+1) &= (\mathbf{F} - \mathbf{gk})\,\tilde{\mathbf{x}}(k) \\ &= \begin{bmatrix} 1 - 0.00484k_1 & 0.0952 - 0.00484k_2 \\ -0.0952k_1 & 0.905 - 0.0952k_2 \end{bmatrix} \tilde{\mathbf{x}}(k)\end{aligned} \tag{7.120}$$

The characteristic equation is

$$|z\mathbf{I} - (\mathbf{F} - \mathbf{gk})| = z^2 + (0.00484k_1 + 0.0952k_2 - 1.905)z + 0.00468k_1 - 0.0952k_2 + 0.905 = 0 \tag{7.121a}$$

We choose the desired characteristic-equation zero locations, to be

$$z_{1,2} = 0.888 \pm j0.173 = 0.905 \angle \pm 11.04^{\circ}$$

Note that this corresponds to $\zeta = 0.46$ and $\omega_n = 2.17$ (refer to Eqns (4.15)):

$$z_{1,2} = e^{-\zeta\omega_n T} e^{\pm j\omega_n T\sqrt{1-\zeta^2}}$$

The desired characteristic equation is given by

$$(z - 0.888 - j0.173)(z - 0.888 + j0.173) = z^2 - 1.776z + 0.819 = 0 \tag{7.121b}$$

Equating coefficients in Eqns (7.121a) and (7.121b) yields the equations

$$0.00484k_1 + 0.0952k_2 = -1.776 + 1.905$$

$$0.00468k_1 - 0.0952k_2 = 0.819 - 0.905$$

These equations are linear in k_1 and k_2 and upon solving, yield

$$k_1 = 4.52,\ k_2 = 1.12$$

The control law is, therefore, given by

$$u(k) = -k_1 \tilde{x}_1(k) - k_2 \tilde{x}_2(k) = -4.52(x_1(k) - \theta_r) - 1.12x_2(k)$$

The implementation of this control law, requires the feedback of the states $x_1(k)$ and $x_2(k)$. If we measure $x_1(k)$ using a shaft encoder and estimate $x_2(k)$ using a state observer, the control configuration will take the form shown in Fig. 7.21.

The state-feedback control has been designed for $\zeta = 0.46$, $\omega_n = 2.17$; $\zeta\omega_n \cong 1$ sec. The reduced-order observer for estimating velocity $x_2(k)$ from measurements of position $x_1(k)$ is a first-order system; we choose the time constant of this system to be 0.5 sec. Hence, the desired pole location[3] in the observer design problem is

$$z = e^{-T/\tau} = e^{-0.1/0.5} = 0.819$$

The observer characteristic equation is then

$$z - 0.819 = 0 \tag{7.122}$$

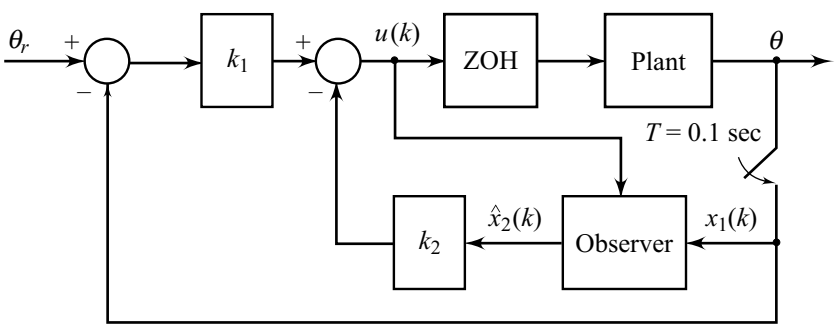


Fig. 7.21 Observer-based digital positioning system

From the plant state equation (7.117), and Eqns (7.91), the partitioned matrices are seen to be

$$f_{11} = 1, f_{1e} = 0.0952, f_{e1} = 0, F_{ee} = 0.905, g_1 = 0.00484, g_e = 0.0952$$

The observer equation is (refer to Eqn. (7.95))

$$\begin{aligned}\hat{x}_2(k+1) &= 0.905\,\hat{x}_2(k) + 0.0952u(k) + m(\theta(k+1) - \theta(k) - 0.00484u(k) - 0.0952\,\hat{x}_2(k)) \\ &= (0.905 - m(0.0952))\,\hat{x}_2(k) + m\theta(k+1) - m\theta(k) + (0.0952 - m(0.00484))u(k)\end{aligned} \tag{7.123}$$

[3] The pole at $s = -1/\tau$ is mapped to $z = e^{-T/\tau}$; T = sampling interval.

The characteristic equation is given by

$$z - (0.905 - 0.0952m) = 0$$

Comparing the coefficients with those of Eqn. (7.122), we obtain

$$m = 0.903$$

Substituting in Eqn. (7.123), we get

$$\hat{x}_2(k+1) = 0.819\,\hat{x}_2(k) + 0.903\theta(k+1) - 0.903\theta(k) + 0.0908u(k)$$

The control system is implemented as follows. A measurement $\theta(k)$ is made at $t = kT$. The observer state is calculated from

$$\hat{x}_2(k) = 0.819\,\hat{x}_2(k-1) + 0.903\theta(k) - 0.903\theta(k-1) + 0.0908u(k-1)$$

Then the control input is calculated, using

$$u(k) = -4.52(\theta(k) - \theta_r) - 1.12\,\hat{x}_2(k)$$

PROBLEMS

7.1 Consider an nth-order Single-Input, Single-Output system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u; \quad y = \mathbf{c}\mathbf{x}$$

and assume that we are using feedback of the form

$$u = -\mathbf{k}\mathbf{x} + r$$

where r is the reference input signal.

Show that the zeros of the system are invariant under state feedback.

7.2 A regulator system has the plant

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u; \quad y = [1 \quad 0 \quad 0]\,\mathbf{x}$$

(a) Design a state-feedback controller which will place the closed-loop poles at $-2 \pm j3.464$, -5. Give a block diagram of the control configuration.

(b) Design a full-order state observer; the observer-error poles are required to be located at $-2 \pm j3.464$, -5. Give all the relevant observer equations and a block diagram description of the observer structure.

(c) The state variable x_1(which is equal to y) is directly measurable and need not be observed. Design a reduced-order state observer for the plant; the observer-error poles are required to be located at $-2 \pm j3.464$. Give all the relevant observer equations.

7.3 A regulator system has the plant

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u; \quad y = \mathbf{c}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -6 \\ 1 & 0 & -11 \\ 0 & 1 & -6 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \mathbf{c} = [0 \quad 0 \quad 1]$$

(a) Compute **k** so that the control law $u = -\mathbf{kx}$, places the closed-loop poles at $-2 \pm j3.464, -5$. Give the state variable model of the closed-loop system.

(b) For the estimation of the state vector **x**, we use an observer defined by

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\,\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y$$

Compute **m** so that the eigenvalues of $(\mathbf{A} - \mathbf{mc})$ are located at $-2 \pm j3.464, -5$.

(c) The state variable x_3 (which is equal to y) is directly measurable and need not be observed. Design a reduced-order observer for the plant; the observer-error poles are required to be located at $-2 \pm j3.464$. Give all the relevant observer equations.

7.4 Consider the system

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}; \quad y = \mathbf{cx} + \mathbf{du}$$

where

$$\mathbf{A} = \begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}; \mathbf{c} = [0 \quad 1]; \mathbf{d} = [2 \quad 0]$$

Design a full-order state observer so that the estimation error will decay in less than 4 seconds.

7.5 Consider the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u; \quad y = [2 \quad -1]\,\mathbf{x}$$

Design a reduced-order state observer that makes the estimation error to decay at least as fast as e^{-10t}.

7.6 Consider the system with the transfer function

$$\frac{Y(s)}{U(s)} = \frac{9}{s^2 - 9}$$

(a) Find (**A**, **b**, **c**) for this system in observable canonical form.

(b) Compute **k** so that the control law $u = -\mathbf{kx}$ places the closed-loop poles at $-3 \pm j3$.

(c) Design a full-order observer such that the observer-error poles are located at $-6 \pm j6$. Give all the relevant observer equations.

(d) Suppose the system has a zero such that

$$\frac{Y(s)}{U(s)} = \frac{9(s+1)}{s^2 - 9}$$

Prove that if $u = -\mathbf{kx} + r$, there is a feedback matrix **k** such that the system is unobservable.

7.7 The equation of motion of an undamped oscillator with frequency ω_0 is

$$\ddot{y} + \omega_0^2 y = u$$

(a) Write the equations of motion in the state variable form with $x_1 = y$ and $x_2 = \dot{y}$ as the state variables.

(b) Find k_1 and k_2 such that $u = -k_1x_1 - k_2x_2$ gives closed-loop characteristic roots with $\omega_n = 2\omega_0$ and $\zeta = 1$.

(c) Design a second-order observer that estimates x_1 and x_2, given measurements of x_1. Pick the characteristic roots of the state-error equation with $\omega_n = 10\omega_0$ and $\zeta = 1$. Give a block diagram of the observer-based state-feedback control system.

(d) Design a first-order observer that estimates x_2, given measurements of x_1. The characteristic root of the state-error equation is required to be located at $-10\omega_0$. Give a block diagram of the observer-based state-feedback control system.

7.8 A regulator system has the plant

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 20.6 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u; \quad y = [1 \quad 0]\, \mathbf{x}$$

(a) Design a control law $u = -\mathbf{kx}$ so that the closed-loop system has eigenvalues at $-1.8 \pm j2.4$.

(b) Design a full-order state observer to estimate the state vector. The observer matrix is required to have eigenvalues at $-8, -8$.

(c) Find the transfer function of the compensator obtained by combining (a) and (b).

(d) Find the state variable model of the complete observer-based state-feedback control system.

7.9 A regulator system has the double integrator plant

$$\frac{Y(s)}{U(s)} = \frac{1}{s^2}$$

(a) Taking $x_1 = y$ and $x_2 = \dot{y}$ as state variables, obtain the state variable model of the plant.

(b) Compute $\mathbf{k}$ such that $u = -\mathbf{kx}$ gives closed-loop characteristic roots with $\omega_n = 1$, $\zeta = \sqrt{2}/2$.

(c) Design a full-order observer that estimates x_1 and x_2, given measurements of x_1. Pick the characteristic roots of the state-error equation with $\omega_n = 5$, $\zeta = 0.5$.

(d) Find the transfer function of the compensator obtained by combining (b) and (c).

(e) Design a reduced-order observer that estimates x_2 given measurements of x_1; place the single observer pole at $s = -5$.

(f) Find the transfer function of the compensator obtained by combining (b) and (e).

7.10 A servo system has the Type-1 plant described by the equation

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{b}u; \quad y = \mathbf{cx}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -2 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \quad 0 \quad 0]$$

(a) If $u = -\mathbf{kx} + Nr$, compute $\mathbf{k}$ and N so that the closed-loop poles are located at $-1 \pm j1, -2$; and $y(\infty) = r$, a constant reference input.

(b) For the estimation of the state vector $\mathbf{x}$, we use a full-order observer

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\, \hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y$$

Compute $\mathbf{m}$ so that observer-error poles are located at $-2 \pm j2, -4$.

(c) Replace the control law in (a) by $u = -\mathbf{k}\,\hat{\mathbf{x}} + Nr$, and give a block diagram of the observer-based servo system.

7.11 A plant is described by the equation

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u; \quad y = [1 \quad 3]\, \mathbf{x}$$

Add to the plant equations an integrator $\dot{z} = y - r$ (r is a constant reference input) and select gains $\mathbf{k}$, k_i so that if $u = -\mathbf{kx} - k_i z$, the closed-loop poles are at $-2, -1 \pm j\sqrt{3}$. Give a block diagram of the control configuration.

7.12 Figure P7.12 shows the block diagram of a position control system employing a dc motor in armature control mode; θ (rad) is the motor shaft position, $\dot{\theta}$(rad/sec) is the motor shaft velocity, i_a(amps) is the armature current, and K_P(volts/rad) is the sensitivity of the potentiometer. Find k_1, k_2 and k_3 so that the dominant poles of the closed-loop system are characterized by $\zeta = 0.5$, $\omega_n = 2$, nondominant pole is at $s = -10$; and the steady-state error to constant reference input is zero.

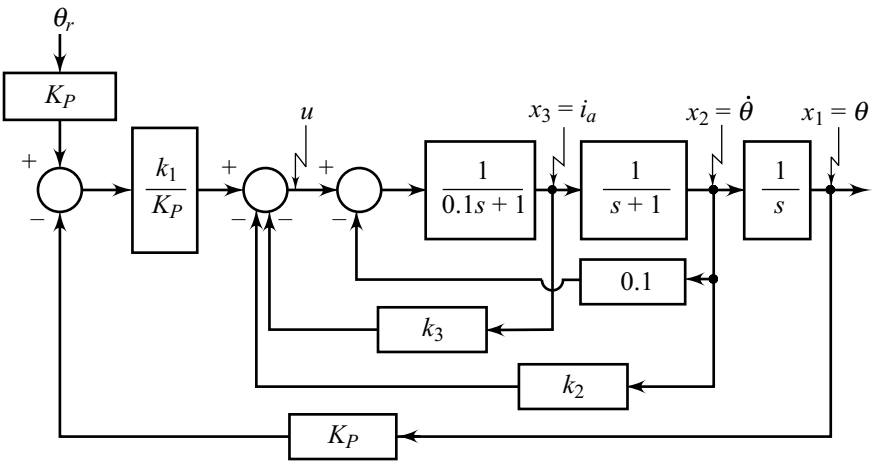


Fig. P7.12

7.13 A dc motor in armature control mode has been used in speed control system of Fig. P7.13 employing state-feedback with integral control; ω (rad/sec) is the motor shaft velocity, i_a(amps) is the armature current and K_t (volts/(rad/sec)) is the tachogenerator constant. Find k_1, k_2 and k_3 so that the closed-loop poles of the system are placed at $-1 \pm j\sqrt{3}$, -10; and the steady-state error to constant reference input is zero.

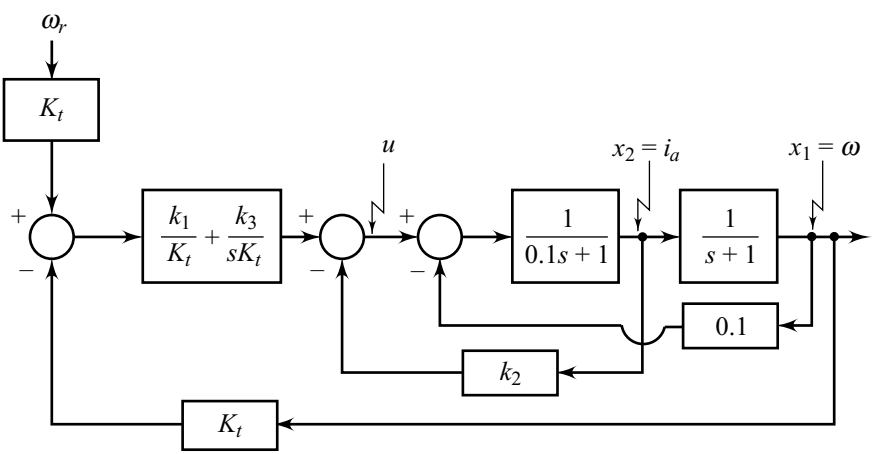


Fig. P7.13

7.14 The control law $u = -\mathbf{kx} - k_1\theta_r$ for position control system of Fig. P7.12 is to be replaced by $u = -\mathbf{k}\hat{\mathbf{x}} + k_1\theta_r$ where $\hat{\mathbf{x}}$ is the estimate of the state vector $\mathbf{x}$ given by the observer system

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}\theta$$

Find the gain matrix $\mathbf{m}$ which places the eigenvalues of $(\mathbf{A} - \mathbf{mc})$ at $-3 \pm j\sqrt{3}, -10$. Give a block diagram of the observer-based position control system.

7.15 Consider the position control system of Fig. P7.15 employing a dc motor in armature control mode with state variables defined on the diagram. Full state-feedback is employed, with position feedback being obtained from a potentiometer, rate feedback from a tachogenerator and current feedback from a voltage sample across a resistance in the armature circuit. K_A is the amplifier gain. Find the adjustable parameters K_A, k_2, and k_3 so that the closed-loop poles of the system are placed at $-3 \pm j3, -20$.

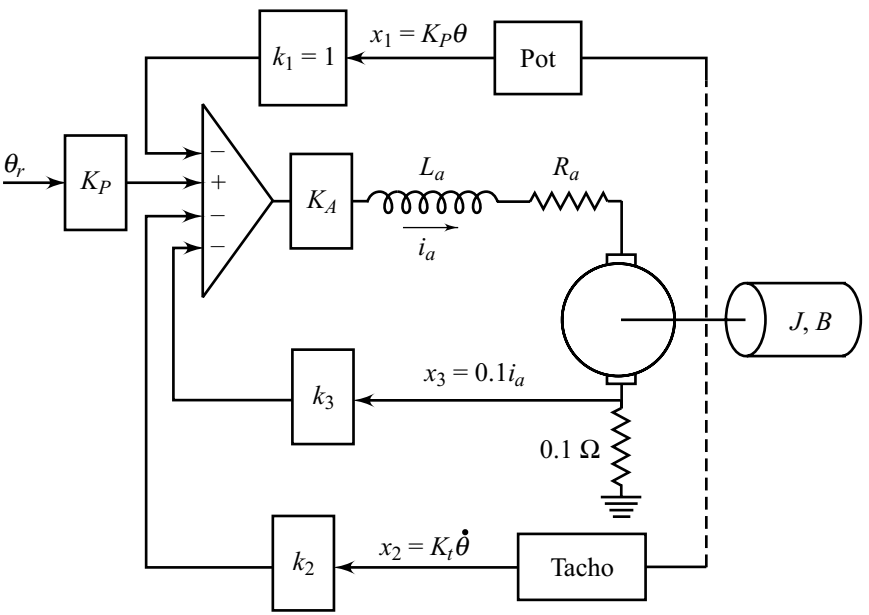


Fig. P7.15

Given:

Potentiometer sensitivity,	$K_P = 1$ volt/rad
Tachogenerator constant,	$K_t = 1$ volt/(rad/sec)
Armature inductance,	$L_a = 0.005$ H
Armature resistance,	$R_a = 0.9\ \Omega$
Moment of inertia of motor and load,	$J = 0.02$ newton-m/(rad/sec^2)
Viscous-friction coefficient of motor and load,	$B = 0$
Back emf constant,	$K_b = 1$ volt/(rad/sec)
Motor torque constant,	$K_T = 1$ newton-m/amp

7.16 Consider the position control system of Fig. P7.16 employing a dc motor in the field control mode, with state variables defined on the diagram. Full state-feedback is employed with position feedback being obtained from a potentiometer, rate feedback from a tachogenerator and current feedback from a voltage sample across a resistor connected in the field circuit. K_A is the amplifier gain.

Find the adjustable parameters K_A, k_2, and k_3 so that the closed-loop system has dominant poles characterized by $\zeta = 0.5$, $\omega_n = 2$, and the third pole at $s = -10$.

Given:

Potentiometer sensitivity,	$K_P = 1$ volt/rad
Tachogenerator constant,	$K_t = 1$ volt/(rad/sec)
Field inductance,	$L_f = 20$ H
Field resistance,	$R_f = 99\ \Omega$
Moment of inertia of motor and load,	$J = 0.5$ newton-m/(rad/sec^2)
Viscous-friction coefficient of motor and load,	$B = 0.5$ newton-m/(rad/sec)
Motor torque constant,	$K_T = 10$ newton-m/amp

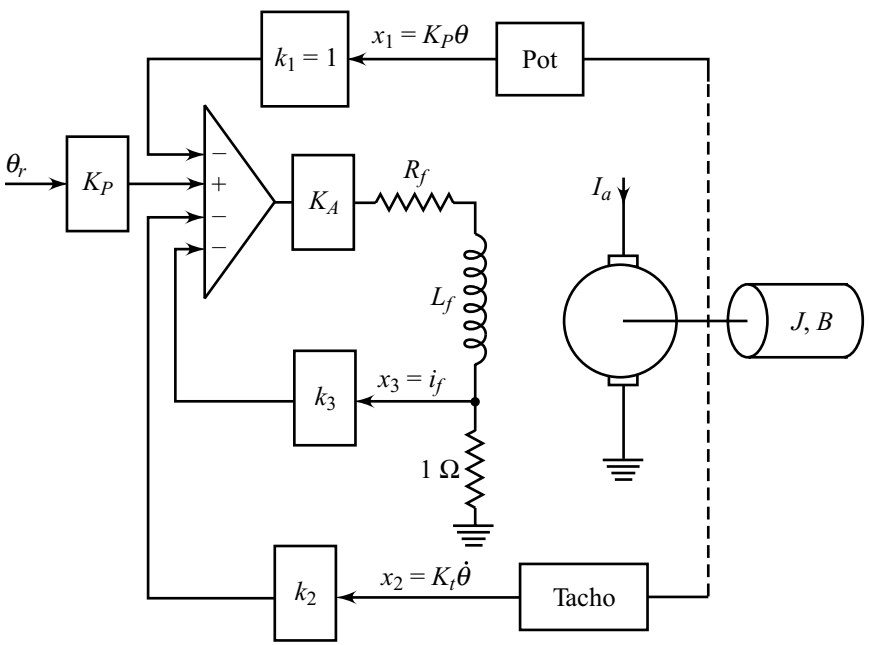


Fig. P7.16

7.17 Figure P7.17 shows control configuration of a Type-1 servo system. Both the state variables x_1 and x_2, are assumed to be measurable. It is desired to regulate the output y to a constant value $r = 5$. Find the values of k_1, k_2 and N so that

(i) $y(\infty) = r = 5$; and

(ii) the closed-loop characteristic equation is

$$s^2 + a_1 s + a_2 = 0.$$

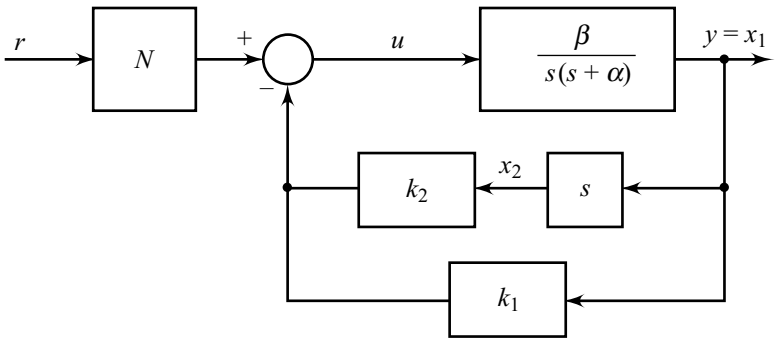


Fig. P7.17

7.18 A speed control system, employing a dc motor in the armature control mode, is described by the following state equations:

$$\frac{d\omega(t)}{dt} = -\frac{B}{J}\omega(t) + \frac{K_T}{J} i_a(t) - \frac{1}{J} T_L$$

$$\frac{di_a(t)}{dt} = -\frac{K_b}{L}\omega(t) - \frac{R}{L} i_a(t) + \frac{1}{L} u(t)$$

where

$i_a(t)$ = armature current, amps;
$u(t)$ = armature applied voltage, volts;
$\omega(t)$ = motor velocity, rad/sec;
B = viscous-friction coefficient of motor and load = 0;
J = moment of inertia of motor and load = 0.02 newton-m/(rad/sec^2);
K_T = motor torque constant = 1 newton-m/amp;
K_b = motor back emf constant = 1 volt/(rad/sec);
T_L = constant disturbance torque (magnitude not known);
L = armature inductance = 0.005 H; and
R = armature resistance = 1 Ω.

The design problem is to find the control $u(t)$ such that

(i) $\lim_{t\to\infty} \frac{di_a(t)}{dt} = 0$ and $\lim_{t\to\infty} \frac{d\omega(t)}{dt} = 0$, and (ii) $\lim_{t\to\infty} \omega(t)$ = constant set-point r.

Show that the control law of the form

$$u(t) = -k_1\omega(t) - k_2 i_a(t) - k_3 \int_0^t (\omega(t) - r)dt$$

can meet these objectives. Find k_1, k_2, and k_3 so that the closed-loop poles are placed at $-10 \pm j10, -300$. Suggest a suitable scheme for implementation of the control law.

7.19 Figure P7.19 shows a process consisting of two interconnected tanks. h_1 and h_2 representing deviations in tank levels from their steady-state values $\bar{H}_1$ and $\bar{H}_2$, respectively; q represents deviation in the flow rate from its steady-state value $\bar{Q}$. The flow rate q is controlled by signal u *via* valve and actuator. A disturbance flow rate w enters the first tank *via* a returns line from elsewhere in the process. The differential equations for levels in the tanks are given by

$$\dot{h}_1 = -3h_1 + 2h_2 + u + w$$
$$\dot{h}_2 = 4h_1 - 5h_2$$

(a) Compute the gains k_1 and k_2 so that the control law $u = -k_1h_1(t) - k_2h_2(t)$ places the closed-loop poles at $-4, -7$.

(b) Show that the steady-state error in the output $y(t) = h_2(t)$, in response to constant disturbance input w, is nonzero.

(c) Add to the plant equations, an integrator $\dot{z}(t) = y(t)$ and select gains k_1, k_2 and k_3 so that the control law $u = -k_1h_1(t) - k_2h_2(t) - k_3z(t)$ places the closed-loop poles at $-1, -2, -7$. Find the steady-state value of the output in response to constant disturbance w. Give a block diagram depicting the control configuration.

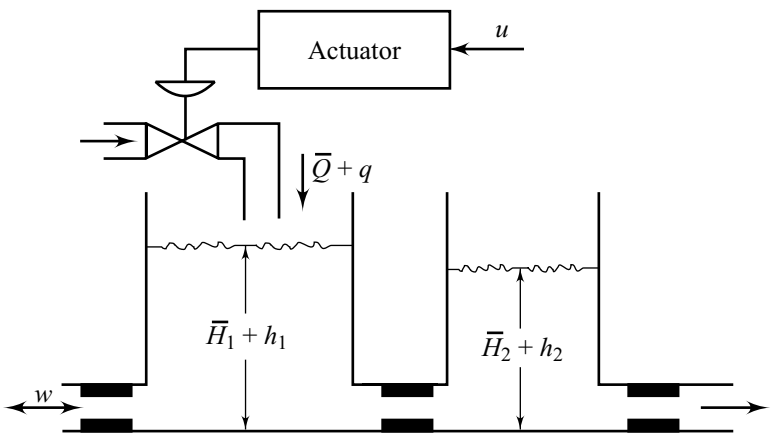


Fig. P7.19

7.20 The plant of a servo system is described by the equations

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u + \mathbf{b}w; \quad y = \mathbf{c}\mathbf{x}$$

where

$$\mathbf{A} = \begin{bmatrix} -3 & 2 \\ 4 & -5 \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mathbf{c} = [0 \quad 1]$$

w is a disturbance input to the system.

A control law of the form $u = -\mathbf{k}\mathbf{x} + Nr$ is proposed; r is a constant reference input.

(a) Compute $\mathbf{k}$ so that the eigenvalues of $(\mathbf{A} - \mathbf{b}\mathbf{k})$ are $-4, -7$.

(b) Choose N so that the system has zero steady-state error to reference input, i.e., $y(\infty) = r$.

(c) Show that the steady-state error to a constant disturbance input w, is nonzero for the above choice of N.

(d) Add to the plant equation, an integrator equation ($z(t)$ being the state of the integrator):

$$\dot{z}(t) = y(t) - r$$

and select gains k_1, k_2 and k_3 so that the control law $u = -k_1x_1(t) - k_2\,x_2(t) - k_3\,z(t)$ places the eigenvalues of closed-loop system matrix at $-1, -2, -7$.

(e) Draw a block diagram of the control scheme employing integral control and show that the steady-state error to constant disturbance input, is zero.

7.21 Consider a plant consisting of a dc motor, the shaft of which has the angular velocity $\omega(t)$ and which is driven by an input voltage $u(t)$. The describing equation is

$$\dot{\omega}(t) = -0.5\,\omega(t) + 100\,u(t) = A\omega(t) + bu(t)$$

It is desired to regulate the angular velocity at the desired value $\omega^0 = r$.

(a) Use control law of the form $u = -K\omega(t) + Nr$. Choose K that results in closed-loop pole with time constant 0.1 sec. Choose N that guarantees zero steady-state error, i.e., $\omega(\infty) = r$.

(b) Show that, if A changes to $A + \delta A$ subject to $(A + \delta A - bK)$ being stable, then the above choice of N will no longer make $\omega(\infty) = r$. Therefore, the system is not robust under changes in system parameters.

(c) The system can be made robust by augmenting it with an integrator

$$\dot{z} = \omega - r$$

where z is the state of the integrator. To see this, first use a feedback of the form $u = -K_1\omega(t) - K_2z(t)$ and select K_1 and K_2 so that the characteristic polynomial of the closed-loop system becomes $\Delta(s) = s^2 + 11s + 50$. Show that the resulting system will have $\omega(\infty) = r$ no matter how the matrix A changes so long as the closed-loop system remains asymptotically stable.

7.22 A discrete-time regulator system has the plant

$$\mathbf{x}(k+1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & -2 & -1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$

Design a state-feedback controller which will place the closed-loop poles at $-\frac{1}{2} \pm j\frac{1}{2}$, 0. Give a block diagram of the control configuration.

7.23 Consider a plant defined by the following state variable model:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k); \quad y(k) = \mathbf{c}\mathbf{x}(k) + \mathbf{d}\mathbf{u}(k)$$

where

$$\mathbf{F} = \begin{bmatrix} \frac{1}{2} & 1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}; \mathbf{G} = \begin{bmatrix} 1 & 4 \\ 0 & 0 \\ -3 & 2 \end{bmatrix}; \quad \mathbf{c} = [1 \quad 0 \quad 0]; \mathbf{d} = [0 \quad 4]$$

Design a prediction observer for the estimation of the state vector $\mathbf{x}$; the observer-error poles are required to lie at $-\frac{1}{2} \pm j\frac{1}{4}$, 0. Give all the relevant observer equations and a block diagram description of the observer structure.

7.24 Consider the system defined by

$$\mathbf{x}(k+1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.5 & -0.2 & 1.1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$

Determine the state-feedback gain matrix **k** such that when the control signal is given by $u(k) = -\mathbf{k}\mathbf{x}(k)$, the closed-loop system will exhibit the deadbeat response to any initial state $\mathbf{x}(0)$. Give the state variable model of the closed-loop system.

7.25 Consider the system

$$\mathbf{x}(k+1) = \begin{bmatrix} 0 & 1 \\ -0.16 & -1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k); \quad y(k) = [1 \quad 1]\, \mathbf{x}(k)$$

Design a current observer for the system; the response to the initial observer error is required to be deadbeat. Give all the relevant observer equations.

7.26 Consider the plant defined in Problem 7.24. Assuming that only $y(k) = x_2(k)$ is measurable, design a reduced-order observer such that the response to the observer error is deadbeat. Give all the relevant observer equations.

7.27 **A** discrete-time regulator system has the plant

$$\mathbf{x}(k+1) \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 4 \\ 3 \end{bmatrix} u(k); \quad y(k) = [1 \quad 1]\, \mathbf{x}(k) + 7u(k)$$

(a) Design a state-feedback control algorithm $u(k) = -\mathbf{k}\mathbf{x}(k)$ which places the closed-loop characteristic roots at $\pm j\frac{1}{2}$.

(b) Design a prediction observer for deadbeat response. Give the relevant observer equations.

(c) Combining (a) and (b), give a block diagram of the control configuration. Also obtain state variable model of the observer-based state-feedback control system.

7.28 A regulator system has the plant with transfer function

$$\frac{Y(z)}{U(z)} = \frac{z^{-2}}{(1+0.8z^{-1})(1+0.2z^{-1})}$$

(a) Find (**F**, **g**, **c**) for the plant in controllable canonical form.

(b) Find k_1 and k_2 such that $u(k) = -k_1x_1(k) - k_2x_2(k)$ gives closed-loop characteristic roots at $0.6 \pm j0.4$.

(c) Design a first-order observer that estimates x_2, given measurements of x_1; the response to initial observer error is required to be deadbeat.

(d) Give a z-domain block diagram of the closed-loop system.

7.29 Consider the system

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k); \quad y(k) = \mathbf{c}\mathbf{x}(k)$$

where

$$\mathbf{F} = \begin{bmatrix} 0.16 & 2.16 \\ -0.16 & -1.16 \end{bmatrix}; \quad \mathbf{g} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}; \quad \mathbf{c} = [1 \quad 1]$$

(a) Design a state-feedback control algorithm which gives closed-loop characteristic roots at $0.6 \pm j\,0.4$.

(b) Design a reduced-order observer for deadbeat response.

(c) Find the transfer function of the compensator obtained by combining (a) and (b). Give a block diagram of the closed-loop system showing the compensator in the control loop.

7.30 A double integrator plant is to be controlled by a digital computer employing state feedback. Figure P7.30 shows a model of the control scheme. Both the state variables x_1 and x_2 are assumed to be measurable.

(a) Obtain the discrete-time state variable model of the plant.

(b) Compute k_1 and k_2 so that the response $y(t)$ of the closed-loop system has the parameters: $\zeta = 0.5$, $\omega_n = 4$.

(c) Assume now that only x_1 is measurable. Design a prediction observer to estimate the state vector **x**; the estimation error is required to decay in a deadbeat manner.

(d) Find the transfer function of the compensator obtained by combining (b) and (c).

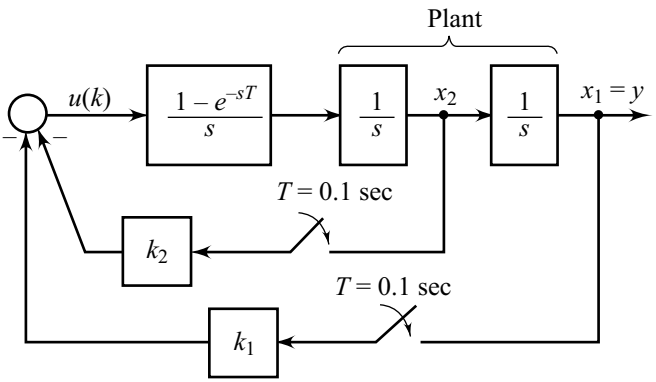


Fig. P 7.30

7.31 Figure P7.31 shows the block diagram of a digital positioning system. The plant is a dc motor driving inertia load. Both the position θ, and velocity $\dot{\theta}$, are measurable.

(a) Obtain matrices (**F**, **g**, **c**) of the discrete-time state variable model of the plant.

(b) Compute k_1 and k_2 so that the closed-loop system positions the load in a deadbeat manner in response to any change in step command θ_r.

(c) Assume now that the position θ is measured by a shaft encoder and a second-order state observer is used to estimate the state vector **x** from plant input u and measurements of θ.

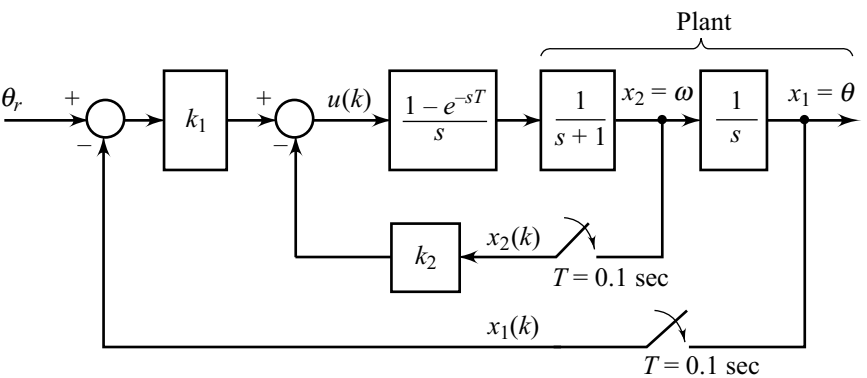


Fig. P7.31

Design a deadbeat observer. Give a block diagram of the observer-based digital positioning system.

(d) Design a first-order deadbeat observer to estimate velocity ω from measurements of position θ.

7.32 A continuous-time plant described by the equation

$$\dot{y} = -y + u + w$$

is to be controlled by a digital computer; y is the output, u is the input, and w is the disturbance signal. Sampling interval $T = 1$ sec.

(a) Obtain a discrete-time state variable model of the plant.

(b) Compute K and N so that the control law

$$u(k) = -Ky(k) + Nr$$

results in a response $y(t)$ with time constant 0.5 sec, and $y(\infty) = r$ (r is a constant reference input).

(c) Show that the steady-state error to a constant disturbance input w is nonzero for the above choice of the control scheme.

(d) Add to the plant equation, an integrator equation ($v(k)$ being the integral state)

$$v(k) = v(k-1) + y(k) - r$$

and select gains K_1 and K_2 so that the control law

$$u = -K_1 y(k) - K_2 v(k)$$

results in a response $y(t)$ with parameters: $\zeta = 0.5$, $\omega_n = 4$.

(e) Give a block diagram depicting the control configuration employing integral control and show that the steady-state error to constant disturbance w, is zero.

Chapter 8

Linear Quadratic Optimal Control through Lyapunov Synthesis

8.1 INTRODUCTION

It should be obvious by now that stability plays a major role in control systems design. We have earlier introduced, in Chapters 2 and 5, the concept of stability based on the dynamic evolution of the system state in response to arbitrary initial state, representing initial energy storage. State variable model

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t);\ \mathbf{x}(t=0) \triangleq \mathbf{x}^0 \tag{8.1}$$

is most appropriate to study dynamic evolution of the state $\mathbf{x}(t)$, in response to the initial state $\mathbf{x}^0$ with zero external input. At the origin of the state space, $\dot{\mathbf{x}}(t) = \mathbf{0}$ for all t; the origin is, thus, the *equilibrium point* of the system and $\mathbf{x}^e = \mathbf{0}$ is the *equilibrium state*. This system is *marginally stable* if, for all possible initial states, $\mathbf{x}^0$, $\mathbf{x}(t)$ remains thereafter within finite bounds for $t > 0$. This is true if none of the eigenvalues of $\mathbf{A}$ are in the right half of the complex plane, and eigenvalues on the imaginary axis, if any, are simple (A multiple eigenvalue on the imaginary axis would have a response that grows in time and could not be stable). Furthermore, the system is *asymptotically stable* if for all possible initial states $\mathbf{x}^0$, $\mathbf{x}(t)$ eventually decays to zero as t approaches infinity. This is true if all the eigenvalues of $\mathbf{A}$ are inside the left half of the complex plane.

A.M. Lyapunov considered the stability of general nonlinear systems described by state equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t));\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{8.2}$$

We assume that the equation has been written so that $\mathbf{x} = \mathbf{0}$ is an *equilibrium point*, which is to say that $\mathbf{f}(\mathbf{0}) = \mathbf{0}$, i.e., the system will continue to be in *equilibrium state* $\mathbf{x}^e = \mathbf{0}$ for all time. This equilibrium point is said to be *stable in the sense of Lyapunov*, if we are able to select a bound on initial condition $\mathbf{x}^0$, that will result in state trajectories $\mathbf{x}(t)$, that remain within a chosen finite limit. The system is *asymptotically stable* at $\mathbf{x} = \mathbf{0}$, if it is stable in the sense of Lyapunov and, in addition, the state $\mathbf{x}(t)$ approaches zero as time t approaches infinity.

No new results are obtained by the use of Lyapunov's method for the stability analysis of linear time-invariant systems. Simple and powerful methods discussed in earlier chapters are adequate for such

systems. However, Lyapunov functions supply certain performance indices and synthesis data for linear time-invariant systems. Chapter 10 will demonstrate the use of Lyapunov functions in variable structure sliding mode control, and model reference adaptive control. In this chapter, we introduce the concept of Lyapunov stability and the role it plays in optimal control design.

The Lyapunov's method of stability analysis, in principle, is the most general method for determination of stability of nonlinear systems. The major drawback which seriously limits its use in practice, is the difficulty often associated with the construction of the Lyapunov function required by the method. Guidelines for construction of Lyapunov functions for nonlinear systems are given in Chapter 9. For linear time-invariant systems of main concern in this chapter, the quadratic function (refer to Section 5.2) is adequate for demonstrating Lyapunov stability. The concept of Lyapunov stability, and working knowledge required for synthesis of linear time-invariant systems, will be provided here, in this chapter, before we start the discussion on optimal control. Detailed account of Lyapunov stability will be given later in Chapter 9.

8.2 THE CONCEPT OF LYAPUNOV STABILITY

8.2.1 Lyapunov Stability Definitions

We shall confine our attention to systems described by state equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)); \mathbf{f}(\mathbf{0}) = \mathbf{0}; \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{8.3}$$

Note that the origin of the state space has been taken as the equilibrium state of the system, i.e.,

$$\mathbf{x}^e = \mathbf{0}$$

The system described by Eqn. (8.3) is *stable in the sense of Lyapunov*, if we are able to select a bound on initial conditions $\mathbf{x}^0$, that will result in state trajectories that remain within a chosen finite limit. More formally, the system described by Eqn. (8.3) is stable in the sense of Lyapunov at $\mathbf{x} = \mathbf{0}$ if, for every real number $\varepsilon > 0$, there exists a real number $\delta > 0$ such that $\|\mathbf{x}(0)\| < \delta$ results in $\|\mathbf{x}(t)\| < \varepsilon$ for all $t \geq 0$; $\|\mathbf{x}\|$ = a *norm* of vector $\mathbf{x}$ (refer to Section 5.2). The system is *asymptotically stable* at $\mathbf{x} = \mathbf{0}$ if it is stable in the sense of Lyapunov and, in addition, the state $\mathbf{x}(t)$ approaches zero as time t approaches infinity. Responses that are stable in the sense of Lyapunov, and asymptotically stable, are shown in Fig. 8.1.

To prove stability results for systems described by equations of the form (8.3), Lyapunov introduced a function that has many of the properties of energy. The basic Lyapunov stability result is given below.

8.2.2 Lyapunov Stability Theorems

For the system (8.3), sufficient conditions of stability are as follows.

Theorem 8.1 Suppose that there exists a scalar function $V(\mathbf{x})$ which satisfies the following properties:

(i) $V(\mathbf{x}) > 0; \mathbf{x} \neq \mathbf{0}$
(ii) $V(\mathbf{0}) = 0$ (8.4)

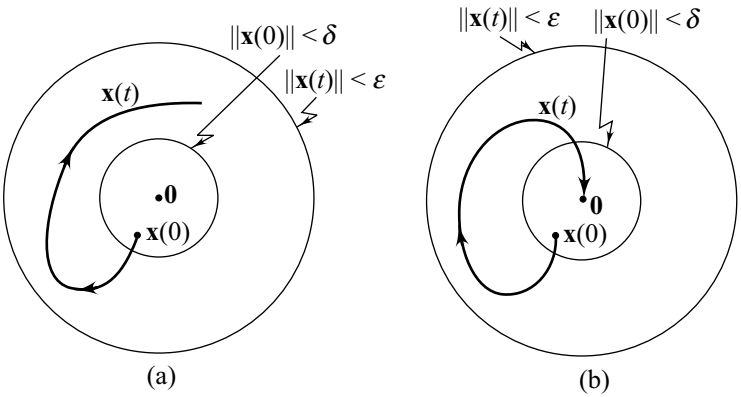


Fig. 8.1 Stability definitions

(iii) $V(\mathbf{x})$ is continuous and has continuous partial derivatives with respect to all components of $\mathbf{x}$.

(iv) $\dot{V}(\mathbf{x}) \leq 0$ along trajectories of Eqn. (8.3).

We call $V(\mathbf{x})$ having these properties, a *Lyapunov function* for the system. Properties (i) and (ii) mean that, like energy, $V(\mathbf{x}) > 0$ if any state is different from zero, but $V(\mathbf{x}) = 0$ when the state is zero. Property (iii) ensures that $V(\mathbf{x})$ is a smooth function and, generally, has the shape of a bowl near the equilibrium.

A visual analysis may be obtained by considering the surface

$$V(x_1, x_2) = \tfrac{1}{2} p_1 x_1^2 + \tfrac{1}{2} p_2 x_2^2; p_1 > 0, p_2 > 0$$

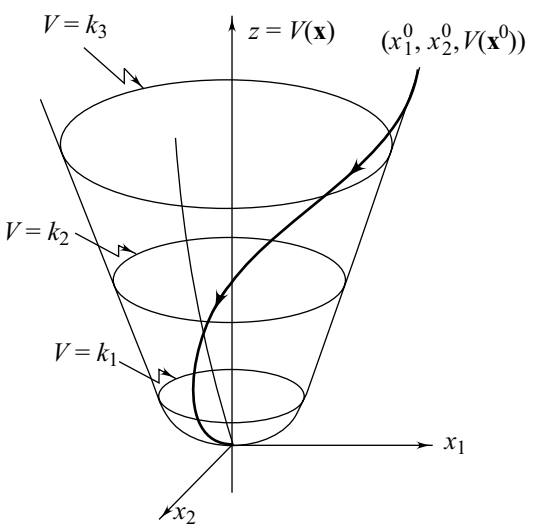


Fig. 8.2 Constant-V curves

This is a paraboloid (a solid generated by rotation of parabola about its axis of symmetry) surface as shown in Fig. 8.2. The value $V(x_1, x_2) = k_i$ (a constant) is represented by the intersection of $V(x_1, x_2)$ surface and the plane $z = k_i$. This intersection results in a closed curve, an oval. If one plots a trajectory from the point $(x_1^0, x_2^0, V(\mathbf{x}^0))$, the trajectory crosses the ovals $V(x_1, x_2) = k_i$ for successively smaller values of $V(x_1, x_2)$, and moves towards the point corresponding to $V(x_1, x_2) = 0$, which is the equilibrium point. Figure 8.2 shows a typical trajectory.

Property (iv) guarantees that any trajectory moves in a way, so as never to climb higher on the bowl, than where it started out. If property (iv) was made stronger so that $\dot{V}(\mathbf{x}) < 0$ for $\mathbf{x} \neq \mathbf{0}$, then the trajectory must be drawn to the origin (The trajectory in Fig. 8.2, in fact, corresponds to this case, i.e., $\dot{V}(\mathbf{x}) < 0$).

The Lyapunov stability theorem states that, given the system of equations $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ *with* $\mathbf{f}(\mathbf{0}) = \mathbf{0}$, *if there exists a Lyapunov function for this equation, then the origin is stable in the sense of Lyapunov; in addition, if* $\dot{V}(\mathbf{x}) < 0$, $\mathbf{x} \neq \mathbf{0}$, *then the stability is asymptotic*.

This theorem on asymptotic stability and stability in the sense of Lyapunov applies in a *local* sense if the region $\|\mathbf{x}(\mathbf{0})\| < \delta$ is small (refer to Fig. 8.1); the theorem applies in the *global* sense when the region includes the entire state space. The value of Lyapunov function for global stability becomes infinite with infinite deviation (i.e., $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$).

The determination of stability through Lyapunov analysis centers around the choice of a Lyapunov function $V(\mathbf{x})$. Unfortunately, there is no universal method for selecting the Lyapunov function which is unique for a given nonlinear system. Several techniques have been devised for the systematic construction of Lyapunov functions; each is applicable to a particular class of systems. If a Lyapunov function cannot be found, it in no way implies that the system is unstable. It only means that our attempt in trying to establish the stability of an equilibrium state has failed. Therefore, faced with specific systems, one has to use experience, intuition, and physical insights to search for an appropriate Lyapunov function. An elegant and powerful Lyapunov analysis may be possible for complex systems if engineering insight and physical properties are properly exploited. In spite of these limitations, Lyapunov's method is the most powerful technique available today for the stability analysis of nonlinear systems.

For linear time-invariant systems of main concern in this chapter, the quadratic function (refer to Section 5.2) is adequate for demonstrating Lyapunov stability. Consider the function

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{P}\mathbf{x} \tag{8.5}$$

where $\mathbf{P}$ is a symmetric positive definite matrix. The quadratic function (8.5) satisfies properties (i), (ii) and (iii) of a Lyapunov function. We need to examine property (iv), the derivative condition, to study the stability properties of the system under consideration.

Discrete-Time Systems: In the following, we extend the Lyapunov stability theorem to discrete-time systems:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)); \ \mathbf{f}(\mathbf{0}) = \mathbf{0} \tag{8.6}$$

Our discussion will be brief because of the strong analogy between the discrete-time and continuous-time cases.

Theorem 8.2 Suppose that there exists a Lyapunov function $V(\mathbf{x}(k))$ which satisfies the following properties:

(i) $V(\mathbf{x}) > 0; \ \mathbf{x} \neq \mathbf{0}$
(ii) $V(\mathbf{0}) = 0$
(iii) $V(\mathbf{x})$ is a smooth function; it is continuous for all $\mathbf{x}$.
(iv) $\Delta V(\mathbf{x}(k)) = [V(\mathbf{x}(k+1)) - V(\mathbf{x}(k))] \leq 0$ along trajectories of Eqn. (8.6).

(8.7)

The *Lyapunov stability theorem states that*, *given the system of equations*, $\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)); \ \mathbf{f}(\mathbf{0}) = \mathbf{0}$, *if there exists a Lyapunov function for this equation*, *then the origin is stable in the sense of Lyapunov*; *in addition*, *if* $\Delta V(\mathbf{x}(k)) < 0$ for $\mathbf{x} \neq \mathbf{0}$, *then the stability is asymptotic*.

8.3 LYAPUNOV FUNCTIONS FOR LINEAR SYSTEMS

As has been said earlier, the Lyapunov theorems give only sufficient conditions on the stability of the equilibrium state of a nonlinear system and, furthermore, there is no unique way of constructing a

Lyapunov function. For a linear system, a Lyapunov function can always be constructed and both the necessary and sufficient conditions on stability established.

8.3.1 Stability of Continuous-Time Linear Systems

Consider a linear system described by the state equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{8.8}$$

where $\mathbf{A}$ is $n \times n$ real constant matrix.

Theorem 8.3 The linear system (8.8) is globally asymptotically stable at the origin if, and only if, for any given symmetric positive definite matrix $\mathbf{Q}$, there exists a symmetric positive definite matrix $\mathbf{P}$, that satisfies the matrix equation

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q} \tag{8.9}$$

Proof Let us first prove the sufficiency of the result. Assume that a symmetric positive definite matrix $\mathbf{P}$ (refer to Section 5.2) exists, which is the unique solution of Eqn. (8.9). Consider the scalar function

$$V(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x}$$

Note that

$$V(\mathbf{x}) > 0 \text{ for } \mathbf{x} \neq \mathbf{0} \quad \text{and} \quad V(\mathbf{0}) = 0$$

The time derivative of $V(\mathbf{x})$ is

$$\dot{V}(\mathbf{x}) = \dot{\mathbf{x}}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}}$$

Using Eqns (8.8) and (8.9), we get

$$\begin{aligned}\dot{V}(\mathbf{x}) &= \mathbf{x}^T\mathbf{A}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}\mathbf{A}\mathbf{x} \\ &= \mathbf{x}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{x} = -\mathbf{x}^T\mathbf{Q}\mathbf{x}\end{aligned}$$

Since $\mathbf{Q}$ is positive definite, $\dot{V}(\mathbf{x})$ is negative definite. Norm of $\mathbf{x}$ may be defined as (Eqn. (5.6b))

$$\|\mathbf{x}\| = (\mathbf{x}^T\mathbf{P}\mathbf{x})^{1/2}$$

Then

$$V(\mathbf{x}) = \|\mathbf{x}\|^2$$

$$V(\mathbf{x}) \to \infty \text{ as } \|\mathbf{x}\| \to \infty$$

Therefore, the system is globally asymptotically stable at the origin.

To prove the necessity of the result, the reader is advised to refer to [105] where the proof has been developed in the following two parts:

(i) If (8.8) is asymptotically stable, then for any $\mathbf{Q}$ there exists a matrix $\mathbf{P}$ satisfying (8.9).

(ii) If $\mathbf{Q}$ is positive definite, then $\mathbf{P}$ is also positive definite.

Comments

(i) The implication of Theorem 8.3 is that if $\mathbf{A}$ is asymptotically stable and $\mathbf{Q}$ is positive definite, then the solution $\mathbf{P}$ of Eqn. (8.9) must be positive definite. Note that it does not say that if $\mathbf{A}$ is asymptotically stable and $\mathbf{P}$ is positive definite, then $\mathbf{Q}$ computed from Eqn. (8.9) is positive definite. For an arbitrary $\mathbf{P}$, $\mathbf{Q}$ may be positive definite (semidefinite) or negative definite (semidefinite).

(ii) Since matrix **P** is known to be symmetric, there are only $n(n + 1)/2$ independent equations in (8.9) rather than n^2.

(iii) In very simple cases, Eqn. (8.9), called the *Lyapunov equation*, can be solved analytically, but usually numerical solution is required. A number of computer programs for this purpose are available [152–154].

(iv) Since Theorem 8.3 holds for any positive definite symmetric matrix **Q**, the matrix **Q** in Eqn. (8.9) is often chosen to be a unit matrix.

(v) If $\dot{V}(\mathbf{x}) = -\mathbf{x}^T\mathbf{Q}\mathbf{x}$ does not vanish identically along any trajectory, then **Q** may be chosen to be positive semidefinite.

A necessary and sufficient condition that $\dot{V}(\mathbf{x})$ does not vanish identically along any trajectory (meaning that $\dot{V}(\mathbf{x}) = 0$ only at $\mathbf{x} = \mathbf{0}$), is that

$$\rho \begin{bmatrix} \mathbf{H} \\ \mathbf{HA} \\ \vdots \\ \mathbf{HA}^{n-1} \end{bmatrix} = n;\ \mathbf{Q} = \mathbf{H}^T\mathbf{H} \tag{8.10}$$

where $\rho(.)$ stands for rank of a matrix.

This can be proved as follows. Since $\dot{V}(\mathbf{x})$ can be written as

$$\dot{V}(\mathbf{x}) = -\mathbf{x}^T\mathbf{Q}\mathbf{x} = -\mathbf{x}^T\mathbf{H}^T\mathbf{H}\mathbf{x},$$

$$\dot{V}(\mathbf{x}) = 0 \text{ means that } \mathbf{H}\mathbf{x} = \mathbf{0}$$

Differentiating with respect to t, gives

$$\mathbf{H}\dot{\mathbf{x}} = \mathbf{H}\mathbf{A}\mathbf{x} = \mathbf{0}$$

Differentiating once again, we get

$$\mathbf{H}\mathbf{A}\dot{\mathbf{x}} = \mathbf{H}\mathbf{A}^2\mathbf{x} = \mathbf{0}$$

Repeating the differentiation process and combining the equations, we obtain

$$\begin{bmatrix} \mathbf{H} \\ \mathbf{HA} \\ \vdots \\ \mathbf{HA}^{n-1} \end{bmatrix} \mathbf{x} = \mathbf{0}$$

A necessary and sufficient condition for $\mathbf{x} = \mathbf{0}$ to be the only solution of this equation is given by (8.10).

Example 8.1

Let us determine the stability of the system described by the following equation:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix}$$

We will first solve Eqn. (8.9) for **P** for an arbitrary choice of real symmetric positive definite matrix **Q**. We may choose **Q** = **I**, the identity matrix. Equation (8.9) then becomes

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{I}$$

or
$$\begin{bmatrix} -1 & 1 \\ -2 & -4 \end{bmatrix}\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}\begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{8.11}$$

Note that we have taken $p_{12} = p_{21}$. This is because the solution matrix **P** is known to be a positive definite real symmetric matrix for a stable system.

From Eqn. (8.11), we get

$$\begin{aligned} -2p_{11} + 2p_{12} &= -1 \\ -2p_{11} - 5p_{12} + p_{22} &= 0 \\ -4p_{12} - 8p_{22} &= -1 \end{aligned}$$

Solving for p_{ij}'s, we obtain

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} = \begin{bmatrix} \frac{23}{60} & -\frac{7}{60} \\ -\frac{7}{60} & \frac{11}{60} \end{bmatrix}$$

Using Sylvester's test (Section 5.2), we find that **P** is positive definite. Therefore, the system under consideration is globally asymptotically stable at the origin.

In order to illustrate the arbitrariness in the choice of **Q**, consider

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{8.12}$$

This is a positive semidefinite matrix. This choice of **Q** is permissible since it satisfies the condition (8.10), as is seen below.

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}[0 \quad 1] = \mathbf{H}^T\mathbf{H}$$

$$\rho\begin{bmatrix} \mathbf{H} \\ \mathbf{H}\mathbf{A} \end{bmatrix} = \rho\begin{bmatrix} 0 & 1 \\ 1 & -4 \end{bmatrix} = 2$$

It can easily be verified that with the choice of **Q** given by Eqn. (8.12), we derive the same conclusion about the stability of the system as obtained earlier with **Q** = **I**.

8.3.2 Stability of Discrete-Time Linear Systems

Consider a linear system described by the state equation

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) \tag{8.13}$$

where **F** is $n \times n$ real constant matrix.

Theorem 8.4 The linear system (8.13) is globally asymptotically stable at the origin if, and only if, for any given symmetric positive definite matrix **Q**, there exists a symmetric positive definite matrix **P**, that satisfies the matrix equation

$$\mathbf{F}^T\mathbf{P}\mathbf{F} - \mathbf{P} = -\mathbf{Q} \tag{8.14}$$

Proof Let us first prove the sufficiency of the result. Assume that a symmetric positive definite matrix **P** exists, which is the unique solution of Eqn. (8.14). Consider the scalar function

$$V(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x}$$

Note that

$$V(\mathbf{x}) > 0 \quad \text{for } \mathbf{x} \neq \mathbf{0} \quad \text{and} \quad V(\mathbf{0}) = 0$$

The difference

$$\begin{aligned}\Delta V(\mathbf{x}) &= V(\mathbf{x}(k+1)) - V(\mathbf{x}(k)) \\ &= \mathbf{x}^T(k+1)\mathbf{P}\mathbf{x}(k+1) - \mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)\end{aligned}$$

Using Eqns (8.13) – (8.14), we get

$$\begin{aligned}\Delta V(\mathbf{x}) &= \mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{F}\mathbf{x}(k) - \mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k) \\ &= \mathbf{x}^T(k)[\mathbf{F}^T\mathbf{P}\mathbf{F} - \mathbf{P}]\mathbf{x}(k) = -\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)\end{aligned}$$

Since **Q** is positive definite, $\Delta V(\mathbf{x})$ is negative definite. Further $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$. Therefore, the system is globally asymptotically stable at the origin.

The proof of necessity is analogous to that of continuous-time case (refer to [105]).

Comments

(i) In very simple cases, Eqn. (8.14), called the *discrete Lyapunov equation*, can be solved analytically, but usually a numerical solution is required. A number of computer programs for this purpose are available [152–154].

(ii) If $\Delta V(\mathbf{x}(k)) = -\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)$ does not vanish identically along any trajectory, then **Q** may be chosen to be positive semidefinite.

A necessary and sufficient condition that $\Delta V(\mathbf{x}(k))$ does not vanish identically along any trajectory (meaning that $\Delta V(\mathbf{x}(k)) = 0$ only at $\mathbf{x} = \mathbf{0}$), is that

$$\rho \begin{bmatrix} \mathbf{H} \\ \mathbf{H}\mathbf{F} \\ \vdots \\ \mathbf{H}\mathbf{F}^{n-1} \end{bmatrix} = n; \ \mathbf{Q} = \mathbf{H}^T\mathbf{H} \tag{8.15}$$

where $\rho(.)$ stands for rank of a matrix.

Example 8.2

Let us determine the stability of the system described by the following equation:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k)$$

with

$$\mathbf{F} = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix}$$

We will first solve Eqn. (8.14) for **P** for an arbitrary choice of real symmetric positive definite matrix **Q**. We may choose **Q** = **I**, the identity matrix. Equation (8.14) then becomes

$$\mathbf{F}^T\mathbf{P}\mathbf{F} - \mathbf{P} = -\mathbf{I}$$

or

$$\begin{bmatrix} -1 & 1 \\ -2 & -4 \end{bmatrix}\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}\begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix} - \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

or

$$-2p_{12} + p_{22} = -1$$

$$2p_{11} + p_{12} - 4p_{22} = 0$$

$$4p_{11} + 16p_{12} + 15p_{22} = -1$$

Solving for p_{ij}'s, we obtain

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} = \begin{bmatrix} -\frac{43}{60} & \frac{11}{30} \\ \frac{11}{30} & -\frac{4}{15} \end{bmatrix}$$

Using Sylvester's test (Section 5.2) we find that **P** is negative definite. Therefore, the system under consideration is unstable.

8.4 PARAMETER OPTIMIZATION AND OPTIMAL CONTROL PROBLEMS

In previous chapters, we encountered various methods for designing feedback control laws, ranging from root-locus and Bode-plot techniques to pole-placement by state feedback and state estimation. In each case, the designer was left with decisions regarding the locations of closed-loop poles. We have given a fairly complete treatment of these design techniques for linear time-invariant single-variable systems.

Here in this chapter, a somewhat different approach to design is taken. The performance of the system is measured with a single scalar quantity—the performance index. A configuration of the controller is selected and free parameters of the controller that optimize (minimize or maximize as the case may be) the performance index are determined. In most industrial control problems, the nature of the performance index is such that the design process requires its minimization.

The design approach based on parameter optimization consists of the following steps:

(i) Compute the performance index J as a function of the free parameters $k_1, k_2, \ldots, k_n$, of the system with fixed configuration:

$$J = J(k_1, k_2, \ldots, k_n) \tag{8.16}$$

(ii) Determine the solution set k_i of the equations:

$$\frac{\partial J}{\partial k_i} = 0;\ i = 1, 2, \ldots, n \tag{8.17}$$

Equations (8.17) give the necessary conditions for J to be minimum. From the solution set of these equations, find the subset that satisfies the sufficient conditions, which require that the *Hessian matrix* given below is positive definite.

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 J}{\partial k_1^2} & \dfrac{\partial^2 J}{\partial k_1 \partial k_2} & \cdots & \dfrac{\partial^2 J}{\partial k_1 \partial k_n} \\ \dfrac{\partial^2 J}{\partial k_2 \partial k_1} & \dfrac{\partial^2 J}{\partial k_2^2} & \cdots & \dfrac{\partial^2 J}{\partial k_2 \partial k_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 J}{\partial k_n \partial k_1} & \dfrac{\partial^2 J}{\partial k_n \partial k_2} & \cdots & \dfrac{\partial^2 J}{\partial k_n^2} \end{bmatrix} \tag{8.18}$$

Since $\dfrac{\partial^2 J}{\partial k_i \partial k_j} = \dfrac{\partial^2 J}{\partial k_j \partial k_i}$,

the matrix **H** is always symmetric.

(iii) If there are two or more sets of k_i satisfying the necessary as well as sufficient conditions of minimization of J, then compute the corresponding J for each set. The set that gives the smallest J is the optimum set.

Selection of an appropriate performance index is as much a part of the design process, as the minimization of the index. We know that the performance of a control system can be adequately specified in terms of settling time, peak overshoot, and steady-state error. The performance index could then be chosen as

$$J \triangleq K_1(\text{settling time}) + K_2(\text{peak overshoot}) + K_3(\text{steady-state error})$$

where the K_i are weighing factors.

Although the criterion seems reasonable, it is not trackable analytically. A compromise must be made between specifying a performance index which includes all the desired system characteristics, and a performance index which can be minimized with a reasonable amount of computation.

In the following, we present several performance indices which include the desired system characteristics and, in addition, have good mathematical trackability. These indices often involve integrating some function of system error over some time interval when the system is subjected to a standard command or disturbance such as step. A common example is the integral of absolute error (IAE) defined by

$$\text{IAE} \triangleq \int_0^\infty |e(t)|\, dt$$

If the index is to be computed numerically, the infinite upper limit can be replaced by the limit t_f, where t_f is large enough so that $e(t)$ is negligible for $t > t_f$. This index is not unreasonable since both the fast but highly oscillatory systems and the sluggish systems will give large IAE value (refer to Fig. 8.3). Minimization of IAE by adjusting system parameters will provide acceptable relative stability and speed of response. Also, a finite value of IAE implies that the steady-state error is zero.

Another similar index is the integral of time multiplied by absolute error (ITAE), which exhibits the additional useful features that the initial large error (unavoidable for a step input) is not heavily weighted, whereas errors that persist are more heavily weighted.

$$\text{ITAE} \triangleq \int_0^\infty t\,|e(t)|\, dt$$

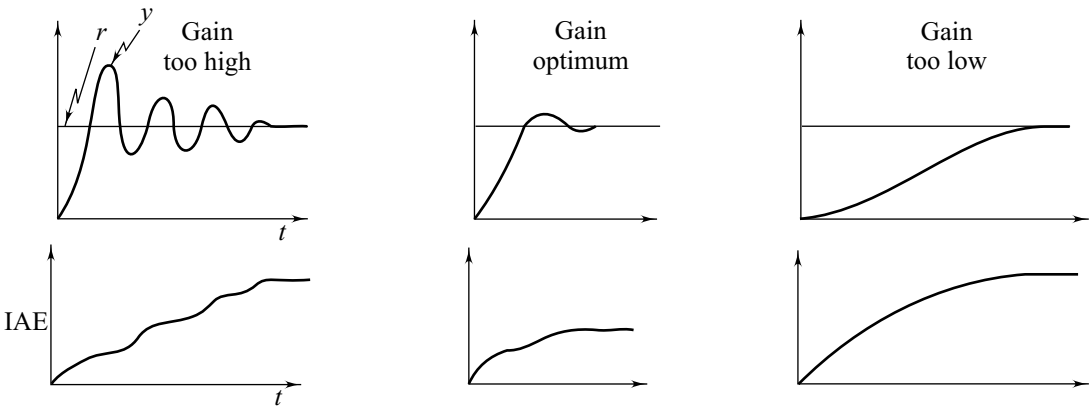


Fig. 8.3 The IAE optimal response criterion

The integral of square error (ISE) and integral of time multiplied by square error (ITSE) indices are analogous to IAE and ITAE criteria, except that the square of the error is employed for three reasons: (i) in some applications, the squared error represents the system's power consumption, (ii) squaring the error weighs large errors more heavily than small errors, and (iii) the squared error is much easier to handle analytically.

$$\text{ISE} \triangleq \int_0^\infty e^2(t)\, dt$$

$$\text{ITSE} \triangleq \int_0^\infty te^2(t)\, dt$$

The system whose design minimizes (or maximizes) the selected performance index with no constraints on controller configuration is, by definition, *optimal*.

The difference between parameter optimization and optimal control problems is that no constraint on controllers is imposed on the latter. In optimal design, the designer is permitted to use controllers of any degree and any configuration, whereas in parameter optimization the configuration and the type of controllers are predetermined. Since there is no constraint imposed on controllers, optimal design results in a better system, i.e., lower value of the performance index.

However, because of considerations other than minimization of the performance index, one may not build an optimal control system. For example, optimal solutions to the problem of control of a linear time-invariant plant may result in a nonlinear and/or time-varying system. Hardware realization of such an optimal control law may be quite difficult and expensive. Also, in many control problems, the optimal solution gives an open-loop control system which is successful only in the absence of meaningful disturbances. In practical systems, then, it may be more sensible to seek *suboptimal* control laws: we select a feedback control configuration and the type of controller, based on considerations of cost, availability of components, etc., and then determine the best possible values of the free parameters of the controller that minimize the given performance index. Modifications in control configuration and the type of controller are made until a satisfactory system is obtained—which has performance characteristics close to the optimal control system we have worked out in theory.

There exists an important class of optimal control problems for which quite general results have been obtained. It involves control of linear systems with the objective of minimizing the integral of a quadratic performance index. An important feature of this class of problems is that optimal control is possible by feedback controllers. For linear time-invariant plants, the optimal control results in a linear time-invariant closed-loop system. The implementation of optimal control is, therefore, simple and less expensive. Many problems of industrial control belong to this class of problems—*linear quadratic optimal control problems.*

As we shall see later in this chapter, the linear quadratic optimal control laws have some computational advantage, and a number of useful properties. The task of the designer shifts to the one of specifying various parameters in the performance index.

In the previous chapters, we have been mostly concerned with the design of single-variable systems. Extensions of the root-locus method and the Bode/Nyquist-plot design to multivariable cases have been reported in the literature. However, the design of multivariable systems using these techniques is much more complicated than the single-variable cases. Design of multivariable systems through pole-placement can also be carried out; the computations required are however highly complicated.

The optimal control theory provides a simple and powerful tool for designing multivariable systems. Indeed, the equations and computations required in the design of optimal single-variable systems and those in the design of optimal multivariable systems are almost identical. We will use, therefore, in this chapter, the Multi-Input, Multi-Output (MIMO) state variable model in the formulation of optimal control problem.

The objective set for the rest of this chapter is the presentation of simple, and analytically solvable, optimal control and parameter optimization problems. This will provide insight into optimal and suboptimal structures and algorithms that may be applied in practical cases. For detailed study, specialized books on optimal control [109–124] should be consulted. A moderate treatment of the subject is also available in reference [105].

Commercially available software [152–154] may be used for solving complex optimal/suboptimal control problems.

8.5 QUADRATIC PERFORMANCE INDEX

A commonly used performance criterion is the integral square error (ISE):

$$J = \int_0^\infty [y(t) - y_r]^2 \, dt \tag{8.19a}$$

$$= \int_0^\infty e^2(t) \, dt \tag{8.19b}$$

where y_r is the command or set-point value of the output, $y(t)$ is the actual output, $e(t) = y(t) - y_r$ is the error of the system.

This criterion, which has good mathematical trackability properties, is acceptable in practice, as a measure of system performance. The criterion penalizes positive and negative errors equally. It penalizes

heavily on large errors; hence, a small J usually results in a system with small overshoot. Since the integration is carried out over $[0, \infty)$, a small J limits the effect of small error lasting for long time and, thus, results in a small settling time. Also, a finite J implies that the steady-state error is zero.

The optimal design obtained by minimizing the performance index given by Eqns (8.19) may be unsatisfactory, because it may lead to excessively large magnitudes of control signals. A more realistic solution to the problem is reached, if the performance index is modified to account for physical constraints like saturation in physical devices. Therefore, a more realistic performance index is of the form

$$J = \int_0^\infty e^2(t)\, dt \tag{8.20a}$$

subject to the following constraint on control signal $u(t)$,

$$\max |u(t)| \leq M \tag{8.20b}$$

for some constant M. The constant M is determined by the linear range of the plant.

Although the criterion expressed in (8.20) can be used in the design, it is not convenient to work with. In a number of problems, $u^2(t)$ is a measure of the instantaneous rate of expenditure of energy. To minimize energy expenditure, we minimize

$$\int_0^\infty u^2(t)\, dt \tag{8.21}$$

We would, very much, like to replace the performance criterion given by (8.20) by the following *quadratic performance index*:

$$J = \int_0^\infty [e^2(t) + u^2(t)]\, dt$$

To allow greater generality, we can insert a real positive constant λ to obtain

$$J = \int_0^\infty [e^2(t) + \lambda\, u^2(t)]\, dt \tag{8.22}$$

By adjusting the *weighting factor* λ, we can weigh the relative importance of the system error and the expenditure of energy. By increasing λ, i.e., by giving sufficient weight to control effort, the amplitude of the control signal, which minimizes the overall performance index, may be kept within practical bounds, although at the expense of the increased system error. Note that as $\lambda \to 0$, the performance index reduces to the integral square error criterion. In this case, the magnitude of $u(t)$ will be very large and the constraint given by (8.20b) may be violated. If $\lambda \to \infty$, the performance index reduces to the one given by Eqn. (8.21), and the optimal system that minimizes this J is one with $u = 0$. From these two extreme cases, we conclude that if λ is properly chosen, then the constraint of Eqn. (8.20b) will be satisfied.

Example 8.3

For the system of Fig. 8.4, let us compute the value of K that minimizes ISE for the unit-step input.

For the system under consideration,

$$\frac{E(s)}{R(s)} = \frac{s}{s+K}$$

For unit-step input,

$$E(s) = \frac{1}{s+K}$$

Therefore, $e(t) = e^{-Kt}$

$$\text{ISE} = \int_0^\infty e^2(t)\, dt = \frac{1}{2K}$$

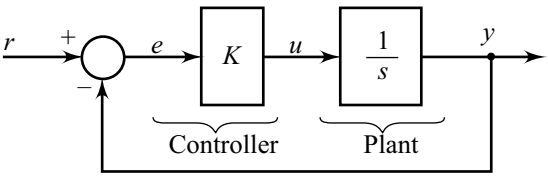


Fig. 8.4 Control system for parameter optimization

Obviously, the minimum value of ISE is obtained as $K \to \infty$. This is an impractical solution, resulting in excessive strain on the physical components of the system.

Sound engineering judgment tells us that we must include the 'cost' of the control effort in our performance index. The quadratic performance index

$$J = \int_0^\infty [e^2(t) + u^2(t)]\, dt$$

may serve the objective.

From Fig. 8.4, $u(t) = K\, e(t) = K\, e^{-Kt}$

Therefore,

$$J = \frac{1}{2K} + \frac{K}{2}$$

The minimum value of J is obtained when

$$\frac{\partial J}{\partial K} = -\frac{1}{2K^2} + \frac{1}{2} = 0 \text{ or } K = 1$$

Note that

$$\frac{\partial^2 J}{\partial K^2} = \frac{1}{K^3} > 0$$

The minimum value of J is 1.

This solution, which weighs error and control effort equally, seems to be acceptable.

The following performance index assigns larger weight to error minimization:

$$J = \int_0^\infty [e^2(t) + \lambda u^2(t)]\, dt;\ \lambda = 0.5$$

For the system under consideration,

$$J = \frac{1}{2K} + \lambda \frac{K}{2}$$

$$\frac{\partial J}{\partial K} = 0 \text{ gives}$$

$$K = \sqrt{2}, J_{\min} = 0.707$$

When λ is greater than unity, it means that more importance is given to the constraint on amplitude of $u(t)$ compared to the performance of the system. A suitable value of λ is chosen so that relative importance of the system performance is contrasted with the importance of the limit on control effort. Figure 8.5 gives a plot of the performance index *versus* K for various values of λ.

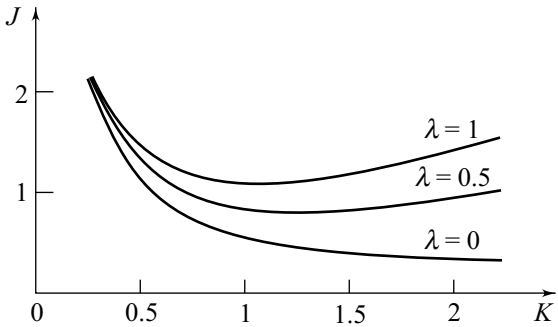


Fig. 8.5 Performance index versus gain for the system shown in Fig. 8.4

Example 8.4

Consider the liquid-level system shown in Fig. 8.6. h represents the deviation of liquid head from the steady-state value $\bar{H}$.

The pump controls the liquid head h by supplying liquid at a rate $(\bar{Q}_i + q_i)$ m^3/sec to the tank. We shall assume that the flow rate q_i is proportional to the error in liquid level (desired level – actual level). Under these assumptions, the system equations are [155]:

(i) $$A\frac{dh}{dt} = q_i - \frac{\rho g h}{R}$$

where A = area of cross-section of the tank;

R = total resistance offered by the tank outlet and pipe ($R \triangleq$ incremental change in pressure across the restriction/incremental change in flow through the restriction);

ρ = density of the liquid; and

g = acceleration due to gravity.

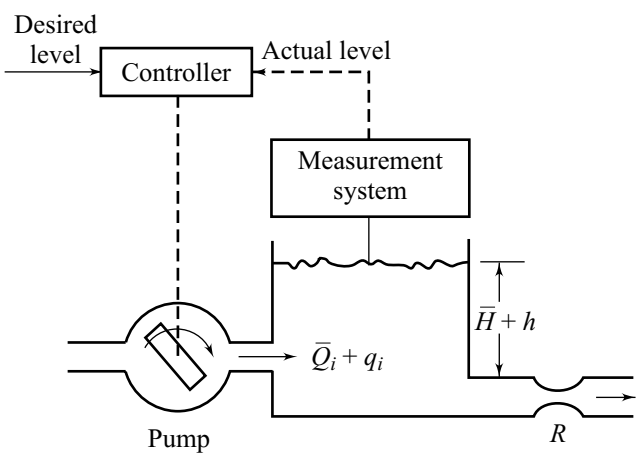


Fig. 8.6 A liquid-level system

(ii) $$q_i = Ke$$

where $e =$ error in liquid level and $K =$ gain constant

Let $A = 1$, and $\dfrac{R}{\rho g} = 1$. Then

$$\frac{H(s)}{Q_i(s)} = \frac{1}{s+1}$$

The block diagram representation is given in Fig. 8.7. The output $y(t) = h(t)$ is the deviation in liquid head from steady-state value. Therefore, the output $y(t)$ is itself the error which is to be minimized. Let us pose the problem of computing the value of K that minimizes the ISE for the initial condition $y(0) = 1$.

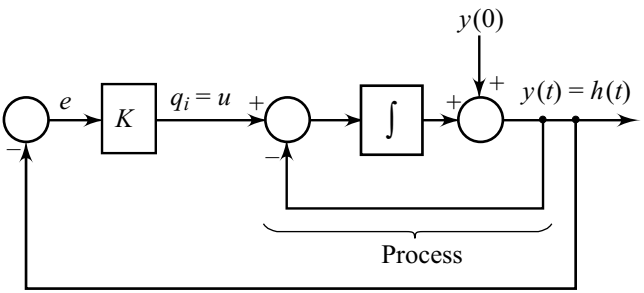


Fig. 8.7 Block diagram representation of the system shown in Fig. 8.6

From Fig. 8.7, we get

$$Y(s) = \frac{s}{s+1+K}\left(\frac{y(0)}{s}\right) = \frac{1}{s+1+K}$$

Therefore,

$$y(t) = e^{-(1+K)t}$$

$$\text{ISE} = \int_0^\infty y^2(t)\,dt = \frac{1}{2(1+K)}$$

Obviously, the minimum value of ISE is obtained as $K \to \infty$.

This is an impractical solution, resulting in excessive strain on the physical components of the system. Increasing the gain means, in effect, increasing the pump size.

Now, consider the problem of minimization of

$$J = \int_0^\infty [y^2(t) + u^2(t)]\,dt$$

From Fig. 8.7, we have

$$u(t) = -Ky(t)$$

Therefore,

$$u(t) = -Ke^{-(1+K)t}$$

$$J = \frac{1}{2(1+K)} + \frac{K^2}{2(1+K)}$$

$$\frac{\partial J}{\partial K} = 0 \text{ gives } K = \left(\sqrt{2}-1\right)$$

Note that

$$\frac{\partial^2 J}{\partial K^2} = \frac{2}{(1+K)^3} > 0$$

The minimum value of J is $\left(\sqrt{2}-1\right)$.

8.5.1 State Regulator Problem

The performance index given by Eqn. (8.22), is a translation of the requirement of regulation of the system output, with constraints on amplitude of the input applied to the plant. We now extend the proposed performance index for the control problem where all the state variables of the system are to be regulated. We use multivariable formulation of the plant model.

Consider the control problem where the objective is to maintain the system state given by the $n \times 1$ state vector $\mathbf{x}(t)$, near the desired state $\mathbf{x}_d$ (which, in many cases, is the equilibrium point of the system) for all time.

Relative to the desired state $\mathbf{x}_d$, $(\mathbf{x}(t) - \mathbf{x}_d)$ can be viewed as the instantaneous system error. If we transform the system coordinates such that the desired state becomes the origin of the state space, then the new state $\mathbf{x}(t)$ is itself the error.

One measure of the magnitude of the state vector $\mathbf{x}(t)$ (or of its distance from the origin) is the norm $\|\mathbf{x}(t)\|$ defined by

$$\|\mathbf{x}(t)\|^2 = \mathbf{x}^T(t)\mathbf{x}(t)$$

Therefore,

$$J = \int_0^\infty [\mathbf{x}^T(t)\mathbf{x}(t)]\, dt = \int_0^\infty [x_1^2(t) + x_2^2(t) + \cdots + x_n^2(t)]\, dt$$

is a reasonable measure of the system transient response.

In practical systems, the control of all the states of the system is not equally important. To be more general,

$$J = \int_0^\infty [\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t)]\, dt \tag{8.23}$$

with $\mathbf{Q}$ as $n \times n$ real, symmetric, positive definite (or positive semidefinite) constant matrix, can be used as a performance measure. The simplest form of $\mathbf{Q}$ one can use is the diagonal matrix:

$$\mathbf{Q} = \begin{bmatrix} q_1 & 0 & \cdots & 0 \\ 0 & q_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & q_n \end{bmatrix}$$

The *i*th entry of **Q** represents the weight the designer places on the constraint on the state variable $x_i(t)$. The larger the value of q_i relative to the other values of q, the more control effort is spent to regular $x_i(t)$.

The design obtained by minimizing the performance index of the form (8.23) may be unsatisfactory in practice. A more realistic solution is obtained if the performance index is modified by adding a penalty term for physical constraints on the $p \times 1$ control vector $\mathbf{u}(t)$. One of the ways of accomplishing this is to introduce the following quadratic control term in the performance index:

$$J = \int_0^\infty [\mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)]\, dt \tag{8.24}$$

where **R** is $p \times p$ real, symmetric, positive definite,[1] constant matrix.

By giving sufficient weight to control terms, the amplitudes of control signals which minimize overall performance index may be kept within practical bounds, although at the expense of increased error in $\mathbf{x}(t)$.

For the state regulator problem, a useful performance measure is, therefore[2],

$$J = \tfrac{1}{2}\int_0^\infty [\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)]\, dt \tag{8.25}$$

8.5.2 Output Regulator Problem

In the state regulator problem, we are concerned with maintaining the $n \times 1$ state vector $\mathbf{x}(t)$ near the origin of the state space for all time. In the output regulator problem, on the other hand, we are concerned with maintaining the $q \times 1$ output vector $\mathbf{y}(t)$ near origin for all time. A useful performance measure for the output regulator problem is

$$J = \tfrac{1}{2}\int_0^\infty [\mathbf{y}^T(t)\mathbf{Q}\mathbf{y}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)]\, dt \tag{8.26a}$$

where **Q** is a $q \times q$ positive definite (or positive semidefinite) real, symmetric constant matrix, and **R** is a $p \times p$ positive definite, real, symmetric, constant matrix.

Substituting $\mathbf{y} = \mathbf{C}\mathbf{x}$ in Eqn. (8.26a), we get

$$J = \tfrac{1}{2}\int_0^\infty (\mathbf{x}^T\mathbf{C}^T\mathbf{Q}\mathbf{C}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})\, dt \tag{8.26b}$$

Comparing Eqn. (8.26b) with Eqn. (8.25) we observe that the two indices are identical in form; **Q** in Eqn. (8.25) is replaced by $\mathbf{C}^T\mathbf{Q}\mathbf{C}$ in Eqn. (8.26b). If we assume that the plant is completely observable, then **C** cannot be zero; $\mathbf{C}^T\mathbf{Q}\mathbf{C}$ will be positive definite (or positive semidefinite) whenever **Q** is positive definite (or positive semidefinite).

Thus, the solution to the output regulator problem directly follows from that of the state regulator problem.

---

[1] As we shall see in Section 8.7, positive definiteness of **R** is a necessary condition for the existence of the optimal solution to the control problem.

[2] Note that multiplication by 1/2 does not affect the minimization problem. The constant helps us in mathematical manipulations as we shall see later in Section 8.7.

8.6 CONTROL CONFIGURATIONS

8.6.1 State Regulator

Consider the plant represented by linear state equations of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{8.27a}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \tag{8.27b}$$

where $\mathbf{x}(t)$ is the $n \times 1$ state vector, $\mathbf{u}(t)$ is the $p \times 1$ input vector, $\mathbf{y}(t)$ is the $q \times 1$ output vector; $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are, respectively, $n \times n$, $n \times p$ and $q \times n$ real constant matrices. We will assume that the null state $\mathbf{x} = \mathbf{0}$ is the desired state; $\mathbf{x}(t)$ is thus system-error vector at time t.

We shall be interested in selecting the controls $\mathbf{u}(t)$ which quickly move the system state $\mathbf{x}(t)$ to the null state $\mathbf{x} = \mathbf{0}$ for any initial perturbation $\mathbf{x}^0$. The control problem is, thus, to determine $\mathbf{u}(t)$ which minimizes performance index of the form

$$J = \frac{1}{2}\int_0^\infty [\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)]\, dt \tag{8.28}$$

where $\mathbf{Q}$ is a positive definite (or positive semidefinite), real, symmetric constant matrix, and $\mathbf{R}$ is a positive definite, real, symmetric, constant matrix.

An important feature (proved later in Section 8.7) of this class of problems, is that optimal control is possible by feedback control law of the form

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) \tag{8.29a}$$

where $\mathbf{K}$ is a $p \times n$ constant matrix, or

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \end{bmatrix} = - \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \vdots & \vdots & & \vdots \\ k_{p1} & k_{p2} & \cdots & k_{pn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{8.29b}$$

If the unknown elements of the matrix $\mathbf{K}$ are determined so as to minimize the performance index given by Eqn. (8.28), then the control law given by Eqns (8.29) is optimal. The configuration of the optimal closed-loop control system is represented by the block diagram of Fig. 8.8. As we shall see later in this chapter, controllability of the plant (8.27) and positive definiteness of matrix $\mathbf{Q}$ in the performance index (8.28), are sufficient conditions for the existence of asymptotically stable ($\mathbf{x}(t) \to 0$ as $t \to \infty$) optimal solution to the control problem.

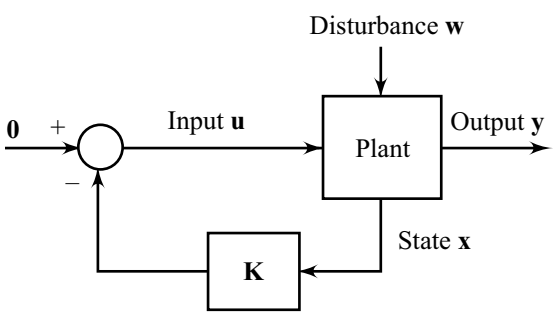


Fig. 8.8 Control configuration of a state regulator

It may be noted that the optimal solution obtained by minimizing the performance index (8.28) may not be the best solution in all circumstances. For example, all the elements of the matrix $\mathbf{K}$ may not be free;

some gains are fixed by the physical constraints of the system and are, therefore, relatively inflexible. Similarly, if all the states $\mathbf{x}(t)$ are not accessible for feedback, one has to go for a state observer whose complexity is comparable to that of the system itself. It is natural to seek a procedure that relies on the use of feedback from the accessible state variables only, constraining the gain elements of matrix $\mathbf{K}$ corresponding to the inaccessible state variables, to have zero value (Section 8.9). Thus, whether one chooses an *optimal* or *suboptimal* solution depends on many factors in addition to the performance required out of the system.

8.6.2 State Observer

Implementation of the optimal control law given by Eqns (8.29) requires the ability to directly measure the entire state vector $\mathbf{x}(t)$. For many systems, full state measurements are not practical. In Section 7.5, we found that the state vector of an observable linear system can be estimated using a state observer which operates on input and output measurements. We assumed that all inputs can be specified exactly and all outputs can be measured with unlimited precision. The dynamic behavior of the observer was assumed to be specified in terms of its characteristic equation.

Here, we are concerned with the optimal design of the state observer for the multivariable system given by Eqns (8.27).

We postulate the existence of an observer of the form

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{M}[\mathbf{y}(t) - \mathbf{C}\hat{\mathbf{x}}(t)] \tag{8.30}$$

where $\hat{\mathbf{x}}$ is the estimate of state $\mathbf{x}$ and $\mathbf{M}$ is an $n \times q$ real constant gain matrix. The observer structure is shown in Fig. 8.9, which is of the same form as that considered in Section 7.5. The estimation error is given by

$$\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t) \tag{8.31a}$$

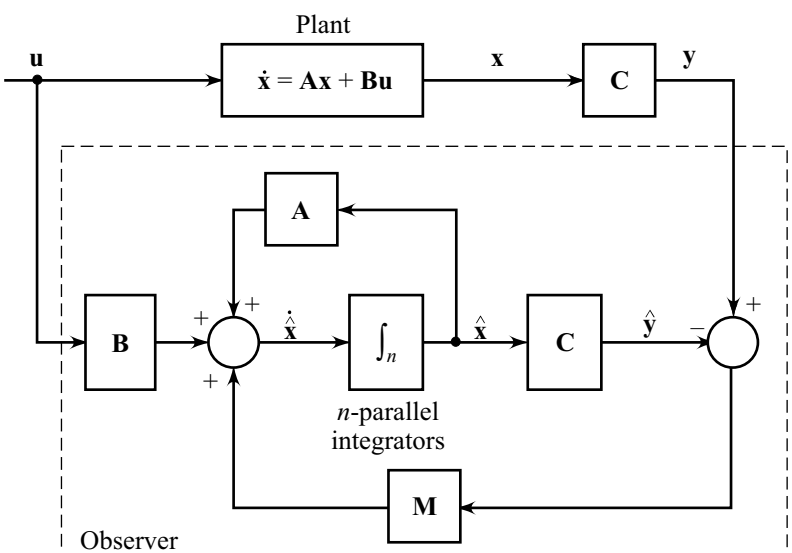


Fig. 8.9 State-observer structure

From Eqns (8.27) and (8.30), we have

$$\dot{\tilde{\mathbf{x}}}(t) = (\mathbf{A} - \mathbf{MC})\,\tilde{\mathbf{x}}(t) \tag{8.31b}$$

To design gain matrix **M**, we may use the duality between control and estimation problems, developed in Section 7.5. (refer to Table 7.1). As per the duality principle, the problem of determination of gain matrix **M** for the optimal state observer, is mathematically equivalent to designing optimal state regulator for the 'transposed auxiliary system' (refer to Eqns (7.36))

$$\dot{\boldsymbol{\zeta}}(t) = \mathbf{A}^T\boldsymbol{\zeta}(t) + \mathbf{C}^T\boldsymbol{\eta}(t) \tag{8.32a}$$

The design problem is to determine $n \times q$ gain matrix **M** such that

$$\boldsymbol{\eta}(t) = -\mathbf{M}^T\boldsymbol{\zeta}(t) \tag{8.32b}$$

minimizes a quadratic performance index of the form

$$J = \tfrac{1}{2}\int_0^\infty (\boldsymbol{\zeta}^T\mathbf{Q}_0\boldsymbol{\zeta} + \boldsymbol{\eta}^T\mathbf{R}_0\boldsymbol{\eta})\,dt \tag{8.33}$$

where $\mathbf{Q}_0$ is a positive definite (or positive semidefinite), real, symmetric constant matrix, and $\mathbf{R}_0$ is a positive definite, real, symmetric, constant matrix.

The solution to this problem exists if the auxiliary system (8.32) is completely controllable. This condition is met, if the original system (8.27) is completely observable.

The *separation principle* (refer to Section 7.6) allows for the separate designs of state-feedback control law and state observer; the control law and the observer are then combined as per the configuration of Fig. 8.10. The weighting matrices $\mathbf{Q}_0$ and $\mathbf{R}_0$, for the observer design, can be assumed to be equal to the weighting matrices **Q** and **R**, respectively, of the control-law design. Generally, however, one would design a faster observer in comparison with the regulator, i.e., for $\mathbf{Q}_0 = \mathbf{Q}$, the elements of $\mathbf{R}_0$ are chosen smaller than those of **R**.

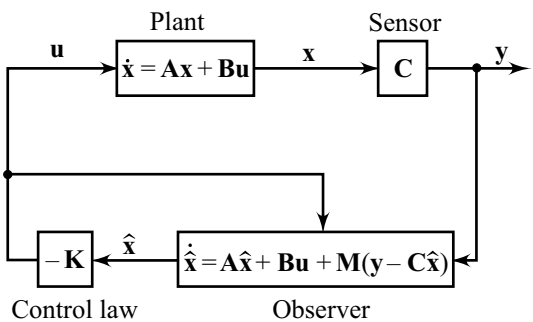


Fig. 8.10 Combined state-feedback control and state estimation

The above solution, to the state estimation problem based on duality between the control and estimation, can be formalized by using "Optimal Filtering Theory". The formal development of the result extends beyond the scope of this text [105]. We may, however, use the term "optimal filter" for the state observer designed by the procedure given here in the text.

An optimal filter whose weighting matrices $\mathbf{Q}_0$ and $\mathbf{R}_0$ are determined by the "spectral properties" of the exogenous noise signals is termed a *Kalman filter* [105].

8.6.3 Servo Systems

The control configuration of Fig. 8.8, implicitly assumes that the null state $\mathbf{x} = \mathbf{0}$ is the desired equilibrium state of the system. It is a state regulator with zero command input.

In servo systems, where the output $\mathbf{y}(t)$ is required to track a constant command input, the equilibrium state is a constant point (other than the origin) in state space. This servo problem can be formulated as a 'shifted regulator problem', by shifting the origin of the state space to the equilibrium point. Formulation of the shifted regulator problem for single-input systems was given in Section 7.7. Extension of the formulation to the multi-input case is straightforward.

8.6.4 State-Feedback with Integral Control

In a state-feedback control system (which is a generalization of proportional plus derivative feedback), it is usually required that the system have one or more integrators within the closed loop. This will lead to zero steady-state error when the command input and disturbance have constant steady-state values. Unless the plant to be controlled has integrating property, it is generally necessary to add one or more integrators within the loop.

For the system (8.27), we can feedback the state $\mathbf{x}$ as well as the integral of the error in output by augmenting the plant state $\mathbf{x}$ with the extra 'integral state'. For single-input systems, the problem of state feedback with integral control was formulated as a state regulator problem in Section 7.8. This was done by augmenting the plant state with 'integral state', and shifting the origin of the state space to the equilibrium point. Multivariable generalization of state feedback with integral control is straightforward.

8.7 OPTIMAL STATE REGULATOR

The parameter optimization problem, as explained earlier, refers to the problem of obtaining the optimum values of free parameters in a predetermined control configuration. In optimal control design, the designer is permitted to use controllers of any degree and any configuration. Optimal solutions to the problem of control of a linear time-invariant plant, may result in a nonlinear and/or time-varying system [105]. Also, in many control problems, the optimal solution gives an open-loop control system [105].

For an important class of control problems, which involves control of linear time-invariant plants with the objective of minimizing the integral of a quadratic performance index, the optimal control is possible by state-feedback control that results in a linear time-invariant closed-loop system. Many problems of industrial control belong to this class of problems—*linear quadratic optimal control problems*.

To prove this important result, we consider the optimal control problem for a linear multivariable completely controllable plant

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{8.34}$$

where $\mathbf{x}$ is the $n \times 1$ state vector, $\mathbf{u}$ a $p \times 1$ input vector; $\mathbf{A}$ and $\mathbf{B}$ are, respectively, $n \times n$ and $n \times p$ real constant matrices, and the null state $\mathbf{x} = \mathbf{0}$ is the desired steady state.

The objective is to find the optimal control law that minimizes the following performance index, subject to the initial conditions $\mathbf{x}(0) \triangleq \mathbf{x}^0$:

$$J = \frac{1}{2}\int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})dt \tag{8.35}$$

where **Q** is $n \times n$ positive definite, real, symmetric, constant matrix, and **R** is $p \times p$ positive definite, real, symmetric, constant matrix.

Since the (**A**, **B**) pair is completely controllable[3], there exists a state-feedback control law

$$\mathbf{u} = -\mathbf{K}\mathbf{x} \tag{8.36}$$

where **K** is $p \times n$ real, constant, unconstrained gain matrix, that results in an *asymptotically* stable closed-loop system (refer to Section 7.3)

$$\dot{\mathbf{x}}(t) = (\mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x}) = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} \tag{8.37}$$

This implies that there is a Lyaponov function $V = \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x}$ for the closed-loop system (8.37); that is, for some positive definite matrix **P**, the time derivative dV/dt evaluated on the trajectories of the closed-loop system is negative definite. We now state and prove a condition for $\mathbf{u} = -\mathbf{K}\mathbf{x}(t)$ to be optimal [35].

Theorem 8.5 If the state-feedback controller $\mathbf{u} = -\mathbf{K}\mathbf{x}(t)$ is such that it minimizes the function

$$f(\mathbf{u}) = \frac{dV}{dt} + \frac{1}{2}\left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}\right), \tag{8.38}$$

and the minimum value of $f(\mathbf{u}) = 0$ for some $V = \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x}$, then the controller is optimal.

Proof We can represent (8.38) as (denoting minimizing **u** by $\mathbf{u}^*$)

$$\left.\frac{dV}{dt}\right|_{\mathbf{u}=\mathbf{u}^*} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^{*T}\mathbf{R}\mathbf{u}^* = 0$$

Hence

$$\left.\frac{dV}{dt}\right|_{\mathbf{u}=\mathbf{u}^*} = -\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^{*T}\mathbf{R}\mathbf{u}^*$$

Integrating both sides with respect to time from 0 to ∞, we obtain

$$V\left(\mathbf{x}(\infty)\right) - V\left(\mathbf{x}(0)\right) = -\frac{1}{2}\int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^{*T}\mathbf{R}\mathbf{u}^*\right)dt$$

Because, by assumption, the closed-loop system is asymptotically stable, we have $\mathbf{x}(\infty) = \mathbf{0}$; therefore,

$$V\left(\mathbf{x}(0)\right) = \frac{1}{2}\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0) = \frac{1}{2}\int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^{*T}\mathbf{R}\mathbf{u}^*\right)dt$$

Thus, if a linear stabilizing controller satisfies the hypothesis of the theorem, then the value of the performance index (8.35) for such a controller is

$$J\left(\mathbf{u}^*\right) = \frac{1}{2}\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0)$$

---

[3] The controllability of the (**A**, **B**) pair is not a necessary condition for the existence of the optimal solution. If the (**A**, **B**) pair is not completely controllable, we can transform the plant model to controllability canonical form given in Eqn. (5.123c). It decomposes the model into two parts: the controllable part and the uncontrollable part. If the uncontrollable part is stable, then the model is said to be stabilizable. Stabilizability of the (**A**, **B**) pair is a necessary condition for the existence of optimal solution.

Since $\mathbf{u}^*$ minimizes the function in (8.38) and the minimum value is zero, for any $\mathbf{u}$ different from $\mathbf{u}^*$, the value of the function will be greater than/equal to zero.

$$\left.\frac{dV}{dt}\right|_{\mathbf{u}=\hat{\mathbf{u}}} + \tfrac{1}{2}\left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \hat{\mathbf{u}}^T\mathbf{R}\hat{\mathbf{u}}\right) \geq 0$$

or

$$\left.\frac{dV}{dt}\right|_{\mathbf{u}=\hat{\mathbf{u}}} \geq -\tfrac{1}{2}\left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \hat{\mathbf{u}}^T\mathbf{R}\hat{\mathbf{u}}\right)$$

Integrating both sides with respect to time from 0 to ∞, yields

$$V\left(\mathbf{x}(0)\right) \leq \tfrac{1}{2}\int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \hat{\mathbf{u}}^T\mathbf{R}\hat{\mathbf{u}}\right) dt$$

implying that

$$J\left(\mathbf{u}^*\right) \leq J\left(\hat{\mathbf{u}}\right)$$

for any $\hat{\mathbf{u}} \neq \mathbf{u}^*$. Therefore, the controller $\mathbf{u}^*$ is optimal.

It follows from the above theorem that the synthesis of optimal control law involves finding an appropriate Lyapunov function, or equivalently, the matrix $\mathbf{P}$. The matrix $\mathbf{P}$ is found by minimizing

$$f\left(\mathbf{u}\right) = \frac{dV}{dt} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u} \tag{8.39}$$

We first apply to (8.39) the necessary conditions for unconstrained minimization.

$$\frac{\partial}{\partial \mathbf{u}}\left(\frac{dV}{dt} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right)\Bigg|_{\mathbf{u}=\mathbf{u}^*} = \mathbf{0}$$

Differentiating the above yields

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{u}}\left(\frac{dV}{dt} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) &= \frac{\partial}{\partial \mathbf{u}}\left(\tfrac{1}{2}\left(\dot{\mathbf{x}}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}}\right) + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) \\
&= \frac{\partial}{\partial \mathbf{u}}\left(\mathbf{x}^T\mathbf{P}\dot{\mathbf{x}} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) \\
&= \frac{\partial}{\partial \mathbf{u}}\left(\mathbf{x}^T\mathbf{P}\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{P}\mathbf{B}\mathbf{u} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\right) \\
&= \mathbf{B}^T\mathbf{P}^T\mathbf{x} + \mathbf{R}\mathbf{u} = \mathbf{0} = \mathbf{B}^T\mathbf{P}\mathbf{x} + \mathbf{R}\mathbf{u}
\end{aligned}$$

Hence, a candidate for an optimal control law has the form

$$\mathbf{u}^* = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}\mathbf{x} = -\mathbf{K}\mathbf{x} \tag{8.40}$$

where $\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}$

Note that $\dfrac{\partial^2}{\partial \mathbf{u}^2} f\left(\mathbf{u}\right) = \dfrac{\partial}{\partial \mathbf{u}}\left(\mathbf{B}^T\mathbf{P}^T\mathbf{x} + \mathbf{R}\mathbf{u}\right) = \mathbf{R}^T = \mathbf{R}$, a positive definite matrix.

Thus, the second-order sufficiency condition, for $\mathbf{u}^*$ to minimize (8.39), is satisfied.

We now turn our attention to finding an appropriate $\mathbf{P}$. The optimal closed-loop system has the form

$$\dot{\mathbf{x}} = \left(\mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}\right)\mathbf{x}; \quad \mathbf{x}(0) \triangleq \mathbf{x}^0$$

Our optimal controller satisfies the equation

$$\left.\frac{dV}{dt}\right|_{\mathbf{u}=\mathbf{u}^*} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^{*T}\mathbf{R}\mathbf{u}^* = 0$$

that is,

$$\mathbf{x}^T\mathbf{P}\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{P}\mathbf{B}\mathbf{u}^* + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^{*T}\mathbf{R}\mathbf{u}^* = 0$$

We substitute $\mathbf{u}^*$ given by (8.40) into the above equation, and represent it as

$$\tfrac{1}{2}\mathbf{x}^T\left(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A}\right)\mathbf{x} - \mathbf{x}^T\mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}\mathbf{x} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}\mathbf{x} = 0$$

Factoring out $\mathbf{x}$ yields

$$\tfrac{1}{2}\mathbf{x}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P})\mathbf{x} = 0$$

The above equation should hold for any $\mathbf{x}$. For this to be true, we have to have

$$\left(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}\right) = \mathbf{0} \tag{8.41}$$

The above equation is referred to as *algebraic Riccati equation*. In conclusion, the synthesis of the optimal linear state-feedback controller, minimizing the performance index

$$J = \tfrac{1}{2}\int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt$$

subject to

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}; \mathbf{x}(0) \triangleq \mathbf{x}^0$$

requires solving the *matrix Riccati equation* given by (8.41).

Controllability of ($\mathbf{A}$, $\mathbf{B}$) pair and positive definiteness of $\mathbf{Q}$ are sufficient conditions for the existence of asymptotically stable optimal solution to the control problem. This implies that there is a Lyapunov function $V = \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x}$ for the closed-loop system (8.37); that is, for positive definite matrix $\mathbf{P}$, the time derivative evaluated on the trajectories of the closed-loop system,

$$\begin{aligned}\frac{dV}{dt} = \dot{V} &= -\tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \tfrac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u} \\ &= -\tfrac{1}{2}\mathbf{x}^T(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x}\end{aligned} \tag{8.42}$$

is always negative definite.

We now study the effect of choosing a positive semidefinite $\mathbf{Q}$ in the performance index J. If $\mathbf{Q}$ is positive semidefinite and in addition the following rank condition is satisfied,

$$\rho \begin{bmatrix} \mathbf{H} \\ \mathbf{HA} \\ \vdots \\ \mathbf{HA}^{n-1} \end{bmatrix} = n; \mathbf{Q} = \mathbf{H}^T\mathbf{H}, \tag{8.43}$$

then $\dot{V}(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{0}$.

We prove this result by contradiction: the rank condition is satisfied but $\dot{V}(\mathbf{x}) = 0$ for some $\mathbf{x} \neq \mathbf{0}$. Substituting $\mathbf{Q} = \mathbf{H}^T\mathbf{H}$ in Eqn. (8.42), we obtain (refer to Eqns (5.6))

$$\dot{V}(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}^T\mathbf{H}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{K}^T\mathbf{R}\mathbf{K}\mathbf{x}) = -\tfrac{1}{2}[\|\mathbf{H}\mathbf{x}\|^2 + \|\mathbf{K}\mathbf{x}\|_{\mathbf{R}}^2]$$

Therefore, $\mathbf{Hx} = \mathbf{0}$ and $\mathbf{Kx} = \mathbf{0}$ should be simultaneously satisfied. $\mathbf{Kx} = \mathbf{0}$ reduces the closed-loop system (8.37) to the open-loop system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

From the condition

$$\mathbf{H}\mathbf{x} = \mathbf{0}$$

we obtain

$$\mathbf{H}\dot{\mathbf{x}} = \mathbf{H}\mathbf{A}\mathbf{x} = \mathbf{0}$$

Continuing the process of taking derivative, we get

$$\begin{aligned} \mathbf{H}\mathbf{A}^2\mathbf{x} &= \mathbf{0} \\ &\vdots \\ \mathbf{H}\mathbf{A}^{n-1}\mathbf{x} &= \mathbf{0} \end{aligned}$$

or

$$\begin{bmatrix} \mathbf{H} \\ \mathbf{HA} \\ \vdots \\ \mathbf{HA}^{n-1} \end{bmatrix} \mathbf{x} = \mathbf{V}\mathbf{x} = \mathbf{0}$$

Since $\rho[\mathbf{V}] = n$, $\mathbf{Vx} = \mathbf{0}$ only when $\mathbf{x} = \mathbf{0}$. This proves the result.

Let us see an alternative interpretation of the rank condition (8.43). The rank condition implies that the system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

with the 'auxiliary output'

$$\boldsymbol{\zeta}(t) = \mathbf{H}\mathbf{x}(t)$$

is completely observable. Since the performance index

$$J = \tfrac{1}{2}\int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})dt = \tfrac{1}{2}\int_0^\infty (\mathbf{x}^T\mathbf{H}^T\mathbf{H}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})\,dt$$

$$= \tfrac{1}{2}\int_0^\infty (\boldsymbol{\zeta}^T\boldsymbol{\zeta} + \mathbf{u}^T\mathbf{R}\mathbf{u})\,dt,$$

the observability of the pair (**A**, **H**) implies that all the modes of the state trajectories are reflected in the performance index. A finite value of J, therefore, ensures that unstable modes (if any) have been stabilized[4] by the control $\mathbf{u} = -\mathbf{Kx}$.

The observability condition is always satisfied when the matrix **Q** is positive definite.

The design steps may now be stated as follows:

(i) Solve the matrix Riccati equation (8.41) for the positive definite matrix **P**.
(ii) Substitute this matrix **P** into Eqn. (8.40); the resulting equation gives optimal control law.

This is a basic and well-known result in the theory of optimal control. Once the designer has specified **Q** and **R**, representing his/her assessment of the relative importance of various terms in the performance index, the solution of Eqn. (8.41) specifies the optimal control law (8.40). This yields the optimal closed-loop system. If the resulting transient response is unsatisfactory, the designer may alter the weighting matrices **Q** and **R**, and try again.

Comments

(i) The matrix **R** has been assumed to be positive definite. This is a necessary condition for the existence of the optimal solution to the control problem, as seen from Eqn. (8.40).
(ii) We have assumed that the plant (8.34) is completely controllable, and the matrix **Q** in performance index J, given by Eqn. (8.35), is positive definite. These are sufficient conditions for the existence of asymptotically stable optimal solution to the control problem. The requirement on matrix **Q**, may be relaxed to a positive semidefinite matrix with the pair (**A**,**H**) completely observable, where $\mathbf{Q} = \mathbf{H}^T\mathbf{H}$.
(iii) It is important to be able to find out whether the sought-after solution exists or not, before we start working out the solution. This is possible only if necessary conditions for the existence of asymptotically stable optimal solution are established. A discussion on this subject entails not only controllability and observability, but also the concepts of *stabilizability* and *detectability*. Basic ideas about these concepts have been given in footnotes of this chapter; a detailed discussion is beyond the scope of this book.
(iv) Equation (8.41) is a set of n^2 nonlinear algebraic equations. Since **P** is a symmetric matrix, we need to solve only $\frac{n(n+1)}{2}$ equations.
(v) The solution of Eqn. (8.41) is not unique. Of the several possible solutions, the desired answer is obtained by enforcing the requirement that **P** be positive definite. The positive definite solution of Eqn. (8.41) is unique.
(vi) In very simple cases, the Riccati equation can be solved analytically, but usually a numerical solution is required. A number of computer programs for the purpose are available [152–154]. Appendix A provides some MATLAB support.

---

[4] Observability canonical form for a state model which is not completely observable, is given in Eqn. (5.124c). It decomposes the model into two parts: the observable part and the unobservable part. If the unobservable part is stable, then the model is said to be *detectable*.

In the optimization problem under consideration, the observability of the pair (**A**, **H**) is not a necessary condition for the existence of a stable solution. If the pair (**A**, **H**) is detectable, then the modes of state trajectories, not reflected in J, are stable and a finite value of J will ensure asymptotic stability of the closed-loop system.

(vii) Note that the optimal state regulator requires that all the parameters of matrix **K** in the control law (8.36), are free parameters. However, all the parameters of matrix **K** may not be available for adjustments. The gain elements of matrix **K**, corresponding to inaccessible state variables, may be constrained to zero value (otherwise, a state observer will be required). Also, some gains may be fixed by physical constraints of the system. This leads to the parameter optimization problem: optimization of free parameters in the matrix **K**. The difference between parameter optimization and optimal control problems is that no constraint on controllers is imposed on the latter. A solution to parameter optimization (suboptimal control) problem will be given in Section 8.9.

Example 8.5

Consider the problem of attitude control of a rigid satellite which was discussed in Example 7.1. An attitude control system for the satellite that utilizes rate feedback is shown in Fig. 8.11; $\theta(t)$ is the actual attitude, $\theta_r(t)$ is the reference attitude which is a step function, and $u(t)$ is the torque developed by the thrusters.

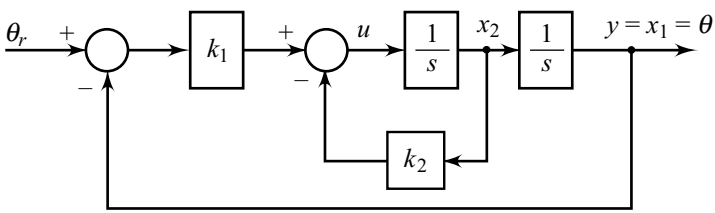


Fig. 8.11 Attitude control of a satellite

The state variable model of the system is

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u \\ y &= \mathbf{C}\mathbf{x} \end{aligned} \tag{8.44}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \ \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \ \mathbf{C} = [1 \quad 0]$$

The problem is to obtain the optimal control law

$$u = -k_1(x_1 - \theta_r) - k_2 x_2; \quad x_1 = \theta, x_2 = \dot{\theta}$$

that minimizes the performance index

$$J = \int_0^\infty [(\theta_r - \theta)^2 + u^2]\, dt \tag{8.45}$$

In terms of the shifted state variables

$$\tilde{x}_1 = x_1 - \theta_r; \ \tilde{x}_2 = x_2$$

the state equation becomes

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}u \tag{8.46}$$

where **A** and **B** are given by Eqns (8.44).

Now, the problem is to find optimal values of the parameters k_1 and k_2, such that the control law

$$u = -k_1 \tilde{x}_1 - k_2 \tilde{x}_2$$

minimizes the performance index

$$J = \int_0^\infty (\tilde{x}_1^2 + u^2)\,dt \tag{8.47}$$

The **Q** and **R** matrices are

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}; \mathbf{R} = 2$$

Note that **Q** is positive semidefinite matrix.

$$\mathbf{Q} = \mathbf{H}^T\mathbf{H} = \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix} [\sqrt{2} \quad 0]$$

The pair (**A**,**H**) is completely observable. Also, the pair (**A**,**B**) is completely controllable. Therefore, sufficient conditions for the existence of asymptotically stable optimal solution are satisfied.

The matrix Riccati equation is

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0}$$

or

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$- \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}[\tfrac{1}{2}][0 \quad 1]\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Upon simplification, we get

$$\frac{-p_{12}^2}{2} + 2 = 0$$

$$p_{11} - \frac{p_{12}p_{22}}{2} = 0$$

$$\frac{-p_{22}^2}{2} + 2p_{12} = 0$$

Solving these three simultaneous equations for p_{11}, p_{12}, and p_{22}, requiring **P** to be positive definite, we obtain

$$\mathbf{P} = \begin{bmatrix} 2\sqrt{2} & 2 \\ 2 & 2\sqrt{2} \end{bmatrix}$$

The optimal control law is given by

$$u = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}\,\tilde{\mathbf{x}}(t) = -\left[\tfrac{1}{2}\right]\ [0 \quad 1]\begin{bmatrix} 2\sqrt{2} & 2 \\ 2 & 2\sqrt{2} \end{bmatrix}\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}$$

$$= -\,\tilde{x}_1(t) - \sqrt{2}\,\tilde{x}_2(t) = -(x_1 - \theta_r) - \sqrt{2}\,x_2$$

It can easily be verified that the closed-loop system is asymptotically stable.

Example 8.6

Consider the liquid-level system of Fig. 8.6. In Example 8.4, we designed an optimal regulator for this process by direct parameter optimization. In the following, we use the Riccati equation for designing the optimal regulator. The state equation of the process is

$$\frac{dy}{dt} = -y + u \tag{8.48}$$

where

y = deviation of the liquid head from the steady state; and

u = rate of liquid inflow.

The performance index

$$J = \int_0^\infty (y^2 + u^2)\,dt$$

For this design problem,

$$A = -1,\ B = 1,\ Q = 2,\ R = 2$$

The Riccati equation is

$$A^TP + PA - PBR^{-1}B^TP + Q = 0$$

or

$$-P - P - \frac{P^2}{2} + 2 = 0$$

Solving for P, requiring it to be positive definite, we get

$$P = 2\left(\sqrt{2} - 1\right)$$

The optimal control law is

$$u = -R^{-1}B^TPy(t) = -\left(\sqrt{2} - 1\right)\,y(t)$$

Substituting in Eqn. (8.48), we get the following equation for the closed-loop system:

$$\frac{dy}{dt} = -\sqrt{2}\,y$$

Obviously, $y(t) \to 0$ for any initial displacement $y(0)$.

Assume now that a constant disturbance due to the pump enters the system as shown in Fig. 8.12. This Type-0 regulator system cannot reject the disturbance; there will be a steady-state offset in the liquid head y. Let us introduce integral control to eliminate this offset.

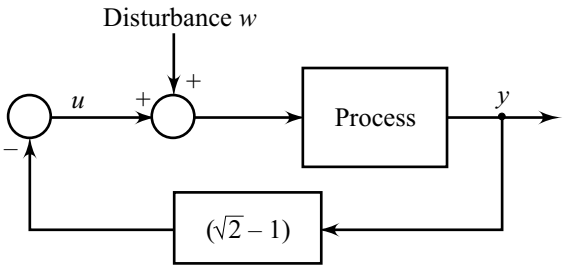


Fig. 8.12 Optimal control of the liquid-level process shown in Fig. 8.6

Defining the integral state z by

$$\dot{z}(t) = y(t),$$

we get the following augmented system:

$$\begin{bmatrix} \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \tag{8.49}$$

Now the design problem is to obtain the control law

$$u = -k_1 y(t) - k_2 z(t)$$

that minimizes

$$J = \int_0^\infty (y^2 + u^2)\, dt$$

The **Q** and **R** matrices are

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{R} = 2$$

The state equation (8.49) is completely controllable, satisfying one of the sufficient conditions for the existence of the optimal solution.

The matrix **Q** is positive semidefinite;

$$\mathbf{Q} = \mathbf{H}^T\mathbf{H} = \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix} [\sqrt{2} \quad 0]$$

The pair $\left(\begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} \sqrt{2} & 0 \end{bmatrix} \right)$ is not completely observable. Therefore, the other sufficient condition for the existence of the asymptotically stable optimal solution is not satisfied. It can easily be verified that a positive definite solution to the matrix Riccati equation does not exist in this case; the chosen matrix **Q** cannot give a closed-loop stable optimal system.

We now modify the performance index to the following:

$$J = \int_0^\infty (y^2 + z^2 + u^2)\, dt$$

The **Q** and **R** matrices are

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \mathbf{R} = 2$$

Since **Q** is positive definite matrix, the asymptotically stable optimal solution exists.

The matrix Riccati equation is

$$\mathbf{A}^T\mathbf{P} + \mathbf{PA} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0}$$

or

$$\begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}$$

$$- \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} [\tfrac{1}{2}] \; [1 \quad 0] \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

From this equation, we obtain the following three simultaneous equations:

$$-2p_{11} + 2p_{12} - \tfrac{1}{2}p_{11}^2 + 2 = 0$$

$$-p_{12} + p_{22} - \frac{p_{11}\,p_{12}}{2} = 0$$

$$\frac{-p_{12}^2}{2} + 2 = 0$$

Solving for p_{11}, p_{12} and p_{22}, requiring **P** to be positive definite, we obtain

$$\mathbf{P} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$$

The gain matrix

$$\mathbf{K} = [k_1 \quad k_2] = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} = [\tfrac{1}{2}] \; [1 \quad 0] \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} = [1 \quad 1]$$

Therefore,

$$u = -y(t) - z(t) = -y(t) - \int_0^\infty y(t)dt$$

The block diagram of Fig. 8.13 shows the configuration of the optimal control system employing state-feedback and integral control. It is a Type-1 regulator system.

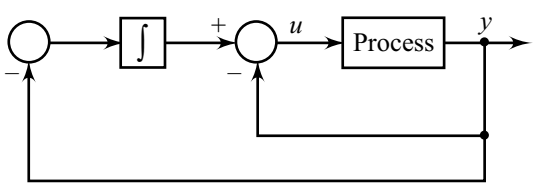


Fig. 8.13 Optimal control system employing state-feedback and integral control

Since at steady state

$$\dot{z}(t) \to 0,$$

therefore,

$$y(t) \to 0,$$

and there will be no steady-state offset in the liquid head y, even in the presence of constant disturbances acting on the plant.

8.8 OPTIMAL DIGITAL CONTROL SYSTEMS

This section covers the key results on the design of optimal controllers for discrete-time systems. Our discussion will be brief because of the strong analogy between the discrete-time and continuous-time cases.

Consider the discretized model of the given plant:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned} \tag{8.50}$$

where $\mathbf{x}$ is the $n \times 1$ state vector, $\mathbf{u}$ is the $p \times 1$ input vector, $\mathbf{y}$ is the $q \times 1$ output vector; $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{C}$ are, respectively, $n \times n$, $n \times p$, and $q \times n$ real constant matrices; and $k = 0, 1, 2, \ldots$. We will assume that the null state $\mathbf{x} = \mathbf{0}$ is the desired steady state; $\mathbf{x}(k)$ is, thus, the system-error vector at $t = kT$, where T is the sampling interval. The state variable model (8.50) is assumed to be completely controllable and observable.

We shall be interested in selecting the controls $\mathbf{u}(k)$; $k = 0, 1, \ldots$, which minimize a performance index of the form

$$J = \tfrac{1}{2} \sum_{k=0}^{\infty} [\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)] \tag{8.51}$$

where $\mathbf{Q}$ is an $n \times n$ positive definite, real, symmetric, constant matrix (or a positive semidefinite, real, symmetric, constant matrix with the restriction that the pair $(\mathbf{F},\mathbf{H})$ is observable, where $\mathbf{H}^T\mathbf{H} = \mathbf{Q}$), and $\mathbf{R}$ is a $p \times p$ positive definite, real, symmetric, constant matrix. This criterion is the discrete analog of that given by Eqn. (8.25); a summation replaces integration.

An important feature (proved later in this section) of this class of problems is that optimal control is possible by feedback control law of the form (refer to Fig. 8.14)

$$\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k) \tag{8.52}$$

where $\mathbf{K}$ is a $p \times n$ constant matrix. If the unknown elements of matrix $\mathbf{K}$ are determined so as to minimize the performance index given by Eqn. (8.51), the control law given by Eqn. (8.52) is optimal.

The control problem stated above, as we know, is a state regulator design problem. The equations developed below for the state regulator design, can also be used for servo design by an appropriate transformation of the state variables (refer to Section 7.9 for details). State regulator design equations can also be used for state-feedback control schemes with integral control. This is done by the augmentation of the plant state with integral state and appropriate transformation of the state variables (refer to Section 7.9 for details).

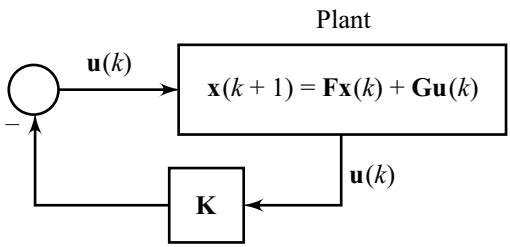


Fig. 8.14 Control configuration of the state regulator for discrete-time systems

In the following, we develop state regulator design equations through Lyapunov synthesis.

With the linear feedback control law (8.52), the closed-loop system is described by

$$\mathbf{x}(k+1) = (\mathbf{F} - \mathbf{G}\mathbf{K})\mathbf{x}(k) \tag{8.53}$$

We will assume that a matrix **K** exists such that (**F** – **GK**) is a stable matrix. The controllability of the model (8.50) is sufficient to ensure this. This implies that there exists a Lyapunov function $V(\mathbf{x}(k)) = \frac{1}{2}\mathbf{x}^T(k)\,\mathbf{P}\mathbf{x}(k)$ for the closed-loop system (8.53). Therefore, the first forward difference, $\Delta V(\mathbf{x}(k)) = V(\mathbf{x}(k+1)) - V(\mathbf{x}(k))$, evaluated on the trajectories of the closed-loop system, is negative definite. We now state and prove a condition for $\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k)$ to be optimal [35].

Theorem 8.6 If the state-feedback controller $\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k)$ is such that it minimizes the function

$$f(\mathbf{u}) = \Delta V\left(\mathbf{x}(k)\right) + \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \tfrac{1}{2}\mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) \tag{8.54}$$

and the minimum value of $f(\mathbf{u}) = 0$ for some $V = \frac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)$, then the controller is optimal.

Proof We can represent (8.54) as (denoting minimizing **u** by $\mathbf{u}^*$)

$$\Delta V\left(\mathbf{x}(k)\right)\Big|_{\mathbf{u}=\mathbf{u}^*} + \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \tfrac{1}{2}\mathbf{u}^{*T}(k)\mathbf{R}\mathbf{u}^*(k) = 0$$

Hence

$$V\left(\mathbf{x}(k+1)\right) - V\left(\mathbf{x}(k)\right)\Big|_{\mathbf{u}=\mathbf{u}^*} = -\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) - \tfrac{1}{2}\mathbf{u}^{*T}(k)\mathbf{R}\mathbf{u}^*(k)$$

We sum both sides from $k = 0$ to ∞ to get

$$V\left(\mathbf{x}(\infty)\right) - V\left(\mathbf{x}(0)\right) = -\tfrac{1}{2}\sum_{k=0}^{\infty}\left(\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^{*T}(k)\mathbf{R}\mathbf{u}^*(k)\right)$$

Because, by assumption, the closed-loop system is stable, we have $\mathbf{x}(\infty) = \mathbf{0}$. Hence

$$V\left(\mathbf{x}(0)\right) = \tfrac{1}{2}\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0) = \tfrac{1}{2}\sum_{k=0}^{\infty}\left(\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^{*T}(k)\mathbf{R}\mathbf{u}^*(k)\right)$$

Thus, if a linear stabilizing controller satisfies the hypothesis of the theorem, then the value of the performance index (8.51) resulting from applying the controller is

$$J\left(\mathbf{u}^*\right) = \tfrac{1}{2}\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0)$$

Since $\mathbf{u}^*$ minimizes the function in (8.54) and the minimum value is zero, for any **u** different from $\mathbf{u}^*$, the value of the function will be greater than/equal to zero.

$$\Delta V\left(\mathbf{x}(k)\right)\Big|_{\mathbf{u}=\hat{\mathbf{u}}} + \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \tfrac{1}{2}\hat{\mathbf{u}}^T(k)\mathbf{R}\hat{\mathbf{u}}(k) \geq 0$$

or,

$$\Delta V\left(\mathbf{x}(k)\right)\Big|_{\mathbf{u}=\hat{\mathbf{u}}} \geq -\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) - \tfrac{1}{2}\hat{\mathbf{u}}^T(k)\mathbf{R}\hat{\mathbf{u}}(k)$$

Summing the above from $k = 0$ to ∞, yields

$$V\left(\mathbf{x}(0)\right) \leq \tfrac{1}{2}\sum_{k=0}^{\infty}\left(\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \tfrac{1}{2}\hat{\mathbf{u}}^T(k)\mathbf{R}\hat{\mathbf{u}}(k)\right);$$

that is,

$$J\left(\mathbf{u}^*\right) \le J\left(\mathbf{u}\right)$$

for any $\hat{\mathbf{u}} \neq \mathbf{u}^*$. Therefore, the controller $\mathbf{u}^*$ is optimal.

Finding an optimal controller involves finding an appropriate quadratic Lyapunov function $V(\mathbf{x}(k)) = \frac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)$, which is used to construct the optimal controller. We first find $\mathbf{u}^*$ that minimizes the function

$$\begin{aligned} f\left(\mathbf{u}(k)\right) &= \Delta V\left(\mathbf{x}(k)\right) + \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \tfrac{1}{2}\mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) \\ &= \tfrac{1}{2}\mathbf{x}^T(k+1)\mathbf{P}\mathbf{x}(k+1) - \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k) + \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \tfrac{1}{2}\mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) \\ &= \tfrac{1}{2}\left(\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)\right)^T \mathbf{P}\left(\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)\right) \\ &\quad - \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k) + \tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \tfrac{1}{2}\mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) \end{aligned} \tag{8.55}$$

Necessary condition for unconstrained minimization is $\left(\dfrac{\partial}{\partial \mathbf{x}}(\mathbf{f}^T\mathbf{g}) = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{g} + \dfrac{\partial \mathbf{g}}{\partial \mathbf{x}}\mathbf{f}\right)$

$$\begin{aligned} \frac{\partial f(\mathbf{u}(k))}{\partial \mathbf{u}(k)} &= \tfrac{1}{2}\frac{\partial (\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k))}{\partial \mathbf{u}(k)}[\mathbf{P}(\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k))] \\ &\quad + \tfrac{1}{2}\frac{\partial (\mathbf{P}(\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)))}{\partial \mathbf{u}(k)}[\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)] + \tfrac{1}{2}\frac{\partial (\mathbf{u}^T(k)\,\mathbf{R}\mathbf{u}(k))}{\partial \mathbf{u}(k)} \\ &= \tfrac{1}{2}\mathbf{G}^T\mathbf{P}(\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)) + \tfrac{1}{2}\mathbf{G}^T\mathbf{P}(\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)) + \mathbf{R}\mathbf{u}(k) \\ &= \mathbf{G}^T\mathbf{P}(\mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)) + \mathbf{R}\mathbf{u}(k) \\ &= \mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k) + (\mathbf{R} + \mathbf{G}^T\mathbf{P}\mathbf{G})\mathbf{u}(k) = \mathbf{0} \end{aligned}$$

The matrix $\mathbf{R} + \mathbf{G}^T\mathbf{P}\mathbf{G}$ is positive definite, and therefore, it is invertible. Hence

$$\mathbf{u}^*(k) = -\left(\mathbf{R} + \mathbf{G}^T\mathbf{P}\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k) = -\mathbf{K}\mathbf{x}(k) \tag{8.56}$$

where

$$\mathbf{K} = \left(\mathbf{R} + \mathbf{G}^T\mathbf{P}\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}$$

We next check if $\mathbf{u}^*$ satisfies the second-order sufficient condition for a relative minimizer of the function (8.55).

$$\begin{aligned} \frac{\partial^2 f(\mathbf{u}(k))}{\partial \mathbf{u}^2(k)} &= \frac{\partial\left[\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)\left(\mathbf{R} + \mathbf{G}^T\mathbf{P}\mathbf{G}\right)\mathbf{u}(k)\right]}{\partial \mathbf{u}(k)} \\ &= \mathbf{R} + \mathbf{G}^T\mathbf{P}\mathbf{G}\text{; a positive-definite matrix} \end{aligned}$$

that is, $\mathbf{u}^*$ satisfies the second-order sufficient condition for a relative minimizer of the function (8.55).

The optimal controller (8.56) can be constructed if we have found an appropriate positive definite matrix $\mathbf{P}$. Our next task is to devise a method that would allow us to compute the desired matrix $\mathbf{P}$. For this, we first find the equation describing the closed-loop system driven by the optimal controller (8.56):

$$\mathbf{x}(k+1) = \left(\mathbf{F} - \mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\right)\mathbf{x}(k) \tag{8.57}$$

where $\mathbf{S} = \mathbf{R} + \mathbf{G}^T\mathbf{P}\mathbf{G}$

Our controller satisfies the hypothesis of Theorem (8.6), that is

$$\tfrac{1}{2}\mathbf{x}^T(k+1)\mathbf{P}\mathbf{x}(k+1)-\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)+\tfrac{1}{2}\mathbf{u}^{*T}(k)\mathbf{R}\mathbf{u}^*(k)=\mathbf{0}$$

or,

$$\tfrac{1}{2}\mathbf{x}^T(k)[\mathbf{F}-\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}]^T\mathbf{P}[\mathbf{F}-\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}]\mathbf{x}(k)-\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)$$
$$+\tfrac{1}{2}\mathbf{x}^T(k)[\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}]\mathbf{x}(k)=0$$

or,

$$\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)-\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)-\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)$$
$$+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)-\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)$$
$$+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{R}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)=0$$

or,

$$\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)-\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)-\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)$$
$$+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}(\mathbf{R}+\mathbf{G}^T\mathbf{P}\mathbf{G})\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)=0$$

or,

$$\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)-\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)-\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)$$
$$+\tfrac{1}{2}\mathbf{x}^T(k)\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}\mathbf{x}(k)=0$$

or,

$$\tfrac{1}{2}[\mathbf{x}^T(k)[\mathbf{F}^T\mathbf{P}\mathbf{F}-\mathbf{P}+\mathbf{Q}-\mathbf{F}^T\mathbf{P}\mathbf{G}\mathbf{S}^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}]\mathbf{x}(k)]=0$$

The above equation should hold for any $\mathbf{x}$. For this to be true we have to have

$$\mathbf{F}^T\mathbf{P}\mathbf{F}-\mathbf{P}+\mathbf{Q}-\mathbf{F}^T\mathbf{P}\mathbf{G}\left(\mathbf{R}+\mathbf{G}^T\mathbf{P}\mathbf{G}\right)^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}=\mathbf{0} \tag{8.58}$$

This equation is called the *discrete algebraic Riccati equation*.

The discrete matrix Riccati equation given in (8.58), is one of the many equivalent forms which satisfy the optimal regulator design. The analytical solution of the discrete Riccati equation is possible only for very simple cases. A number of computer programs for the solution of the discrete Riccati equation are available [152–154]. Appendix A provides some MATLAB support.

Example 8.7

Consider the problem of digital control of a plant described by the transfer function

$$G(s)=\frac{1}{s+1}$$

Discretization of the plant model gives

$$G_{h0}G(z)=\frac{Y(z)}{U(z)}=\mathscr{Z}\left[\left(\frac{1-e^{-sT}}{s}\right)\left(\frac{1}{s+1}\right)\right]=(1-z^{-1})\,\mathscr{Z}\left[\frac{1}{s(s+1)}\right]=\frac{1-e^{-T}}{z-e^{-T}}$$

For a sampling interval $T=1$ sec,

$$G_{h0}G(z)=\frac{0.632}{z-0.368}$$

The difference equation model of the plant is

$$y(k+1) = 0.368\, y(k) + 0.632\, u(k) \tag{8.59}$$

The design specifications are given below.

(i) Minimize

$$J = \tfrac{1}{2} \sum_{k=0}^{\infty} [\tilde{y}^2(k) + \tilde{u}^2(k)]$$

where $\tilde{y}$ and $\tilde{u}$ are, respectively, the deviations of the output and the control signal from their steady-state values.

(ii) For a constant y_r, $y(\infty) = y_r$, i.e., there is zero steady-state error.

For this design problem, we select the feedback plus feedforward control scheme shown in Fig. 8.15. The feedback gain K is obtained from the solution of the shifted regulator problem, as is seen below.

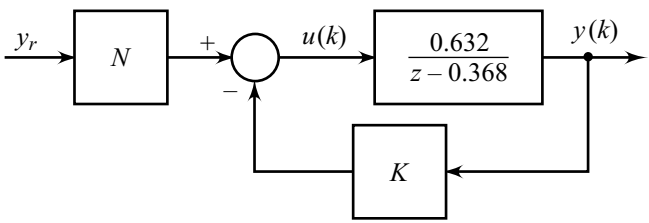


Fig. 8.15 Feedback plus feedforward control scheme

Let y_s and u_s be the steady-state values of the output and the control signal, respectively. Equation (8.59) at steady state becomes

$$y_s = 0.368\, y_s + 0.632\, u_s$$

The state equation (8.59) may equivalently be expressed as

$$\tilde{y}(k+1) = 0.368\ \tilde{y}(k) + 0.632\ \tilde{u}(k)$$

where

$$\tilde{y} = y - y_s;\ \tilde{u} = u - u_s$$

In terms of this equivalent formulation, the optimal control problem is to obtain

$$\tilde{u}(k) = -K\, \tilde{y}(k)$$

so that

$$J = \tfrac{1}{2} \sum_{k=0}^{\infty} [\tilde{y}^2(k) + \tilde{u}^2(k)]$$

is minimized.

For this shifted regulator problem,

$$F = 0.368,\ G = 0.632,\ Q = 1,\ R = 1$$

The Riccati equation (8.58) gives

$$\begin{aligned} P &= Q + F^T PF - F^T PG(R + G^T PG)^{-1} G^T PF \\ &= 1 + (0.368)^2 P - (0.368)P(0.632)[1 + (0.632)^2 P]^{-1}(0.632)P(0.368) \\ &= 1 + 0.135P - \frac{0.054\,P^2}{1+0.4\,P} \end{aligned}$$

Solving for P, requiring it to be positive definite, we obtain,

$$P = 1.11$$

Feedback gain (refer to Eqn. (8.56))

$$K = (R + G^T PG)^{-1} G^T PF = 0.18$$

The feedforward gain (refer to Eqn. (7.105)) N, is given by

$$\frac{1}{N} = -\,C(F - GK - I)^{-1} G = -\,[0.368 - 0.632\,(0.18) - 1]^{-1}\,(0.632)$$

or

$$N = \frac{0.746}{0.632} = 1.18$$

The optimal control sequence (refer to Eqn. (7.106))

$$u(k) = -\,Ky(k) + Ny_r = -\,0.18\,y(k) + 1.18\,y_r$$

Substituting in Eqn. (8.59), we obtain

$$y(k+1) = 0.254\,y(k) + 0.746\,y_r$$

At steady state,

$$y(\infty) = y_s = \frac{0.746}{0.746}\,y_r = y_r$$

8.9 CONSTRAINED STATE FEEDBACK CONTROL

In classical control theory, as we have seen in Chapters 1–4, output feedback controller is the common control structure. These controllers are designed in the context of an input-output transfer function model. When we move to a state variable model for the system, we have time-domain information about the internal structure of the system, available in the form of a state vector. To investigate a link between the state information and output feedback, we set up an output feedback regulator system.

The system equations are

$$\begin{aligned} \dot{\mathbf{x}} &= \underset{(n\times n)}{\mathbf{A}}\ \underset{(n\times 1)}{\mathbf{x}} + \underset{(p\times 1)}{\mathbf{B}}\ \underset{(p\times 1)}{\mathbf{u}}\ ;\mathbf{x}(0) \triangleq \mathbf{x}^0 \\ \underset{(q\times 1)}{\mathbf{y}} &= \mathbf{C}\mathbf{x} \end{aligned} \tag{8.60}$$

and the output feedback control law is

$$\mathbf{u}(t) = -\mathbf{K}_0\ \mathbf{y}(t) \tag{8.61}$$

If we now progress this analysis for the closed-loop system, we substitute for the control law in system equations. This gives

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\mathbf{K}_0\mathbf{C})\mathbf{x} \tag{8.62}$$

The design problem can be stated through the closed-loop characteristic polynomial $\Delta(s)$ that specifies the poles of the closed-loop system (eigenvalues of the matrix $(\mathbf{A} - \mathbf{B}\mathbf{K}_0\mathbf{C})$):

$$\Delta(s) = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n$$

The problem is to choose the available controller gain matrix $\mathbf{K}_0$ so that the specified characteristic polynomial $\Delta(s)$ equals the characteristic polynomial of the matrix $(\mathbf{A} - \mathbf{B}\mathbf{K}_0\mathbf{C})$:

$$s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n = \left| s\mathbf{I} - \left(\mathbf{A} - \mathbf{B}\mathbf{K}_0\mathbf{C}\right)\right| \tag{8.63}$$

The output feedback in a state variable framework does not necessarily have sufficient degrees of freedom to satisfy this design requirement. The output feedback gain $\mathbf{K}_0$ will have $p \times q$ parameters to tune n coefficients of the closed-loop characteristic polynomial. In most real systems, the order of the system n will be very much greater than the number of measurements q and/or control p. The important issue here is that the output vector is only a partial view of the state vector.

With full state feedback, a closed-form solution to the pole-placement problem exists under mildly restrictive condition on controllability of the system. In case of single-input systems, the solution is unique (given earlier in Chapter 7). In case of multi-input systems, the solution is not unique. In fact, there is lot of freedom available in the choice of state-feedback gain matrix; this freedom is utilized to serve other objectives on system performance [105].

With partial state feedback (output feedback), a closed-form solution to the pole-placement problem may not exist. The designer often solves the problem numerically; tuning the output feedback gain matrix by trial and error to obtain approximate pole-placement solution and checking the acceptability of the approximation by simulation.

The output feedback law is restricted in design achievements, while the state-feedback law is able to give total control over system dynamics. In fact, as we have seen, the design flexibility of the state-feedback law is supported by deep technical results to guarantee the design properties. However, this design flexibility of state feedback is achieved because it has been assumed that we can access each state variable. In practice, this means that we must have a more complicated system where we include an observer which provides us with the state information. Since an observer incorporates model of the process, dependence on very accurate representation of the process being controlled for accurate state information, is obvious. We know that industrial process models are not usually so well known or accurate. Therefore, inclusion of an observer is bound to deteriorate the robustness properties of the feedback system.

Therefore, in spite of excellent design flexibility in state feedback, we are, many a times, forced to look at the alternatives, not so excellent in terms of design flexibility but not dependent on inclusion of an observer. Constrained state feedback is an interesting alternative. Here, we set the gains of the state-feedback gain matrix corresponding to unmeasurable states, to zero and try to exploit the rest of the gains to achieve the desired properties of the closed-loop system. This, in fact, may also be viewed as an output feedback problem where the state $\mathbf{x}$ passes through an appropriately selected output matrix $\mathbf{C}$ to give the output variables in the vector $\mathbf{y}$.

The constrained state-feedback control law is not supported by deep technical results and does not guarantee the design properties; nonetheless, it yields robust feedback control systems for many industrial control problems. Existence of a control law that stabilizes the feedback system is a pre-requisite of the design algorithm. Unfortunately, general conclusions for existence of a stabilizing control law with constrained state feedback cannot be laid down; therefore, one resorts to numerical methods to establish the existence.

In the following, we outline a procedure for the design of constrained state-feedback control law that minimizes a quadratic performance index. By constrained state feedback we mean that all the parameters of the matrix **K** are not available for adjustments. Some of them may have zero values (output feedback). The procedure described below is equally applicable to situations wherein some of the parameters of **K** have fixed values.

Let us consider the system (8.60). It is desired to minimize the following performance index:

$$J = \frac{1}{2}\int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})\, dt \tag{8.64}$$

where **Q** is $n \times n$ positive definite, real, symmetric, constant matrix, and **R** is $p \times p$ positive definite, real, symmetric, constant matrix.

We shall obtain a direct relationship between Lyapunov functions and quadratic performance measures, and solve the constrained parameter-optimization problem using this relationship. We select the feedback control configuration described by the control law

$$\mathbf{u} = -\mathbf{K}_0\mathbf{C}\,\mathbf{x} = -\mathbf{K}\mathbf{x} \tag{8.65}$$

where **K** is $p \times n$ matrix which involves adjustable parameters. With this control law, the closed-loop system becomes

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x} = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} \tag{8.66}$$

All the parameters of the matrix **K** are not available for adjustments. Some of them have fixed values (zero values). We will assume that a matrix **K** satisfying the imposed constraints on its parameters exists such that (**A** – **BK**) is a stable matrix.

The optimization problem is to determine the values of free parameters of the matrix **K** so as to minimize the performance index given by Eqn. (8.64).

Substituting the control vector **u** from Eqn. (8.65) in the performance index J of Eqn. (8.64), we have

$$\begin{aligned} J &= \frac{1}{2}\int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{x}^T\mathbf{K}^T\mathbf{R}\mathbf{K}\mathbf{x})\,dt \\ &= \frac{1}{2}\int_0^\infty \mathbf{x}^T(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x}\,dt \end{aligned} \tag{8.67}$$

Let us assume a Lyapunov function

$$V(\mathbf{x}(t)) = \frac{1}{2}\int_0^\infty \mathbf{x}^T(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x}\,dt$$

Note that the value of the performance index for system trajectory starting at $\mathbf{x}(0)$ is $V(\mathbf{x}(0))$.

The time derivative of the Lyapunov function is

$$\dot{V}(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T(\mathbf{Q}+\mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x}\Big|_t^\infty$$

$$= \tfrac{1}{2}\mathbf{x}^T(\infty)[\mathbf{Q}+\mathbf{K}^T\mathbf{R}\mathbf{K}]\mathbf{x}(\infty) - \tfrac{1}{2}\mathbf{x}^T(t)[\mathbf{Q}+\mathbf{K}^T\mathbf{R}\mathbf{K}]\mathbf{x}(t)$$

Assuming that the matrix $(\mathbf{A} - \mathbf{B}\mathbf{K})$ is stable, we have from Eqn. (8.66),

$$\mathbf{x}(\infty) \to \mathbf{0}$$

Therefore,

$$\dot{V}(\mathbf{x}) = -\tfrac{1}{2}\mathbf{x}^T(\mathbf{Q}+\mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x} \tag{8.68}$$

Since $\dot{V}(\mathbf{x})$ is quadratic in $\mathbf{x}$ and the plant equation is linear, let us assume that $V(\mathbf{x})$ is also given by the quadratic form:

$$V(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} \tag{8.69}$$

where $\mathbf{P}$ is a positive definite real, symmetric, constant matrix. Therefore,

$$\dot{V}(\mathbf{x}) = \tfrac{1}{2}(\dot{\mathbf{x}}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}})$$

Substituting for $\dot{\mathbf{x}}$ from Eqn. (8.66), we get

$$\dot{V}(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T[(\mathbf{A}-\mathbf{B}\mathbf{K})^T\mathbf{P} + \mathbf{P}(\mathbf{A}-\mathbf{B}\mathbf{K})]\mathbf{x}$$

Comparison of this result with Eqn. (8.68) gives

$$\tfrac{1}{2}\mathbf{x}^T[(\mathbf{A}-\mathbf{B}\mathbf{K})^T\mathbf{P} + \mathbf{P}(\mathbf{A}-\mathbf{B}\mathbf{K})]\mathbf{x} = -\tfrac{1}{2}\mathbf{x}^T(\mathbf{Q}+\mathbf{K}^T\mathbf{R}\mathbf{K})\mathbf{x}$$

Since the above equality holds for arbitrary $\mathbf{x}(t)$, we have

$$(\mathbf{A}-\mathbf{B}\mathbf{K})^T\mathbf{P} + \mathbf{P}(\mathbf{A}-\mathbf{B}\mathbf{K}) + \mathbf{K}^T\mathbf{R}\mathbf{K} + \mathbf{Q} = \mathbf{0} \tag{8.70}$$

This equation is of the form of Lyapunov equation defined in Section 8.3. In Eqn. (8.70) we have n^2 nonlinear algebraic equations. However, since $n \times n$ matrix $\mathbf{P}$ is symmetric, we need to solve only $\frac{n(n+1)}{2}$ equations for the elements p_{ij} of the matrix $\mathbf{P}$. The solution will give p_{ij} as functions of the feedback matrix $\mathbf{K}$.

As pointed out earlier, $V(\mathbf{x}(0))$ is the value of the performance index for the system trajectory starting at $\mathbf{x}(0)$. From Eqn. (8.69), we get,

$$J = \tfrac{1}{2}\mathbf{x}^T(0)\,\mathbf{P}\mathbf{x}(0) \tag{8.71}$$

A *suboptimal control law* may be obtained by minimizing J with respect to the available elements k_{ij} of $\mathbf{K}$, i.e., by setting

$$\frac{\partial[\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0)]}{\partial k_{ij}} = 0 \tag{8.72}$$

If for the suboptimal solution thus obtained, the matrix $(\mathbf{A} - \mathbf{B}\mathbf{K})$ is stable, then the minimization of J as per the procedure described above gives the correct result. From Eqn. (8.68) we observe that for a

positive definite **Q**, $\dot{V}(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{0}$ (note that $\mathbf{K}^T\mathbf{RK}$ is non-negative definite). Also Eqn. (8.69) shows that $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq 0$ if **P** is positive definite. Therefore, minimization of J with respect to k_{ij} (Eqn. (8.72)) will lead to a stable closed-loop system if the optimal k_{ij} result in a positive definite matrix **P**.

One would like to examine the existence of a solution to the optimization problem before actually starting the optimization procedure. For the problem under consideration, existence of **K** that minimizes J is ensured if, and only if, there exists a **K** satisfying the imposed constraints on the parameters, such that (**A** – **BK**) is asymptotically stable. The question of existence of **K** has, as yet, no straightforward answer. We resort to numerical trial-and-error to find a stabilizing matrix **K** (such a matrix is required by numerical algorithms for optimization of J [105]). Failure to do so does not mean that a suboptimal solution does not exist.

Also note that the solution is dependent on initial condition (Eqn. (8.72)). If a system is to operate satisfactorily for a range of initial disturbances, it may not be clear which is the most suitable for optimization.

The dependence on initial conditions can be avoided by averaging the performance obtained for a linearly independent set of initial conditions. This is equivalent to assuming the initial state $\mathbf{x}(0)$ to be a random variable, uniformly distributed on the surface of an n-dimensional sphere, i.e.,

$$E\{\mathbf{x}(0)\mathbf{x}^T(0)\} = \mathbf{I}$$

where $E\{.\}$ denotes the expected value.

We define a new performance index

$$\begin{aligned} \hat{J} = E\{J\} &= E\left\{\tfrac{1}{2}\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0)\right\} \\ &= \tfrac{1}{2}E\{trace\ (\mathbf{P}\mathbf{x}(0)\mathbf{x}^T(0))\} \\ &= \tfrac{1}{2}trace\ (\mathbf{P}E\{\mathbf{x}(0)\mathbf{x}^T(0)\}) \\ &= \tfrac{1}{2}trace\ \mathbf{P} \end{aligned} \tag{8.73}$$

Reference [105] describes a numerical algorithm for the minimization of $\hat{J}$. When feedback matrix **K** is unconstrained, the resulting value of J is optimal; J(optimal) < J (suboptimal).

As we have seen earlier in this chapter, the optimal solution is independent of initial conditions. It is computationally convenient to use Riccati equation (8.41) for obtaining optimal control law (8.40).

Consider now the discretized model of the given plant:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k);\ \mathbf{x}(0) \triangleq \mathbf{x}^0 \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned} \tag{8.74}$$

where **x** is the $n \times 1$ state vector, **u** is the $p \times 1$ input vector, **y** is the $q \times 1$ output vector; **F**, **G**, and **C** are, respectively, $n \times n$, $n \times p$, and $q \times n$ real constant matrices; and $k = 0, 1, 2, \ldots$.

We shall be interested in selecting the controls $\mathbf{u}(k)$; $k = 0, 1, \ldots$, which minimize a performance index of the form

$$J = \tfrac{1}{2}\sum_{k=0}^{\infty}[\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)] \tag{8.75}$$

where $\mathbf{Q}$ is an $n \times n$ positive definite, real, symmetric, constant matrix, and $\mathbf{R}$ is a $p \times p$ positive definite, real, symmetric, constant matrix.

The constrained state-feedback control law is

$$\mathbf{u}(k) = -\mathbf{K}_0\mathbf{C}\,\mathbf{x}(k) = -\mathbf{K}\mathbf{x}(k) \tag{8.76}$$

where $\mathbf{K}$ is a $p \times n$ constant matrix.

With the linear feedback control law (8.76), the closed-loop system is described by

$$\mathbf{x}(k+1) = (\mathbf{F} - \mathbf{G}\mathbf{K})\mathbf{x}(k) \tag{8.77}$$

We will assume that a matrix $\mathbf{K}$ exists such that $(\mathbf{F} - \mathbf{G}\mathbf{K})$ is a stable matrix.

Substituting for the control vector $\mathbf{u}(k)$ from Eqn. (8.76) in the performance index J given by Eqn. (8.75), we get

$$J = \tfrac{1}{2}\sum_{k=0}^{\infty} \mathbf{x}^T(k)\,[\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}]\mathbf{x}(k) \tag{8.78}$$

Let us assume a Lyapunov function

$$V(\mathbf{x}(k)) = \tfrac{1}{2}\sum_{i=k}^{\infty} \mathbf{x}^T(i)\,[\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}]\mathbf{x}(i) \tag{8.79}$$

Note that the value of the performance index for system trajectory starting at $\mathbf{x}(0)$ is $V(\mathbf{x}(0))$. The difference

$$V(\mathbf{x}(k+1)) - V(\mathbf{x}(k)) = \Delta V(\mathbf{x}(k)) = -\tfrac{1}{2}\,\mathbf{x}^T(k)[\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}]\mathbf{x}(k) \tag{8.80}$$

(Note that $\mathbf{x}(\infty)$ has been taken as zero under the assumption of asymptotic stability of the closed-loop system).

Since $\Delta V(\mathbf{x}(k))$ is quadratic in $\mathbf{x}(k)$ and the plant equation is linear, let us assume that $V(\mathbf{x}(k))$ is also given by the quadratic form

$$V(\mathbf{x}(k)) = \tfrac{1}{2}\,\mathbf{x}^T(k)\,\mathbf{P}\mathbf{x}(k) \tag{8.81}$$

where $\mathbf{P}$ is a positive definite, real, symmetric, constant matrix.

Therefore,

$$\Delta V(\mathbf{x}(k)) = \tfrac{1}{2}\,\mathbf{x}^T(k+1)\mathbf{P}\mathbf{x}(k+1) - \tfrac{1}{2}\,\mathbf{x}^T(k)\mathbf{P}\mathbf{x}(k)$$

Substituting for $\mathbf{x}(k+1)$ from Eqn. (8.77), we get

$$\Delta V(\mathbf{x}(k)) = \tfrac{1}{2}\,\mathbf{x}^T(k)\,[(\mathbf{F} - \mathbf{G}\mathbf{K})^T\mathbf{P}(\mathbf{F} - \mathbf{G}\mathbf{K}) - \mathbf{P}]\mathbf{x}(k)$$

Comparing this result with Eqn. (8.80), we obtain

$$(\mathbf{F} - \mathbf{G}\mathbf{K})^T\mathbf{P}(\mathbf{F} - \mathbf{G}\mathbf{K}) - \mathbf{P} + \mathbf{K}^T\mathbf{R}\mathbf{K} + \mathbf{Q} = \mathbf{0} \tag{8.82}$$

This equation is of the form of Lyapunov equation defined in Section 8.3.

Since $V(\mathbf{x}(0))$ is the value of the performance index, we have

$$J = \tfrac{1}{2}\,\mathbf{x}^T(0)\mathbf{P}\mathbf{x}(0) \tag{8.83}$$

Assuming $\mathbf{x}(0)$ to be a random variable uniformly distributed on the surface of n-dimensional sphere, the problem reduces to minimization of

$$\hat{J} = \tfrac{1}{2} trace\, \mathbf{P} \tag{8.84}$$

Reference [105] describes a numerical algorithm for this minimization problem.

REVIEW EXAMPLES

Review Example 8.1

Using the Lyapunov equation, determine the stability range for the gain K of the system shown in Fig. 8.16.

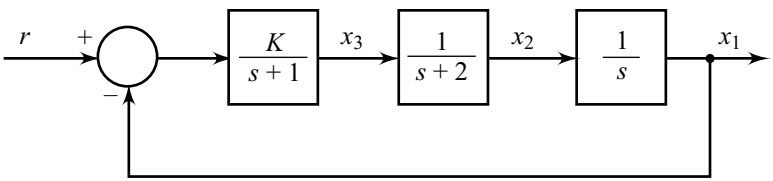


Fig. 8.16

Solution The state equation of the system is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 1 \\ -K & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ K \end{bmatrix} r$$

For the investigation of asymptotic stability, we consider the system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 1 \\ -K & 0 & -1 \end{bmatrix}$$

Clearly, the equilibrium state is the origin.

Let us choose a Lyapunov function

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$$

where $\mathbf{P}$ is to be determined from the Lyapunov equation

$$\mathbf{A}^T \mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q}$$

The matrix **Q** could be chosen as identity matrix. However, we make the following choice for **Q**:

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This is a positive semidefinite matrix which satisfies the condition (8.10) as is seen below.

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} [0 \quad 0 \quad 1] = \mathbf{H}^T\mathbf{H}$$

$$\rho \begin{bmatrix} \mathbf{H} \\ \mathbf{HA} \\ \mathbf{HA}^2 \end{bmatrix} = \rho \begin{bmatrix} 0 & 0 & 1 \\ -K & 0 & -1 \\ K & -K & 1 \end{bmatrix} = 3$$

With this choice of **Q**, as we shall see, manipulation of the Lyapunov equation for its analytical solution becomes easier.

Now let us solve the Lyapunov equation

$$\mathbf{A}^T\mathbf{P} + \mathbf{PA} = -\mathbf{Q}$$

or
$$\begin{bmatrix} 0 & 0 & -K \\ 1 & -2 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{12} & p_{22} & p_{23} \\ p_{13} & p_{23} & p_{33} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{12} & p_{22} & p_{23} \\ p_{13} & p_{23} & p_{33} \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 1 \\ -K & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Solving this equation for p_{ij}'s, we obtain

$$\mathbf{P} = \begin{bmatrix} \dfrac{K^2 + 12K}{12 - 2K} & \dfrac{6K}{12 - 2K} & 0 \\ \dfrac{6K}{12 - 2K} & \dfrac{3K}{12 - 2K} & \dfrac{K}{12 - 2K} \\ 0 & \dfrac{K}{12 - 2K} & \dfrac{6}{12 - 2K} \end{bmatrix}$$

For **P** to be positive definite, it is necessary and sufficient that

$$(12 - 2K) > 0 \text{ and } K > 0 \quad \text{or} \quad 0 < K < 6$$

Thus for $0 < K < 6$, the system is asymptotically stable.

Review Example 8.2

Consider the system described by the equations

$$x_1(k+1) = 2x_1(k) + 0.5x_2(k) - 5$$
$$x_2(k+1) = 0.8x_2(k) + 2$$

Investigate the stability of the equilibrium state using Lyapunov equation.

Solution The equilibrium state $\mathbf{x}^e = \begin{bmatrix} x_1^e \\ x_2^e \end{bmatrix}$ can be determined from the equations

$$x_1^e = 2x_1^e + 0.5x_2^e - 5$$
$$x_2^e = 0.8x_2^e + 2$$

Solving, we get

$$\begin{bmatrix} x_1^e \\ x_2^e \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$$

Define

$$\tilde{x}_1(k) = x_1(k) - x_1^e$$
$$\tilde{x}_2(k) = x_2(k) - x_2^e$$

In terms of the shifted variables, the system equations become

$$\begin{bmatrix} \tilde{x}_1(k+1) \\ \tilde{x}_2(k+1) \end{bmatrix} = \begin{bmatrix} 2 & 0.5 \\ 0 & 0.8 \end{bmatrix} \begin{bmatrix} \tilde{x}_1(k) \\ \tilde{x}_2(k) \end{bmatrix}$$

or

$$\tilde{\mathbf{x}}(k+1) = \mathbf{F}\tilde{\mathbf{x}}(k) \tag{8.85}$$

Clearly $\tilde{\mathbf{x}} = \mathbf{0}$ is the equilibrium state of this autonomous system.

Let us choose a Lyapunov function

$$V(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}^T \mathbf{P} \tilde{\mathbf{x}}$$

where $\mathbf{P}$ is to be determined from the Lyapunov equation

$$\mathbf{F}^T\mathbf{P}\mathbf{F} - \mathbf{P} = -\mathbf{I}$$

or

$$\begin{bmatrix} 2 & 0 \\ 0.5 & 0.8 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 2 & 0.5 \\ 0 & 0.8 \end{bmatrix} - \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Solving for p_{ij}'s, we get

$$\mathbf{P} = \begin{bmatrix} -\frac{1}{3} & \frac{5}{9} \\ \frac{5}{9} & \frac{1225}{324} \end{bmatrix}$$

By applying the *Sylvester's test* for positive definiteness, we find that the matrix $\mathbf{P}$ is not positive definite. Therefore, the origin of the system (8.85) is not asymptotically stable.

In terms of the original state variables, we can say that the equilibrium state

$\mathbf{x}^e = [0 \quad 10]^T$ of the given system is not asymptotically stable.

Review Example 8.3

Referring to the block diagram of Fig. 8.17, consider that $G(s) = 100/s^2$ and $R(s) = 1/s$. Determine the optimal values of parameters k_1 and k_2 such that

$$J = \int_0^\infty [e^2(t) + 0.25u^2(t)]dt$$

is minimized.

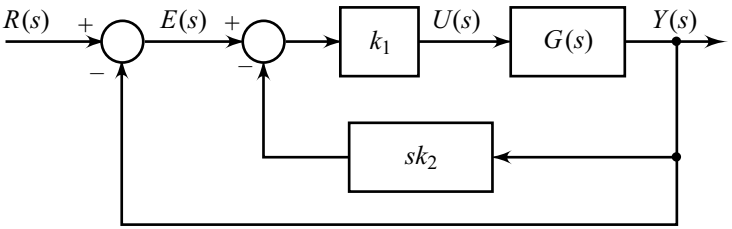


Fig. 8.17

Solution From Fig. 8.17, we obtain

$$\ddot{y}(t) = 100u(t); y(0) = \dot{y}(0) = 0$$
$$u(t) = k_1[r - y(t) - k_2\dot{y}(t)]$$

In terms of the state variables

$$\tilde{x}_1(t) = y(t) - r$$
$$\tilde{x}_2(t) = \dot{y}(t),$$

the state variable model of the system becomes

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} 0 \\ 100 \end{bmatrix} u;\ \tilde{\mathbf{x}}(0) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$
$$u = -k_1\tilde{x}_1 - k_1k_2\tilde{x}_2 = -\mathbf{K}\tilde{\mathbf{x}} \tag{8.86}$$

where

$$\mathbf{K} = [k_1 \quad k_1k_2]$$

The optimization problem is to find **K** such that

$$J = \int_0^\infty (\tilde{x}_1^2 + 0.25u^2)dt$$

is minimized.

Note that

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} 0 \\ 100 \end{bmatrix}; \mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}; \mathbf{R} = 0.5$$

The matrix Riccati equation is

$$\mathbf{A}^T\mathbf{P} + \mathbf{PA} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0}$$

or

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$
$$- \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 100 \end{bmatrix} \left[\tfrac{1}{0.5}\right] [0 \quad 100] \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Solving for p_{11}, p_{12}, and p_{22}, requiring **P** to be positive definite, we obtain

$$\mathbf{P} = \begin{bmatrix} 2\times10^{-1} & 10^{-2} \\ 10^{-2} & 10^{-3} \end{bmatrix}$$

The feedback gain matrix

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}$$

$$= \tfrac{1}{0.5}\ [0 \quad 100] \begin{bmatrix} 2\times10^{-1} & 10^{-2} \\ 10^{-2} & 10^{-3} \end{bmatrix} = [2 \quad 0.2]$$

From Eqn. (8.86), we obtain

$$[k_1 \quad k_1k_2] = [2 \quad 0.2] \quad \text{or} \quad k_1 = 2,\ k_2 = 0.1$$

Review Example 8.4

Figure 8.18 shows the optimal control configuration of a position servo system.

Both the state variables—angular position θ and angular velocity $\dot{\theta}$—are assumed to be measurable.

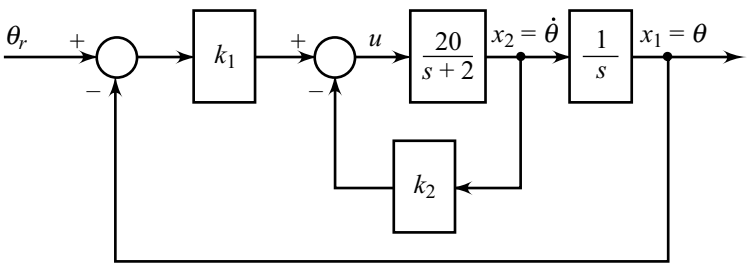


Fig. 8.18 A position servo system

It is desired to regulate the angular position to a unit-step function θ_r. Find the optimum values of the gains k_1 and k_2 that minimize

$$J = \int_0^\infty [(x_1 - \theta_r)^2 + u^2]dt$$

Solution The state variable description of the system, obtained from Fig. 8.18, is given by

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 20 \end{bmatrix} u$$

$$y = x_1$$

In terms of the shifted state variables

$$\tilde{x}_1 = x_1 - \theta_r$$

$$\tilde{x}_2 = x_2,$$

the state variable model becomes

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}\,\tilde{\mathbf{x}} + \mathbf{B}u \tag{8.87}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 20 \end{bmatrix}$$

The design problem is to determine optimal control

$$u = -k_1 \tilde{x}_1 - k_2 \tilde{x}_2 \tag{8.88}$$

that minimizes

$$J = \int_0^\infty (\tilde{x}_1^2 + u^2)\,dt$$

For this J,

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}; \mathbf{R} = 2$$

The matrix $\mathbf{Q}$ is positive semidefinite;

$$\mathbf{Q} = \mathbf{H}^T\mathbf{H} = \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix} [\sqrt{2} \quad 0]$$

The pair ($\mathbf{A}$,$\mathbf{H}$) is completely observable. Also, the pair ($\mathbf{A}$,$\mathbf{B}$) is completely controllable. Therefore, the sufficient conditions for the existence of asymptotically stable optimal closed-loop system are satisfied.

The matrix Riccati equation is

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = \mathbf{0}$$

or

$$\begin{bmatrix} 0 & 0 \\ 1 & -2 \end{bmatrix}\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix}$$

$$- \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}\begin{bmatrix} 0 \\ 20 \end{bmatrix}[\tfrac{1}{2}][0 \quad 20]\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

or

$$\begin{bmatrix} 0 & p_{11} - 2p_{12} \\ p_{11} - 2p_{12} & 2p_{12} - 4p_{22} \end{bmatrix} - \begin{bmatrix} 200p_{12}^2 & 200p_{12}p_{22} \\ 200p_{12}p_{22} & 200p_{22}^2 \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

or

$$2 - 200\,P_{12}^2 = 0$$

$$2p_{12} - 4p_{22} - 200p_{22}^2 = 0$$

$$p_{11} - 2p_{12} - 200p_{12}p_{22} = 0$$

The positive definite solution of the Riccati equation is

$$\mathbf{P} = \begin{bmatrix} 0.664 & 0.1 \\ 0.1 & 0.0232 \end{bmatrix}$$

The optimal gain matrix

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} = [1 \quad 0.232]$$

The minimum value of J for an initial condition $\tilde{\mathbf{x}}(0) = [-1 \quad 0]^T$, is

$$J = \tfrac{1}{2}\,\tilde{\mathbf{x}}^T(0)\mathbf{P}\,\tilde{\mathbf{x}}(0) = \frac{p_{11}}{2} = 0.332$$

It can easily be verified that the optimal closed-loop system is stable.

Review Example 8.5

Figure 8.19a illustrates a typical sampled-data system. The transfer functions $G(s)$ and $G_{h0}(s)$ of the controlled plant and the hold circuit, respectively, are known. The data-processing unit $D(z)$ which operates on the sampled error signal $e(k)$ is to be designed.

Assuming the processing unit $D(z)$ to be simply an amplifier of gain K, let us find K so that the sum square error

$$J = \sum_{k=0}^{\infty} [e^2(k) + 0.75u^2(k)]$$

is minimized.

Solution From Fig. 8.19a, we have

$$G_{h0}(s)G(s) = \frac{1 - e^{-sT}}{s^2}$$

Therefore,

$$G_{h0}G(z) = (1 - z^{-1})\,\mathcal{Z}\left[\frac{1}{s^2}\right] = \frac{1}{z-1}$$

Figure 8.19b shows an equivalent block diagram of the sampled-data system.

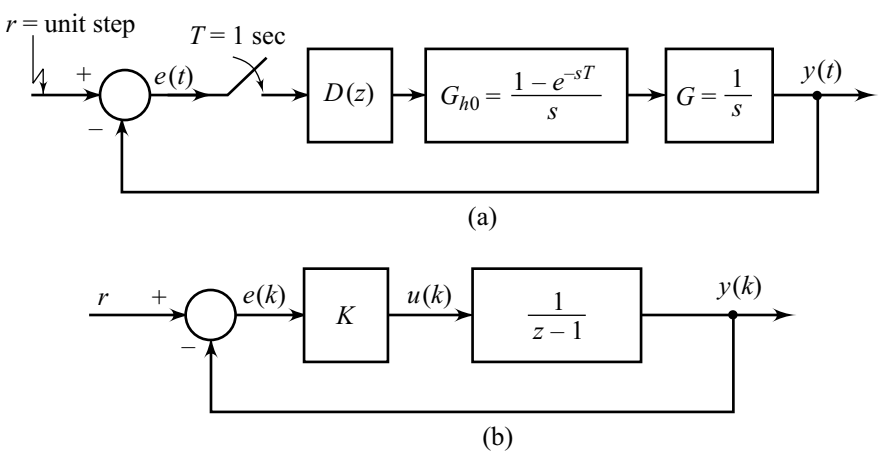


Fig. 8.19 A sampled-data system

From this figure, we obtain the following state variable model:

$$\begin{aligned} y(k+1) &= y(k) + u(k) \\ u(k) &= -K[y(k) - r] \end{aligned} \tag{8.89}$$

In terms of the shifted state variable,

$$\tilde{x}(k) = y(k) - r,$$

the state equation becomes

$$\tilde{x}(k+1) = \tilde{x}(k) + u(k) \tag{8.90}$$

The problem is to obtain optimal control sequence

$$u(k) = -K\tilde{x}(k)$$

that minimizes the performance index

$$J = \sum_{k=0}^{\infty} [\tilde{x}^2(k) + 0.75u^2(k)]$$

For this problem,

$$F = 1,\ G = 1,\ Q = 2,\ R = 1.5$$

The Riccati equation is (refer to Eqn. (8.58))

$$\begin{aligned} P &= Q + F^T PF - F^T PG(R + G^T PG)^{-1} G^T PF \\ &= 2 + P - \frac{P^2}{1.5 + P} \end{aligned}$$

Solving for P, requiring it to be positive definite, we get

$$P = 3$$

The optimal control (refer to Eqn. (8.56))

$$u(k) = -K\tilde{x}(k)$$

where

$$K = (R + G^T PG)^{-1} G^T PF = \frac{P}{1.5 + P} = \frac{2}{3}.$$

PROBLEMS

8.1 Consider the linear system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \mathbf{x}$$

Using Lyapunov analysis, determine the stability of the equilibrium state.

8.2 Using Lyapunov analysis, determine the stability of the equilibrium state of the system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix}$$

8.3 Consider the system described by the equations

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 - x_2 + 2$$

Investigate the stability of the equilibrium state. Use Lyapunov analysis.

8.4 A linear system is described by the state equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

where

$$\mathbf{A} = \begin{bmatrix} -4K & 4K \\ 2K & -6K \end{bmatrix}$$

Using Lyapunov analysis, find restrictions on the parameter K to guarantee the stability of the system.

8.5 Consider the system of Fig. P8.5. Find the restrictions on the parameter K to guarantee system stability. Use Lyapunov's analysis.

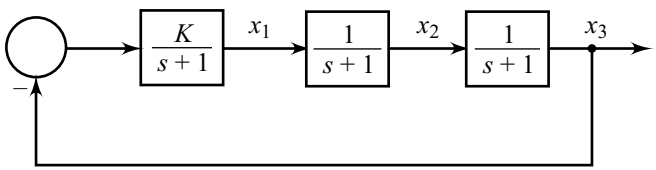


Fig. P8.5

8.6 Consider the linear system

$$\mathbf{x}(k+1) = \begin{bmatrix} 0.5 & 1 \\ -1 & -1 \end{bmatrix} \mathbf{x}(k)$$

Using Lyapunov analysis, determine the stability of the equilibrium state.

8.7 Using Lyapunov analysis, determine the stability of the equilibrium state of the system

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k)$$

with

$$\mathbf{F} = \begin{bmatrix} 0 & 0.5 \\ -0.5 & -1 \end{bmatrix}$$

8.8 Consider the system shown in Fig. P8.8. Determine the optimal feedback gain matrix **K**, such that the following performance index is minimized:

$$J = \tfrac{1}{2}\int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + 2u^2)\,dt;\ \mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

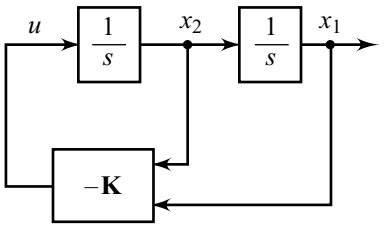


Fig. P8.8

8.9 The matrix **Q** in Problem 8.8 is replaced by the following positive semidefinite matrix:

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

Show that sufficient conditions for the existence of the asymptotically stable optimal control solution are satisfied. Find the optimal feedback matrix **K**.

8.10 Test whether sufficient conditions for the existence of the asymptotically stable optimal control solution for the plant

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u$$

with the performance index

$$J = \int_0^\infty (x_1^2 + u^2)\, dt$$

are satisfied. Find the optimal closed-loop system, if it exists, and determine its stability.

8.11 Consider the plant

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

with the performance index

$$J = \int_0^\infty (x_1^2 + u^2)\, dt$$

Test whether an asymptotically stable optimal solution exists for this control problem.

8.12 Consider the system described by the state model:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & -2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 20 \end{bmatrix} u$$

$$y = [1 \quad 0]\, \mathbf{x}$$

Find the optimal control law that minimizes

$$J = \tfrac{1}{2} \int_0^\infty [(y(t) - 1)^2 + u^2]\, dt$$

8.13 Determine the optimal control law for the system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{x}$$

such that the following performance index is minimized:

$$J = \int_0^\infty (y_1^2 + y_2^2 + u^2) dt$$

8.14 Consider the plant

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$

$$y = \mathbf{C}\mathbf{x}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{C} = [1 \quad 0]$$

with the performance index

$$J = \int_0^\infty (x_1^2 + x_2^2 + u^2)\, dt$$

Choose a control law that minimizes J. Design a state observer for implementation of the control law; both the poles of the state observer are required to lie at $s = -3$.

8.15 Figure P8.15 shows the optimal control configuration of a position servo system. Both the state variables—angular position θ and angular velocity $\dot{\theta}$—are assumed to be measurable.

It is desired to regulate the angular position to a constant value $\theta_r = 5$. Find optimum values of the gains k_1 and k_2 that minimize

$$J = \int_0^\infty [(x_1 - \theta_r)^2 + \tfrac{1}{2} u^2]\, dt$$

What is the minimum value of J?

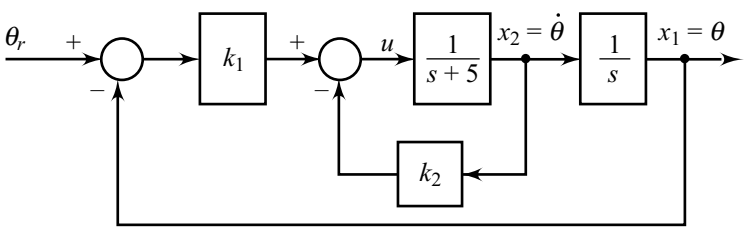


Fig. P8.15

8.16 Consider now that for the position servo system of Problem 8.15, the performance index is

$$J = \int_0^\infty [(x_1 - \theta_r)^2 + \rho u^2] dt$$

For $\rho = \frac{1}{10}$, $\frac{1}{100}$, and $\frac{1}{1000}$, find the optimal control law that minimizes the given J. Determine closed-loop poles for various values of ρ and comment on your result.

8.17 In the control scheme of Fig. P8.17, the control law of the form $u = -Ky + Ny_r$ has been used; y_r is the constant command input.

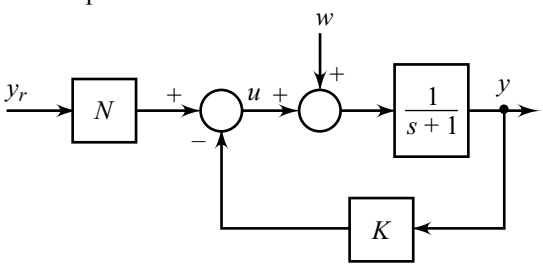


Fig. P8.17

(a) Find K such that

$$J = \int_0^\infty (\tilde{y}^2 + \tilde{u}^2)\, dt$$

is minimized; $\tilde{y}$ and $\tilde{u}$ are, respectively, the deviations of the output and the control signal from their steady-state values.

(b) Choose N so that the system has zero steady-state error, i.e., $y(\infty) = y_r$.

(c) Show that the steady-state error to a constant disturbance input w, is nonzero for the above choice of N.

(d) Add to the plant equation, an integrator equation ($z(t)$ being the state of the integrator),

$$\dot{z}\,(t) = y(t) - y_r$$

and select gains K and K_1 so that if $u = -Ky - K_1 z$, the performance index

$$J = \int_0^\infty (\tilde{y}^2 + \tilde{z}^2 + \tilde{u}^2)\, dt$$

is minimized.

(e) Draw a block diagram of the control scheme employing integral control and show that the steady-state error to constant disturbance w, is zero.

8.18 Consider a plant consisting of a dc motor, the shaft of which has the angular velocity $\omega(t)$, and which is driven by the input voltage $u(t)$. The describing equation is

$$\dot{\omega}(t) = -0.5\omega(t) + 100u(t) = A\omega(t) + Bu(t)$$

It is desired to regulate the angular velocity at the desired value $\omega^0 = r$.

(a) Use the control law of the form $u(t) = -K\omega(t) + Nr$.

Choose K that minimizes $J = \int_0^\infty (\tilde{\omega}^2 + 100\tilde{u}^2)\, dt$; $\tilde{\omega}$ and $\tilde{u}$ are, respectively, the deviations of the output and the control signal, from their steady-state values.

Choose N that guarantees zero steady-state error, i.e.,

$$\omega(\infty) = \omega^0 = r.$$

(b) Show that if A changes to $A + \delta A$, subject to $(A + \delta A - BK)$ being stable, then the above choice of N will no longer make $\omega(\infty) = r$. Therefore, the feedforward-feedback control system is not robust under changes in system parameters.

(c) The system can be made robust by augmenting it with an integrator:

$$\dot{z} = \omega - r$$

where z is the state of the integrator. To see this, first use the feedback of the form $u = -K\omega(t) - K_1 z(t)$ and select K and K_1 so that

$$J = \int_0^\infty (\tilde{\omega}^2 + \tilde{z}^2 + 100\tilde{u}^2)\, dt$$

is minimized. Show that the resulting system will have $\omega(\infty) = r$, no matter how the matrix A changes, so long as the closed-loop system remains asymptotically stable.

8.19 Consider the system

$$x(k+1) = 0.368\, x(k) + 0.632\, u(k)$$

Using the discrete matrix Riccati equation, find the control sequence

$$u(k) = -Kx(k)$$

that minimizes the performance index

$$J = \sum_{k=0}^{\infty} [x^2(k) + u^2(k)]$$

8.20 Consider the sampled-date system shown in Fig. P8.20.

(a) Find K so that

$$J = \tfrac{1}{2} \sum_{k=0}^{\infty} [\tilde{y}^2(k) + \tilde{u}^2(k)]$$

is minimized; $\tilde{y}$ and $\tilde{u}$ are, respectively, the deviations of the output and the control signal, from their steady-state values.

(b) Find the steady-state value of the output.

(c) To eliminate steady-state error, introduce a feedforward controller. The control scheme now becomes $u(k) = -Ky(k) + Nr$. Find the value of N so that $y(\infty) = r$.

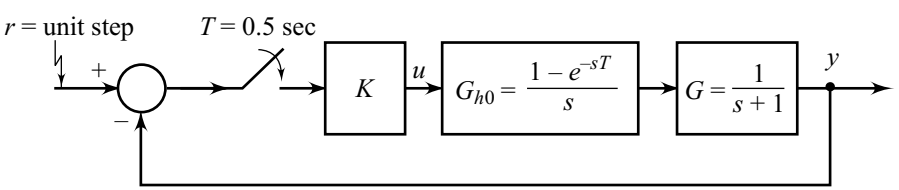


Fig. P8.20

8.21 A plant is described by the state equation

$$x(k+1) = 0.5x(k) + 2u(k) = Fx(k) + Gu(k)$$

(a) Find K such that if $u(k) = -Kx(k) + Nr$, the performance index

$$J = \frac{1}{2}\sum_{k=0}^{\infty}[\tilde{x}^2(k) + \tilde{u}^2(k)]$$

is minimized; r is a constant reference input, and $\tilde{x}$ and $\tilde{u}$ are, respectively, the deviations in state and control signal, from their steady-state values.

(b) Find N so that $x(\infty) = r$, i.e., there is no steady-state error.

(c) Show that the property of zero steady-state error is not robust with respect to changes in F.

(d) In order to obtain robust steady-state accuracy with respect to changes in F, we may use integral control in addition to state feedback. Describe through block diagram, the structure of such a control scheme.

Part III

Nonlinear Control Systems: Conventional and Intelligent

In this part of the book, we will explore tools and techniques for attacking control problems that contain significant nonlinearities.

Nonlinear control system design has been dominated by linear control techniques, which rely on the key assumption of a small range of operation for the linear model to be valid. This tradition has produced many reliable and effective control systems. However, the demand for nonlinear control methodologies has recently been increasing for several reasons.

First, modern technology, such as applied in high-performance aircraft and high-speed high-accuracy robots, demands control systems with much more stringent design specifications, which are able to handle nonlinearities of the controlled systems more accurately. When the required operation range is large, a linear controller is likely to perform very poorly or to be unstable, because the nonlinearities in the system cannot be properly compensated for. Nonlinear controllers, on the other hand, may directly handle the nonlinearities in large range operation. Also, in control systems there are many nonlinearities whose *discontinuous* nature does not allow linear approximation.

Second, controlled systems must be able to reject disturbances and uncertainties confronted in real-world applications. In designing linear controllers, it is usually necessary to assume that the parameters of the system model are reasonably well known. However, many control problems involve uncertainties in the model parameters. This may be due to a slow time variation of the parameters (e.g., of ambient air pressure during an aircraft flight), or to an abrupt change in parameters (e.g., in the inertial parameters of a robot when a new object is grasped). A linear controller, based on inaccurate values of the model parameters, may exhibit significant performance degradation or even instability. Nonlinearities can be intentionally introduced into the controller part of a control system, so that model uncertainties can be tolerated.

Third, advances in computer technology have made the implementation of nonlinear controllers a relatively simple task. The challenge for control design is to fully utilize this technology to achieve the best control system performance possible.

Thus, the subject of nonlinear control is an important area of automatic control. Learning basic techniques of nonlinear control analysis and design can significantly enhance the ability of control engineers to deal with practical control problems, effectively. It also provides a sharper understanding of the real world, which is inherently nonlinear.

NONLINEAR SYSTEMS ANALYSIS

No universal technique works for the analysis of all nonlinear control systems. In linear control, one can analyze a system in the time domain or in the frequency domain. However, for nonlinear control systems, none of these standard approaches can be used, since direct solutions of nonlinear differential equations are generally difficult, and frequency-domain transformations do not apply.

While the analysis of nonlinear control systems is difficult, serious efforts have been made to develop appropriate theoretical tools for it. Many methods of nonlinear control system analysis have been proposed. Let us briefly describe some of these methods before discussing their details in the following chapters.

Phase-Plane Analysis

Phase-plane analysis, discussed in Chapter 9, is a method of studying second-order nonlinear systems. Its basic idea is to solve a second-order differential equation and graphically display the result as a family of system motion trajectories on a two-dimensional plane, called the phase plane, which allow us to visually observe the motion patterns of the system. While phase-plane analysis has a number of important advantages, it has the fundamental disadvantage of being applicable only to systems which can be well approximated by a second-order dynamics. Because of its graphical nature, it is frequently used to provide intuitive insights about nonlinear effects.

Lyapunov Theory

In using the Lyapunov theory to analyze the stability of a nonlinear system, the idea is to construct a scalar energy-like function (a Lyapunov function) for the system, and to see whether it decreases. The power of this method comes from its generality; it is applicable to all kinds of control systems. Conversely, the limitation of the method lies in the fact that it is often difficult to find a Lyapunov function for a given system.

Although Lyapunov's method is originally a method of stability analysis, it can be used for synthesis problems. One important application is the design of nonlinear controllers. The idea is to somehow formulate a scalar positive definite function of the system states, and then choose a control law to make this function decrease. A nonlinear control system thus designed, will be guaranteed to be stable. Such a design approach has been used to solve many complex design problems, e.g., in adaptive control and in sliding mode control (discussed in Chapter 10). The basic concepts of Lyapunov theory have earlier been presented in Chapter 8. Lyapunov theory is elaborate in Chapter 9, wherein guidelines for construction of Lyapunov functions for nonlinear systems are given.

Describing Functions

The describing function method, discussed in Chapter 9, is an approximate technique for studying nonlinear systems. The basic idea of the method is to approximate the nonlinear components in nonlinear control systems by linear "equivalents", and then use frequency-domain techniques to analyze the resulting systems. Unlike the phase-plane method, it is not restricted to second-order systems. Rather,

the accuracy of describing function analysis improves with an increase in the order of the system. Unlike Lyapunov method, whose applicability to a specific system hinges on the success of a trial-and-error search for a Lyapunov function, its application is straightforward for a specific class of nonlinear systems.

NONLINEAR CONTROL DESIGN

As in the analysis of nonlinear control systems, there is no general method for designing nonlinear controllers. What we have is a rich collection of alternative and complementary techniques, each of them best applicable to particular classes of nonlinear control problems.

Trial-and-Error

Based on the analysis methods, one can use trial-and-error to synthesize controllers. The idea is to use the analysis tools to guide the search for a controller, which can then be justified by analysis and simulations. The phase-plane method, the describing function method, and Lyapunov analysis can all be used for this purpose. Experience and intuition are critical in this process. However, for complex systems, trial-and-error often fails.

Feedback Linearization

Feedback linearization discussed in Chapter 10, can be used as a nonlinear design methodology. The basic idea is to first transform a nonlinear system into a linear system using feedback, and then use the well-known and powerful linear design techniques to complete the control design. The approach has been used to solve a number of practical nonlinear control problems. It applies to important classes of nonlinear systems.

Variable Structure Sliding Mode Control

In pure model-based nonlinear control (such as the basic feedback linearization control approach), the control law is designed based on a nominal model of the physical system. How the control system will behave in the presence of model uncertainties is not clear at the design stage. In robust nonlinear control (e.g., variable structure sliding mode control), on the other hand, the controller is designed based on the consideration of both the nominal model and some characterization of the model uncertainties. Sliding mode control techniques, discussed in Chapter 10, have proven very effective in a variety of practical control problems.

Adaptive Control

Adaptive control is an approach to deal with uncertain systems or time-varying systems. Although the term "adaptive" can have broad meanings, current adaptive control designs apply mainly to systems with known dynamic structure but unknown constant or slowly-varying parameters. Adaptive controllers, whether developed for linear systems or for nonlinear systems, are inherently nonlinear.

Systematic theories exist for the adaptive control of linear systems. Frequently used adaptive control structures are discussed in Chapter 10.

Gain-Scheduling

Gain-scheduling is an attempt to apply the well-developed linear control methodology to the control of nonlinear systems. The idea of gain-scheduling is to select a number of *operating points* which cover the range of system operation. Then, at each of these points, the designer makes a linear time-invariant approximation to the plant dynamics, and designs a linear controller for each linearized plant. Between operating points, the parameters of the compensators are then interpolated, or *scheduled*; thus resulting in a global compensator.

Intelligent Control

In order to handle the complexities of nonlinearities and accommodate the demand for high-performance control systems, intelligent control takes advantage of the computational structures—fuzzy systems and neural networks—which are inherently nonlinear; a very important property, particularly if the underlying physical mechanisms for the systems are highly nonlinear. Whereas classical control is rooted in the theory of differential equations, intelligent control is largely rule-based because the dependencies involved in its deployment are much too complex to permit an analytical representation. To deal with such dependencies, the mathematics of fuzzy systems and neural networks integrates the experience and knowledge gained in the operation of a similar plant, into control algorithm. The power of fuzzy systems lies in their ability (i) to quantify linguistic inputs, and (ii) to quickly give a working approximation of complex, and often unknown, system input-output rules. The power of neural networks is in their ability to learn from data. There is a natural synergy between neural networks and fuzzy systems that makes their hybridization a powerful tool for intelligent control and other applications. Intelligent control is one of the most serious candidates for the future control of the large class of nonlinear, partially known, and time-varying systems.

With no agreed upon scientific definition of *intelligence*, and due to space limitations, we will not venture into the discussion of what intelligence is. Rather, we will confine our brief exposition in Chapters 11-14 to intelligent machines—neural networks, support vector machines, fuzzy interference systems, genetic algorithms, reinforcement learning—in the context of applications in control.

Chapter 9

Nonlinear Systems Analysis

9.1 INTRODUCTION

Because nonlinear systems can have much richer and complex behaviors than linear systems, their analysis is much more difficult. Mathematically, this is reflected in two aspects. Firstly, nonlinear equations, unlike linear ones, cannot, in general, be solved analytically, and therefore, a complete understanding of the behavior of a nonlinear system is very difficult. Secondly, powerful mathematical tools like Laplace and Fourier transforms do not apply to nonlinear systems. As a result, there are no systematic tools for predicting the behavior of nonlinear systems. Instead, there is a rich inventory of powerful analysis tools, each best applicable to a particular class of nonlinear control problems [125–129].

Perhaps the single most valuable asset to the field of engineering is the simulation tool—constructing a model of the proposed or actual system and using a numerical solution of the model to reveal the behavior of the system. Simulation is the only general method of analysis applicable to finding solutions of linear and nonlinear differential and difference equations. Of course, simulation finds specific solutions; that is, solutions to the equations with specific inputs, initial conditions, and parametric conditions. It is for this reason that simulation is not a substitute for other forms of analysis. Important properties such as stability and conditional stability are not proven with simulations. When the complexity of a system precludes the use of any analytical approach to establish proof of stability, simulations will be the only way to obtain necessary information for design purposes. A partial list of the simulation programs available today is contained in references [151–154].

This chapter also does not provide a magic solution to the analysis problem. In fact, no universal analytical technique exists that can cater to our demand on analysis of the effects of nonlinearities. Our focus in this chapter, is only on some important categories of nonlinear systems for which significant analysis (and design) can be done.

Describing Function Analysis

For the so-called *separable* systems, which comprise a linear part defined by its transfer function, and a nonlinear part defined by a time-independent relationship between its input and output variables, the *describing function method* is most practically useful for analysis. It is an approximate method but

experience with real systems and computer simulation results shows adequate accuracy in many cases. Basically, the method is an approximate extension of frequency response methods (including Nyquist stability criterion) to nonlinear systems.

In terms of mathematical properties, nonlinearities may be categorized as *continuous* and *discontinuous*. Because discontinuous nonlinearities cannot be locally approximated by linear functions, they are also called "hard" nonlinearities. Hard nonlinearities (such as saturation, backlash, or coulomb friction) are commonly found in control systems, both in small range and large range operations. Whether a system in small range operation should be regarded as nonlinear or linear depends on the magnitude of the hard nonlinearities and on the extent of their effects on the system performance.

The continuous or so-called "soft" nonlinearities are present in every control system, though not visible because these are not separable. Throughout the book, we have neglected these nonlinearities in our derivations of transfer function and state variable models. For example, we have assumed linear restoring force of a spring, a constant damping coefficient independent of the position of the mass, etc. In practice, none of these assumptions is true for a large range operation. Also, there are situations, not covered in this book, wherein the linearity assumption gives too small range of operation to be useful; linear design methods cannot be applied for such systems. If the controlled systems are not too complex and the performance requirements are not too stringent, the linearity assumptions made in this book, give satisfactory results in practice.

Describing function analysis is applicable to separable hard nonlinearities. For this category of nonlinear systems, as we shall see later in this chapter, the predictions of describing function analysis usually are a good approximation to actual behavior when the linear part of the system provides a sufficiently strong filtering effect. Filtering characteristics of the linear part of a system improve as the order of the system goes up. The 'low pass filtering' requirement is never completely satisfied; for this reason, the describing function method is mainly used for stability analysis and is not directly applied to the optimization of system design.

Phase-Plane Analysis

Another practically useful method for nonlinear system analysis is the *phase-plane method*. While phase-plane analysis does not suffer from any approximations and hence can be used for stability analysis as well as optimization of system design, its main limitation is that it is applicable to systems which can be well approximated by second-order dynamics. Its basic idea is to solve second-order differential equation and graphically display the result as a family of system motion trajectories on a two-dimensional plane, called the *phase plane*, which allows us to visually observe the motion patterns of the system. The method is equally applicable to both hard and soft nonlinearities.

Lyapunov Stability Analysis

The most fundamental analysis tool is the concept of a Lyapunov function and its use in nonlinear stability analysis. The power of the method comes from its generality. It is applicable to all kinds of control systems; systems with hard or soft nonlinearities, and of second-order or higher-order. The limitation of the method lies in the fact that it is often difficult to find a Lyapunov function for a given system.

The aim of this chapter is to introduce the two classical, yet practically important tools—the describing function method and the phase-plane method—for a class of nonlinear systems. The two methods are complementary to a large extent, each being available for the study of the systems which are most likely to be beyond the scope of the other. The phase-plane analysis applies primarily to systems described by second-order differential equations. Systems of order higher than the second are likely to be well filtered and tractable by the describing function method.

The use of Lyapunov functions for stability analysis of nonlinear systems is also given in this chapter.

9.2 SOME COMMON NONLINEAR SYSTEM BEHAVIORS

As a minimum, it is important to be aware of the main characteristics of nonlinear behavior, only to permit recognition if these are encountered experimentally or in system simulations.

The previous chapters have been predominantly concerned with the study of linear time-invariant control systems. We have observed that these systems have quite simple properties, such as the following:

- a linear system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, with $\mathbf{x}$ being the vector of states and $\mathbf{A}$ being the system matrix, has a *unique equilibrium point* (if $\mathbf{A}$ is nonsingular; normally true for feedback system matrices);
- the equilibrium point is stable if all eigenvalues of $\mathbf{A}$ have negative real parts, *regardless of initial conditions*;
- the transient response is composed of the natural modes of the system, and the general solution can be solved analytically; and
- in the presence of the external input $u(t)$, the system response has a number of interesting properties: (i) it satisfies the principle of superposition, (ii) the asymptotic stability of the system implies bounded-input, bounded-output stability, and (iii) a sinusoidal input leads to a sinusoidal output of the same frequency.

The behavior of nonlinear systems, however, is much more complex. Due to the lack of superposition property, nonlinear systems respond to external inputs and initial conditions quite differently from linear systems. Some common nonlinear system properties are as follows [126]:

(i) Nonlinear systems frequently have more than one equilibrium point. For a linear system, stability is seen by noting that for *any* initial condition, the motion of a stable system always converges to the equilibrium point. However, a nonlinear system may converge to an equilibrium point starting with one set of the initial conditions, and may go to infinity starting with another set of initial conditions. This means that the stability of nonlinear systems may depend on initial conditions. In the presence of a bounded external input, unlike linear systems, stability of nonlinear systems may also be dependent on the input value.

(ii) Nonlinear systems can display oscillations of fixed amplitude and fixed period without external excitation. These oscillations are called *limit cycles*. Consider the well-known *Van der Pol's* differential equation

$$M\ddot{y} + B(y^2 - 1)\dot{y} + Ky = 0;\ M > 0,\ B > 0,\ K > 0 \tag{9.1}$$

which describes physical situations in many nonlinear systems. It can be regarded as describing a mass-spring-damper system with a position-dependent damping coefficient $B(y^2 - 1)$. For

large values of y, the damping coefficient is positive and the damper removes energy from the system. This implies that the system motion has a convergent tendency. However, for small values of y, the damping coefficient is negative and the damper adds energy into the system. This suggests that the system motion has a divergent tendency. Therefore, because the nonlinear damping varies with y, the system motion can neither grow unboundedly nor decay to zero. Instead, it displays a sustained oscillation independent of initial conditions, as illustrated in Fig. 9.1.

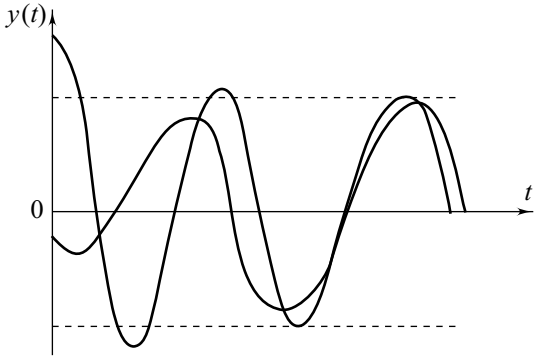


Fig. 9.1 Responses of Van der Pol oscillator

Of course, sustained oscillations can also be found in linear systems, e.g., in the case of marginally stable linear systems. However, the oscillation of a marginally stable linear system has its amplitude determined by its initial conditions, and such a system is very sensitive to changes in system parameters (a slight change in parameters is capable of leading either to stable convergence or to instability). In nonlinear systems, on the other hand, the amplitude of sustained oscillations is independent of the initial conditions, and limit cycles are not easily affected by parameter changes.

(iii) A nonlinear system with a periodic input may exhibit a periodic output whose frequency is either a subharmonic or a harmonic of the input frequency. For example, an input of frequency of 10 Hz may result in an output of 5 Hz for the subharmonic case or 30 Hz for the harmonic case.

(iv) A nonlinear system can display *jump resonance*, a form of hysteresis, in its frequency response. Consider a mass-spring-damper system

$$M\ddot{y} + B\dot{y} + K_1 y + K_2 y^3 = F \cos \omega t;\ M > 0,\ B > 0,\ K_1 > 0,\ K_2 > 0 \tag{9.2}$$

Note that the restoring force of the spring is assumed to be nonlinear. If in an experiment, the frequency ω is varied and the input amplitude F is held constant, frequency-response curve of the form shown in Fig. 9.2, may be obtained. As the frequency ω is increased, the response y follows the curve through the points A, B and C. At point C, a small change in frequency results in a discontinuous jump to point D. The response then follows the curve to point E upon further increase in frequency. As the frequency is decreased from point E, the response follows the curve through points D and F. At point F, a small change in frequency results in a

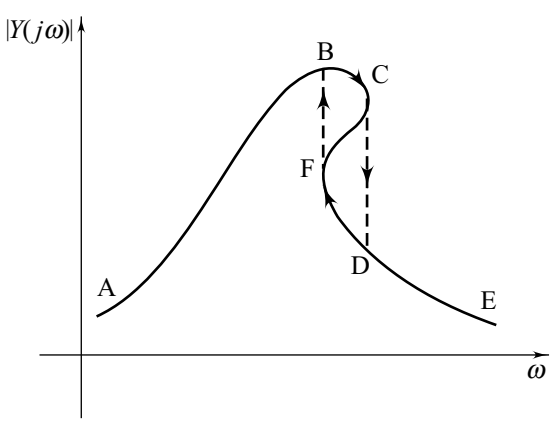


Fig. 9.2 Frequency response of a system with jump resonance

discontinuous jump to point B. The response follows the curve to point A for further decrease in frequency. Observe from this description that the response never actually follows the segment CF. This portion of the curve represents a condition of unstable equilibrium.

9.3 COMMON NONLINEARITIES IN CONTROL SYSTEMS

In this section, we take a closer look at the nonlinearities found in control systems. Consider the typical block diagram of closed-loop system shown in Fig. 9.3. It is composed of four parts: a plant to be controlled, sensors for measurements, actuators for control action, and a control law usually implemented on a computer. Nonlinearities may occur in any part of the system.

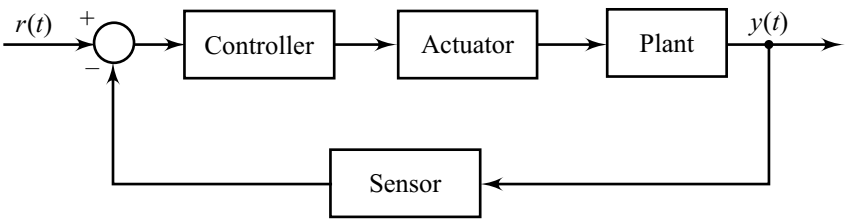


Fig. 9.3 General diagram of a control system

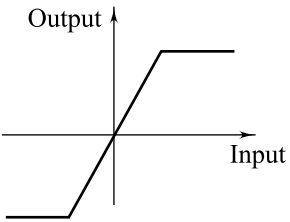
Saturation

Saturation is probably the most commonly encountered nonlinearity in control systems. It is often associated with amplifiers and actuators. In transistor amplifiers, the output varies linearly with the input, only for small amplitude limits. When the input amplitude gets out of the linear range of the amplifier, the output changes very little and stays close to its maximum value. Figure 9.4a shows a linear-segmented approximation of saturation nonlinearity.

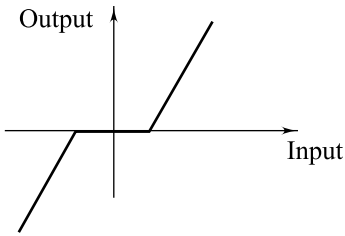
Most actuators display saturation characteristics. For example, the output torque of a servo motor cannot increase infinitely, and tends to saturate due to the properties of the magnetic material. Similarly, valve-controlled hydraulic actuators are saturated by the maximum flow rate.

Deadzone

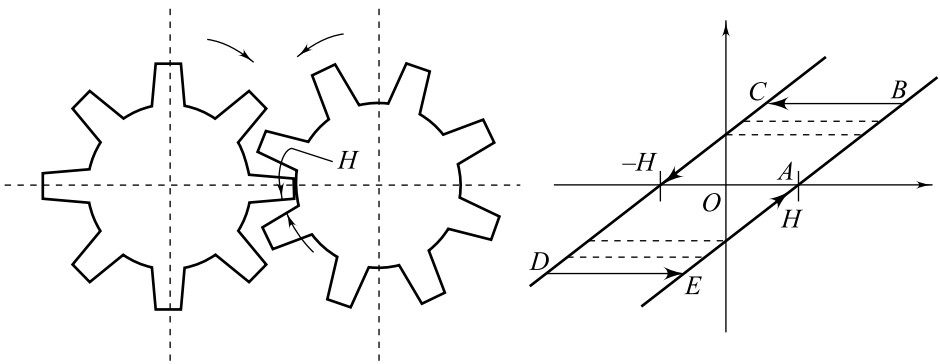
A deadzone nonlinearity may occur in sensors, amplifiers and actuators. In a dc motor, we assume that any voltage applied to the armature windings will cause the armature to rotate if the field current is maintained constant. In reality, due to static friction at the motor shaft, rotation will occur only if the torque provided by the motor is sufficiently large. This corresponds to a so-called deadzone for small voltage signals. Similar deadzone phenomena occur in valve-controlled pneumatic and hydraulic actuators. Figure 9.4b shows linear-segmented approximation of deadzone nonlinearity.



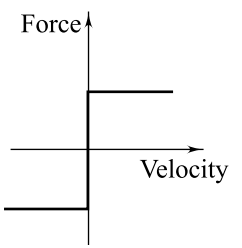
(a) Saturation nonlinearity



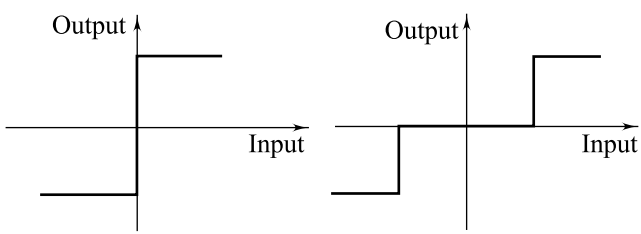
(b) Deadzone nonlinearity



(c) Backlash nonlinearity



(d) Coulomb friction nonlinearity



(e) On–Off (two position/three position) control nonlinearities

Fig. 9.4

Backlash

A backlash nonlinearity commonly occurs in mechanical components of control systems. In gear trains, small gaps exist between a pair of mating gears (refer to Fig. 9.4c). As a result, when the driving gear rotates a smaller angle than the gap H, the driven gear does not move at all, which corresponds to the deadzone (OA segment in Fig. 9.4c); after contact has been established between the two gears, the driven gear follows the rotation of the driving gear in a linear fashion (AB segment). When the driving gear rotates in the reverse direction, by a distance of $2H$, the driven gear again does not move, corresponding to the segment BC in Fig. 9.4c. After the contact between the two gears is re-established, the driven gear linearly follows the rotation of the driving gear in the reverse direction (CD segment). Therefore, if the driving gear is in periodic motion, the driven gear will move in the fashion represented by the closed path EBCD.

A critical feature of backlash, a form of hysteresis, is its multivalued nature. Corresponding to each input, two output values are possible; which one of the two occurs depends on the history of the input.

Coulomb Friction

In any system where there is a relative motion between contacting surfaces, there are several types of friction: all of them nonlinear—except the viscous components. Coulomb friction is, in essence, a drag (reaction) force which opposes motion, but is essentially constant in magnitude, regardless of velocity (Fig. 9.4d). The common example is an electric motor, in which we find Coulomb friction drag due to the rubbing contact between the brushes and the commutator.

On–Off Nonlinearity

In this book we have primarily covered the following three modes of control:

(i) proportional control;
(ii) integral control; and
(iii) derivative control.

Another important mode of feedback control is the *on–off* control. This class of controllers have only two fixed states rather than a continuous output. In its wider application, the states of an *on–off controller* may not, however, be simply on and off but could represent any two values of a control variable. Oscillatory behavior is a typical response characteristic of a system under *two-position control*, also called *bang-bang control*. The oscillatory behavior may be avoided using a *three-position control* (*on–off controller with a deadzone*). Figure 9.4e shows the characteristics of on–off controllers.

The on–off mode of control results in a *variable structure system* whose structure changes in accordance with the current value of its state. A variable structure system can be viewed as a system composed of independent structures, together with a switching logic between each of the structures. With appropriate switching logic, a variable structure system can exploit the desirable properties of each of the structures the system is composed of. Even more, a variable structure system may have a property that is not a property of any of its structures. The variable structure sliding mode control law is usually implemented on a computer. The reader will be exposed to simple variable structure systems in this chapter; details to follow in Chapter 10.

We may classify the nonlinearities as *inherent* and *intentional*. Inherent nonlinearities naturally come with the system's hardware (saturation, deadzone, backlash, Coulomb friction). Usually such nonlinearities have undesirable effects, and control systems have to properly compensate for them. Intentional nonlinearities, on the other hand, are artificially introduced by the designer. Nonlinear control laws, such as bang-bang optimal control laws and adaptive control laws (refer to Chapter 10), are typical examples of intentional nonlinearities.

9.4 DESCRIBING FUNCTION FUNDAMENTALS

Of all the analytical methods developed over the years for nonlinear systems, the describing function method is generally agreed upon as being the most practically useful. It is an approximate method,

but experience with real systems and computer simulation results, shows adequate accuracy in many cases. The method predicts whether limit cycle oscillations will exist or not, and gives numerical estimates of oscillation frequency and amplitude when limit cycles are predicted. Basically, the method is an approximate extension of frequency-response methods (including Nyquist stability criterion) to nonlinear systems.

To discuss the basic concept underlying the describing function analysis, let us consider the block diagram of a nonlinear system shown in Fig. 9.5, where the blocks $G_1(s)$ and $G_2(s)$ represent the linear elements, while the block N represents the nonlinear element.

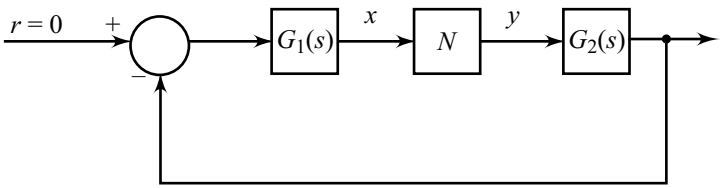


Fig. 9.5 A nonlinear system

The describing function method provides a "linear approximation" to the nonlinear element based on the assumption that the input to the nonlinear element is a sinusoid of known, constant amplitude. The fundamental harmonic of the element's output is compared with the input sinusoid, to determine the steady-state amplitude and phase relation. This relation is the describing function for the nonlinear element. The method can, thus, be viewed as 'harmonic linearization' of a nonlinear element.

The describing function method is based on the Fourier series. A review of the Fourier series will be in order here.

9.4.1 Fourier Series

We begin with the definition of a periodic signal. A signal $y(t)$ is said to be periodic with the period T if $y(t + T) = y(t)$ for every value of t. The smallest positive value of T for which $y(t + T) = y(t)$, is called fundamental period of $y(t)$. We denote the fundamental period as T_0. Obviously, $2T_0$ is also a period of $y(t)$, and so is any integer multiple of T_0.

A periodic signal $y(t)$ may be represented by the series [31]:

$$y(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos n\omega_0 t + b_n \sin n\omega_0 t] \tag{9.3a}$$

$$= \frac{a_0}{2} + \sum_{n=1}^{\infty} Y_n \sin(n\omega_0 t + \phi_n) \tag{9.3b}$$

where

$$a_n = \frac{2}{T_0} \int_0^{T_0} y(t) \cos n\omega_0 t \, dt; \; n = 0, 1, 2, \ldots \tag{9.3c}$$

$$b_n = \frac{2}{T_0} \int_0^{T_0} y(t) \sin n\omega_0 t \, dt; \; n = 1, 2, \ldots \tag{9.3d}$$

$$Y_n = \sqrt{a_n^2 + b_n^2} \tag{9.3e}$$

$$\phi_n = \tan^{-1}\left(\frac{a_n}{b_n}\right) \tag{9.3f}$$

In Eqn. (9.3b), the term for $n = 1$ is called *fundamental* or *first harmonic*, and always has the same frequency as the repetition rate of the original periodic waveform; whereas $n = 2, 3, ...$, give second, third, and so forth harmonic frequencies as integer multiples of the fundamental frequency.

Introducing a change of variable to $\psi = \omega_0 t$, we obtain the following alternative equations for the coefficients of Fourier series ($\omega_0 = 2\pi/T_0$):

$$a_n = \frac{1}{\pi}\int_0^{2\pi} y(t)\cos n\omega_0 t\, d(\omega_0 t);\ n = 0, 1, 2, ... \tag{9.4a}$$

$$b_n = \frac{1}{\pi}\int_0^{2\pi} y(t)\sin n\omega_0 t\, d(\omega_0 t);\ n = 1, 2, ... \tag{9.4b}$$

Certain simplifications are possible when $y(t)$ has a symmetry of one type or another.

(i) Even symmetry: $y(t) = y(-t)$ results in

$$b_n = 0;\ n = 1, 2, ... \tag{9.4c}$$

(ii) Odd symmetry: $y(t) = -y(-t)$ results in

$$a_n = 0;\ n = 0, 1, 2, ... \tag{9.4d}$$

(iii) Odd half-wave symmetry: $y(t \pm T_0/2) = -y(t)$ results in

$$a_n = b_n = 0;\ n = 0, 2, 4, ... \tag{9.4e}$$

9.4.2 Describing Function for the Nonlinear Element

Let us assume that input x to the nonlinearity in Fig. 9.5 is sinusoidal, i.e.,

$$x = X\sin\omega t$$

With such an input, the output y of the nonlinear element will, in general, be a nonsinusoidal periodic function which may be expressed in terms of Fourier series as follows (refer to Eqns (9.3)–(9.4)):

$$y = Y_0 + A_1\cos\omega t + B_1\sin\omega t + A_2\cos 2\omega t + B_2\sin 2\omega t + \cdots$$

The nonlinear characteristics listed in the previous section, are all odd-symmetrical/odd half-wave symmetrical; the mean value Y_0 for all such cases is zero and therefore, the output

$$y = A_1\cos\omega t + B_1\sin\omega t + A_2\cos 2\omega t + B_2\sin 2\omega t + \cdots$$

In the absence of an external input (i.e., $r = 0$ in Fig. 9.5), the output y of the nonlinear element N is fed back to its input, through the linear elements $G_2(s)$ and $G_1(s)$ in tandem. If $G_2(s)G_1(s)$ has *low-pass characteristics* (this is usually the case in control systems), it can be assumed, to a good degree of approximation, that all the higher harmonics of y are filtered out in the process, and the input x to the nonlinear element N is mainly contributed by the fundamental component (first harmonic) of y, i.e., x remains sinusoidal. Under such conditions, the second and higher harmonics of y can be thrown away for

the purpose of analysis, and the fundamental component of y, i.e.,

$$y_1 = A_1 \cos \omega t + B_1 \sin \omega t$$

need only be considered.

The above procedure heuristically linearizes the nonlinearity since, for a sinusoidal input, only a sinusoidal output of the same frequency is now assumed to be produced. This type of linearization, called the *first-harmonic approximation*, is valid for large signals as well, so long as the filtering condition is satisfied.

We can write $y_1(t)$ in the form

$$y_1(t) = A_1 \sin(\omega t + 90°) + B_1 \sin \omega t = Y_1 \sin(\omega t + \phi_1) \tag{9.5a}$$

where, by using phasors,

$$Y_1 \angle \phi_1 = B_1 + jA_1 = \sqrt{B_1^2 + A_1^2} \angle \tan^{-1}(A_1/B_1) \tag{9.5b}$$

The coefficients A_1 and B_1 of the Fourier series are given by (refer to Eqns 9.3)

$$A_1 = \frac{1}{\pi} \int_0^{2\pi} y \cos \omega t \, d(\omega t) \tag{9.5c}$$

$$B_1 = \frac{1}{\pi} \int_0^{2\pi} y \sin \omega t \, d(\omega t) \tag{9.5d}$$

As we shall see shortly, the amplitude Y_1 and the phase shift ϕ_1 are both functions of X, but independent of ω. We may combine the amplitude ratio and the phase shift in a *complex equivalent gain* $N(X)$, such that

$$N(X) = \frac{Y_1(X)}{X} \angle \phi_1(X) = \frac{B_1 + jA_1}{X} \tag{9.6}$$

Under first-harmonic approximation, the nonlinear element is completely characterized by the function $N(X)$; this function is usually referred to as the *describing function* of the nonlinearity.

The describing function differs from a linear system transfer function, in that its numerical value will vary with input amplitude X. Also, it does not depend on frequency ω (there are, however, a few situations in which the describing function for the nonlinearity is a function of both, the input amplitude X, and the frequency ω (refer to [128–129]). When embedded in an otherwise linear system (Fig. 9.6), the describing function can be combined with the 'ordinary' sinusoidal transfer function of the rest of the system, to obtain the complete open-loop function. However, we will get a different open-loop function for every different amplitude X. We can check all of these open-loop functions for closed-loop stability, using Nyquist stability criterion.

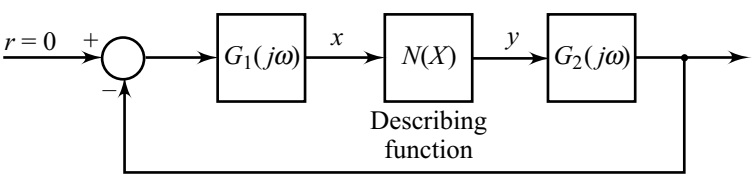


Fig. 9.6 Nonlinear system with nonlinearity replaced by describing function

It is important to remind ourselves here that the simplicity in analysis of nonlinear systems using describing functions, has been achieved at the cost of certain limitations; the foremost being the assumption that in traversing the path through the linear parts of the system from nonlinearity output back to nonlinearity input, the higher harmonics will have been effectively low-pass filtered, relative to the first harmonic. When the linear part of the system does indeed provide a sufficiently strong filtering effect, then the predictions of describing function analysis, usually, are a good approximation to actual behavior. Filtering characteristics of the linear part of the system improve as the order of the system goes up.

The 'low-pass filtering' requirement is never completely satisfied; for this reason, the describing function method is mainly used for stability analysis and is not directly applied to the optimization of system design. Usually, the describing function analysis will correctly predict the existence and characteristics of limit cycles. However, false indications cannot be ruled out; therefore, the results must be verified by simulation. Simulation, in fact, is an almost indispensable tool for analysis and design of nonlinear systems; describing function and other analytical methods, provide the background for intelligent planning of the simulations.

We will limit our discussion to separable nonlinear systems with reference input $r = 0$, and with symmetrical nonlinearities (listed in Section 9.3) in the loop. Refer to [128–129] for situations wherein dissymmetrical nonlinearities are present, and/or the reference input is nonzero.

9.5 DESCRIBING FUNCTIONS OF COMMON NONLINEARITIES

Before coming to the stability study by the describing function method, it is worthwhile to derive the describing functions of some common nonlinearities. Our first example is an on–off controller with a deadzone as in Fig. 9.7. If X is less than deadzone Δ, then the controller produces no output; the first harmonic component of the Fourier series is, of course, zero, and the describing function is also zero. If $X > \Delta$, the controller produces the 'square wave' output y. One cycle of this periodic function of period 2π is described as follows:

$$y = \begin{cases} 0 & ; 0 \le \omega t < \alpha \\ M & ; \alpha \le \omega t < (\pi - \alpha) \\ 0 & ; (\pi - \alpha) \le \omega t < (\pi + \alpha) \\ -M & ; (\pi + \alpha) \le \omega t < (2\pi - \alpha) \\ 0 & ; (2\pi - \alpha) \le \omega t \le 2\pi \end{cases} \tag{9.7}$$

where $X \sin\alpha = \Delta$; or $\alpha = \sin^{-1}(\Delta/X)$.

This periodic function has odd symmetry:

$$y(\omega t) = -y(-\omega t)$$

Therefore, the fundamental component of y is given by (refer to Eqn. (9.4d))

$$y_1 = B_1 \sin\omega t$$

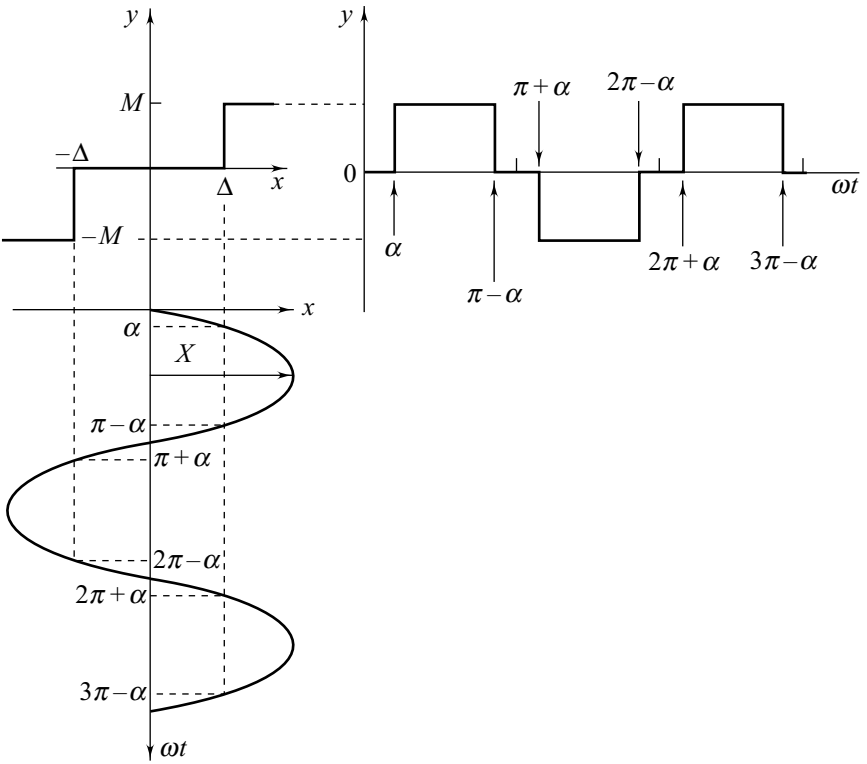


Fig. 9.7 Fourier series analysis of an on–off controller with deadzone

where
$$B_1 = \frac{1}{\pi}\int_0^{2\pi} y \sin\omega t \, d(\omega t)$$

Due to the symmetry of y (refer to Fig. 9.7), the coefficient B_1 can be calculated as follows:

$$B_1 = \frac{4}{\pi}\int_0^{\frac{\pi}{2}} y \sin\omega t \, d(\omega t) = \frac{4M}{\pi}\int_\alpha^{\frac{\pi}{2}} \sin\omega t \, d(\omega t) = \frac{4M}{\pi}\cos\alpha \tag{9.8}$$

Since A_1 (the Fourier series cosine coefficient) is zero, the first harmonic component of y is exactly in phase with $X\sin\omega t$, and the describing function $N(X)$ is given by (refer to Eqns (9.6)–(9.8))

$$N(X) = \begin{cases} 0 & ; X < \Delta \\ \dfrac{4M}{\pi X}\sqrt{1-\left(\dfrac{\Delta}{X}\right)^2} & ; X \geq \Delta \end{cases} \tag{9.9}$$

For a given controller, M and Δ are fixed and the describing function is a function of input amplitude X, which is graphed in Fig. 9.8a, together with peak location and value, found by standard calculus

maximization procedure. Note that for a given X, $N(X)$ is just a pure real positive number, and thus, plays the role of a steady-state gain in a block diagram of the form shown in Fig. 9.6. However, this gain term is unusual in that it changes when X changes.

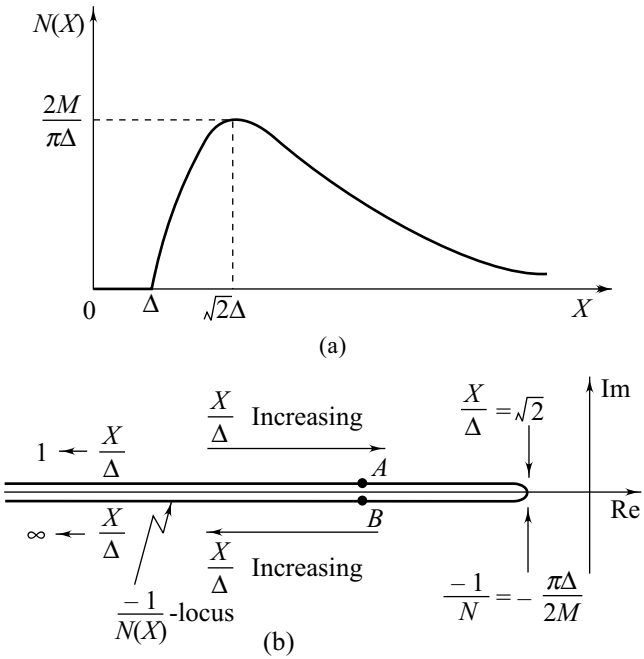


Fig. 9.8 Describing function of an on–off controller with deadzone

A describing function $N(X)$ may be equivalently represented by a plot of

$$-\frac{1}{N(X)} = \left|-\frac{1}{N(X)}\right| \angle(-1/N(X)) \tag{9.10}$$

as a function of X on the polar plane. We will use this form of representation in the next section for stability analysis.

Rearrangement of Eqn. (9.9) gives

$$-\frac{1}{N(X)} = -\frac{\pi\Delta}{4M}\frac{(X/\Delta)^2}{\sqrt{(X/\Delta)^2 - 1}} \tag{9.11}$$

Figure 9.8b gives the representation on the polar plane, of the describing function for an on–off controller with deadzone. It may be noted that though the points A and B lie at the same place on the negative real axis, they belong to different values of X/Δ.

We choose as another example the backlash, since its behavior brings out certain features not encountered in our earlier example. The characteristics of backlash nonlinearity, and its response to sinusoidal input,

are shown in Fig. 9.9. The output y is again a periodic function of period 2π; one cycle of this function is described as follows:

$$y = \begin{cases} x - H \ ; \ 0 \le \omega t < \pi/2 \\ X - H \ ; \ \pi/2 \le \omega t < (\pi - \beta) \\ x + H \ ; \ (\pi - \beta) \le \omega t < 3\pi/2 \\ -X + H \ ; \ 3\pi/2 \le \omega t < (2\pi - \beta) \\ x - H \ ; \ (2\pi - \beta) \le \omega t \le 2\pi \end{cases} \tag{9.12}$$

where $X \sin\beta = X - 2H$; or $\beta = \sin^{-1}\left(1 - \dfrac{2H}{X}\right)$.

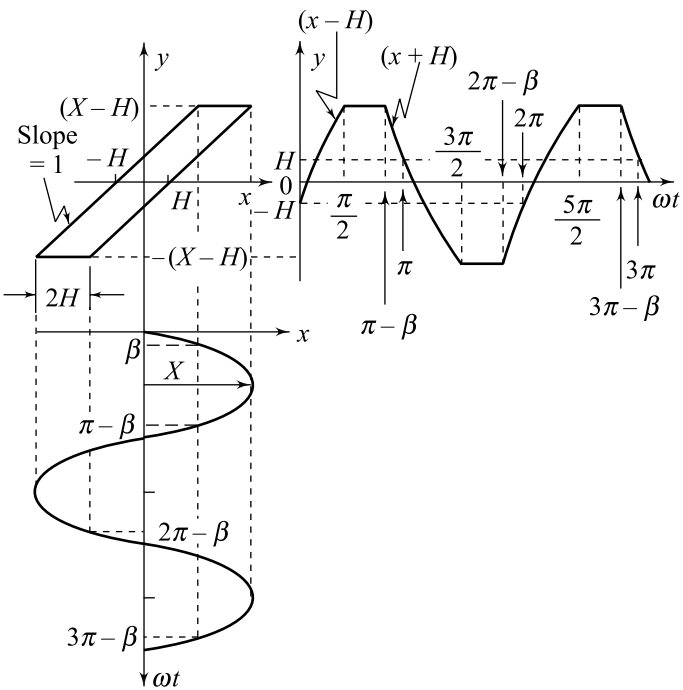


Fig. 9.9 Fourier-series analysis of backlash

The periodic function does not possess odd symmetry:

$$y(\omega t) \neq -y(-\omega t),$$

but possesses odd half-wave symmetry:

$$y(\omega t \pm \pi) = -y(\omega t)$$

Therefore, the fundamental component of y is given by (refer to Eqn. (9.4e))

$$y_1 = A_1 \cos\omega t + B_1 \sin\omega t$$

where

$$A_1 = \frac{1}{\pi} \int_0^{2\pi} y \cos\omega t \, d(\omega t)$$

$$B_1 = \frac{1}{\pi}\int_0^{2\pi} y \sin\omega t\, d(\omega t)$$

Due to the symmetry of y, only the positive half-wave need be considered (Fig. 9.9):

$$\begin{aligned}
A_1 &= \frac{2}{\pi}\left[\int_0^{\frac{\pi}{2}} (X\sin\omega t - H)\cos\omega t\, d(\omega t) + \int_{\frac{\pi}{2}}^{(\pi-\beta)} (X - H)\cos\omega t\, d(\omega t)\right.\\
&\quad \left. + \int_{(\pi-\beta)}^{\pi} (X\sin\omega t + H)\cos\omega t\, d(\omega t)\right]\\
&= \frac{2X}{\pi}\int_0^{\frac{\pi}{2}} \sin\theta\cos\theta\, d\theta - \frac{2H}{\pi}\int_0^{\frac{\pi}{2}} \cos\theta\, d\theta + \frac{2(X-H)}{\pi}\int_{\frac{\pi}{2}}^{(\pi-\beta)} \cos\theta\, d\theta\\
&\quad + \frac{2X}{\pi}\int_{(\pi-\beta)}^{\pi} \sin\theta\cos\theta\, d\theta + \frac{2H}{\pi}\int_{(\pi-\beta)}^{\pi} \cos\theta\, d\theta\\
&= -\frac{3X}{2\pi} + \frac{2(X-2H)}{\pi}\sin\beta + \frac{X}{2\pi}\cos 2\beta\\
&= -\frac{3X}{2\pi} + \frac{2X}{\pi}\sin^2\beta + \frac{X}{2\pi}\cos 2\beta = -\frac{X}{\pi}\cos^2\beta
\end{aligned} \tag{9.13a}$$

$$\begin{aligned}
B_1 &= \frac{2}{\pi}\left[\int_0^{\frac{\pi}{2}} (X\sin\omega t - H)\sin\omega t\, d(\omega t) + \int_{\frac{\pi}{2}}^{(\pi-\beta)} (X - H)\sin\omega t\, d(\omega t)\right.\\
&\quad \left. + \int_{(\pi-\beta)}^{\pi} (X\sin\omega t + H)\sin\omega t\, d(\omega t)\right]\\
&= \frac{2X}{\pi}\int_0^{\frac{\pi}{2}} \sin^2\theta\, d\theta - \frac{2H}{\pi}\int_0^{\frac{\pi}{2}} \sin\theta d\theta + \frac{2(X-H)}{\pi}\int_{\frac{\pi}{2}}^{(\pi-\beta)} \sin\theta d\theta\\
&\quad + \frac{2X}{\pi}\int_{(\pi-\beta)}^{\pi} \sin^2\theta d\theta + \frac{2H}{\pi}\int_{(\pi-\beta)}^{\pi} \sin\theta d\theta\\
&= \frac{X}{\pi}\left[\frac{\pi}{2} + \beta\right] + \frac{2(X-2H)}{\pi}\cos\beta - \frac{X}{2\pi}\sin 2\beta\\
&= \frac{X}{\pi}\left[\frac{\pi}{2} + \beta\right] + \frac{2X}{\pi}\sin\beta\cos\beta - \frac{X}{2\pi}\sin 2\beta = \frac{X}{\pi}\left[\frac{\pi}{2} + \beta + \frac{1}{2}\sin 2\beta\right]
\end{aligned} \tag{9.13b}$$

It is clear that the fundamental component of y will have a phase shift with respect to $X\sin\omega t$ (a feature not present in our earlier example). The describing function $N(X)$ is given by (refer to Eqns (9.6), (9.12), (9.13))

$$N(X) = \frac{1}{X}(B_1 + jA_1) = \frac{1}{\pi}\left[\frac{\pi}{2} + \beta + \frac{1}{2}\sin 2\beta - j\cos^2\beta\right] \tag{9.14}$$

$$\beta = \sin^{-1}\left(1 - \frac{2H}{X}\right)$$

Note that $N(X)$ is a function of the nondimensional ratio H/X; we can thus tabulate or plot a single graph of $N(X)$ that will be usable for any numerical value of H (refer to Table 9.1, and Fig. 9.10).

Table 9.1 Describing function for backlash

H/X	$\lvert -1/N(X)\rvert$	$\angle(-1/N(X))$
0.000	1.000	180.0
0.050	1.017	183.5
0.125	1.066	188.5
0.200	1.134	193.4
0.300	1.259	199.7
0.400	1.435	206.0
0.500	1.687	212.5
0.600	2.072	219.3
0.700	2.720	226.7
0.800	4.024	235.1
0.850	5.330	239.9
0.900	7.946	245.6
0.925	10.560	248.9
0.950	15.800	252.8
0.975	31.500	257.9

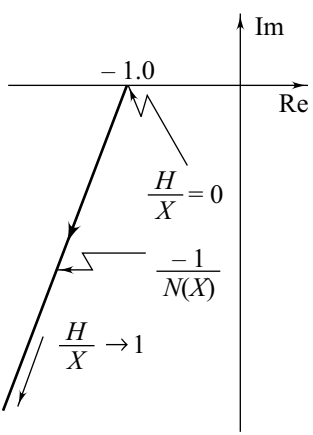
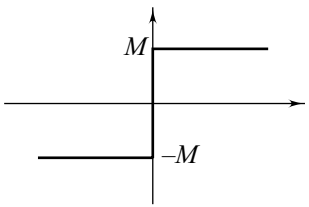
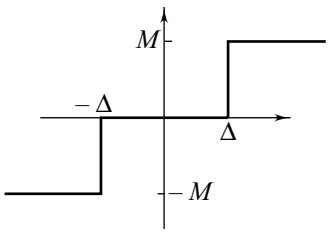
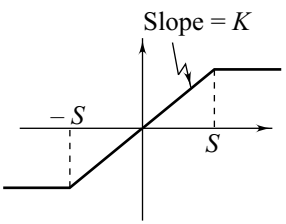
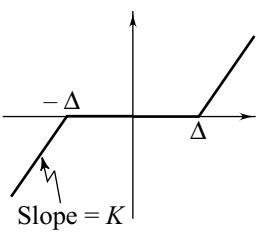
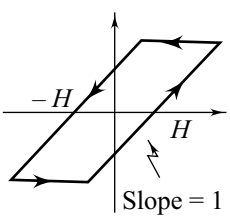


Fig. 9.10 Describing function for backlash

We have so far given illustrative derivations of describing functions for on–off controller with deadzone, and backlash. By similar procedures, the describing functions of other common nonlinearities can be derived; some of these are tabulated in Table 9.2.

Table 9.2 Describing functions of common nonlinearities

Nonlinearity	*Describing function (input = $X\sin\omega t$)*
1. 	$N(X) = \dfrac{4M}{\pi X}$
2. 	$N(X) = \begin{cases} 0 & ; X < \Delta \\ \dfrac{4M}{\pi X}\sqrt{1-\left(\dfrac{\Delta}{X}\right)^2} & ; X \geq \Delta \end{cases}$
3. 	$N(X) = \begin{cases} K & ; X < S \\ \dfrac{2K}{\pi}\left[\sin^{-1}\dfrac{S}{X} + \dfrac{S}{X}\sqrt{1-\left(\dfrac{S}{X}\right)^2}\right] & ; X \geq S \end{cases}$
4. 	$N(X) = \begin{cases} 0 & ; X < \Delta \\ \dfrac{2K}{\pi}\left[\dfrac{\pi}{2} - \sin^{-1}\dfrac{\Delta}{X} - \dfrac{\Delta}{X}\sqrt{1-\left(\dfrac{\Delta}{X}\right)^2}\right] & ; X \geq \Delta \end{cases}$
5. 	$N(X) = \dfrac{1}{\pi}\left[\dfrac{\pi}{2} + \beta + \dfrac{1}{2}\sin 2\beta - j\cos^2\beta\right]$ $\beta = \sin^{-1}\left(1 - \dfrac{2H}{X}\right)$

9.6 STABILITY ANALYSIS BY THE DESCRIBING FUNCTION METHOD

Consider the linear system of Fig. 9.11a. Application of Nyquist stability criterion[1] to this system involves the following steps.

(i) Define the Nyquist contour in the *s*-plane that encloses the entire right-hand side (unstable region) of the *s*-plane (Fig. 9.11b).

(ii) Sketch the Nyquist plot, which is the locus of $KG(s)H(s)$, when *s* takes on values along the Nyquist contour (Fig. 9.11c).

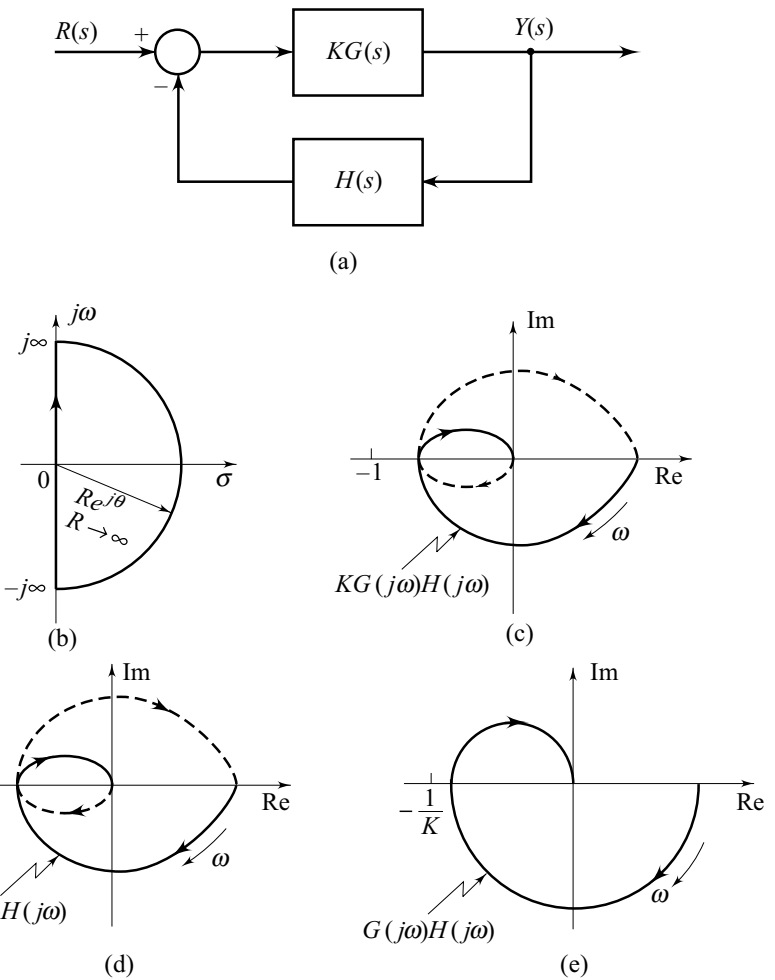


Fig. 9.11 Application of the Nyquist stability criterion

[1] Chapter 10 of reference [155].

(iii) The characteristic equation of the system is

$$1 + KG(s)H(s) = 0$$

or

$$KG(s)H(s) = -1 \tag{9.15}$$

The stability of the closed-loop system is determined by investigating the behavior of the Nyquist plot of $KG(s)H(s)$ with respect to the *critical point* $(-1 + j0)$ in the $KG(s)H(s)$-plane.

For the predominant case of systems wherein open-loop transfer function $KG(s)H(s)$ has no poles in the right half of the s-plane, the Nyquist stability criterion is stated below as

If the Nyquist plot of the open-loop transfer function $KG(s)H(s)$ corresponding to the Nyquist contour in the s-plane, does not encircle the critical point $(-1 + j0)$, the closed-loop system is stable.

(iv) The characteristic equation (9.15) may be rearranged as follows:

$$G(s)H(s) = -1/K \tag{9.16}$$

For the linear system with open-loop transfer function $KG(s)H(s)$, we can count the number of encirclements of $(-1/K + j0)$ point if the Nyquist plot of $G(s)H(s)$ is constructed (Fig. 9.11d).

(v) When the Nyquist plot of $G(s)H(s)$ passes through $(-1/K + j0)$ point, the number of encirclements is indeterminate. This corresponds to the condition where $1 + KG(s)H(s)$ has zeros on the imaginary axis (i.e., the closed-loop system has poles on the imaginary axis). The gain corresponding to this situation, will yield oscillatory behavior (we have assumed that the zeros are nonrepeated).

(vi) The most commonly occurring situation in control system design is that the system becomes unstable if the gain increases past a certain critical value. Stability condition for such systems, becomes

$$|G(j\omega)H(j\omega)| < 1/K \text{ at } \angle G(j\omega)H(j\omega) = -180^\circ$$

The stability may, therefore, be examined from polar plot (plot of $G(j\omega)H(j\omega)$ on polar plane with ω varying from 0 to ∞) only (Fig. 9.11e).

Consider now a nonlinear system of Fig. 9.12. $N(X)$ is the describing function of the nonlinear element and $G(s)$ is the transfer function of the linear part of the system. $G(s)$ is assumed to have no poles in the right half of the s-plane.

The validity of the block diagram shown in Fig. 9.12 is based on the assumption that the input to the nonlinearity is a pure sinusoid $x = X\sin\omega t$. This necessarily requires that r is zero, since nonzero values of the system input, usually result in the nonlinearity input signal containing components in addition to the assumed sine wave. So, the describing function approach is applicable when the input r is zero and the system is excited by some initial conditions. For different values of the initial conditions, a signal of the form $x = X\sin\omega t$ will be generated at the input of an odd-symmetrical/odd half-wave symmetrical nonlinearity, with X varying from 0 to ∞. This is true only if the linear part of the system possesses the required low-pass characteristics. For situations where the nonlinearity input

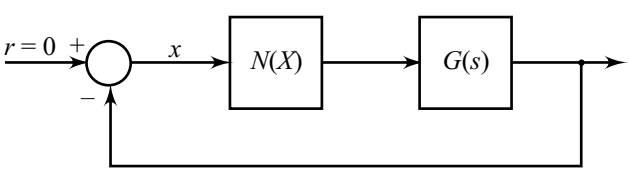


Fig. 9.12 A nonlinear system

signal contains components in addition to $X\sin\omega t$ (such as r not being zero), the method of dual-input describing functions may be useful (refer to [128–129]).

For a given X, $N(X)$ in Fig. 9.12 is just a real/complex number; the condition (9.16) therefore, becomes

$$G(s) = -1/N(X) \tag{9.17}$$

This modified condition differs from the condition (9.16), in the fact that the critical point $(-1/K + j0)$, now becomes the critical locus $-1/N(X)$ as a function of X. The stability analysis can be carried out by examining the relative position of the following plots on polar plane.

(i) Plot of $G(j\omega)$ with ω varying from 0 to ∞, called the *polar plot* of $G(j\omega)$ (note that the Nyquist plot is the plot of $G(j\omega)$ with ω varying from $-\infty$ to $+\infty$).

(ii) Plot of $-1/N(X)$ with X varying from 0 to ∞.

When the critical points of $-1/N(X)$ lie to the left of the polar plot of $G(j\omega)$ (or are not encircled by the Nyquist plot of $G(j\omega)$), the closed-loop system is stable; any disturbances which appear in the system will tend to die out. Conversely, if any part of the $-1/N(X)$ locus lies to the right of the polar plot of $G(j\omega)$ (or is enclosed by the Nyquist plot of $G(j\omega)$), it implies that any disturbances which are characterized by the values of X corresponding to the enclosed critical points, will provide unstable operations. The intersection of $G(j\omega)$ and $-1/N(X)$ loci, corresponds to the possibility of a periodic oscillation (limit cycle) characterized by the value of X on the $-1/N(X)$ locus, and the value of ω on the $G(j\omega)$ locus.

Figure 9.13a shows a $G(j\omega)$ plot superimposed on a $-1/N(X)$ locus. The values of X, for which the $-1/N(X)$ locus lies in the region to the right of an observer traversing the polar plot of $G(j\omega)$ in the direction of increasing ω, correspond to unstable conditions. Similarly, the values of X, for which the $-1/N(X)$ locus lies in the region to the left of an observer traversing the polar plot of $G(j\omega)$ in the direction of increasing ω, correspond to the stable conditions. The locus of $-1/N(X)$ and the polar plot of $G(j\omega)$ intersect at the point $A(\omega = \omega_2, X = X_2)$, which corresponds to the condition of limit cycle. The system is unstable for $X < X_2$ and is stable for $X > X_2$. The stability of the limit cycle can be judged by the perturbation technique described below.

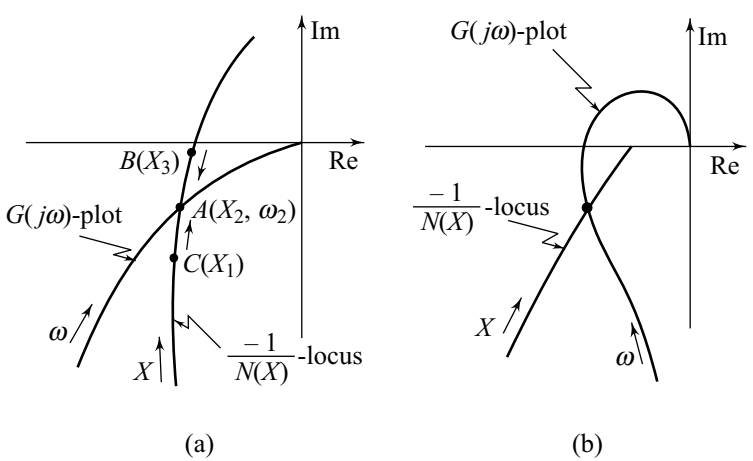


Fig. 9.13 Prediction and stability of limit cycles

Suppose that the system is originally operating at A under the state of a limit cycle. Assume that a slight perturbation is given to the system, so that the input to the nonlinear element increases to X_3, i.e., the operating point is shifted to B. Since B is in the range of stable operation, the amplitude of the input to the nonlinear element progressively decreases, and hence the operating point moves back towards A. Similarly, a perturbation which decreases the amplitude of input to the nonlinearity, shifts the operating point to C which lies in the range of unstable operation. The input amplitude now progressively increases and the operating point again returns to A. Therefore, the system has a stable limit cycle at A.

Figure 9.13b shows the case of an unstable limit cycle. For systems having $G(j\omega)$ plots and $-1/N(X)$ loci as shown in Figs 9.14a and 9.14b, there are two limit cycles; one stable and the other unstable.

Describing function method usually gives sufficiently accurate information about stability and limit cycles. This analysis is invariably followed by a simulation study.

The transfer function $G(s)$ in Fig. 9.12, when converted to state variable formulation, takes the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t);\ \mathbf{x}(0) \triangleq \mathbf{x}^0$$
$$y(t) = \mathbf{c}\mathbf{x}(t)$$

where

$\mathbf{x}(t) = n \times 1$ state vector for nth order $G(s)$;
$u(t)$ = input to $G(s)$;
$y(t)$ = output of $G(s)$;
$\mathbf{A} = n \times n$ matrix;
$\mathbf{b} = n \times 1$ column matrix; and
$\mathbf{c} = 1 \times n$ row matrix.

In Appendix A we use MATLAB Software SIMULINK to obtain response of nonlinear systems of the form given in Fig. 9.12 with zero reference input and initial state $\mathbf{x}^0$.

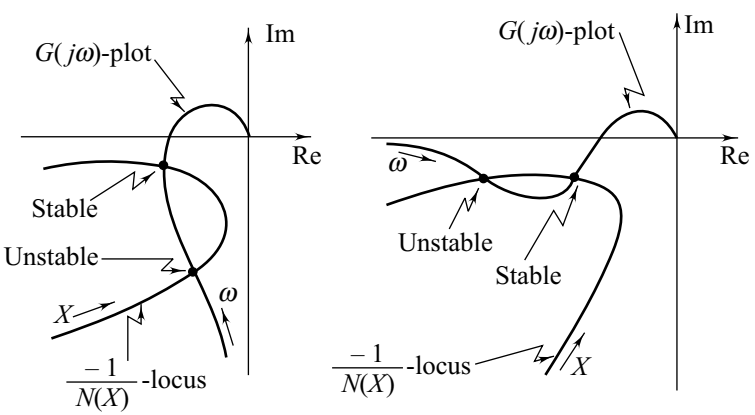


Fig. 9.14 Prediction and stability of limit cycles

Example 9.1

Let us investigate the stability of a with system on–off controller, shown in Fig. 9.15. Using the describing function of an on–off nonlinearity given in Table 9.2, we have

$$-\frac{1}{N(E)} = -\frac{\pi E}{4} \tag{9.18}$$

where E is the maximum amplitude of the sinusoidal signal e. Figure 9.16 shows the locus of $-1/N(E)$ as a function of E, and the plot of $G(j\omega)$ for $K = 5$. Equation (9.17) is satisfied at A since the two graphs intersect at this point.

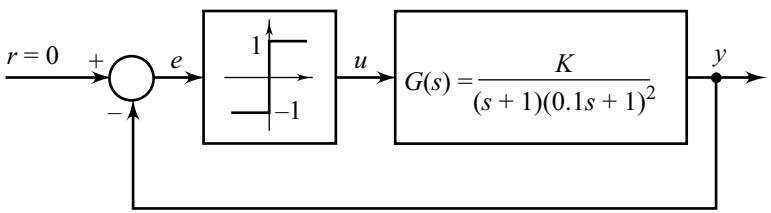


Fig. 9.15 A system with on–off controller

The point of intersection on the $G(j\omega)$ plot gives a numerical value ω_1 for the frequency of the limit cycle; whereas, the same point on the $-1/N(E)$ locus gives us the predicted amplitude E_1 of the oscillation. As an observer traverses the $G(j\omega)$ plot in the direction of increasing ω, the portion O-A of the $-1/N(E)$ locus lies to its right and the portion A-C lies to its left. Using the arguments presented previously, we can conclude that the limit cycle is a stable one.

Since $-1/N(E)$ is a negative real number, it is clear that intersection occurs at $-180°$ phase angle. The frequency ω_1 that gives $\angle G(j\omega_1) = -180°$ is 10.95 rad/sec. Furthermore, at point A

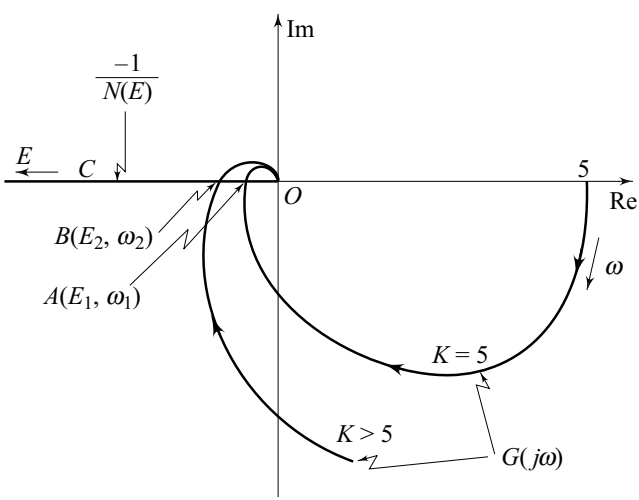


Fig. 9.16 Stability analysis of the system in Fig. 9.15

$$|G(j\omega_1)| = \left| -\frac{1}{N(E_1)} \right|$$

At $\omega_1 = 10.95$, $|G(j\omega_1)| = 0.206$ and, therefore, (refer to Eqn. (9.18)).

$$\left| -\frac{1}{N(E_1)} \right| = \frac{\pi E_1}{4} = 0.206$$

This gives $E_1 = 0.262$.

The describing function analysis, thus predicts a *limit cycle* (sustained oscillation)

$$y(t) = -e(t) = -0.262 \sin 10.95t$$

For $K > 5$, the intersection point shifts to B (Fig. 9.16) resulting in a limit cycle of amplitude $E_2 > E_1$ and frequency $\omega_2 = \omega_1$. It should be observed that the system has a limit cycle for all positive values of gain K.

To gain some further insight into on–off control behavior and describing function analysis, let us modify the system of Fig. 9.15 by letting the linear portion be of second-order with

$$G(s) = \frac{5}{(s+1)(0.1s+1)}$$

Figure 9.17 shows the plot of $G(j\omega)$ superimposed on the locus of $-1/N(E)$. The intersection of the two graphs is now impossible, since the phase angle of neither $G(j\omega)$ nor $-1/N(E)$, can be more lagging than –180°. Describing function analysis thus seems to predict no limit cycling, whereas, the fact that the control signal u must be either +1.0 or –1.0 dictates that the system oscillate. One possible interpretation to this analysis would be that the second-order linear system provides less of the low-pass filtering assumed in the describing function method, than did the third-order system and thus the approximation has become inaccurate, to the point of predicting no limit cycle when actually one occurs. Another interpretation would be that the curves actually do 'intersect' at the origin, predicting a limit cycle of infinite frequency and infinitesimal amplitude. This latter interpretation, even though it predicts a physically impossible result, agrees with the rigorous mathematical solution of the differential equations: for some nonzero initial value of y, we find that $y(t)$ oscillates about zero, with ever-decreasing amplitude and ever-increasing frequency. We will examine this solution on the phase plane in a later section.

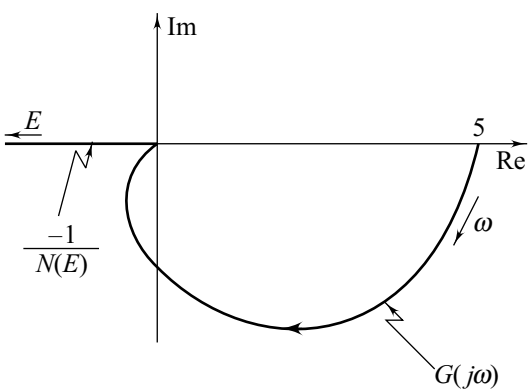


Fig. 9.17 Describing function analysis of a system with second–order plant and on–off controller

Let us now modify the system of Fig. 9.15 by giving the controller a deadzone Δ, as shown in Fig. 9.18. The $-1/N(E)$ locus for this type of controller is given by Fig. 9.8b. Plots of $G(j\omega)$ for different values of K, superimposed on $-1/N(E)$ locus, are shown in Fig. 9.19. From this figure, we observe

that for $K = K_1$, the $G(j\omega)$ plot crosses the negative real axis at a point to the right of $-\pi\Delta/2$, such that no intersection takes place between the graphs of $G(j\omega)$ and $-1/N(E)$, and therefore, no limit cycle results. With such a gain, the $-1/N(E)$ locus lies entirely to the left of the $G(j\omega)$ plot; the system is, therefore, stable, i.e., it has effectively positive damping.

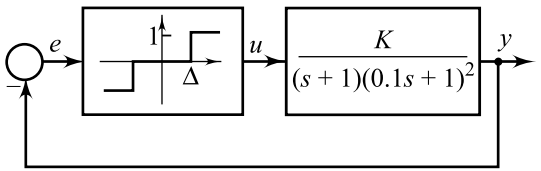


Fig. 9.18 A system controlled by on–off controller with deadzone

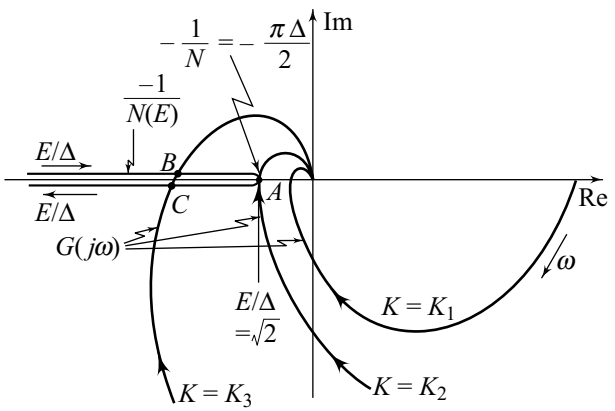


Fig. 9.19 Stability analysis of the system of Fig. 9.18

If the gain K is now increased to a value K_2, such that the $G(j\omega)$ plot intersects the $-1/N(E)$ locus at the point A (i.e., on the negative real axis at $-\pi\Delta/2$), then there exists a limit cycle. Now, suppose that the system is operating at the point A. Any increase in the amplitude of E takes the operating point to the left so that it is not enclosed by the $G(j\omega)$ plot, which means that the system has positive damping. This reduces E till the operating point comes back to A. Any decrease in the amplitude of E again takes the operating point to the left of the $G(j\omega)$ plot, i.e., the system has positive damping which further reduces E, finally driving the system to rest. Since random disturbances are always present in any system, the system under discussion cannot remain at A. Therefore, the limit cycle represented by A is unstable.

When the gain K is further increased to K_3, the graphs of $G(j\omega)$ and $-1/N(E)$ intersect at two points B and C. By arguments similar to those advanced earlier, it can be shown that the point B represents an unstable limit cycle and C represents a stable limit cycle. It may be noted that though the points B and C lie at the same place on the negative real axis, they belong to different values of E/Δ.

It is also clear that limit cycling is predicted only for deadzone Δ smaller than the value given by

$$\frac{\pi\Delta}{2} = |G(j\omega_1)|$$

where ω_1 is the frequency at which the plot $G(j\omega)$ intersects the negative real axis. A deadzone in on–off controllers appears to be a desirable feature to avoid limit cycling. However, as we shall see later in this chapter, a large value of Δ would cause the steady-state performance of the system to deteriorate.

Example 9.2

Figure 9.20a shows a block diagram for a servo system consisting of an amplifier, a motor, a gear train, and a load (gear 2 shown in the diagram includes the load element). It is assumed that the inertia of the gears and load element is negligible compared with that of the motor, and backlash exists between gear 1 and gear 2. The gear ratio between gear 1 and gear 2 is unity.

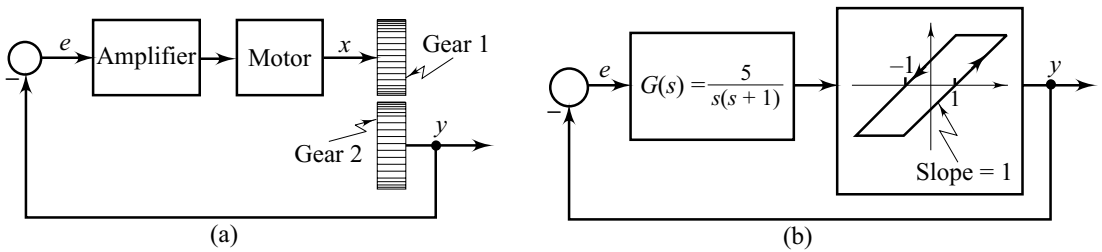


Fig. 9.20 A servo system with backlash in gears

The transfer function of the amplifier-motor combination, is given by $5/s(s + 1)$ and the backlash amplitude is given as unity ($H = 1$).

From the problem statement, the block diagram of the system may be redrawn as shown in Fig. 9.20b. Let us investigate the stability of this system. The $-1/N(X)$ locus for the backlash nonlinearity is given by Fig. 9.10 (Table 9.1). Plot of $G(j\omega)$ superimposed on $-1/N(X)$ locus is shown in Fig. 9.21. As seen from this figure, there are two intersections of the two loci. Applying the stability test for the limit cycle reveals that point A corresponds to a stable limit cycle and point B corresponds to an unstable limit cycle. The stable limit cycle has a frequency of 1.6 rad/sec and an amplitude of 2 (the unstable limit cycle cannot physically occur). To avoid limit-cycle behavior, the gain of the amplifier must be decreased sufficiently, so that the entire $G(j\omega)$ plot lies to the left of $-1/N(X)$ locus.

Note that checking for an intersection must be done graphically/numerically, since no analytical solution for limit-cycle amplitude or frequency is possible. A computer program that tabulates $G(j\omega)$ and $-1/N(X)$ is useful in searching for intersections and is not difficult to write. Once the general region of an intersection is found, we can use smaller increments of H/X and ω to pinpoint the intersection, as accurately as we wish.

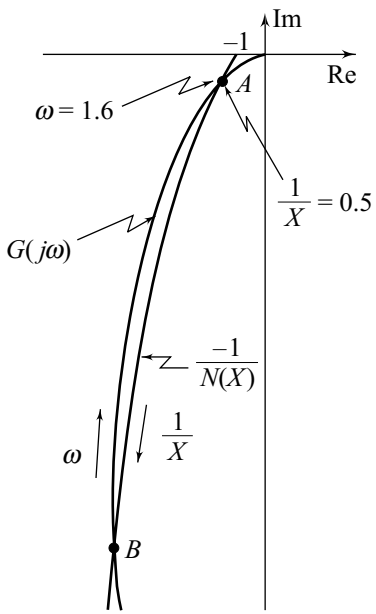


Fig. 9.21 Stability analysis of the system of Fig. 9.20

9.7 CONCEPTS OF PHASE-PLANE ANALYSIS

The free motion of any second-order nonlinear system can always be described by an equation of the form

$$\ddot{y} + g(y, \dot{y})\dot{y} + h(y, \dot{y})y = 0 \tag{9.19}$$

The state of the system, at any moment, can be represented by a point of coordinates $(y, \dot{y})$ in a system of rectangular coordinates. Such a coordinate plane is called a 'phase plane'.

In terms of the state variables

$$x_1 = y,\ x_2 = \dot{y}, \tag{9.20a}$$

second-order system (9.19) is equivalent to the following canonical set of state equations:

$$\begin{aligned}\dot{x}_1 &= \frac{dx_1}{dt} = x_2 \\ \dot{x}_2 &= \frac{dx_2}{dt} = -g(x_1, x_2)x_2 - h(x_1, x_2)x_1\end{aligned} \tag{9.20b}$$

By division, we obtain a first-order differential equation relating the variables x_1 and x_2:

$$\frac{dx_2}{dx_1} = -\frac{g(x_1, x_2)x_2 + h(x_1, x_2)x_1}{x_2} \tag{9.21}$$

Thus, we have eliminated the independent variable t from the set of first-order differential equations given by (9.20b). In Eqn. (9.21), we consider x_1 and x_2 as independent and dependent variables, respectively.

For a given set of initial conditions $\{x_1(0), x_2(0)\}$, the solution to Eqn. (9.21) may be represented by a single curve in the phase plane, for which the coordinates are x_1 and x_2. The curve traced out by the state point $\{x_1(t), x_2(t)\}$, as time t is varied from 0 to ∞, is called the *phase trajectory*, and the family of all possible curves for different initial conditions is called the *phase portrait*. Normally, a finite number of trajectories, defined in a finite region, is considered a portrait.

One may obviously raise the question that when time solutions $x_1(t)$ and $x_2(t)$, as time t is varied from 0 to ∞, may be obtained by direct integration of Eqns (9.20b) analytically or numerically, where is the necessity of drawing phase portraits? In fact, as we shall see, the phase portraits provide a powerful qualitative aid for investigating system behavior and the design of system parameters, to achieve a desired response. Furthermore, the existence of limit cycles is sharply brought into focus by the phase portrait.

Figure 9.22a shows the output response, and the corresponding phase trajectory, for a linear second-order servo system described by the differential equation

$$\ddot{y} + 2\zeta\dot{y} + y = 0;\ y(0) = y^0,\ \dot{y}(0) = 0,\ 0 < \zeta < 1$$

In terms of the state variables $x_1 = y$ and $x_2 = \dot{y}$, the system model is given by the equations

$$\dot{x}_1 = x_2;\quad \dot{x}_2 = -2\zeta x_2 - x_1;\ x_1(0) = y^0,\ x_2(0) = 0$$

The origin of the phase plane ($x_1 = 0$, $x_2 = 0$) is the *equilibrium point* of the system since, at this point, the derivatives $\dot{x}_1$ and $\dot{x}_2$ are zero (the system continues to lie at the equilibrium point unless otherwise disturbed). The nature of the transient can be readily inferred from the phase trajectory of Fig. 9.22; starting from the point P, i.e., with initial deviation but no initial velocity, the system returns to rest, i.e., to the origin, with damped oscillatory behavior.

Consider now the well-known *Van der Pol's differential equation* (refer to Eqn. (9.1))

$$\ddot{y} - \mu(1 - y^2)\dot{y} + y = 0$$

which describes physical situations in many nonlinear systems. It terms of the state variables $x_1 = y$ and $x_2 = \dot{y}$, we obtain

$$\dot{x}_1 = x_2;\ \dot{x}_2 = \mu(1 - x_1^2)x_2 - x_1$$

Origin of the phase plane is the equilibrium point of the system. Figure 9.23 shows phase portraits for (i) $\mu > 0$; and (ii) $\mu < 0$. In the case of $\mu > 0$, we observe that for large values of $x_1(0)$, the system response

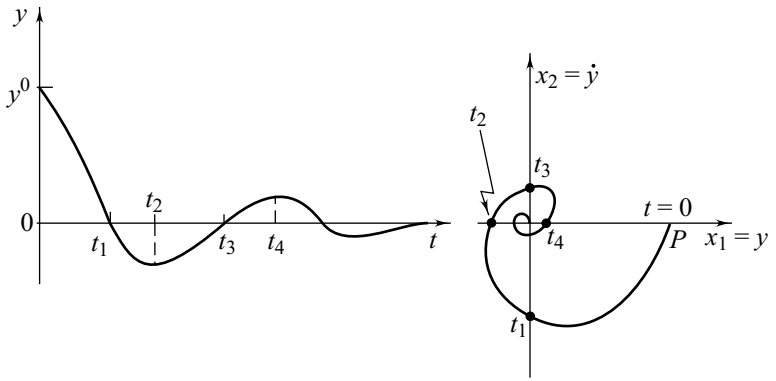


Fig. 9.22 A second-order linear system on the phase plane

is damped and the amplitude of $x_1(t) = y(t)$ decreases till the system state enters the limit cycle, as shown by the outer trajectory. On the other hand, if initially $x_1(0)$ is small, the damping is negative, hence the amplitude of $x_1(t) = y(t)$ increases till the system state enters the limit cycle, as shown by the inner trajectory. The limit cycle is a stable one, since the paths in its neighborhood converge towards the limit cycle. Figure 9.23 shows an unstable limit cycle for $\mu < 0$.

The *phase plane* for second-order systems is indeed a special case of *phase space* or *state space* defined for *n*th-order systems. Much work has been done to extend this approach of analysis to third-order systems. Though a phase trajectory for a third-order system can be graphically visualized through its projections on two planes, say (x_1, x_2) and (x_2, x_3) planes, this complexity causes the technique to lose its major power of quick graphical visualization of the total system response. The phase trajectories are, therefore, generally restricted to second-order systems only.

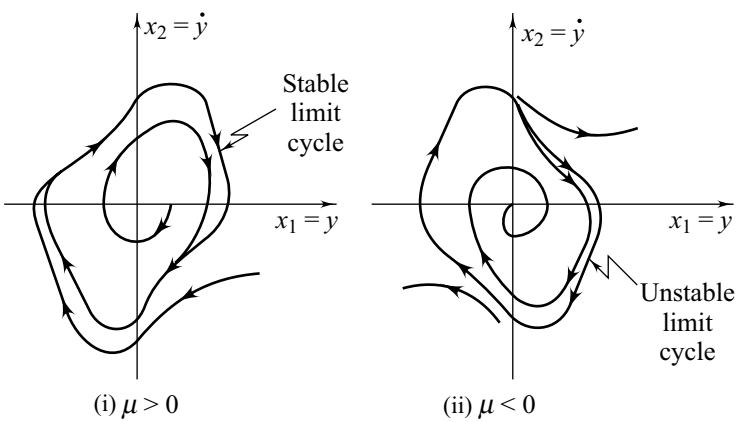


Fig. 9.23 A second-order nonlinear system on the phase plane

For time-invariant systems, the entire phase plane is covered with trajectories with one, and only one, curve passing through each point of the plane, except for certain critical points through which, either infinite number or none of the trajectories pass. Such points (called *singular points*) are discussed later in Section 9.9.

If the parameters of a system vary with time, or if a time-varying driving function is imposed, two or more trajectories may pass through a single point in a phase plane. In such cases, the *phase portrait* becomes complex and more difficult to work with and interpret. Therefore, the use of phase-plane analysis is restricted to second-order systems with constant parameters and constant or zero input. However, it may be mentioned that investigators have made fruitful use of the phase-plane method in investigating second-order time-invariant systems under simple time-varying inputs, such as ramp. Some simple time-varying systems have also been analyzed by this method. Our discussion will be limited to second-order time-invariant systems with constant or zero input.

From the above discussion, we observe that the phase-plane analysis applies primarily to systems described by second-order differential equations. In the case of feedback control systems, systems of order higher than the second, are likely to be well filtered and tractable by the describing-function method discussed earlier in this chapter. The two methods of the phase plane and of the describing function are, therefore, complementary to a large extent; each being available for the study of the systems which are most likely to be beyond the scope of the other.

9.8 CONSTRUCTION OF PHASE PORTRAITS

Today, phase portraits are routinely computer-generated. However, of course (as, for example, in the case of root locus for linear systems), it is still practically useful to learn how to roughly sketch phase portraits or quickly verify the plausibility of computer outputs.

For some special nonlinear systems, particularly piecewise linear systems (whose phase portraits can be constructed by piecing together the phase portraits of the related linear systems), phase portraits can be constructed analytically. Analytical methods are useful for systems modeled by differential equations that can be easily solved. If the system of differential equations cannot be solved analytically, we can use graphical methods. A number of graphical methods for constructing phase-plane trajectories are now available; we will describe in this section, the method of isoclines.

9.8.1 Analytical Method

Most nonlinear systems cannot be easily solved by analytical techniques. However for piecewise linear systems, an important class of nonlinear systems, this method can be conveniently used, as shown in the following examples.

Example 9.3

In this example, we consider a model of a satellite shown in Fig. 7.3. We assume that the satellite is rigid and is in a frictionless environment. It can rotate about the reference axis as a result of torque T applied to the satellite about its mass center by firing the thrusters ($T = Fd$). The system input is the applied torque T and the system output is the attitude angle θ. The satellite's moment of inertia is J. The input-output model of the system is

$$J\frac{d^2\theta}{dt^2} = T \tag{9.22a}$$

We assume that when the thrusters fire, the thrust is constant; that is, $T = A$, a constant greater than or less than zero. In terms of output variable $y\,(= \theta)$, we obtain the equation

$$J\ddot{y} = A \tag{9.22b}$$

In terms of the state variables $x_1 = y$ and $x_2 = \dot{y}$, the state equations become

$$\dot{x}_1 = x_2;\ \dot{x}_2 = \frac{A}{J} \tag{9.22c}$$

Elimination of t by division, yields the equation of the trajectories:

$$\frac{dx_2}{dx_1} = \frac{A}{Jx_2} \tag{9.23}$$

or

$$J\,x_2\,dx_2 = A\,dx_1$$

This equation is easily integrated; the general solution is

$$x_1(t) = \frac{Jx_2^2(t)}{2A} + C \tag{9.24a}$$

where C is a constant of integration and is determined by initial conditions, i.e.,

$$C = x_1(0) - \frac{Jx_2^2(0)}{2A} \tag{9.24b}$$

For an initial state point $(x_1(0), x_2(0))$, the trajectory is a parabola passing through the point $x_1 = C$ on the x_1-axis where C is defined by Eqn. (9.24b).

A family of parabolas in the (x_1, x_2) plane is shown in Fig. 9.24a for $A > 0$. As time t increases, each trajectory is described in the clockwise direction, as indicated by the arrows. The direction of the phase trajectories is dictated by the relationship $\dot{x}_1 = x_2$; x_1 increases with time in the upper half of the phase plane and the state point, therefore, moves from left to right ($\rightarrow$); in the lower half of the phase plane, x_1 decreases with time and the state point must, therefore, move from right to left ($\leftarrow$).

The time interval between two points of a trajectory is given by $\Delta t = \Delta x_1 / x_{2av}$. The trajectories may be provided with a time scale by means of this equation. This operation is, however, often unnecessary since the phase portrait is mainly used to display the general features of the system transients.

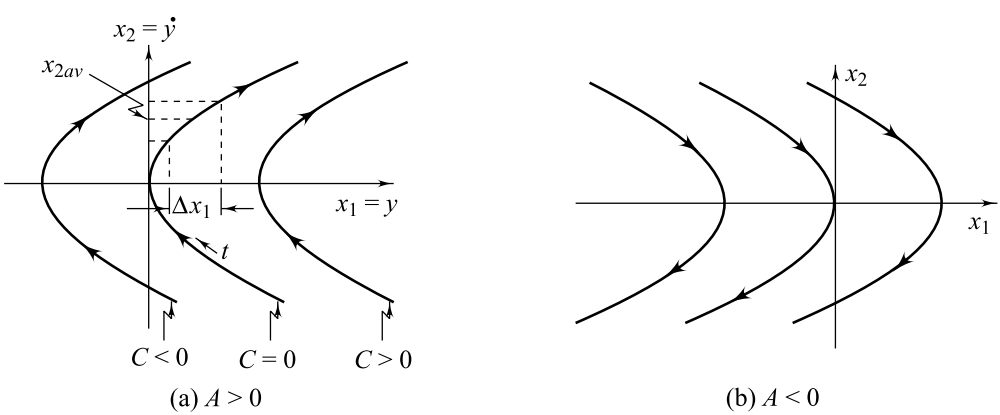


Fig. 9.24 Phase potraits for system (9.22)

The phase portrait for $A < 0$ is shown in Fig. 9.24b. In the special case of $A = 0$ (no driving torque), the integration of the trajectory equation (9.23) gives $x_2(t) = x_2(0)$. The trajectories are, therefore, straight lines parallel to x_1-axis.

Example 9.4

Consider now the equation

$$J\ddot{\theta} + B\dot{\theta} = T \tag{9.25a}$$

corresponding to a torque T driving a load comprising inertia J and viscous friction B. For a constant torque, the equation may be expressed as

$$\tau\ddot{y} + \dot{y} = A \tag{9.25b}$$

where $y = \theta$ is the system output and A represents normalized torque; a constant greater than or less than zero. The equivalent system is

$$\dot{x}_1 = x_2; \quad \tau\dot{x}_2 = A - x_2 \tag{9.25c}$$

Let us take a new variable z such that

$$A - x_2 = z;\ dx_2 = -dz$$

Eliminating the time variable by division, we obtain

$$\frac{1}{\tau}\frac{dx_1}{dz} = -\frac{A - z}{z} = 1 - \frac{A}{z}$$

This first-order equation is readily integrated.

$$\frac{1}{\tau}x_1 = z - A\ \ln z + C$$

or

$$\frac{1}{\tau}\,x_1(t) = A - x_2(t) - A\ \ln(A - x_2(t)) + C \tag{9.26a}$$

where the constant of integration C is determined by the initial conditions, i.e.,

$$C = \frac{1}{\tau}\,x_1(0) - A + x_2(0) + A\ \ln(A - x_2(0)) \tag{9.26b}$$

Therefore, the trajectory equation becomes

$$\frac{1}{\tau}\,(x_1 - x_1(0)) = -\,(x_2 - x_2(0)) - A\ \ln\left(\frac{A - x_2}{A - x_2(0)}\right) \tag{9.26c}$$

The phase portrait is shown in Fig. 9.25a for $A > 0$.

For the case of initial state point at the origin ($x_1(0) = x_2(0) = 0$), Eqn. (9.26c) reads

$$\frac{1}{\tau}x_1 = -x_2 - A\ln\left(\frac{A - x_2}{A}\right) \tag{9.26d}$$

The phase trajectory described by this equation is shown in Fig. 9.25a as the curve Γ_0. It is seen that the trajectory is asymptotic to the line $x_2 = A$, which is the final velocity.

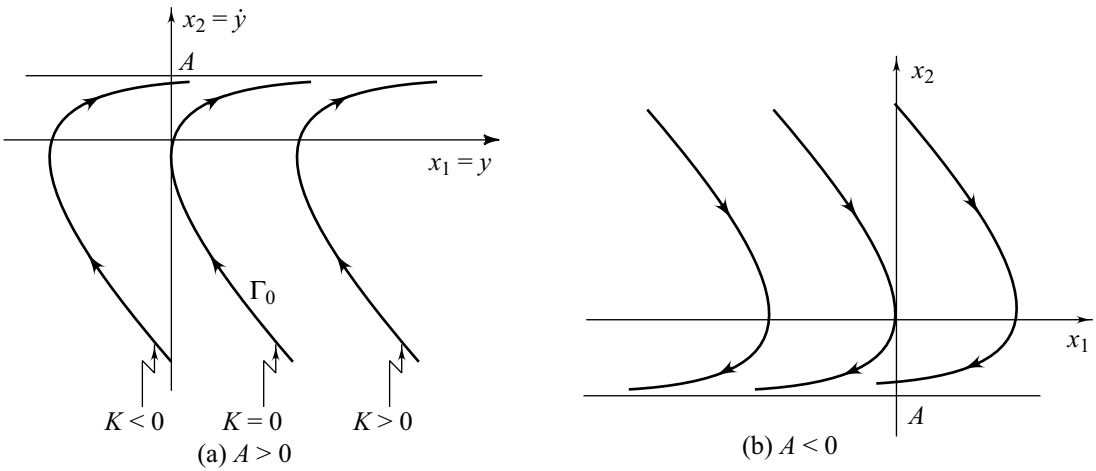


Fig. 9.25 Phase portraits for system (9.25)

For an initial state point ($x_1(0)$, $x_2(0)$), the trajectory will have the same shape as the curve Γ_0, except that it is shifted horizontally—so that it passes through the point ($x_1(0)$, $x_2(0)$). This is obvious from Eqn. (9.26c) which can be written as

$$\frac{1}{\tau}(x_1 - K) = -x_2 - A\ln\left(\frac{A - x_2}{A}\right)$$

where

$$K = x_1(0) + \tau x_2(0) + \tau A\ln\left(\frac{A - x_2(0)}{A}\right)$$

For an initial state point ($x_1(0)$, $x_2(0)$), the trajectory is Γ_0, shifted horizontally by K units. The phase portrait for $A < 0$ is shown in Fig. 9.25b. In the special case of $A = 0$, the phase portrait consists of a family of straight lines of slope $-1/\tau$.

9.8.2 The Method of Isoclines

Consider a time-invariant second-order system described by equations of the form (refer to Eqns (9.20b))

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x_1, x_2)\end{aligned} \tag{9.27}$$

The equation of the trajectories is

$$\frac{dx_2}{dx_1} = \frac{f(x_1, x_2)}{x_2} \tag{9.28}$$

At a point (x_1^*, x_2^*) in the phase plane, the slope m^* of the tangent to the trajectory can be determined from

$$\frac{f(x_1^*, x_2^*)}{x_2^*} = m^* \tag{9.29}$$

An *isocline* is defined to be the locus of the points corresponding to a given constant slope m of the trajectories, on the phase plane. All trajectories passing through the points on the curve

$$f(x_1, x_2) = mx_2 \tag{9.30}$$

will have the same tangent slope m at the points on the curve; the curve, thus, represents an isocline corresponding to trajectories of slope m. All trajectories crossing this isocline will have tangent slope m at the points on the isocline.

The idea of the method of isoclines is to construct several isoclines and a field of local tangents m. Then, the trajectory passing through any given point in the phase plane, is obtained by drawing a continuous curve following the directions of the field.

Consider the Van der Pol equation (refer to Eqn. (9.1))

$$\ddot{y} + \mu(y^2 - 1)\dot{y} + y = 0 \tag{9.31}$$

With $x_1 = y$ and and $x_2 = \dot{y}$, the equation of the trajectories becomes

$$\frac{dx_2}{dx_1} = \frac{-\mu(x_1^2 - 1)x_2 - x_1}{x_2}$$

Therefore, the points on the curve

$$\frac{-\mu(x_1^2 - 1)x_2 - x_1}{x_2} = m$$

all have the same slope m. The isocline equation becomes

$$x_2 = \frac{x_1}{(\mu - \mu x_1^2) - m}$$

By taking m of different values, different isoclines can be obtained. Short line segments are drawn on the isoclines to generate a field of tangent directions. A trajectory starting at any point can be constructed by drawing short lines from one isocline to another at average slope corresponding to the two adjoining isoclines, as shown in Fig. 9.26.

Of course, the construction is much simpler if isoclines are straight lines.

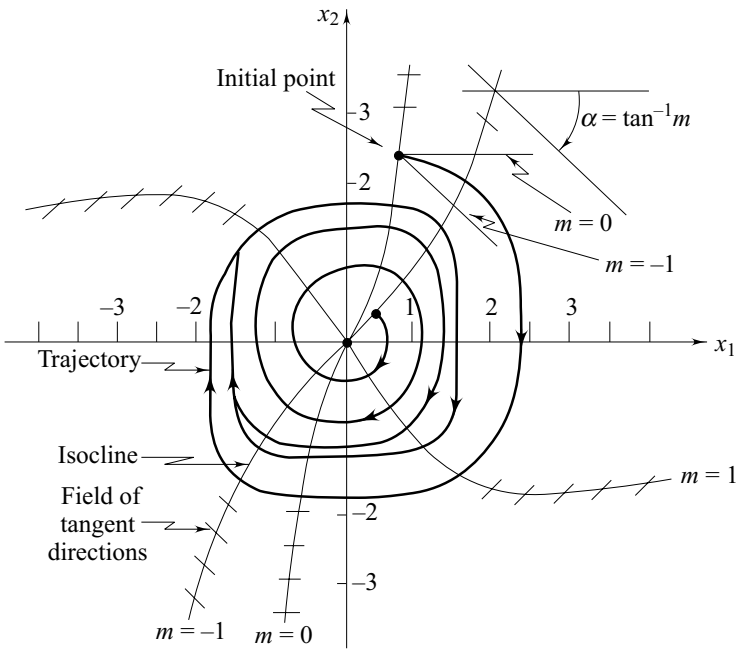


Fig. 9.26 Phase portrait of the Van der Pol equation

Example 9.5

The satellite in Example 9.3, is now placed in a feedback configuration in order to maintain the attitude θ at 0°. This feedback control system, called an attitude control system, is shown in Fig. 9.27. When θ is other than 0°, the appropriate thruster will fire to force θ towards 0°. When $\theta = x_1$ is greater than 0°, u (torque T) = $-U$, and the trajectories of Fig. 9.24b (corresponding to $A < 0$) apply. When $\theta = x_1$ is less than 0°, u (torque T) = U, and the trajectories of Fig. 9.24a (corresponding to $A > 0$) apply. Note that the switching of u occurs at $x_1 = 0$. Thus the line $x_1 = 0$ (the x_2-axis) is called the *switching line*. From this discussion we see that Fig. 9.28a illustrates a typical trajectory for the system corresponding to the initial condition (x_1^0, x_2^0). The system response is thus a periodic motion. Figure 9.28b shows many closed curves for different initial conditions.

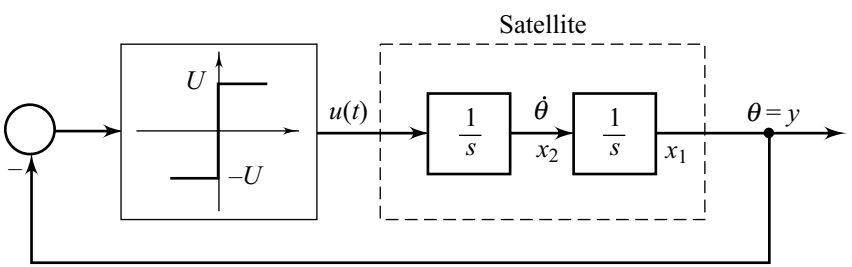


Fig. 9.27 A satellite-attitude control system

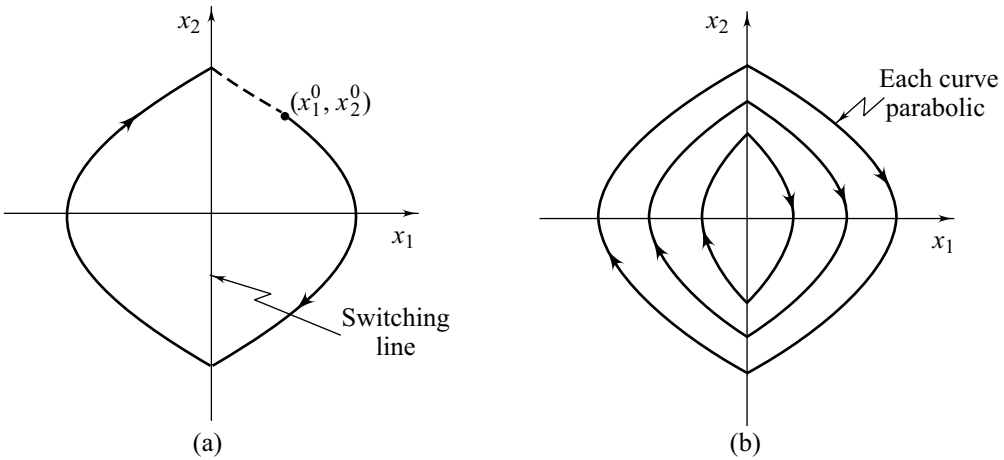


Fig. 9.28 Typical trajectories for the system of Fig. 9.27

By controlling the switching line in the phase plane, we can control the performance of the attitude control system. This simple control strategy leads to a robust nonlinear control structure: the *variable structure sliding mode control*. The details will follow later in this chapter.

In the following, we obtain the phase portrait of the closed-loop system of Fig. 9.27 using the method of isoclines. The purpose here is to illustrate the method of isoclines.

The state equations are

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -U \operatorname{sgn} x_1$$

where

$$\operatorname{sgn} x_1 = \begin{cases} 1; x_1 > 0 \\ -1; x_1 < 0 \end{cases}$$

Then

$$m = \frac{dx_2}{dx_1} = \frac{-U \operatorname{sgn} x_1}{x_2}$$

Suppose that U is normalized to a value of unity for convenience. Then

$$x_2 = -\frac{1}{m} \operatorname{sgn} x_1$$

For $x_1 > 0$, sgn $x_1 = 1$, and the isocline equation is

$$x_2 = -\frac{1}{m}\,; x_1 > 0$$

For $x_1 < 0$, sgn $x_1 = -1$, and the isocline equation is

$$x_2 = \frac{1}{m}; x_1 < 0$$

Given in Fig. 9.29, is the phase plane showing the isoclines and a typical phase trajectory. Note the parabolic shape, as was determined analytically earlier in this example.

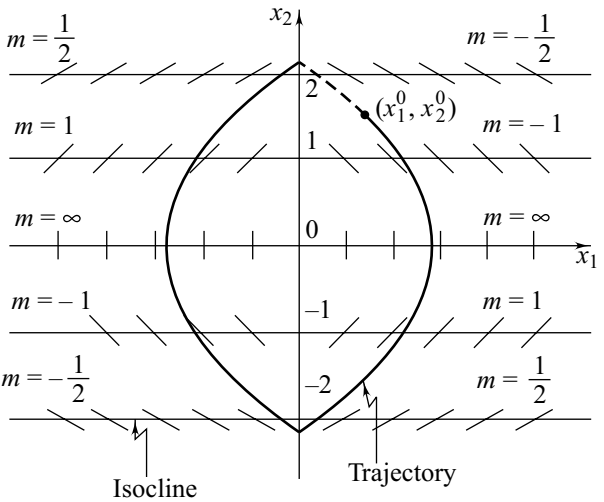


Fig. 9.29 The isoclines and a typical trajectory for the system of Fig. 9.27

9.9 SYSTEM ANALYSIS ON THE PHASE PLANE

In the phase-plane analysis of nonlinear systems, two points should be kept in mind:

- Phase-plane analysis of nonlinear systems is related to that of linear systems because the local behavior of a nonlinear system can be approximated by a linear system behavior.
- Yet, nonlinear systems can display much more complicated patterns on the phase plane, such as multiple equilibrium points, and limit cycles.

Consider a time-invariant second-order system described by equations of the form (refer to Eqns (9.27))

$$\dot{x}_1 = x_2 \; ; \; \dot{x}_2 = f(x_1, x_2) \tag{9.32}$$

Elimination of independent variable t gives the equation of the trajectories of phase plane (refer to Eqn. (9.28)):

$$\frac{dx_2}{dx_1} = \frac{f(x_1, x_2)}{x_2} \tag{9.33}$$

In this equation, x_1 and x_2 are independent and dependent variables, respectively. Integration of the equation, analytically, graphically or numerically, for various initial conditions, yields a family of phase trajectories which displays the general features of the system transients.

9.9.1 Singular Points

Every point (x_1, x_2) of the phase plane has associated with it, the slope of the trajectory which passes through that point. The slope m at the point (x_1, x_2) is given by the equation

$$m = \frac{dx_2}{dx_1} = \frac{f\left(x_1, x_2\right)}{x_2}$$

With the function $f(x_1, x_2)$ assumed to be single valued, there is usually a definite value for this slope at any given point in phase plane. This implies that the phase trajectories will not intersect. The only exceptions are the *singular points* at which the trajectory slope is indeterminate:

$$\frac{dx_2}{dx_1} = \frac{0}{0} = \frac{f(x_1, x_2)}{x_2} \tag{9.34a}$$

Many trajectories may intersect at such points. This indeterminacy of the slope accounts for the adjective 'singular'.

Singular points are very important features on the phase plane. Examination of the singular points can reveal a great deal of information about the properties of a system. In fact, the stability of linear systems is uniquely characterized by the nature of their singular points. For nonlinear systems, besides singular points, there may be more complex features such as limit cycles.

We need to know the following:

(i) Where will the singular points be and how many will be there?
(ii) What is the behavior of trajectories (i.e., the system) in the vicinity of a singular point?

The first question is answered by our definition of the singular point. There will be singular points at all the points of the phase plane for which the slope of the trajectory is undefined. These points are given by the solution of the equations

$$x_2 = 0; f(x_1, x_2) = 0 \tag{9.34b}$$

Singular points of the nonlinear system (9.32), thus, lie on the x_1-axis of the phase plane.

Since at singular points on the phase plane, $\dot{x}_1 = \dot{x}_2 = 0$, these points, in fact, correspond to the *equilibrium states* of the nonlinear system. We know a nonlinear system often has multiple equilibrium states.

To determine the behavior of the trajectories in the vicinity of a singular point (equilibrium state of the nonlinear system), we first linearize the nonlinear equations at the singular point, and then determine the nature of phase trajectories around the singular point by linear system analysis. If the singular point of interest is not at the origin, by defining the difference between the original state and the singular point as a new set of state variables, one can always shift the singular point to the origin. Therefore, without loss of generality, we can simply consider Eqns (9.32) with a singular point at **0**. Using Taylor series expansion, Eqns (9.32) can be rewritten as

$$\dot{x} = x_2$$
$$\dot{x}_2 = ax_1 + bx_2 + g_2(x_1, x_2)$$

where $g_2(\bullet)$ contains higher-order terms.

In the vicinity of the origin, the higher-order terms can be neglected and, therefore, the nonlinear system trajectories essentially satisfy the linearized equations

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = ax_1 + bx_2$$

Transforming these equations into a scalar second-order equation, we get

$$\ddot{x}_1 = ax_1 + b\dot{x}_1$$

Therefore, we will simply consider the second-order linear system described by

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = 0 \tag{9.35a}$$

The characteristic roots of this equation are assumed to be λ_1 and λ_2:

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = (s - \lambda_1)(s - \lambda_2) = 0 \tag{9.35b}$$

The corresponding canonical state model is

$$\dot{x}_1 = x_2; \quad \dot{x}_2 = -2\zeta\omega_n x_2 - \omega_n^2 x_1 \tag{9.35c}$$

and the differential equation of the trajectories is

$$\frac{dx_2}{dx_1} = \frac{-2\zeta\omega_n x_2 - \omega_n^2 x_1}{x_2} \tag{9.35d}$$

By inspection of this equation, it is easily seen that at $x_1 = x_2 = 0$, the slope dx_2/dx_1 is indeterminate:

$$\frac{dx_2}{dx_1} = \frac{0}{0}$$

In the following, we discuss the behavior of the trajectories in the vicinity of this point with undefined slope (the singular point).

According to the values of λ_1 and λ_2, one is led to distinguish between the six types of singular points shown in Fig. 9.30. Let us examine each of these cases in detail.

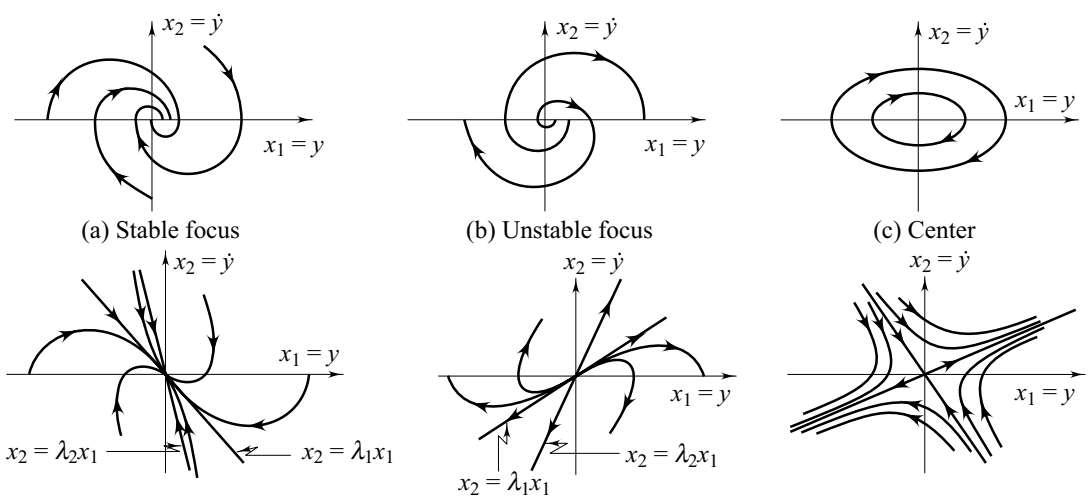
Stable System with Complex Roots

$$\lambda_1 = -\alpha + j\beta, \quad \lambda_2 = -\alpha - j\beta; \quad \alpha > 0, \beta > 0$$

The response

$$y(t) = C_1 e^{-\alpha t}\sin(\beta t + C_2) \tag{9.36}$$

where the constants C_1 and C_2 are determined by the initial conditions.



(a) Stable focus (b) Unstable focus (c) Center

(d) Stable node (e) Unstable node (f) Saddle point

Fig. 9.30 Phase portraits for system (9.35)

Using Eqn. (9.36), we can construct a phase portrait on the (x_1, x_2)-plane with $x_1 = y$ and $x_2 = \dot{y}$. A typical phase trajectory is shown in Fig. 9.30a which is a logarithmic spiral into the singular point. This type of singular point is called a *stable focus*.

Unstable System with Complex Roots

$$\lambda_1 = \alpha + j\beta, \quad \lambda_2 = \alpha - j\beta; \quad \alpha > 0, \beta > 0$$

The response
$$y(t) = C_1 e^{\alpha t} \sin(\beta t + C_2) \tag{9.37}$$

The transient is an exponentially increasing sinusoid; the phase trajectory on the $(x_1 = y, x_2 = \dot{y})$-plane is a logarithmic spiral expanding out of the singular point (Fig. 9.30b). This type of singular point is called an *unstable focus*.

Marginally Stable System with Complex Roots

$$\lambda_1 = j\beta, \lambda_2 = -j\beta; \quad \beta > 0$$

The response
$$y(t) = C_1 \sin(\beta t + C_2) \tag{9.38}$$

The phase trajectories are closed curves (elliptical), concentric with the singular point (Fig. 9.30c). This type of singular point is called a *center*, or a *vortex*.

Stable System with Real Roots

Assume that λ_1 and λ_2 are two real, distinct roots in the left half of the s-plane; λ_1 is the root with the smaller modulus. The response

$$y(t) = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t} \tag{9.39a}$$

It is an overdamped system. The phase portrait in the vicinity of the singular point on the $(x_1 = y, x_2 = \dot{y})$-plane is shown in Fig. 9.30d. Such a singular point is called a *stable node*.

The phase portrait has two straightline trajectories, defined by the equations

$$x_2(t) = \lambda_1 x_1(t); \quad x_2(t) = \lambda_2 x_1(t) \tag{9.39b}$$

It can easily be verified that these trajectories satisfy the differential equation of the given system.

The transient term $e^{\lambda_2 t}$ decays faster than the term $e^{\lambda_1 t}$. Therefore, as t increases indefinitely, $x_1 \to C_1 e^{\lambda_1 t} \to 0$, and $x_2 \to \lambda_1 C_1 e^{\lambda_1 t} \to 0$, so that all the trajectories are tangential at the origin to the straightline trajectory $x_2(t) = \lambda_1 x_1(t)$. The other straightline trajectory, $x_2(t) = \lambda_2 x_1(t)$, is described only if the initial conditions are such that $x_2(0) = \lambda_2 x_1(0)$.

For stable systems with repeated real roots, the two straightline trajectories coalesce into a single trajectory, again with the slope determined by the root value.

Unstable System with Positive Real Roots

Assume that λ_1 and λ_2 are two real distinct roots in the right half of the s-plane; λ_1 is the smaller root. The phase portrait in the vicinity of the singular point on the $(x_1 = y, x_2 = \dot{y})$-plane is shown in Fig. 9.30e. Such a singular point is called an *unstable node*.

All trajectories emerge from the singular point and go to infinity. The trajectories are tangential at the origin to the straightline trajectory, $x_2(t) = \lambda_1 x_1(t)$.

Unstable System with One Negative Real Root and One Positive Real Root

The phase portrait in the vicinity of the singular point on the $(x_1 = y, x_2 = \dot{y})$-plane is shown in Fig. 9.30f. Such a singular point is called a *saddle*.

There are two straightline trajectories with slopes defined by the root values. The straightline due to the negative root, provides a trajectory that *enters* the singular point, while the straightline trajectory due to the positive root, leaves the singular point. All other trajectories approach the singular point adjacent to the incoming straightline, then curve away and leave the vicinity of the singular point, eventually approaching the second straightline asymptotically.

Example 9.6

Consider the nonlinear system shown in Fig. 9.31. The nonlinear element is an on–off controller with deadzone whose characteristics are shown in Fig. 9.31.

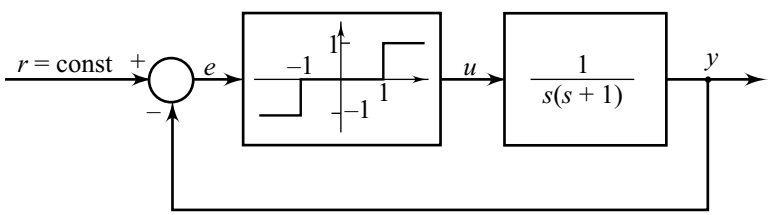


Fig. 9.31 A system controlled by on–off controller with deadzone

The differential equation governing the dynamics of the system is given by

$$\ddot{y} + \dot{y} = u; \qquad \text{or} \qquad \ddot{e} + \dot{e} = -\phi(e) \tag{9.40}$$

where $e = r - y$; r is constant, and

$$\phi(e) = \begin{cases} +1\,; & e > 1 \\ 0\,; & -1 < e < 1 \\ -1\,; & e < -1 \end{cases}$$

Choosing the state variables $x_1 = e$ and $x_2 = \dot{e}$, we obtain the following first-order equations:

$$\dot{x}_1 = x_2; \quad \dot{x}_2 = -x_2 - \phi(x_1)$$

These equations are same as Eqns (9.25c) with $\tau = 1$ and $A = -\phi(x_1)$.

The phase plane may be divided into three regions:

(i) *Region* I (defined by $x_1 > 1$): The trajectories in this region are given by the equation (refer to Eqn. (9.26c): $\tau = 1$, $A = -1$)

$$x_1 - x_1(0) = -(x_2 - x_2(0)) + \ln\left(\frac{1 + x_2}{1 + x_2(0)}\right) \tag{9.41a}$$

The trajectories are asymptotic to the ordinate -1.

(ii) *Region* II (defined by $-1 < x_1 < 1$): The trajectories in this region are given by the equation (refer to Eqn. (9.26c): $\tau = 1$, $A = 0$)

$$x_1 - x_1(0) = -(x_2 - x_2(0)) \tag{9.41b}$$

The trajectories are straightlines of slope -1.

(iii) *Region* III (defined by $x_1 < -1$): The trajectories in this region are given by the equation (refer to Eqn. (9.26c): $\tau = 1$, $A = 1$)

$$x_1 - x_1(0) = -(x_2 - x_2(0)) - \ln\left(\frac{1 - x_2}{1 - x_2(0)}\right) \tag{9.41c}$$

The trajectories are asymptotic to the ordinate $+1$.

For a step input $r = 3$ and zero initial conditions, the initial point of the phase trajectory is located at P in Fig. 9.32. The figure also shows a phase trajectory, constructed using Eqns (9.41).

It is important to note that a small deadzone region is not always undesirable in on–off controllers. Let us investigate the behavior of the system of Fig. 9.31 using on–off with (no deadzone) as a controller. For such a controller, the width of region II (corresponding to deadzone) in the phase plane, reduces to zero. The phase trajectory of such a system with $r = 3$ is shown in Fig. 9.33 : $e(t)$ oscillates about the origin, with ever-decreasing amplitude and ever-increasing frequency.

Comparison of Figs 9.32 and 9.33 reveals that deadzone in on–off controller characteristic helps to reduce system oscillations, thereby reducing settling time. However, the on–off controller with deadzone drives the system to a point within the deadzone width. A large deadzone would of course cause the steady-state performance of the system to deteriorate.

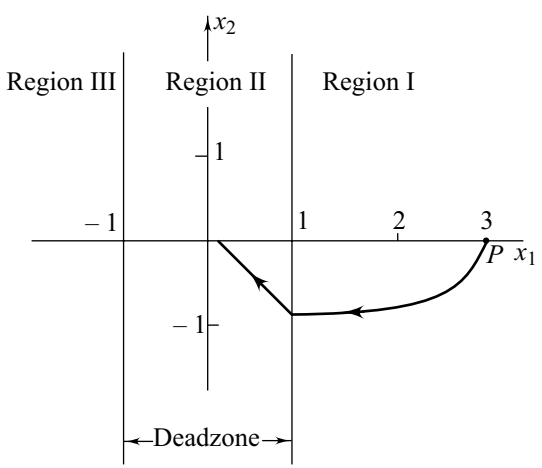


Fig. 9.32 A typical trajectory for the system in Fig. 9.31

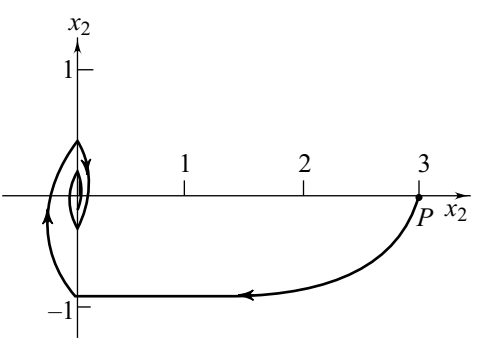


Fig. 9.33 Phase trajectory for the system in Fig. 9.31 when the deadzone is absent

Example 9.7

Let us investigate the performance of a second-order position control system with Coulomb friction. Figure 9.34 is a model for a motor position servo with Coulomb friction on the motor shaft. The dynamics of the system is described by the following differential equation:

$$Ke - T_c \operatorname{sgn}(\dot{y}) = J\ddot{y} + B\dot{y}$$

where T_c is the Coulomb frictional torque.

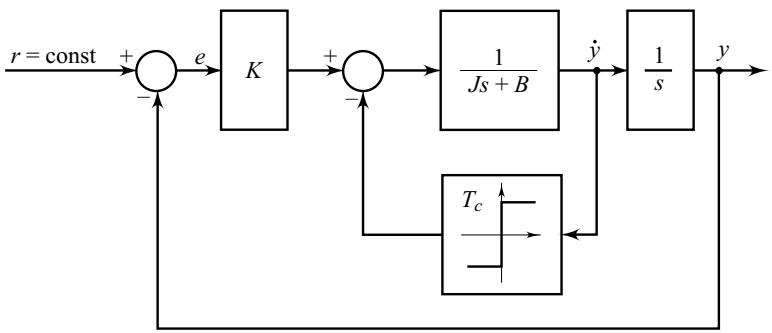


Fig. 9.34 A motor position servo with Coulomb friction on the motor shaft

For constant input r, $\dot{y} = -\dot{e}$ and $\ddot{y} = -\ddot{e}$.

Therefore,

$$J\ddot{e} + B\dot{e} + T_c \operatorname{sgn}(\dot{e}) + Ke = 0$$

or

$$\frac{J}{B}\ddot{e} + \dot{e} + \frac{T_c}{B}\operatorname{sgn}(\dot{e}) + \frac{K}{B}e = 0 \tag{9.42}$$

Letting $J/B = \tau$, we get

$$\tau\ddot{e} + \dot{e} + \frac{T_c}{B}\operatorname{sgn}(\dot{e}) + \frac{K}{B}e = 0 \tag{9.43}$$

In terms of state variables

$$x_1 = e;\ x_2 = \dot{e}$$

we get the following description of the system:

$$\dot{x}_1 = x_2; \quad \tau\dot{x}_2 = -\frac{K}{B}x_1 - x_2 - \frac{T_c}{B}\operatorname{sgn}(x_2) \tag{9.44}$$

The singular points are given by the solution of the equations (refer to Eqns (9.34b))

$$0 = x_2; \quad 0 = -\frac{K}{B}x_1 - x_2 - \frac{T_c}{B}\operatorname{sgn}(x_2)$$

The solution gives

$$x_1 = -\frac{T_c}{K}\operatorname{sgn}(x_2)$$

Thus, there are two singular points. Their location can be interpreted physically—they are at a value of $e = x_1$, such that $|Ke| = |T_c|$, i.e., the drive torque is exactly equal to the Coulomb-friction torque. We note that both the singular points are on the x_1-axis ($x_2 \equiv 0$), and that the singular point given by $x_1 = T_c/K$ is

related to the lower-half phase plane (x_2 negative), and the singular point given by $x_1 = -T_c/K$ is related to the upper-half phase plane (x_2 positive).

Let us now investigate the stability of the singular points. For $\dot{e} > 0$, Eqn. (9.43) may be expressed as

$$\tau \frac{d^2}{dt^2}\left(e + \frac{T_c}{K}\right) + \frac{d}{dt}\left(e + \frac{T_c}{K}\right) + \frac{K}{B}\left(e + \frac{T_c}{K}\right) = 0 \tag{9.45}$$

This is a linear second-order system with the singular point at $(-T_c/K, 0)$ on the $(e, \dot{e})$-plane. The characteristic equation of this system is given by

$$\lambda^2 + \frac{1}{\tau}\lambda + \frac{K}{\tau B} = 0$$

Let us assume the following parameter values for the system under consideration:

$$(K/B) = 5,\ \tau = 4 \tag{9.46}$$

With these parameters, the roots of the characteristic equation are complex-conjugate with negative real parts; the singular point is, therefore, a stable focus (refer to Fig. 9.30a).

Let us now investigate the system behavior when large inputs are applied. Phase trajectories may be obtained by solving the following second-order differential equations for given initial state points (refer to Eqns (9.35b)–(9.36)).

Region I (defined by $x_2 > 0$):

$$4\ddot{z} + \dot{z} + 5z = 0;\ z = x_1 + \frac{T_c}{K};\ x_2 = \dot{x}_1$$

Region II (defined by $x_2 < 0$):

$$4\ddot{z} + \dot{z} + 5z = 0;\ z = x_1 - \frac{T_c}{K};\ x_2 = \dot{x}_1$$

Figure 9.35 shows a few phase trajectories. It is observed that for small as well as large inputs, the resulting trajectories terminate on a line along the x_1-axis from $-T_c/K$ to $+T_c/K$, i.e., the line joining the

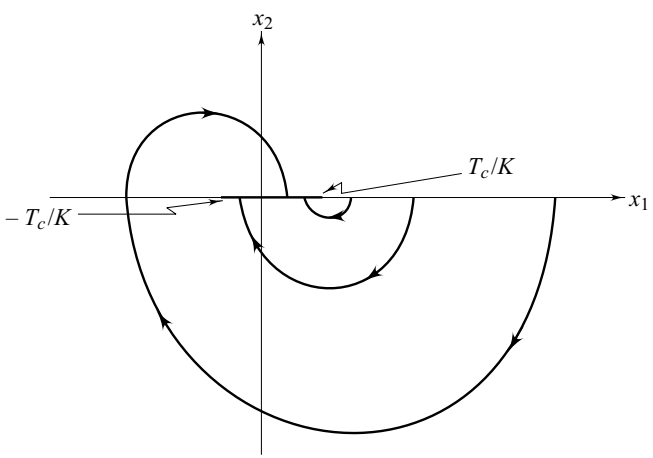


Fig. 9.35 Phase portrait for the system in Fig. 9.34

singular points. Therefore, the system with Coulomb friction is stable; however, there is a possibility of large steady-state error.

9.9.2 Limit Cycles

In the phase portrait of the nonlinear Van der Pol equation, shown in Fig. 9.26, one observes that there is a closed-curve in the phase portrait. Trajectories inside the curve—and those outside the curve—all tend to this curve, while a motion started on this curve will stay on it forever. This curve is an instance of the so-called 'limit cycle' phenomenon. Limit cycles are unique features of nonlinear systems.

On the phase plane, a *limit cycle is defined as an isolated closed curve*. The trajectory has to be both closed, indicating the periodic nature of motion, and isolated, indicating the limiting nature of the cycle (with neighboring trajectories converging to or diverging from it). Thus, while there are many closed curves in the satellite system in Example 9.5, these are not limit cycles because they are not isolated.

A limit cycle is stable if all trajectories in the vicinity of the limit cycle converge to it as $t \to \infty$ (Fig. 9.23a). A limit cycle is unstable if all trajectories in the vicinity of the limit cycle diverge from it as $t \to \infty$ (Fig. 9.23b).

9.10 SIMPLE VARIABLE STRUCTURE SYSTEMS

The purpose of this section is to informally introduce the reader to variable structure sliding mode control systems. Formal introduction to sliding mode control will appear later in Section 10.5.

A *variable structure system* is a dynamical system, whose structure changes in accordance with the current value of its state. A variable structure system can be viewed as a system composed of independent structures, together with a switching logic between each of the structures. With appropriate switching logic, a variable structure system can exploit the desirable properties of each of the structures the system is composed of. Even more, a variable structure system may have a property that is not a property of any of its structures. We illustrate the above ideas with two numerical examples.

Example 9.8

We consider a double integrator model

$$\ddot{e} = -u \tag{9.47}$$

having two structures corresponding to $u = -1$ and $u = +1$ (refer to Fig. 9.36).

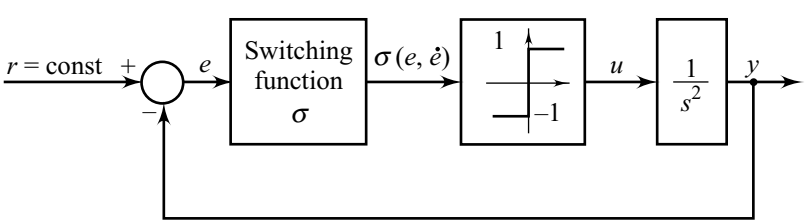


Fig. 9.36 A simple variable structure system

Choosing $x_1 = e$ and $x_2 = \dot{e}$ as state variables, we have

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -u \end{aligned} \tag{9.48}$$

The trajectories corresponding to the structure $u = -1$ are given by (refer to Eqns (9.24))

$$x_1(t) = \tfrac{1}{2}x_2^2(t) + x_1(0) - \tfrac{1}{2}x_2^2(0) \tag{9.49a}$$

and the trajectories corresponding to the structure $u = +1$ are given by

$$x_1(t) = -\tfrac{1}{2}x_2^2(t) + x_1(0) + \tfrac{1}{2}x_2^2(0) \tag{9.49b}$$

The phase-plane portraits of the two structures are shown in Figs 9.37a and 9.37b; the individual structures are families of parabolas. Neither of the structures is asymptotically stable; each structure is unstable. However, by choosing a suitable switching logic between the two structures, we can make the resulting variable structure system, asymptotically stable.

Suppose the structure of the system is changed at any time the system's trajectory crosses the vertical axis of the state plane, that is,

$$u = \begin{cases} +1 & \text{if } x_1 > 0 \\ -1 & \text{if } x_1 < 0 \end{cases} \tag{9.50}$$

A phase portrait of the system (9.48) with the switching logic specified by (9.50) is shown in Fig. 9.37c; the system always enters into a limit cycle (In fact, we are familiar with the switching function given by (9.50); it is on–off switching.

To achieve asymptotic stability, we redesign the switching logic. We note that one trajectory of each family in Figs 9.37a and 9.37b goes through the origin. Segments *A-O* and *B-O* of these two trajectories terminating at the origin form the curve shown by the thick line in Fig. 9.38.

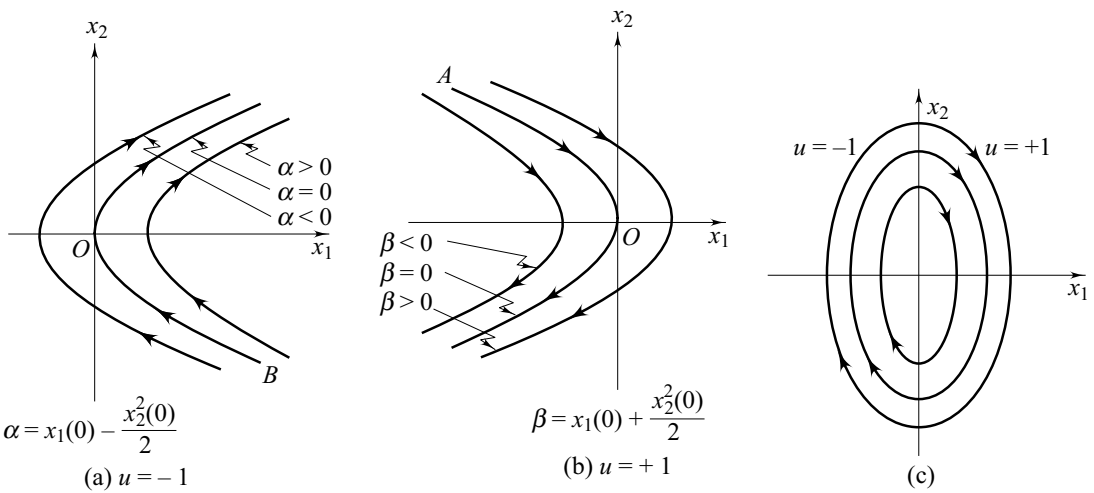


Fig. 9.37 Phase trjectories for the system of Fig. 9.36

The following switching logic seems to give an optimal control performance.

(i) If the initial state point lies at P_1 on the segment *A-O* (Fig. 9.38), the state point $(x_1(t), x_2(t))$ is driven to the origin along a segment of a parabola corresponding to $u = +1$.

(ii) If the initial point lies at P_2 on the segment *B-O* (Fig. 9.38), the state point $(x_1(t), x_2(t))$ is driven to the origin along a segment of a parabola corresponding to $u = -1$.

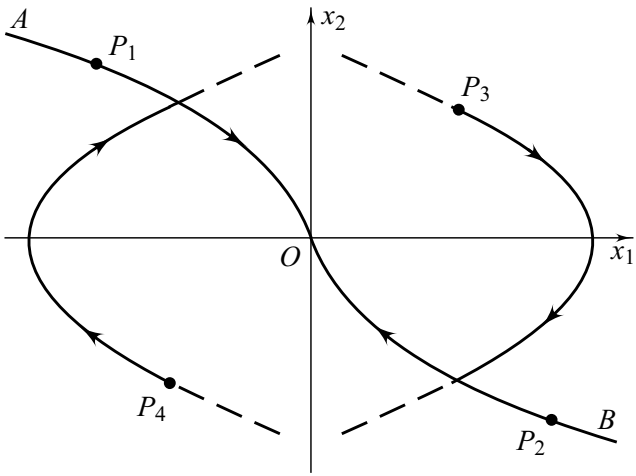


Fig. 9.38 Optimal switching curve

(iii) If the initial state point lies above or below the curve *A-O-B*, then only one switching is required to drive the state point to the origin. Consider the initial state point at P_3, which is above the curve *A-O-B* (Fig. 9.38). The state point $(x_1(t), x_2(t))$ follows a parabola corresponding to $u = +1$ till it reaches the segment *B-O*. This is followed by switching of the control to -1, and driving of the state point to the origin along *B-O* with $u = -1$.

(iv) Consider the initial point at P_4, which is below the curve *A-O-B* (Fig. 9.38). The state point $(x_1(t), x_2(t))$ follows a parabola corresponding to $u = -1$ till it reaches the segment *A-O*. This is followed by switching of the control to $+1$, and driving of the state point to the origin along *A-O* with $u = +1$.

The double integrator model with the switching strategy described above is in fact an on–off closed-loop control system shown in Fig. 9.36, in which the controller is actuated by a signal which is a function $\sigma(e, \dot{e})$ of the error and its first derivative. The differential equation describing the dynamics of the system is given by

$$\ddot{y} = u \text{ or } \ddot{e} = -u \tag{9.51a}$$

where $e = r - y$; r is constant, $x_1 = e$ and $x_2 = \dot{e}$ are state variables and

$$u = \begin{cases} +1; & \sigma(x_1, x_2) > 0 \\ -1; & \sigma(x_1, x_2) < 0 \end{cases} \tag{9.51b}$$

It is also clear from Fig. 9.38 that, for all initial conditions, the state point is driven to the origin along the shortest-time path with no oscillations (the output reaches the final value in minimum time with no ripples, and stays there; this type of response is commonly called a *deadbeat response*). Such bang-bang control systems provide optimal control (minimum-time control) [105].

The equation of *optimal switching curve A-0-B* can be obtained from Eqns (9.49) by setting (refer to Fig. 9.37)

$$x_1(0) - \frac{x_2^2(0)}{2} = x_1(0) + \frac{x_2^2(0)}{2} = 0$$

This gives
$$x_1(t) = -\tfrac{1}{2} x_2(t)\left|x_2(t)\right| \tag{9.52a}$$

Let us define the switching function $\sigma(e, \dot{e}) = \sigma(x_1, x_2)$ as
$$\sigma(x_1, x_2) = x_1(t) + \tfrac{1}{2} x_2(t)\left|x_2(t)\right| \tag{9.52b}$$

$\sigma(x_1, x_2) > 0$ implies that the state point (x_1, x_2) lies above the curve *A-O-B*. $\sigma(x_1, x_2) = 0$ and $x_2 > 0$ implies that the state point (x_1, x_2) lies on the segment *A-O*. $\sigma(x_1, x_2) = 0$ and $x_2 < 0$ implies that the state point (x_1, x_2) lies on the segment *B-O*. $\sigma(x_1, x_2) < 0$ implies that the state point (x_1, x_2) lies below the segment *A-O-B*.

In terms of the *optimal switching function* $\sigma(x_1, x_2)$, the control law becomes
$$u(t) = \begin{cases} +1 & \text{when } \sigma(x_1, x_2) > 0 \\ -1 & \text{when } \sigma(x_1, x_2) = 0 \text{ and } x_2(t) < 0 \\ -1 & \text{when } \sigma(x_1, x_2) < 0 \\ +1 & \text{when } \sigma(x_1, x_2) = 0 \text{ and } x_2(t) > 0 \end{cases} \tag{9.53}$$

The optimal switching may be realized by a computer. It accepts the state point (x_1, x_2) and computes the switching function given by Eqn. (9.52b). It then manipulates the on–off controller to produce the optimal control components according to Eqn. (9.53).

Example 9.9

In this example, we discuss a suboptimal method of switching the on–off controller in Fig. 9.36. The advantage of the method described below, is that implementation of the switching function is simple. The cost paid is in terms of increase in settling time compared to the optimal solution.

Consider the following suboptimal switching function:
$$\sigma(e, \dot{e}) = e + K_D \dot{e}$$

The system equations now become
$$\ddot{e} = -u; \quad u = \text{sgn}(e + K_D \dot{e})$$

In the state variable form $(x_1 = e, x_2 = \dot{e})$,
$$\dot{x}_1 = x_2; \dot{x}_2 = -\text{sgn}(x_1 + K_D x_2)$$

The phase plane is divided into two regions by the switching line
$$x_1 + K_D x_2 = 0 \tag{9.54}$$

The trajectory equation for the region defined by $x_1 + K_D x_2 < 0$, is (refer to Eqn. (9.49a))
$$x_1(t) = \tfrac{1}{2} x_2^2(t) + x_1(0) - \tfrac{1}{2} x_2^2(0)$$

and the trajectory equation for the region defined by $x_1 + K_D x_2 > 0$, is (refer to Eqn. (9.49b))
$$x_1(t) = -\tfrac{1}{2} x_2^2(t) + x_1(0) + \tfrac{1}{2} x_2^2(0)$$

In each half of the phase plane separated by the switching line (9.54), the system trajectories would be parabolas. Assume that the system under consideration starts with initial conditions corresponding to point A in Fig. 9.39. The on–off controller switches when the representative point reaches B. By geometry of the situation, we see that the trajectory resulting from the reversal of the drive at point B, will bring the representative point on a parabola passing much closer to the origin. This will continue until the trajectory intersects the switching line at a point closer to the origin than the points A_1 and A_2 which are points of intersection of the switching line with parabola passing through the origin. In Fig. 9.39, point C corresponds to this situation. Here, an instant after the on–off controller is switched, the system trajectory will recross the switching line and the on–off controller must switch back. The on–off controller will, thus, chatter while the system stays on the switching line. In a second-order system, the chattering frequency will be infinite and amplitude will be zero; the representative point, thus, slides along the switching line. It can easily be verified that with $K_D = 0$ (on–off controller switching on the vertical axis), the system always enters into a limit cycle. The switching process given by the switching line (9.54) has, thus, converted the oscillating system into an asymptotically stable one; though the goal has been achieved in a suboptimal way.

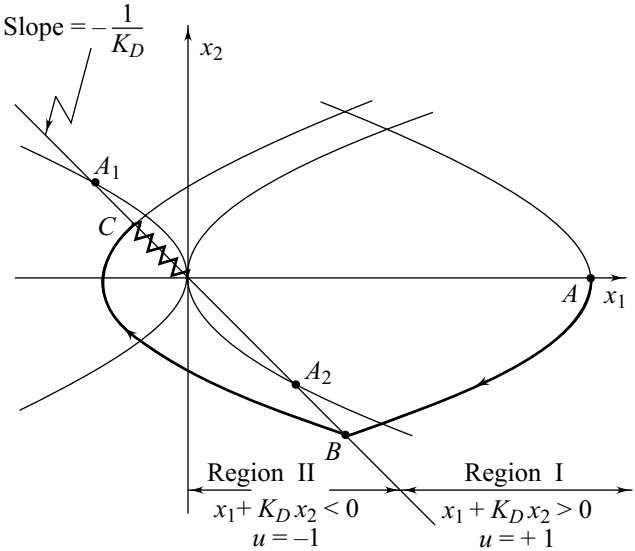


Fig. 9.39 Suboptimal switching curve

9.11 LYAPUNOV STABILITY DEFINITIONS

The general state equation for a nonlinear system can be expressed as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t); \mathbf{x}(t_0) \triangleq \mathbf{x}^0 \tag{9.55}$$

where $\mathbf{x}$ is the $n \times 1$ state vector, $\mathbf{u}$ is the $p \times 1$ input vector, and

$$\mathbf{f}(\cdot) = \begin{bmatrix} f_1(\cdot) \\ f_2(\cdot) \\ \vdots \\ f_n(\cdot) \end{bmatrix}$$

is the $n \times 1$ function vector.

Suppose that all the states of the system (9.55) settle to constant values (not necessarily zero values) for a constant input vector $\mathbf{u}(t) = \mathbf{u}^c$. The system is then said to be in an *equilibrium state* corresponding to

the input $\mathbf{u}^c$. The state trajectories converge to a point in state space, called the *equilibrium point*. At this point, no states vary with time. Thus, we have the following definition of equilibrium point (equilibrium state).

If for any constant input vector $\mathbf{u}(t) = \mathbf{u}^c$, there exists a point $\mathbf{x}(t) = \mathbf{x}^e$ = constant in state space, such that at this point $\dot{\mathbf{x}}(t) = \mathbf{0}$ for all t, then this point is called an equilibrium point of the system corresponding to the input $\mathbf{u}^c$. Applying this definition to the system (9.55), any equilibrium point must satisfy

$$\mathbf{f}(\mathbf{x}^e, \mathbf{u}^c, t) = \mathbf{0} \qquad \text{for all } t \tag{9.56}$$

The number of solutions depends entirely upon the nature of $\mathbf{f}(\cdot)$ and no general statement is possible.

Example 9.10

Consider the nonlinear system described by the state equations:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 - x_1^2 - x_2$$

The equilibrium states of this system are given by the solutions of the following set of equations (refer to Eqn. (9.56)):

$$\dot{x}_1^e = x_2{}^e = 0$$
$$\dot{x}_2^e = -x_1^e - (x_1^e)^2 - x_2^e = 0$$

From the first equation, x_2^e is equal to zero. From the second equation,

$$(x_1^e)^2 + x_1^e = x_1^e(x_1^e + 1) = 0$$

which has the solutions $x_1^e = 0$ and $x_1^e = -1$. Thus, there are two equilibrium states, given by

$$\mathbf{x}^{e1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}^{e2} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

In the stability analysis of a system, we are usually concerned with the following two notions of stability:

(i) when a relaxed system ($\mathbf{x}(t_0) = \mathbf{0}$) is excited by a bounded input, the output must be bounded; and
(ii) in an unforced system ($\mathbf{u} = \mathbf{0}$) with arbitrary initial conditions, the system state must tend towards the equilibrium point in state space.

We have seen earlier in Chapter 5 that the two notions of stability defined above are essentially equivalent for linear time-invariant systems.

Unfortunately in nonlinear systems, there is no definite correspondence between the two notions. Most of the important results obtained, thus far, concern the stability of nonlinear *autonomous*[2] systems:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)); \quad \mathbf{x}(t_0) \triangleq \mathbf{x}^0 \tag{9.57}$$

in the sense of second notion above.

[2] An unforced (i.e., $\mathbf{u} = \mathbf{0}$) and time-invariant system is called an autonomous system.

It may be noted that even for this class of systems, the concept of stability is not clear cut. The linear autonomous systems have only one equilibrium state (the origin of the state space), and their behavior about the equilibrium state completely determines the qualitative behavior in the entire state space. In nonlinear systems, on the other hand, system behavior for small deviations about the equilibrium point may be different from that for large deviations. Therefore, *local stability* does not imply stability in the overall state space, and the two concepts should be considered separately.

Secondly, the set of nonlinear equations (refer to Eqns (9.56)–(9.57)),

$$\mathbf{f}(\mathbf{x}^e) = \mathbf{0} \tag{9.58}$$

may result in a number of solutions (equilibrium points). Due to the possible existence of multiple equilibrium states, the system trajectories may move away from one equilibrium state to the other as time progresses. Thus, it appears that in the case of nonlinear systems, it is simpler to speak of system stability relative to the equilibrium state rather than using the general term 'stability of a system'.

We shall confine our attention to nonlinear autonomous systems described by state equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)); \; \mathbf{f}(\mathbf{0}) = \mathbf{0}; \; \mathbf{x}(0) \triangleq \mathbf{x}^0 \tag{9.59}$$

Note that the origin of the state space has been taken as the equilibrium state of the system. There is no loss in generality in this assumption, since any nonzero equilibrium state can be shifted to the origin by appropriate transformation. Further, we have taken $t_0 = 0$ in Eqn. (9.59), which is a convenient choice for time-invariant systems.

For nonlinear autonomous systems, local stability may be investigated through linearization in the neighborhood of the equilibrium point. The validity of determining the stability of the unperturbed solution near the equilibrium points from the linearized equations was developed independently by Poincaré and Lyapunov in 1892. Lyapunov designated this as the *first method*. This stability determination is applicable only in a small region near the equilibrium point.

The region of validity of local stability is generally not known. In some cases, the region may be too small to be of any use practically; while in others the region may be much larger than the one assumed by the designer—giving rise to systems that are too conservatively designed. We, therefore, need information about the domain of stability. The 'second method of Lyapunov' (also called the 'direct method of Lyapunov') is used to determine *stability in-the-large*. We first present direct method of Lyapunov; the linearization method is described in Section 9.14.

The concept of stability formulated by Russian mathematician A.M. Lyapunov is concerned with the following question:

If a system with zero input is perturbed from the equilibrium point $\mathbf{x}^e$ at $t = 0$, will the state $\mathbf{x}(t)$ return to $\mathbf{x}^e$, remain 'close' to $\mathbf{x}^e$, or diverge from $\mathbf{x}^e$?

Lyapunov stability analysis is, thus, concerned with the boundedness of the free (unforced) response of a system. The free response of a system is said to be *stable in the sense of Lyapunov* at the equilibrium point $\mathbf{x}^e$ if, for every initial state $\mathbf{x}(t_0)$ which is sufficiently close to $\mathbf{x}^e$, $\mathbf{x}(t)$ remains near $\mathbf{x}^e$ for all t. It is *asymptotically stable* at $\mathbf{x}^e$ if $\mathbf{x}(t)$, in fact, approaches $\mathbf{x}^e$ as $t \to \infty$.

In the following, we give mathematically precise definitions of different types of stability with respect to the system described by Eqn. (9.59).

Stability in the Sense of Lyapunov

The system described by Eqn. (9.59) is *stable in the sense of Lyapunov* at the origin if, for every real number $\varepsilon > 0$, there exists a real number $\delta(\varepsilon) > 0$ such that $\|\mathbf{x}(0)\| < \delta$ results in $\|\mathbf{x}(t)\| < \varepsilon$ for all $t \geq 0$.

This definition uses the concept of the vector norm. The Euclidean norm for a vector with n components $x_1, x_2, \ldots, x_n$ is (refer to Eqn. (5.6a))

$$\|\mathbf{x}\| = (x_1^2 + x_2^2 + \cdots + x_n^2)^{1/2}$$

$\|\mathbf{x}\| \leq R$ defines a hyper-spherical region $S(R)$ of radius R, surrounding the equilibrium point $\mathbf{x}^e = \mathbf{0}$. In terms of the Euclidean norm, the above definition of stability implies that for any $S(\varepsilon)$ that we may designate, the designer must produce $S(\delta)$ so that the system state, initially in $S(\delta)$, will never leave $S(\varepsilon)$. This is illustrated in Fig. 9.40a.

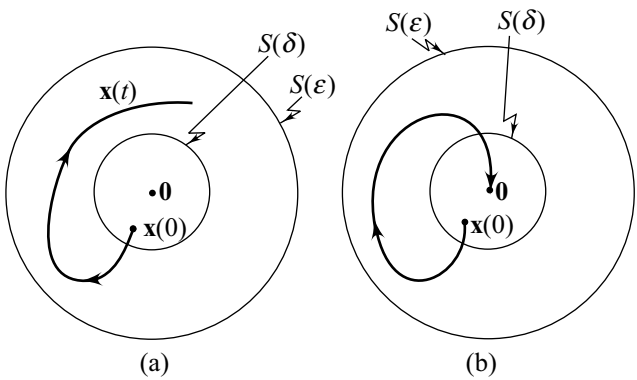


Fig. 9.40 Stability definitions

Note that this definition of stability permits the existence of continuous oscillation about the equilibrium point. The state-space trajectory for such an oscillation is a closed path. The amplitude and frequency of the oscillation may influence whether it represents acceptable performance.

Example 9.11

Consider a linear oscillator described by the differential equation

$$\ddot{y}(t) + \omega^2 y(t) = 0$$

where ω is the frequency of oscillations.

Define the state variables as

$$x_1(t) = y(t),\ x_2(t) = \dot{y}(t)$$

This gives the state equations

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = -\omega^2 x_1(t)$$

From these equations, we obtain the following equation for state trajectory:

$$\frac{dx_2}{dx_1} = -\omega^2 \frac{x_1}{x_2} \quad \text{or} \quad x_2^2 + \omega^2 x_1^2 = c^2; \quad c = \text{constant}$$

Several state trajectories for various values of c, corresponding to various initial conditions of x_1 and x_2, are shown in Fig. 9.41. For a specified value of ε, we can find a closed state trajectory whose maximum distance from the origin is ε. We then select a value of δ which is less than the minimum distance from that curve to the origin. The $\delta(\varepsilon)$ so chosen, will satisfy the conditions that guarantee stability in the sense of Lyapunov.

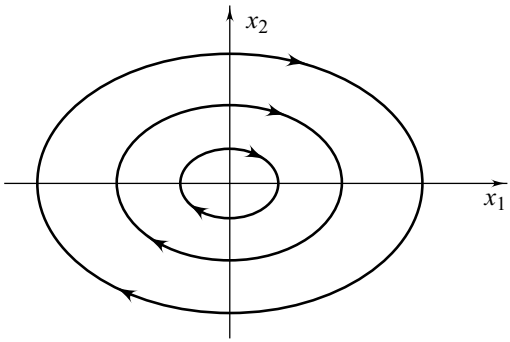


Fig. 9.41 State trajectories

Asymptotic Stability

The system (9.59) is asymptotically stable at the origin if

(i) it is stable in the sense of Lyapunov, i.e., for each $S(\varepsilon)$ there is a region $S(\delta)$ such that trajectories, starting within $S(\delta)$, do not leave $S(\varepsilon)$ as $t \to \infty$; and

(ii) each trajectory starting within $S(\delta)$ converges to the origin as $t \to \infty$ (Fig. 9.40b).

Local and Global Stability

The definitions of asymptotic stability and stability in the sense of Lyapunov apply in a *local* sense (*stability in-the-small*) if the region $S(\delta)$ is small. When the region $S(\delta)$ includes the entire state space, the definitions of asymptotic stability and stability in the sense of Lyapunov are said to apply in a *global* sense (*stability in-the-large*).

9.12 LYAPUNOV STABILITY THEOREMS

The Lyapunov stability analysis is based upon the concept of energy, and the relation of stored energy with system stability. We first give an example to motivate the discussion.

Consider the spring-mass-damper system of Fig. 9.42. The governing equation of the system is

$$\ddot{x}_1 + B\dot{x}_1 + Kx_1 = 0$$

A corresponding state variable model is

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -Kx_1 - Bx_2 \end{aligned} \tag{9.60}$$

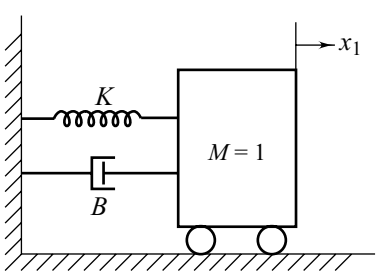


Fig. 9.42 A spring-mass-damper system

At any instant, the total energy V in the system consists of the kinetic energy of the moving mass, and the potential energy stored in the spring.

$$V(x_1, x_2) = \tfrac{1}{2}x_2^2 + \tfrac{1}{2}Kx_1^2 \tag{9.61}$$

Thus
$$V(\mathbf{x}) > 0 \text{ when } \mathbf{x} \neq 0$$
$$V(\mathbf{0}) = 0$$

This means that the total energy is positive unless the system is at rest at the equilibrium point $\mathbf{x}^e = \mathbf{0}$, where the energy is zero.

The rate of change of energy is given by

$$\dot{V}(x_1, x_2) = \frac{d}{dt}V(x_1, x_2) = \frac{\partial V}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial V}{\partial x_2}\frac{dx_2}{dt} = -Bx_2^2 \tag{9.62}$$

Case I: Positive Damping ($B > 0$)

Let (x_1^0, x_2^0) be an arbitrary initial state of the system of Fig. 9.42. The solution of the differential equations (9.60) corresponding to this initial state, gives the state trajectory $\mathbf{x}(t)$ for $t > 0$. Since the linear system (9.60) is stable under the condition of positive damping, $\mathbf{x}(t) \to \mathbf{0}$ as $t \to \infty$.

Let us study the relation of stored energy with system stability. The initial energy in the system is

$$V(x_1^0, x_2^0) = \tfrac{1}{2}(x_2^0)^2 + \tfrac{1}{2}K(x_1^0)^2$$

As per Eqn. (9.62), the rate of change of energy is negative and, therefore, system energy $V(x_1, x_2)$ continually decreases along the trajectory $\mathbf{x}(t)$, $t > 0$. There is only one exception; when the representative point $\mathbf{x}(t)$ of the trajectory reaches $x_2 = 0$ points in the state plane, the rate of change of energy becomes zero. However, as seen from Eqns (9.60), $\dot{x}_2 = -Kx_1$ at the points where $x_2 = 0$. The representative point $\mathbf{x}(t)$, therefore, cannot stay at the points in the state plane where $x_2 = 0$ (except at the origin). It immediately moves to the points at which the rate of change of energy is negative and the system energy, therefore, continually decreases from its initial value $V(x_1^0, x_2^0)$ along the trajectory $\mathbf{x}(t)$, $t > 0$, till it reaches a value $V = 0$ at the equilibrium point $\mathbf{x}^e = \mathbf{0}$.

A visual analogy may be obtained by considering the surface

$$V(x_1, x_2) = \tfrac{1}{2}x_2^2 + \tfrac{1}{2}Kx_1^2 \tag{9.63}$$

This is paraboloid (a solid generated by rotation of parabola about its axis of symmetry) surface as shown in Fig. 9.43. The value $V(x_1, x_2) = k_i$ (a constant) is represented by the intersection of the surface $V(x_1, x_2)$ and the plane $z = k_i$. The projection of this intersection on the (x_1, x_2)-plane is a closed curve, an oval, around the origin. There is a family of such closed curves in the (x_1, x_2)-plane for different values of k_i. The closed curve corresponding to $V(x_1, x_2) = k_1$, lies entirely inside the closed curve corresponding to $V(x_1, x_2) = k_2$ if $k_1 < k_2$. The value $V(x_1, x_2) = 0$ is the point at the origin. It is the innermost curve of the family of closed curves, representing different levels on the paraboloid for $V(x_1, x_2) = k_i$.

If one plots a state-plane trajectory starting from the point (x_1^0, x_2^0), the representative point $\mathbf{x}(t)$ crosses the ovals for successively smaller values of $V(x_1, x_2)$, and moves towards the point corresponding to $V(x_1, x_2) = 0$, which is the equilibrium point. Figure 9.43 shows a typical trajectory.

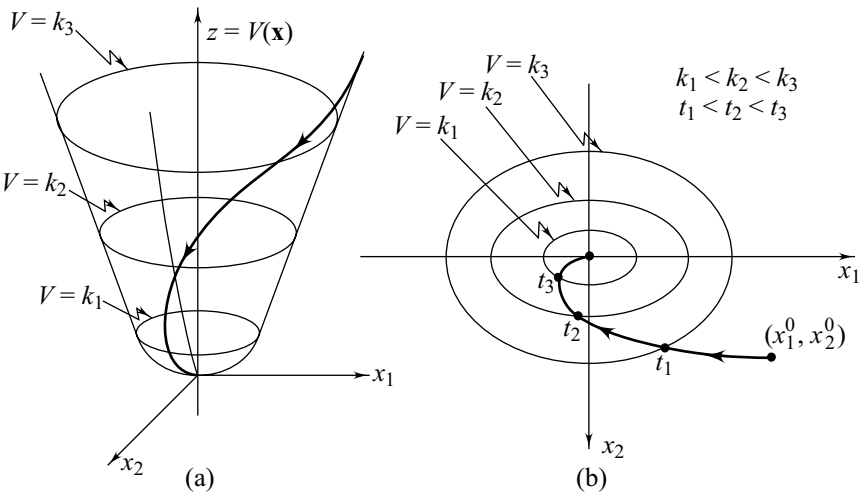


Fig. 9.43 Constant-V curves

Note also that $V(\mathbf{x})$ given by Eqn. (9.63) is *radially unbounded*[3], i.e., $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$. The ovals on the (x_1, x_2)-plane extend over the entire state plane and, therefore, for any initial state $\mathbf{x}^0$ in the entire state plane, the system energy continually decreases from the value $V(\mathbf{x}^0)$ to zero.

Case II: Zero Damping ($B = 0$)

Under the condition of zero damping, Eqns (9.60) become

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -K & 0 \end{bmatrix}$$

The eigenvalues of **A** lie on the imaginary axis in the complex plane; the system response is, therefore, oscillatory in nature.

From Eqn. (9.62) we observe that when $B = 0$, the rate of change of energy $\dot{V}(x_1, x_2)$ vanishes identically along any trajectory; the system energy $V(x_1, x_2) = V(x_1^0, x_2^0)$ for all $t \geq 0$. The representative point $\mathbf{x}(t)$ cannot cross the V-contours in Fig. 9.43; it simply moves along one of these contours.

In the example given above, it was easy to associate the energy function V with the given system. However, in general, there is no obvious way of associating an energy function with a given set of equations describing a system. In fact, there is nothing sacred or unique about the total energy of the system which allows us to determine system stability in the way described above. Other scalar functions of the system state can also answer the question of stability. This idea was introduced and formalized by the mathematician A.M. Lyapunov. The scalar function is now known as the *Lyapunov function* and the method of investigating stability using Lyapunov's function is known as the *Lyapunov's direct method*.

[3] Use of the norm definition given by Eqn. (5.6b) immediately proves this result.

In Section 5.2, we introduced the concept of sign definiteness of scalar functions. Let us examine here the scalar function $V(x_1, x_2, ..., x_n) \triangleq V(\mathbf{x})$ for which $V(\mathbf{0}) = 0$ and the function is continuous in a certain region surrounding the origin in state space. Due to the manner in which these V-functions are used later, we define the sign definiteness with respect to a region around the origin represented as $\|\mathbf{x}\| \leq K$ (a positive constant) where $\|\mathbf{x}\|$ is the norm of $\mathbf{x}$.

Positive Definiteness of Scalar Functions A scalar function $V(\mathbf{x})$ is said to be *positive definite* in the region $\|\mathbf{x}\| \leq K$ (which includes the origin of the state space), if $V(\mathbf{x}) > 0$ at all points of the region except at the origin, where it is zero.

Negative Definiteness of Scalar Functions A scalar function $V(\mathbf{x})$ is said to be *negative definite* if $[-V(\mathbf{x})]$ is positive definite.

Positive Semidefiniteness of Scalar Functions A scalar function $V(\mathbf{x})$ is said to be *positive semidefinite* in the region $\|\mathbf{x}\| < K$, if its value is positive at all points of the region except at finite number of points, including origin, where it is zero.

Negative Semidefiniteness of Scalar Functions A scalar function $V(\mathbf{x})$ is said to be *negative semidefinite* if $[-V(\mathbf{x})]$ is positive semidefinite.

Indefiniteness of Scalar Functions A scalar function $V(\mathbf{x})$ is said to be indefinite in the region $\|\mathbf{x}\| < K$, if it assumes both positive and negative values, within this region.

Examples For all $\mathbf{x}$ in the state plane:

(i) $V(\mathbf{x}) = x_1^2 + x_2^2$ is positive definite;
(ii) $V(\mathbf{x}) = (x_1 + x_2)^2$ is positive semidefinite;
(iii) $V(\mathbf{x}) = -x_1^2 - (x_1 + x_2)^2$ is negative definite; and
(iv) $V(\mathbf{x}) = x_1 x_2 + x_2^2$ is indefinite.

An important class of scalar functions is a quadratic form:

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$$

where $\mathbf{P}$ is a real, symmetric constant matrix. In this form, the definiteness of V is usually attributed to $\mathbf{P}$. We speak of the positive (negative) definite and the positive (negative) semidefinite $\mathbf{P}$ depending upon the definiteness of $V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$.

Tests for checking definiteness of a matrix were described in Section 5.2. We consider an example here:

$$\mathbf{P} = \left[\begin{array}{c:c:c} 10 & 1 & -2 \\ \hdashline 1 & 4 & -1 \\ \hdashline -2 & -1 & 1 \end{array}\right]$$

As per Sylvester's test, the necessary and sufficient condition for $\mathbf{P}$ to be positive definite is that all the successive principal minors of $\mathbf{P}$ be positive.

Applying Sylvester's test to the given **P**, we obtain

$$10 > 0$$

$$\begin{vmatrix} 10 & 1 \\ 1 & 4 \end{vmatrix} > 0$$

$$\begin{vmatrix} 10 & 1 & -2 \\ 1 & 4 & -1 \\ -2 & -1 & 1 \end{vmatrix} > 0$$

Since all the successive principal minors of the matrix **P** are positive, $V(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x}$ is a positive definite function.

In the following, we state, without proof, the basic Lyapunov stability results. For proof refer to [105].

Theorem 9.1 For the autonomous system (9.59), sufficient conditions of stability are as follows:

Suppose that there exists a scalar function $V(\mathbf{x})$ which, for some real number $\varepsilon > 0$, satisfies the following properties for all $\mathbf{x}$ in the region $\|\mathbf{x}\| \le \varepsilon$:

(i) $V(\mathbf{x}) > 0$; $\mathbf{x} \neq \mathbf{0}$
(ii) $V(\mathbf{0}) = 0$

(i.e., $V(\mathbf{x})$ is positive definite function)

(iii) $V(\mathbf{x})$ has continuous partial derivatives with respect to all components of $\mathbf{x}$.
Then the equilibrium state $\mathbf{x}^e = \mathbf{0}$ of the system (9.59) is

(iva) *asymptotically stable* if $\dot{V}(\mathbf{x}) < 0$, $\mathbf{x} \neq \mathbf{0}$, i.e., $\dot{V}(\mathbf{x})$ is a negative definite function; and

(ivb) *asymptotically stable in-the-large* if $\dot{V}(\mathbf{x}) < 0$, $\mathbf{x} \neq \mathbf{0}$, and in addition $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$.

Example 9.12

Consider a nonlinear system described by the equations

$$\begin{aligned} \dot{x}_1 &= x_2 - x_1(x_1^2 + x_2^2) \\ \dot{x}_2 &= -x_1 - x_2(x_1^2 + x_2^2) \end{aligned} \tag{9.64}$$

Clearly, the origin is the only equilibrium state.

Let us choose the following positive definite scalar function as a possible Lyapunov function:

$$V(\mathbf{x}) = x_1^2 + x_2^2 \tag{9.65}$$

Time derivative of $V(\mathbf{x})$ along any trajectory, is given by

$$\begin{aligned} \dot{V}(\mathbf{x}) &= \frac{dV(x_1, x_2)}{dt} = \frac{\partial V}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial V}{\partial x_2}\frac{dx_2}{dt} \\ &= 2x_1\dot{x}_1 + 2x_2\dot{x}_2 = -2(x_1^2 + x_2^2)^2 \end{aligned} \tag{9.66}$$

which is negative definite. This shows that $V(\mathbf{x})$ is continually decreasing along any trajectory; hence $V(\mathbf{x})$ is a Lyapunov function. By Theorem 9.1, the equilibrium state (at the origin) of the system (9.64) is asymptotically stable.

Further, $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$, i.e., $V(\mathbf{x})$ becomes infinite with infinite deviation from the equilibrium state. Therefore, as per condition (ivb) of Theorem 9.1, the equilibrium state of the system (9.64) is asymptotically stable in-the-large.

Although Theorem 9.1 is a basic theorem of Lyapunov stability analysis, it is somewhat restrictive because $\dot{V}(\mathbf{x})$ must be negative definite. This requirement can be relaxed to $\dot{V}(\mathbf{x}) \leq 0$ (a negative semidefinite $\dot{V}(\mathbf{x})$) under proper conditions. This relaxed requirement is sufficient if it can be shown that no trajectory can stay forever at the points or on the line other than the origin, at which $\dot{V}(\mathbf{x}) = 0$. This is the case for the system of Fig. 9.42 as described at the beginning of this section.

If, however, there exists a positive definite function $V(\mathbf{x})$ such that $\dot{V}(\mathbf{x})$ is identically zero along a trajectory, the system will remain in that trajectory and will not approach the origin. The equilibrium state at the origin, in this case, is said to be stable in the sense of Lyapunov.

Theorem 9.2 For the autonomous system (9.59), sufficient conditions of stability are as follows.

Suppose that there exists a scalar function $V(\mathbf{x})$ which, for some real number $\varepsilon > 0$, satisfies the following properties for all $\mathbf{x}$ in the region $\|\mathbf{x}\| \leq \varepsilon$:

(i) $V(\mathbf{x}) > 0; \mathbf{x} \neq \mathbf{0}$
(ii) $V(\mathbf{0}) = 0$
} (i.e, $V(\mathbf{x})$ is positive definite function)

(iii) $V(\mathbf{x})$ has continuous partial derivatives with respect to all components of $\mathbf{x}$.
Then the equilibrium state $\mathbf{x}^e = \mathbf{0}$ of the system (9.59) is

(iva) *asymptotically stable* if $\dot{V}(\mathbf{x}) < 0$, $\mathbf{x} \neq \mathbf{0}$, i.e., $\dot{V}(\mathbf{x})$ is a negative definite function; or if $\dot{V}(\mathbf{x}) \leq 0$ (i.e., $\dot{V}(\mathbf{x})$ is negative semidefinite) and no trajectory can stay forever at the points or on the line other than the origin, at which $\dot{V}(\mathbf{x}) = 0$;

(ivb) *asymptotically stable in-the-large* if conditions (iva) are satisfied, and in addition $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$; and

(ivc) *stable in the sense of Lyapunov* if $\dot{V}(\mathbf{x})$ is identically zero along a trajectory.

Example 9.13

Consider the linear feedback system shown in Fig. 9.44 with $r(t) = 0$. We know that the closed-loop system will exhibit sustained oscillations.

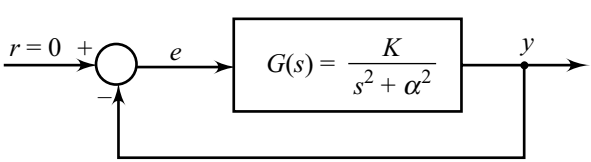


Fig. 9.44 Linear feedback system

The differential equation for the error signal is

$$\ddot{e} + \alpha^2 e = Ky = -Ke$$

Taking e and $\dot{e}$ as state variables x_1 and x_2, respectively, we obtain the following state equations:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -(K + \alpha^2)x_1 \end{aligned} \tag{9.67}$$

Let us choose the following scalar positive definite function as a possible Lyapunov function:

$$V(\mathbf{x}) = x_1^2 + x_2^2 \tag{9.68}$$

Then $\dot{V}(\mathbf{x})$ becomes

$$\dot{V}(\mathbf{x}) = 2x_1\dot{x}_1 + 2x_2\dot{x}_2 = 2[1 - (K + \alpha^2)]x_1x_2$$

$\dot{V}(\mathbf{x})$ is indefinite. This implies that $V(\mathbf{x})$, given by Eqn. (9.68), is not a Lyapunov function and stability cannot be determined by its use (the system is known to be stable in the sense of Lyapunov as per the stability definition given in Section 9.11).

We now test

$$V(\mathbf{x}) = p_1x_1^2 + p_2x_2^2;\ p_1 > 0, p_2 > 0$$

for Lyapunov properties. Conditions (i)–(iii) of Theorem 9.2 are obviously satisfied.

$$\dot{V}(\mathbf{x}) = 2p_1x_1x_2 - 2p_2(K + \alpha^2)x_1x_2$$

If we set $p_1 = p_2(K + \alpha^2)$, $\dot{V}(\mathbf{x}) = 0$ and, as per Theorem 9.2, the equilibrium state of the system (9.67) is stable in the sense of Lyapunov.

Example 9.14

Reconsider the system of Fig. 9.44 with

$$G(s) = \frac{K}{s(s + \alpha)}$$

If the reference variable $r(t) = 0$, then the differential equation for the actuating error will be

$$\ddot{e} + \alpha\dot{e} + Ke = 0$$

Taking e and $\dot{e}$ as state variables x_1 and x_2 respectively, we obtain the following state equations:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -Kx_1 - \alpha x_2 \end{aligned} \tag{9.69}$$

A candidate for a Lyapunov function is

$$V(\mathbf{x}) = p_1x_1^2 + p_2x_2^2;\ p_1 > 0, p_2 > 0,$$

which is a positive definite function.

Its derivative is

$$\dot{V}(\mathbf{x}) = 2(p_1x_1\dot{x}_1 + p_2x_2\dot{x}_2) = 2(p_1 - p_2K)x_1x_2 - 2p_2\alpha x_2^2$$

If we take $p_1 = Kp_2$ with $K > 0$, $\alpha > 0$, we obtain

$$\dot{V}(\mathbf{x}) = -2\alpha p_2x_2^2$$

which is negative semidefinite.

The condition $\dot{V}(\mathbf{x}) = 0$ exists along the x_1-axis where $x_2 = 0$. A way of showing that $\dot{V}(\mathbf{x})$, being negative semidefinite, is sufficient for asymptotic stability is to show that x_1-axis is not a trajectory of the system differential equation (9.69). The first equation yields $\dot{x}_1 = 0$ or $x_1 = c$. The x_1-axis can be a trajectory only

if $x_2 = 0$ and $\dot{x}_2 = 0$. Since on x_1-axis, $\dot{x}_2 = -Kc \neq 0$, x_1-axis is not a trajectory, and the equilibrium state at the origin of the system (9.69) is asymptotically stable.

Further, since $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$, the equilibrium state is asymptotically stable in-the-large.

This result, obtained by Lyapunov's direct method, is readily recognized as being correct either from the Routh stability criterion or from the root locus.

Example 9.15

Consider the system described by the state equations

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 - x_2$$

Let us choose,

$$V(\mathbf{x}) = x_1^2 + x_2^2$$

which is a positive definite function; $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$.

This gives

$$\dot{V}(\mathbf{x}) = 2x_1\dot{x}_1 + 2x_2\dot{x}_2 = -2x_2^2$$

which is negative semidefinite. As per the procedure described in the earlier example, it can be established that $\dot{V}(\mathbf{x})$ vanishes identically only at the origin. Hence, by Theorem 9.2, the equilibrium state at the origin is asymptotically stable in-the-large.

To show that a different choice of a Lyapunov function yields the same stability information, let us choose the following positive definite function as another possible Lyapunov function:

$$V(\mathbf{x}) = \tfrac{1}{2}[(x_1 + x_2)^2 + 2x_1^2 + x_2^2]$$

Then $\dot{V}(\mathbf{x})$ becomes

$$\begin{aligned}\dot{V}(\mathbf{x}) &= (x_1 + x_2)(\dot{x}_1 + \dot{x}_2) + 2x_1\dot{x}_1 + x_2\dot{x}_2 \\ &= (x_1 + x_2)(x_2 - x_1 - x_2) + 2x_1x_2 + x_2(-x_1 - x_2) = -(x_1^2 + x_2^2)\end{aligned}$$

which is negative definite. Since $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$, by Theorem 9.2, the equilibrium state at the origin is asymptotically stable in-the-large.

Instability

It may be noted that instability in a nonlinear system can be established by direct recourse to the instability theorem of the direct method. The basic instability theorem is presented below.

Theorem 9.3 For the autonomous system (9.59), sufficient conditions for instability are as follows.

Suppose that there exists a scalar function $W(\mathbf{x})$ which, for some real number $\varepsilon > 0$, satisfies the following properties for all $\mathbf{x}$ in the region $\|\mathbf{x}\| \leq \varepsilon$:

(i) $W(\mathbf{x}) > 0$; $\mathbf{x} \neq \mathbf{0}$;

(ii) $W(\mathbf{0}) = 0$; and

(iii) $W(\mathbf{x})$ has continuous partial derivatives with respect to all components of $\mathbf{x}$.

Then the equilibrium state $\mathbf{x}^e = \mathbf{0}$ of the system (9.59) is *unstable* if $\dot{W}(\mathbf{x}) > 0$, $\mathbf{x} \neq \mathbf{0}$, i.e., $\dot{W}(\mathbf{x})$ is a positive definite function.

Note that it requires as much ingenuity to devise a suitable W function, as to devise a Lyapunov function V. In the stability analysis of nonlinear systems, it is valuable to establish conditions for which the system is unstable. Then the regions of asymptotic stability need not be sought for such conditions, and the analyst is saved from this fruitless effort.

9.13 LYAPUNOV FUNCTIONS FOR NONLINEAR SYSTEMS

The determination of stability through Lyapunov's direct method centers around the choice of a positive definite function $V(\mathbf{x})$, called the *Lyapunov function*. Unfortunately, there is no universal method for selecting the Lyapunov function which is unique for a given nonlinear system. Some Lyapunov functions may provide better answers than others. Several techniques have been devised for the systematic construction of Lyapunov functions; each is applicable to a particular class of systems.

In addition, if a Lyapunov function cannot be found, it in no way implies that the system is unstable (stability theorems presented in the earlier section, merely provide *sufficient conditions* for stability). It only means that our attempt in trying to establish the stability of an equilibrium state of the system has failed. Also, if a certain Lyapunov function provides stability for a specified parameter region, this does not necessarily mean that leaving that region will result in system instability. Another choice of Lyapunov function may lead to a larger stability region.

Further, for a given V-function, there is no general method which will allow us to ascertain whether it is positive definite. However, if $V(\mathbf{x})$ is in quadratic form in x_i's, we can use simple tests given in Section 5.2 to ascertain definiteness of the function.

In spite of all these limitations, Lyapunov's direct method is the most powerful technique available today for the stability analysis of nonlinear systems. Many mathematical methods for constructing Lyaunov functions are available in the literature. Two of these methods—"Krasovskiis Method", and the "Variable Gradient Method"—are simple and systematic ways of constructing Lyapunov functions. However, the mathematical approach of these methods, though effective for simple systems, is often of little use for complex dynamic equations.

Therefore, faced with specific systems, one has to use experience, intuition, and physical insights to search for an appropriate Lyapunov function. An elegant and powerful Lyapunov analysis may be possible for complex systems if engineering insight and physical properties are properly exploited [126].

Example 9.16

Consider a nonlinear system governed by the equations:

$$\dot{x}_1 = -x_1 + 2x_1^2 x_2$$
$$\dot{x}_2 = -x_2$$

Note that $\mathbf{x} = \mathbf{0}$ is the equilibrium point.

A candidate for a Lyapunov function is

$$V = p_{11}x_1^2 + p_{22}x_2^2;\ p_{11} > 0, p_{22} > 0$$

which is a positive definite function.

Then

$$\begin{aligned}\frac{dV}{dt} &= 2p_{11}x_1\dot{x}_1 + 2p_{22}x_2\dot{x}_2 \\ &= 2p_{11}x_1(-x_1 + 2x_1^2x_2) + 2p_{22}x_2(-x_2) \\ &= -2p_{11}x_1^2(1 - 2x_1x_2) - 2p_{22}x_2^2\end{aligned}$$

dV/dt is negative definite if

$$1 - 2x_1x_2 > 0 \tag{9.70}$$

Therefore, for asymptotic stability we require that the condition (9.70) is satisfied. The region of state space where this condition is not satisfied is *possibly* the region of instability. Let us concentrate on the region of state space where this condition is satisfied. The limiting condition for such a region is

$$1 - 2x_1x_2 = 0$$

The dividing lines lie in the first and the third quadrants and are rectangular hyperbolas as shown in Fig. 9.45. In the second and the fourth quadrants, the inequality is satisfied for all values of x_1 and x_2. Figure 9.45 shows the regions of stability and possible instability. Since the choice of the Lyapunov function is not unique, it may be possible to choose another Lyapunov function for the system under consideration which yields a larger region of stability.

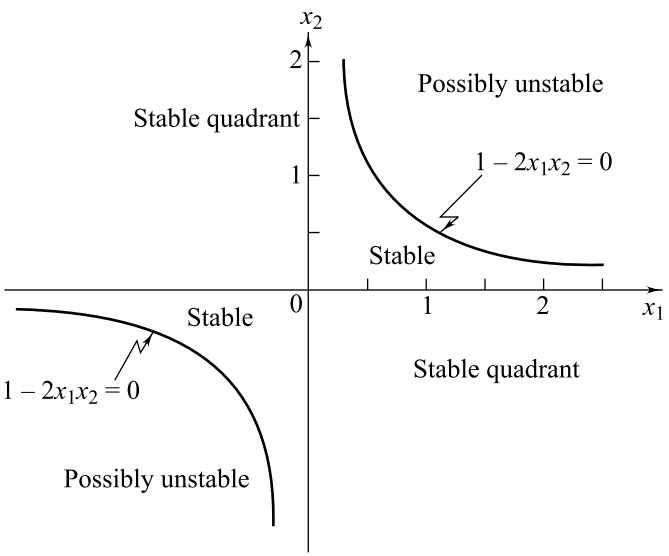


Fig. 9.45 Stability regions for the nonlinear system of Example 9.16

9.13.1 The Krasovskii Method

In the following, we describe the *Krasovskii Method* of constructing Lyapunov functions for nonlinear systems [105].

Consider the nonlinear autonomous system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x});\ \mathbf{f}(\mathbf{0}) = \mathbf{0} \tag{9.71}$$
$$\mathbf{f} = [f_1 \quad f_2 \cdots f_n]^T;\ \mathbf{x} = [x_1 \quad x_2 \cdots x_n]^T$$

We assume that $\mathbf{f}(\mathbf{x})$ has continuous first partial derivatives.

We define a Lyapunov function as

$$V(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\mathbf{P}\,\mathbf{f}(\mathbf{x}) \tag{9.72}$$

where $\mathbf{P}$ = a symmetric positive definite matrix.

Now,

$$\dot{V}(\mathbf{x}) = \dot{\mathbf{f}}^T(\mathbf{x})\mathbf{P}\,\mathbf{f}(\mathbf{x}) + \mathbf{f}^T(\mathbf{x})\mathbf{P}\,\dot{\mathbf{f}}(\mathbf{x}) \tag{9.73a}$$

where
$$\dot{\mathbf{f}}(\mathbf{x}) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\frac{d\mathbf{x}}{dt} = \mathbf{J}(\mathbf{x})\mathbf{f}(\mathbf{x});$$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{9.73b}$$

is the *Jacobian matrix* of $\mathbf{f}(\mathbf{x})$.

Substituting $\dot{\mathbf{f}}(\mathbf{x})$ in Eqn. (9.73a), we have

$$\dot{V}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})[\mathbf{J}^T(\mathbf{x})\mathbf{P} + \mathbf{P}\mathbf{J}(\mathbf{x})]\mathbf{f}(\mathbf{x}) \tag{9.74a}$$

Let
$$\mathbf{Q} = -[\mathbf{J}^T(\mathbf{x})\mathbf{P} + \mathbf{P}\mathbf{J}(\mathbf{x})] \tag{9.74b}$$

Since $V(\mathbf{x})$ is positive definite, for the system to be asymptotically stable at the origin, $\dot{V}(\mathbf{x})$ should be negative definite, or equivalently, $\mathbf{Q}$ should be positive definite. If, in addition, $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$, the system is asymptotically stable in-the-large.

Example 9.17

As an illustration of the Krasovskii method, consider the nonlinear system shown in Fig. 9.46, where the nonlinear element is described as

$$u = g(e) = e^3$$

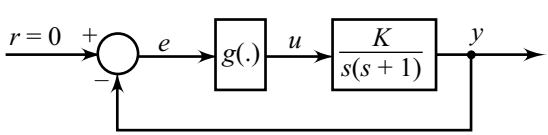


Fig. 9.46 A nonlinear system

The system is described by the differential equation

$$\ddot{e} + \dot{e} = -Ke^3;\ K > 0$$

Defining $x_1 = e$ and $x_2 = \dot{e}$, we get the following state equations:

$$\begin{aligned} \dot{x}_1 &= f_1(\mathbf{x}) = x_2 \\ \dot{x}_2 &= f_2(\mathbf{x}) = -x_2 - Kx_1^3 \end{aligned} \tag{9.75}$$

The equilibrium point lies at the origin.

Now $$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -3Kx_1^2 & -1 \end{bmatrix}$$

Let $$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}$$

For **P** to be positive definite,

$$p_{11} > 0 \tag{9.76a}$$

$$p_{11}p_{22} - p_{12}^2 > 0 \tag{9.76b}$$

The matrix

$$\begin{aligned} \mathbf{Q} &= -[\mathbf{J}^T(\mathbf{x})\mathbf{P} + \mathbf{P}\mathbf{J}(\mathbf{x})] \\ &= -\left\{ \begin{bmatrix} 0 & -3Kx_1^2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -3Kx_1^2 & -1 \end{bmatrix} \right\} \\ &= -\begin{bmatrix} -6p_{12}Kx_1^2 & p_{11} - p_{12} - 3p_{22}Kx_1^2 \\ p_{11} - p_{12} - 3p_{22}Kx_1^2 & 2(p_{12} - p_{22}) \end{bmatrix} \end{aligned}$$

For the system (9.75) to be asymptotically stable at the origin, **Q** should be positive definite, i.e.,

$$6p_{12}Kx_1^2 > 0 \tag{9.76c}$$

$$-12p_{12}Kx_1^2(p_{12} - p_{22}) - (p_{11} - p_{12} - 3p_{22}Kx_1^2)^2 > 0$$

or

$$12p_{12}Kx_1^2(p_{22} - p_{12}) > (p_{11} - p_{12} - 3p_{22}Kx_1^2)^2 \tag{9.76d}$$

Choose $p_{12} > 0$. Inequality (9.76c) then yields the condition $x_1^2 > 0$, which is always met.

Choose, $p_{11} = p_{12}$, and $p_{22} = \beta p_{12}$ with $\beta > 1$. Inequalities (9.76a) and (9.76b) are satisfied, and inequality (9.76d) gives the condition

$$12(\beta - 1) > 9\beta^2 Kx_1^2 \quad \text{or} \quad x_1^2 < \frac{4}{3K}\left(\frac{1}{\beta} - \frac{1}{\beta^2}\right)$$

It can easily be shown that the largest value of x_1 occurs when $\beta = 2$. Therefore,

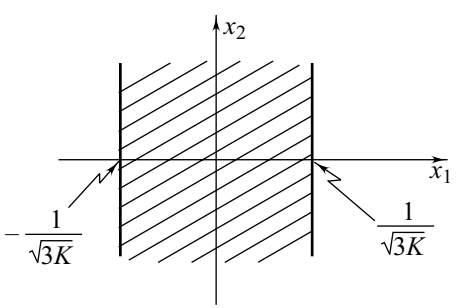


Fig. 9.47 Stability region of the nonlinear system of Fig. 9.46

$$x_1^2 < \frac{1}{3K} \quad \text{or} \quad -\frac{1}{\sqrt{3K}} < x_1 < \frac{1}{\sqrt{3K}}$$

This region of asymptotic stability is illustrated in Fig. 9.47.

9.13.2 The Variable Gradient Method

In searching for a Lyapunov function, we can approach the problem in a backward manner. We begin with an assumed form for the derivative $\dot{V}(\mathbf{x})$, and go back to choose the parameters of $V(\mathbf{x})$ so as to make $\dot{V}(\mathbf{x})$ negative definite. This is a useful idea in searching for a Lyapunov function. A procedure that exploits this idea is known as the *variable gradient method*.

To describe the procedure, let $V(\mathbf{x})$ be a scalar function of $\mathbf{x}$, and the vector function

$$\mathbf{g}(\mathbf{x}) = \text{gradient of } V(\mathbf{x}) = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial V}{\partial x_1} \\ \frac{\partial V}{\partial x_2} \\ \vdots \\ \frac{\partial V}{\partial x_n} \end{bmatrix} = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_n(\mathbf{x}) \end{bmatrix} \tag{9.77}$$

The derivative $\dot{V}(\mathbf{x})$ along the trajectories of Eqn. (9.71) is given by

$$\begin{aligned} \dot{V}(\mathbf{x}) &= \frac{\partial V}{\partial x_1}\dot{x}_1 + \frac{\partial V}{\partial x_2}\dot{x}_2 + \cdots + \frac{\partial V}{\partial x_n}\dot{x}_n \\ &= (\mathbf{g}(\mathbf{x}))^T \dot{\mathbf{x}} = (\mathbf{g}(\mathbf{x}))^T \mathbf{f}(\mathbf{x}) \end{aligned} \tag{9.78}$$

The idea now is to try to choose a vector function $\mathbf{g}(\mathbf{x})$, such that it would be a gradient of a positive definite scalar function $V(\mathbf{x})$ and, at the same time, $\dot{V}(\mathbf{x})$ would be negative definite. However, for a vector function $\mathbf{g}(\mathbf{x})$ to be gradient of a scalar $V(\mathbf{x})$, the Jacobian matrix of $\mathbf{g}(\mathbf{x})$ (refer to Eqn. (9.73b))

$$\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \cdots & \frac{\partial g_n}{\partial x_n} \end{bmatrix}$$

must be symmetric, i.e.,

$$\frac{\partial g_i}{\partial x_j} = \frac{\partial g_j}{\partial x_i} \forall i, j = 1, 2, ..., n \tag{9.79}$$

This is so because the Hessian matrix (refer to Eqn.(8.18))

$$\frac{\partial}{\partial \mathbf{x}}\left(\frac{\partial V}{\partial \mathbf{x}}\right)=\frac{\partial^2 V(\mathbf{x})}{\partial \mathbf{x}^2}=\begin{bmatrix}\dfrac{\partial^2 V}{\partial x_1^2} & \dfrac{\partial^2 V}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 V}{\partial x_1 \partial x_n}\\ \vdots & \vdots & & \vdots\\ \dfrac{\partial^2 V}{\partial x_n \partial x_1} & \dfrac{\partial^2 V}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 V}{\partial x_n \partial x_n}\end{bmatrix}$$

is always symmetric $\left(\dfrac{\partial^2 V}{\partial x_i \partial x_j}=\dfrac{\partial^2 V}{\partial x_j\, \partial x_i}\right)$.

Under the constraints (9.79), we start by choosing $\mathbf{g}(\mathbf{x})$ such that $(\mathbf{g}(\mathbf{x}))^T\mathbf{f}(\mathbf{x})$ is negative definite. The function $V(\mathbf{x})$ is then computed from the integral

$$V(\mathbf{x})=\int_{\mathbf{0}}^{\mathbf{x}}\mathbf{g}^T(\mathbf{y})d\mathbf{y}=\int_{0}^{\mathbf{x}}\sum_{i=1}^{n}g_i\left(y_1,y_2,...,y_n\right)dy_i \tag{9.80a}$$

The integration is taken over any path joining the origin to $\mathbf{x}$ (The line integral of a gradient vector is independent of the path [125]). Usually, this is done along the axes; that is

$$\begin{aligned}V(\mathbf{x})=&\int_0^{x_1}g_1\left(y_1,0,0,...,0\right)dy_1+\int_0^{x_2}g_2\left(x_1,y_2,0,...,0\right)dy_2\\&+\cdots+\int_0^{x_n}g_n\left(x_1,x_2,...,x_{n-1},y_n\right)dy_n\end{aligned} \tag{9.80b}$$

By leaving some parameters of $\mathbf{g}(\mathbf{x})$ undetermined, one would try to choose them to ensure that $V(\mathbf{x})$ is positive definite.

Example 9.18

Let us use the variable gradient method to find a Lyapunov function for the nonlinear system

$$\begin{aligned}\dot{x}_1&=-x_1\\ \dot{x}_2&=-x_2+x_1x_2^2\end{aligned} \tag{9.81}$$

We assume the following form for the gradient of the undetermined Lyapunov function.

$$\mathbf{g}(\mathbf{x})=\begin{bmatrix}g_1(\mathbf{x})\\ g_2(\mathbf{x})\end{bmatrix}=\begin{bmatrix}a_{11}x_1+a_{12}x_2\\ a_{21}x_1+a_{22}x_2\end{bmatrix};\ a_{ij}\text{ may be functions of }\mathbf{x} \tag{9.82}$$

The function has to satisfy the constraints (9.79):

$$\frac{\partial g_1}{\partial x_2}=\frac{\partial g_2}{\partial x_1},\ \text{i.e.,}\ \ a_{12}+x_2\frac{\partial a_{12}}{\partial x_2}=a_{21}+x_1\frac{\partial a_{21}}{\partial x_1}$$

If the coefficients are chosen to be

$$a_{11}=a_{22}=1;\ a_{12}=a_{21}=0$$

then

$$g_1(\mathbf{x}) = x_1 \text{ and } g_2(\mathbf{x}) = x_2$$

and

$$\begin{aligned}\dot{V}(\mathbf{x}) &= (\mathbf{g}(\mathbf{x}))^T \mathbf{f}(\mathbf{x}) \\ &= -x_1^2 - x_2^2 (1 - x_1 x_2)\end{aligned} \tag{9.83}$$

Thus, if $(1 - x_1x_2) > 0$, then $\dot{V}$ is negative definite. The function $V(\mathbf{x})$ can be computed as

$$V(\mathbf{x}) = \int_0^{x_1} y_1 dy_1 + \int_0^{x_2} y_2 dy_2 = \frac{x_1^2 + x_2^2}{2} \tag{9.84}$$

This is a positive definite function and, therefore, the asymptotic stability of the origin in the region $1 > x_1x_2$ is guaranteed.

Note that (9.84) is not the only Lyapunov function obtainable by the variable gradient method. A different choice of a_{ij}'s may lead to another Lyapunov function for the system.

9.14 LYPUNOV'S LINEARIZATION METHOD AND LOCAL STABILITY

Lyapunov's original work, first published in 1892, included two methods for stability analysis; the so-called *Lyapunov's first method* (*linearization method*) and Lyapunov's second method (*direct method*). The linearization method draws conclusions about a nonlinear system's local stability around an equilibrium point from the stability properties of its linear approximation. The direct method is not restricted to local motion and determines the stability of a nonlinear system (directly without linearization) by constructing a scalar function for a system and examining the function's time variation. We have so far discussed Lyapunov's second method/direct method. In the following discussion, the key results of Lyapunov's first method/linearization method are presented.

(i) If the linearized system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ is strictly stable (i.e., if all the eigenvalues of $\mathbf{A}$ are strictly in the left half of the complex plane), then the equilibrium point of the actual nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ is locally asymptotically stable.

(ii) If the linearized system is unstable (i.e., if at least one eigenvalue of $\mathbf{A}$ is strictly in the right half of the complex plane), then the equilibrium point is locally unstable for the nonlinear system.

(iii) If the linearized system has all eigenvalues of $\mathbf{A}$ in the left half of the complex plane, but also has at least one eigenvalue having zero real part, then one cannot conclude anything from the linear approximation (the equilibrium point may be stable in the sense of Lyapunov, locally asymptotically stable or locally unstable for the nonlinear system.

While the proof of these results are not detailed here, *intuitive* justification is obvious. If the linearized system is strictly stable or strictly unstable, then, since the approximation is valid 'not too far' from the equilibrium, the nonlinear system itself is locally stable or locally unstable. However, if the linearized system has roots with zero real parts, the higher-order terms in Taylor series expansion (refer to Eqns (5.11)) can have decisive effect on whether the nonlinear system is stable or unstable. Simple nonlinear systems may be globally asymptotically stable while their linear approximations have roots

with zero real parts; one simply cannot infer any stability property of a nonlinear system from its linear approximation with roots having zero real parts.

Today, Lyapunov's linearization method has come to represent the theoretical justification of linear control, while Lyapunov's direct method has become the most important tool for nonlinear system analysis and design. Lyapunov's linearization method shows that linear control design is a matter of *consistency*; one must design a controller such that the system remain in its 'linear range'. It also stresses on major limitations of linear design, i.e., how large is the linear range from stability considerations? Such questions motivate a deeper approach to nonlinear system analysis.

REVIEW EXAMPLES

Review Example 9.1

Figure 9.48 shows the input-output waveforms of a saturating element or a limiter.

For small input signals ($X < S$), the output is proportional to the input. However, if the input amplitude is sufficiently large to cause saturation ($X > S$), the output is a clipped sine wave. One cycle of the output, which is a periodic function of period 2π, is described as follows:

$$y = \begin{cases} Kx; & 0 \le \omega t < \alpha \\ KS; & \alpha \le \omega t < (\pi - \alpha) \\ Kx; & (\pi - \alpha) \le \omega t < (\pi + \alpha) \\ -KS; & (\pi + \alpha) \le \omega t < (2\pi - \alpha) \\ Kx; & (2\pi - \alpha) \le \omega t \le 2\pi \end{cases}$$

where $\alpha = \sin^{-1}(S/X)$

This periodic function has odd symmetry:

$$y(\omega t) = -y(-\omega t)$$

Therefore, the fundamental component of y is given by (refer to Eqn. (9.4d))

$$y_1 = B_1 \sin \omega t$$

where

$$B_1 = \frac{1}{\pi} \int_0^{2\pi} y \sin \omega t \, d(\omega t)$$

Due to symmetry of y (refer to Fig. 9.48), the coefficient B_1 can be calculated as follows:

$$B_1 = \frac{4}{\pi} \int_0^{\frac{\pi}{2}} y \sin \theta \, d\theta = \frac{4K}{\pi} \left[\int_0^{\alpha} X \sin^2 \theta \, d\theta + \int_{\alpha}^{\frac{\pi}{2}} S \sin \theta \, d\theta \right]$$

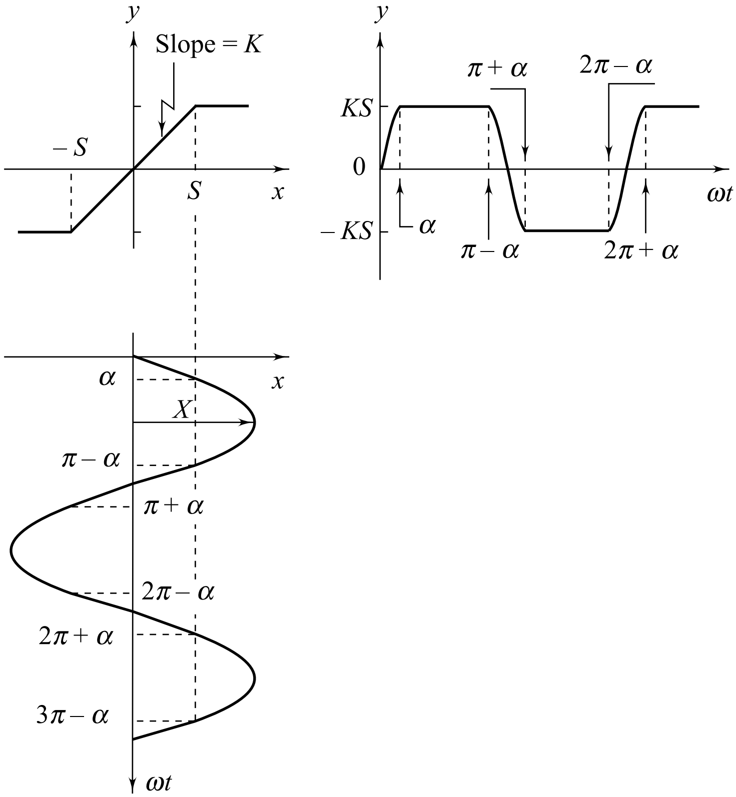


Fig. 9.48 Fourier-series analysis of a saturating element

$$= \frac{4K}{\pi}\left[\frac{X}{2}(\alpha - \sin\alpha\cos\alpha) + S\cos\alpha\right]$$

$$\frac{B_1}{X} = \frac{2K}{\pi}\left[\sin^{-1}\frac{S}{X} - \frac{S}{X}\sqrt{1-\left(\frac{S}{X}\right)^2} + \frac{2S}{X}\sqrt{1-\left(\frac{S}{X}\right)^2}\right]$$

Therefore,

$$N(X) = \begin{cases} \dfrac{2K}{\pi}\left[\sin^{-1}\dfrac{S}{X} + \dfrac{S}{X}\sqrt{1-\left(\dfrac{S}{X}\right)^2}\right]; & X \geq S \\ K & ; X < S \end{cases} \tag{9.85}$$

The describing functions given by Eqn. (9.85) and nonlinearity 4 in Table 9.2, have a common term of the form

$$N_c(z) = \frac{2}{\pi}\left[\sin^{-1}\frac{1}{z} + \frac{1}{z}\sqrt{1-\left(\frac{1}{z}\right)^2}\right] \tag{9.86}$$

In terms of $N_c(z)$, the describing function (9.85) may be expressed as

$$N(X) = KN_c\left(\frac{X}{S}\right) \tag{9.87}$$

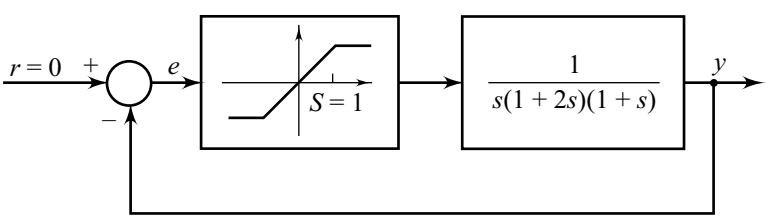
The function $N_c(z)$ is listed in Table 9.3.

Table 9.3 Values of $N_c(z)$ given by Eqn. (9.86)

z	$N_c(z)$	z	$N_c(z)$	z	$N_c(z)$
1.0	1.000	6.0	0.211	11.0	0.116
1.5	0.781	6.5	0.195	11.5	0.111
2.0	0.609	7.0	0.181	12.0	0.106
2.5	0.495	7.5	0.169	12.5	0.102
3.0	0.416	8.0	0.159	13.0	0.0978
3.5	0.359	8.5	0.149	14.0	0.0909
4.0	0.315	9.0	0.141	15.0	0.0848
4.5	0.281	9.5	0.134	19.0	0.0670
5.0	0.253	10.0	0.127	25.0	0.0509
5.5	0.230	10.5	0.121	30.0	0.0424
				50.0	0.0255
				100.0	0.0127

Review Example 9.2

Consider the nonlinear system of Fig. 9.49a, with a saturating amplifier having gain K in its linear region. Determine the largest value of gain K for the system to stay stable. What would be the frequency, amplitude and nature of the limit cycle for a gain $K = 3$?



(a)

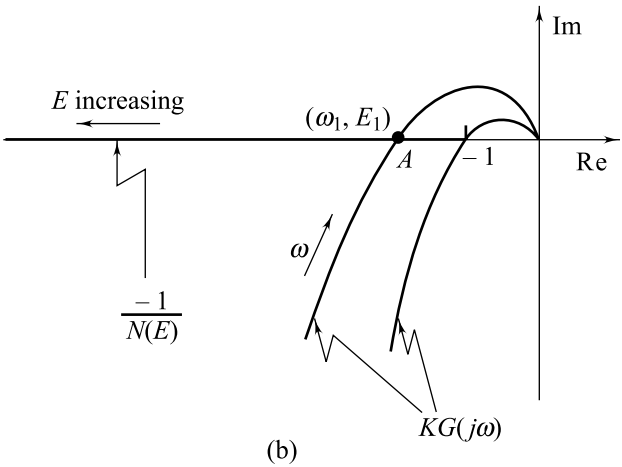


Fig. 9.49 Nonlinear system with saturating amplifier

Solution It is convenient to regard the amplifier to have unit gain and the gain K to be attached to the linear part. From Eqn. (9.87), we obtain, for $S = 1$ and $K = 1$, $N(E) = N_c(E)$; the function $N_c(E)$ is listed in Table 9.3.

The locus of $-1/N(E)$ thus starts from $(-1 + j0)$ and travels along the negative real axis for increasing E, as shown in Fig. 9.49b. Now, for the equation

$$KG(j\omega) = -1/N(X)$$

to be satisfied, $G(j\omega)$ must have an angle of $-180°$:

$$\angle G(j\omega) = -90° - \tan^{-1} 2\omega - \tan^{-1}\omega = -180°$$

This gives

$$\frac{2\omega + \omega}{1 - 2\omega^2} = \tan 90° = \infty \text{ or } \omega = 1/\sqrt{2} \text{ rad/sec.}$$

The largest value of K for stability is obtained when $KG(j\omega)$ passes through $(-1 + j0)$, i.e.,

$$|KG(j\omega)|_{\omega=1/\sqrt{2}} = 1 \quad \text{or} \quad \frac{K}{\left(\frac{1}{\sqrt{2}}\right)\left(\sqrt{3}\right)\left(\frac{\sqrt{3}}{\sqrt{2}}\right)} = 1 \text{ or } K = 3/2$$

For $K = 3$, $KG(j\omega)$ plot intersects $-1/N(X)$ locus, resulting in a limit cycle at (ω_1, E_1) where $\omega_1 = 1/\sqrt{2}$, while E_1 is obtained from the relation

$$|-1/N(E_1)| = |3G(j\omega_1)| = 2 \quad \text{or} \quad |N(E_1)| = 0.5$$

From Table 9.3, we obtain

$$E_1 \cong 2.5$$

Applying the stability test for the limit cycle reveals that point A in Fig. 9.49b corresponds to a stable limit cycle.

Review Example 9.3

Consider the servomechanism of Fig. 9.50, having a deadzone nonlinearity in the feedback loop. With $x_1 = y$ and $x_2 = \dot{y}$, we get

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \begin{cases} -x_2; -1 < x_1 < 1 \\ -x_2 - 2(x_1 - 1); x_1 > 1 \\ -x_2 - 2(x_1 + 1); x_1 < -1 \end{cases}$$

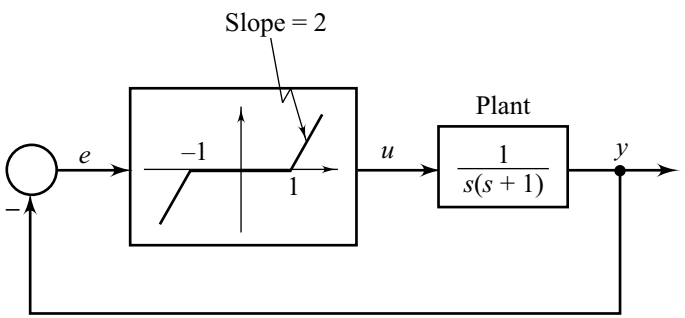


Fig. 9.50 A system with deadzone nonlinearity

The phase plane may be divided into three regions:

Region I (defined by $-1 < x_1 < 1$) The isocline equation is

$$m = \frac{-x_2}{x_2} = -1$$

Since m is the slope of phase trajectories, all trajectories in Region I have a slope of -1. Typical trajectories are shown in Fig. 9.51.

Region II (defined by $x_1 > 1$) The isocline equation is

$$m = \frac{-x_2 - 2(x_1 - 1)}{x_2}$$

or

$$x_2 = \frac{-2(x_1 - 1)}{m + 1}$$

The isoclines are straightlines intersecting the x_1-axis at $x_1 = 1$, with slope equal to $-2/(m + 1)$. Some of these isoclines are shown in Fig. 9.51.

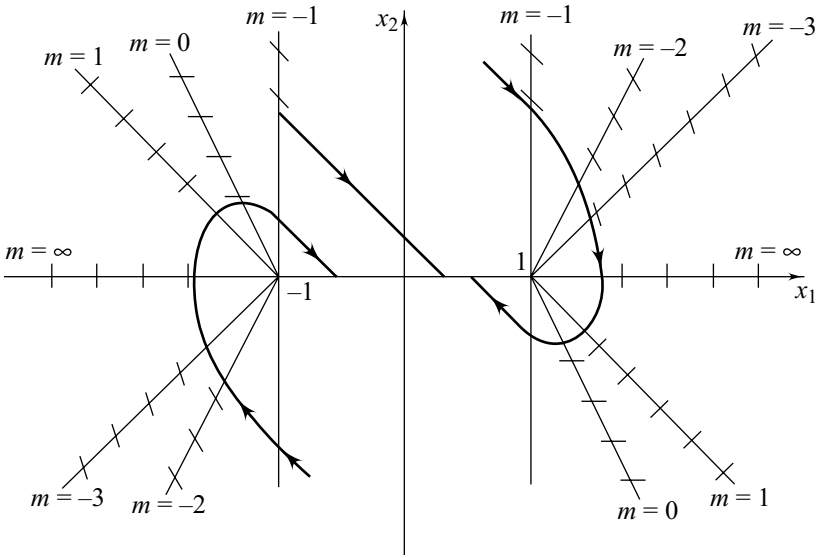


Fig. 9.51 Isoclines and typical trajectories for the system of Fig. 9.50

Region III (defined by $x_1 < -1$) The isocline equation is

$$x_2 = \frac{-2(x_1+1)}{m+1}$$

These isoclines are straightlines intersecting the x_1-axis at $x_1 = -1$, with slope equal to $-2/(m+1)$. Some of these isoclines are shown in Fig. 9.51.

Some typical phase trajectories are also shown in Fig. 9.51. Note that for the given system, we have an *equilibrium zone*, $-1 \le x_1 \le 1$, with $x_2 = 0$. The system comes to rest at any point within this zone.

Review Example 9.4

Consider the nonlinear system

$$\dot{x}_1 = -x_1 - x_2^2$$
$$\dot{x}_2 = -x_2$$

Investigate the stability of the equilibrium points.

Solution The given system is

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

or

$$\dot{x}_1 = f_1(\mathbf{x}) = -x_1 - x_2^2$$
$$\dot{x}_2 = f_2(\mathbf{x}) = -x_2$$

Clearly, $\mathbf{x} = \mathbf{0}$ is the only equilibrium point.

In the following, we apply Krasovskii method to determine sufficient conditions for asymptotic stability, in the vicinity of the equilibrium point.

A candidate for a Lyapunov function is

$$V(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\mathbf{P}\mathbf{f}(\mathbf{x})$$

Selecting $\mathbf{P} = \mathbf{I}$ may lead to a successful determination of the conditions for asymptotic stability in the vicinity of the equilibrium point.

With this choice of Lyapunov function, we have (refer to Eqns (9.74))

$$\dot{V}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})[\mathbf{J}^T(\mathbf{x}) + \mathbf{J}(\mathbf{x})]\mathbf{f}(\mathbf{x})$$

where

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -1 & -2x_2 \\ 0 & -1 \end{bmatrix}$$

The matrix

$$\mathbf{Q} = -[\mathbf{J}^T(\mathbf{x}) + \mathbf{J}(\mathbf{x})] = \begin{bmatrix} 2 & 2x_2 \\ 2x_2 & 2 \end{bmatrix}$$

Using Sylvester's criterion (Section 5.2), we find that the matrix $\mathbf{Q}$ is positive definite if

$$4 - 4x_2^2 > 0 \quad \text{or} \quad |x_2| < 1$$

The shaded region in Fig. 9.52 is the region of asymptotic stability. It is, however, not the largest region. Another choice of Lyapunov function for the system under consideration, may lead to a larger region of asymptotic stability in the vicinity of the equilibrium point.

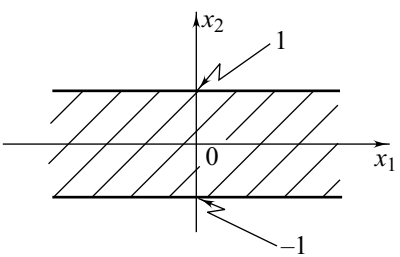


Fig. 9.52

PROBLEMS

9.1 For a sinusoidal input $x = X \sin \omega t$, find the output waveforms for each of the nonlinearities listed in Table 9.2. By Fourier-series analysis of the output waveforms, derive the describing function for each entry of the table.

9.2 Consider the system shown in Fig. P9.2. Using the describing-function analysis, show that a stable limit cycle exists for all values of $K > 0$. Find the amplitude and frequency of the limit cycle when $K = 4$, and plot $y(t)$ versus t.

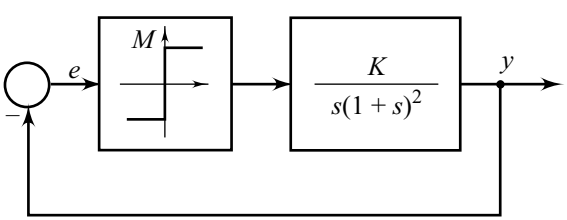


Fig. P9.2

9.3 Consider the system shown in Fig. P9.3. Use the describing function technique

to investigate the possibility of limit cycles in this system. If a stable limit cycle is predicted, determine its amplitude and frequency.

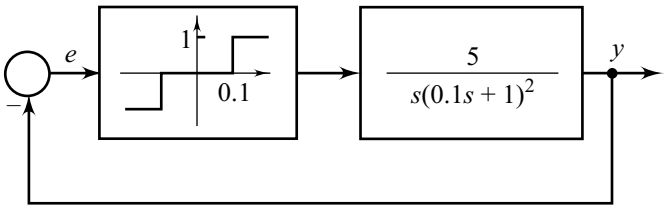


Fig. P9.3

9.4 Using the describing-function analysis, prove that no limit cycle exists in the system shown in Fig. P9.4. Find the range of values of the deadzone of the on–off controller for which limit cycling will be predicted.

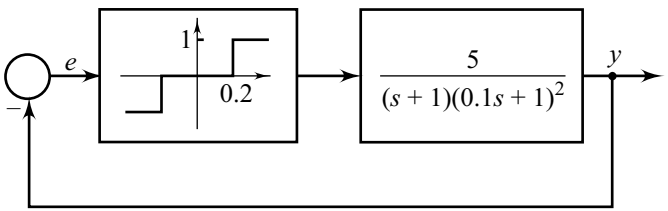


Fig. P9.4

9.5 Consider the system shown in Fig. P9.5. Using the describing-function technique, show that a stable limit cycle cannot exist in this system for any $K > 0$.

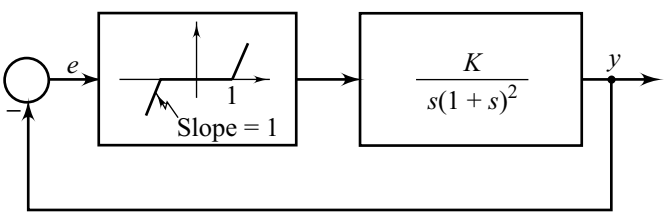


Fig. P9.5

9.6 Consider the system shown in Fig. P9.6. Using the describing-function analysis, investigate the possibility of a limit cycle in the system. If a limit cycle is predicted, determine its amplitude and frequency, and investigate its stability.

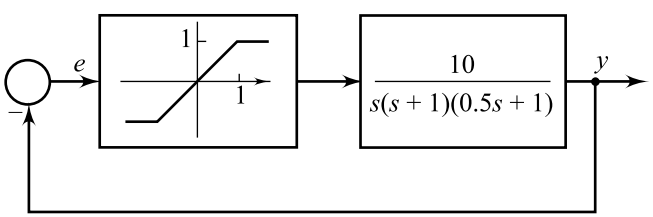


Fig. P9.6

9.7 An instrument servo system used for positioning a load, may be adequately represented by the block diagram in Fig. P9.7a. The backlash characteristic is shown in Fig. P9.7b. Show that the system is stable for $K = 1$. If the value of K is now raised to 2, show that limit cycles exist. Investigate the stability of these limit cycles. Determine the amplitude and frequency of the stable limit cycle.

Given:

H/X	0	0.1	0.2	0.5	0.7	0.8	0.9	0.95	1.0
$\lvert N(X)\rvert$	1	0.954	0.882	0.593	0.367	0.249	0.126	0.063	0
$\angle N(X)$	0	–6.7°	–13.4°	–32.5°	–46.6°	–55.1°	–65.6°	–72.8°	–90°

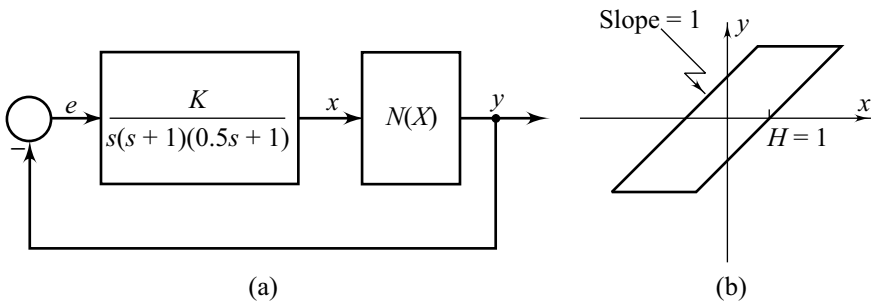


Fig. P9.7

9.8 Determine the kind of singularity for each of the following differential equations. Also locate the singular points on the phase plane.

(a) $\ddot{y} + 3\dot{y} + 2y = 0$ (b) $\ddot{y} + 5\dot{y} + 6y = 6$ (c) $\ddot{y} - 8\dot{y} + 17y = 34$

9.9 An undamped pendulum is described by the differential equation

$$ml^2\ddot{\theta} + mgl \sin\theta = 0$$

where mg is the weight of the pendulum, and l is its length. Show that this nonlinear system has two equilibrium points: $\theta = 0$, and $\theta = \pi$. Develop linear state models using Taylor series approximation around the two equilibrium points, and therefrom identify the kind of singularity at each point.

9.10 A linear second-order servo is described by the equation

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = \omega_n^2$$

where $\omega_n = 1, y(0) = 2.0, \dot{y}(0) = 0$

Determine the singular points when (i) $\zeta = 0$, (ii) $\zeta = 0.15$. Construct the phase trajectory in each case.

9.11 Consider the block diagram of a system, shown in Fig. P9.11.

(a) Derive state variable model with $x_1 = e$ and $x_2 = \dot{e}$.

(b) Write equations of the isoclines on the x_1 versus x_2 plane.

(c) Given: $x_1(0) = 1, x_2(0) = 0$, obtain a trajectory on the x_1 versus x_2 plane. Show singular points (if any) and some isoclines.

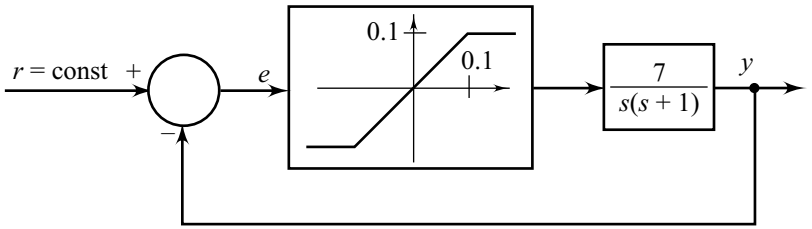


Fig. P9.11

9.12 Consider the block diagram of a system, shown in Fig. P9.12.

(a) Derive state variable model with $x_1 = e$ and $x_2 = \dot{e}$.

(b) Write equations of the isoclines on the x_1 versus x_2 plane.

(c) Given: $x_1(0) = 1$, $x_2(0) = 0$, obtain a trajectory on the x_1 versus x_2 plane. Show singular points (if any) and some isoclines.

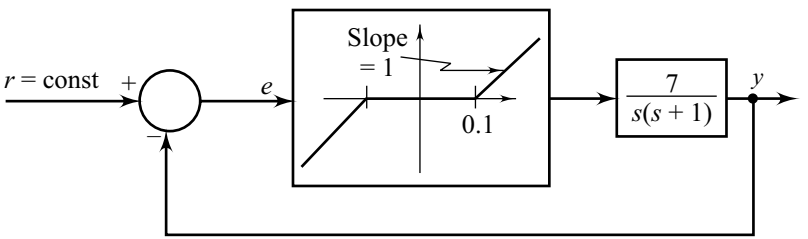


Fig. P9.12

9.13 Consider the block diagram of a system, shown in Fig. P9.13.

(a) Derive state variable model with $x_1 = e$ and $x_2 = \dot{e}$.

(b) Write equations of the isoclines on the x_1 versus x_2 plane.

(c) Given: $x_1(0) = 1$, $x_2(0) = 0$, obtain a trajectory on the x_1 versus x_2 plane. Show singular points (if any) and some isoclines.

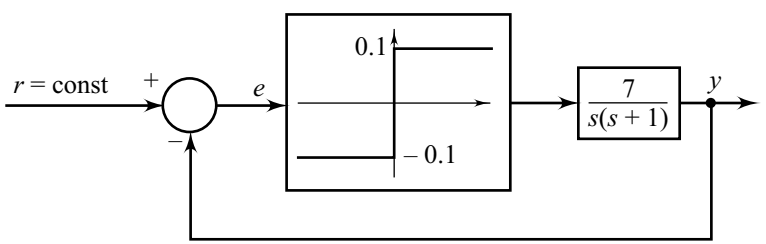


Fig. P9.13

9.14 Consider the block diagram of a system, shown in Fig. P9.14.

(a) Derive state variable model with $x_1 = e$ and $x_2 = \dot{e}$.

(b) Write equations of the isoclines on the x_1 versus x_2 plane.

(c) Given: $x_1(0) = 1$, $x_2(0) = 0$, obtain a trajectory on the x_1 versus x_2 plane. Show singular points (if any) and some isoclines.

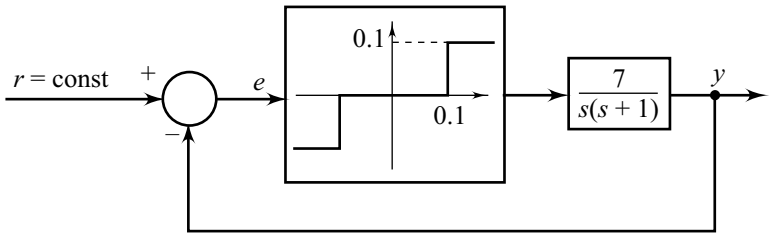


Fig. P9.14

9.15 Consider the block diagram of a system, shown in Fig. P9.15.

(a) Derive state variable model with $x_1 = e$ and $x_2 = \dot{e}$.

(b) Write equations of the isoclines on the x_1 versus x_2 plane.

(c) Given: $x_1(0) = 1$, $x_2(0) = 0$, obtain a trajectory on the x_1 versus x_2 plane. Show singular points (if any) and some isoclines.

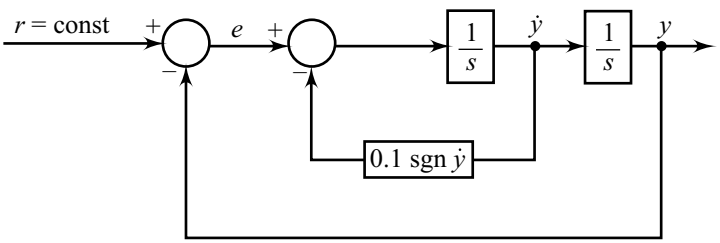


Fig. P9.15

9.16 The position control system shown in Fig. P9.16 has Coulomb friction T_c sgn$(\dot{\theta})$ at the output shaft. Prove that the phase trajectories on $(e,\ \dot{e}/\omega_n)$-plane are semicircles, with the center on horizontal axis at $+T_c/K$ for $\dot{e} < 0$ and $-T_c/K$ for $\dot{e} > 0$.

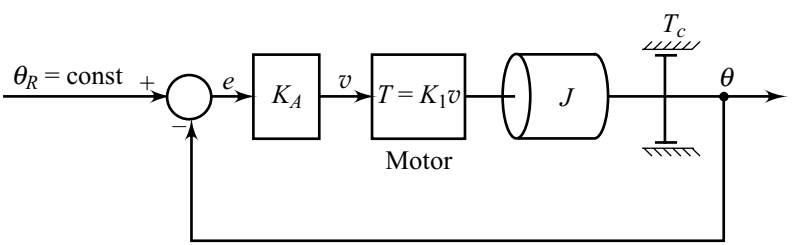


Fig. P9.16

Examine the phase trajectory corresponding to θ_R = unit step, $\dot{\theta}(0) = \theta(0) = 0$; and find the value of the steady-state error. What is the largest possible steady-state error which the system in Fig. P9.16 can possess?

Given:

$$\omega_n = \sqrt{K/J} = 1.2 \text{ rad/sec}; \quad T_c/K = 0.3 \text{ rad}$$

where $K = K_A K_1$.

9.17 Consider the nonlinear system with deadzone shown in Fig. P9.17. Using the method of isoclines, sketch some typical phase trajectories with and without deadzone, and comment upon the effect of deadzone on transient and steady-state behavior of the system.

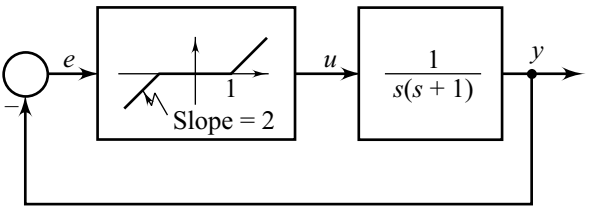


Fig. P9.17

9.18 Consider the system shown in Fig. P9.18 in which the nonlinear element is a power amplifier with gain equal to 1.0, which saturates for error magnitudes greater than 0.4. Given the initial condition: $e(0) = 1.6$, $\dot{e}(0) = 0$, plot phase trajectories with and without saturation, and comment upon the effect of saturation on the transient behavior of the system. Use the method of isoclines for construction of phase trajectories.

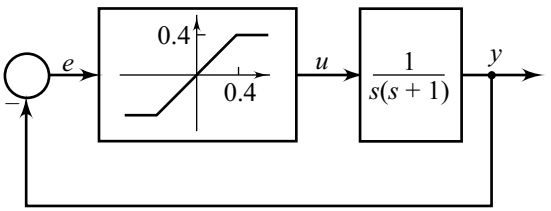


Fig. P9.18

9.19 (a) A plant with model $G(s) = 1/s^2$ is placed in a feedback configuration as in Fig. P9.19a. Construct a trajectory on the $(e, \dot{e})$ plane with $r = 2$ and $y(0) = \dot{y}(0) = 0$. Show that the system response is a limit cycle. What is the switching line of this variable structure system?

(b) To the feedback control system of Fig. P9.19a, we add a derivative feedback with gain K_D as in Fig. P9.19b. Show that the limit cycle gets eliminated by the introduction of derivative-control term. What is the switching line of this variable structure system?

(c) Show that if K_D is large, the trajectory may slide along the switching line towards the origin.

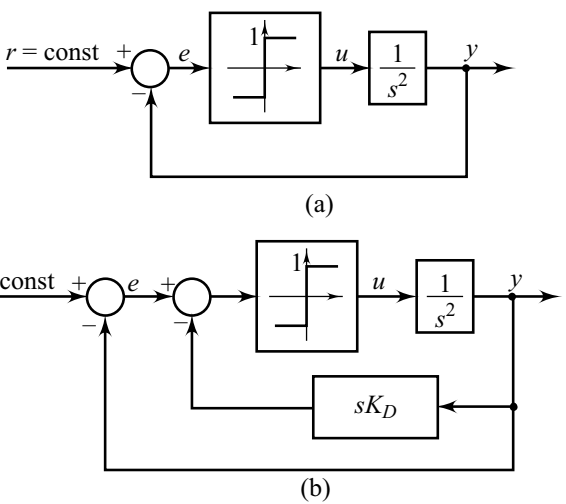


Fig. P9.19

9.20 A position control system comprises of a dc servo motor, potentiometer error detector, an on–off controller, and a tachogenerator coupled to the motor shaft.

The following equations describe the system:

Reaction torque = $\ddot{\theta} + 0.5\dot{\theta}$

Drive torque = $2\,\text{sgn}(e + 0.5\,\dot{e})$; $e = \theta_R - \theta$

(a) Make a sketch of the system showing how the hardware is connected.

(b) Construct a phase trajectory on $(e, \dot{e})$-plane with $e(0) = 2$ and $\dot{e}(0) = 0$, and comment upon the transient and steady-state behavior of the system.

(c) What is the switching line of the variable structure system?

9.21 (a) Describe the system of Fig. P9.21a on the $(e,\dot{e})$-plane, and show that with the on–off controller switching on the vertical axis of the phase plane, the system oscillates with increasing frequency and decreasing amplitude. Obtain a phase trajectory with $(e(0) = 1.4, \dot{e}(0) = 0)$ as the initial state point.

(b) Introduce now a deadzone of ± 0.2 in the on–off controller characteristic. Obtain a phase trajectory for the modified system with $(e(0) = 1.4, \dot{e}(0) = 0)$ as the initial state point and comment upon the effect of deadzone.

(c) The on–off controller with deadzone is now controlled by the signal $\left(e + \frac{1}{3}\dot{e}\right)$, combining proportional and derivative control (Fig. P9.21b). Draw the switching line on the phase plane and construct a phase trajectory with $(e(0) = 1.4, \dot{e}(0) = 0)$ as the initial state point. What is the effect of the derivative-control action?

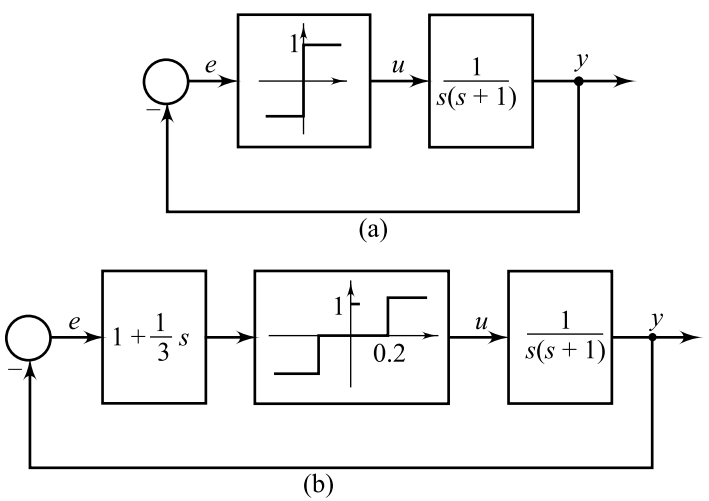


Fig. P9.21

9.22 Consider the second-order system

$$\dot{x}_1 = x_2; \quad \dot{x}_2 = -u$$

It is desired to transfer the system to the origin in minimum time from an arbitrary initial state. Use the bang-bang control strategy with $|u| = 1$. Derive an expression for the optimum switching curve. Construct a phase portrait showing a few typical minimum-time trajectories.

9.23 A plant with model $G(s) = \dfrac{1}{s(s+1)}$ is placed in a feedback configuration as shown in Fig. P9.23. It is desired to transfer the system from any initial state to the origin in minimum time. Derive an expression for optimum switching curve and construct a phase portrait on the $(e,\dot{e})$-plane showing a few typical minimum-time trajectories.

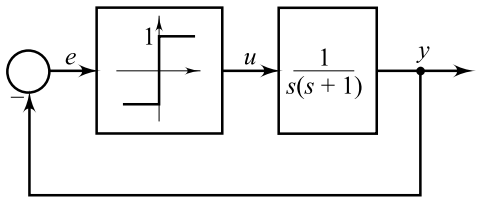


Fig. P9.23

9.24 Consider the nonlinear system described by the equations

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -(1 - |x_1|)x_2 - x_1$$

Find the region in the state plane for which the equilibrium state of the system is asymptotically stable.

9.25 Check the stability of the equilibrium state of the system described by

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 - x_1^2 x_2$$

Show that the Lyapunovs linearization method fails while the Lyapunovs direct method can easily solve this problem.

9.26 Consider a nonlinear system described by the equations

$$\dot{x}_1 = -3x_1 + x_2$$
$$\dot{x}_2 = x_1 - x_2 - x_2^3$$

Using the Krasovskii method for constructing the Lyapunov function with **P** as identity matrix, investigate the stability of the equilibrium state.

Find a region of asymptotic stability using Krasovskii method.

9.27 Check the stability of the system described by

$$\dot{x}_1 = -x_1 + 2x_1^2 x_2$$
$$\dot{x}_2 = -x_2$$

by use of the variable gradient method.

9.28 Develop a linearized state model for the Van der Pol's differential equation

$$\ddot{y} + \mu (y^2 - 1)\ \dot{y} + y = 0;\ \mu = 1$$

and therefrom determine the local stability of the nonlinear system using Lyapunov's first method.

9.29 Consider the nonlinear system described by the equations

$$\dot{x}_1 = x_1 (x_1^2 + x_2^2 - 1) - x_2$$
$$\dot{x}_2 = x_1 + x_2 (x_1^2 + x_2^2 - 1)$$

Investigate the stability of this nonlinear system around its equilibrium point at the origin.

9.30 Consider a nonlinear system described by the differential equation

$$\ddot{x} + [K_1 + K_2(\dot{x})^2]\ \dot{x} + x = 0$$

Check the stability of the equilibrium state of this system when (i) $K_1 > 0$ and $K_2 > 0$; (ii) $K_1 < 0$ and $K_2 < 0$; and (iii) $K_1 > 0$ and $K_2 < 0$.

Chapter 10

Nonlinear Control Structures

10.1 INTRODUCTION

In the previous chapter, we covered tools and techniques for the analysis of control systems containing nonlinearities. In the present chapter, we will discuss the deliberate introduction of nonlinearities into the controller for performance improvement over that of a simpler linear controller. Although there are numerous techniques, several of the most common are illustrated with examples, to give the reader an idea of the general approach to designing nonlinear controllers.

If the system is only mildly nonlinear, the simplest approach might be to *ignore the nonlinearity* in designing the controller (i.e., to omit the nonlinearity in the design model), but to include its effect in evaluating the system performance. The inherent robustness of the feedback control law designed for the approximating nonlinear system is relied upon to carry it over to the nonlinear system.

When a system is significantly nonlinear, it is traditionally dealt with by *linearization* (refer to Eqns (5.11)) about a selected operating point *using Taylor series*. We design a linear controller based on first-order approximation. If the controller works effectively, the perturbations in actual state about the equilibrium state will be small; if the perturbations are small, the neglected higher-order terms will be small and can be regarded as a disturbance. Since the controller is designed to counteract the effects of disturbances, the presence of higher-order terms should cause no problems. This reasoning cannot be justified rigorously, but, nevertheless, it usually works. Needless to say, it may not always work; so it is necessary to test the design that emerges, for stability and performance—analytically, by Lyapunov's stability analysis for example, and/or by simulation.

In many systems, the nonlinearity inherent in the plant is so dominant that the linearization approach described above can hardly meet the stringent requirements on systems' performance. This reality, inevitably, promotes the endeavor to develop control approaches that will more or less incorporate the nonlinear dynamics into the design process. One such approach is *feedback linearization*. Unlike the first-order approximation approach, wherein the higher-order terms of the plant are ignored, this approach utilizes the feedback to render the given system, a linear input-output dynamics. On the basis of the linear system thus obtained, linear control techniques can be applied to address design issues.

The roughness of the linearization approach based on first-order approximation, can be viewed from two perspectives. First, it neglects all higher-order terms. Second, the linear terms depend on the equilibrium

point. These two uncertainties may explain why this linearization approach is incapable of dealing with the situation where the system operates over wide dynamic regions. Although the feedback linearization may overcome the first drawback, its applicability is limited, mainly because it rarely leads to a design that guarantees the system performance over the whole dynamic regime. This is because feedback linearization is often performed locally—around a specific equilibrium point. The resulting controller is of local nature. Another remedy to linearization based on first-order approximation, is to design several control laws corresponding to several operating points that cover the whole dynamics of the system. Then these linear controllers are pieced together to obtain a nonlinear control law. This approach is often called *gain scheduling*. Though this approach does not account for the higher-order terms, it does accommodate the variation of the first-order terms with respect to the equilibrium points.

Adaptive control theory provides an effective tool for the design of uncertain systems. Unlike *fixed-parameter controllers* (e.g., H_∞-theory-based robust controller), *adaptive controllers* adapt (adjust) their behavior on-line to the changing properties of the controlled processes.

If a fixed-parameter automatic control system is used, the plant-parameter variations directly affect the capability of the design to meet the performance specifications under all operating conditions. If an adaptive controller is used, the plant-parameter variations are accounted for at the price of increased complexity of the controller. Adaptive control is certainly more complex than fixed-parameter control, and carries with it more complex failure mechanisms. In addition, adaptive control is both time-varying and nonlinear, increasing the difficulty of stability and performance analysis. It is this tradeoff, of complexity versus performance, that must be examined carefully in choosing the control structure.

The main distinctive feature of *variable structure systems* (VSS), setting them apart as an independent class of control systems, is that changes can occur in the structure of the system during the transient process. The structure of a VSS is changed intentionally in accordance with some law of structural change; the times at which these changes occur (and the type of structure formed) are determined not by a fixed program, but in accordance with the current value of the states of the system.

The basic idea of feedback linearization in Section 10.2 is the algebraic transformation of the dynamics of a nonlinear system to that of a linear system, on which linear control designs can in turn be applied. Sections 10.3–10.5 show how to reduce, or practically eliminate, the effects of model uncertainties on the stability and performance of feedback controllers using so-called adaptive and variable structure systems. The chapter concentrates on nonlinear systems represented in continuous-time form. Even though most control systems are implemented digitally, nonlinear physical systems are continuous in nature and are hard to discretize meaningfully, while digital control systems may be treated as continuous-time systems in analysis and design if high sampling rates are used. Thus, we perform nonlinear system analysis and controller design in continuous-time form. However, of course, the control law is generally implemented digitally.

The objective set for this chapter is to make the reader aware of the nonlinear control structures commonly used for dealing with practical control problems in industry. The chapter is not intended to train the reader on designing nonlinear control systems. For a comprehensive treatment of the subject, refer to Slotine and Li[126].

10.2 FEEDBACK LINEARIZATION

In this section, we describe basic concepts of feedback linearization intuitively with the help of an example. Mathematical tools from differential geometry, which are useful to generalize these concepts to a broad class of nonlinear systems, are not presented here for want of space; Slotine and Li [126], and Zak [35] are good references for this subject.

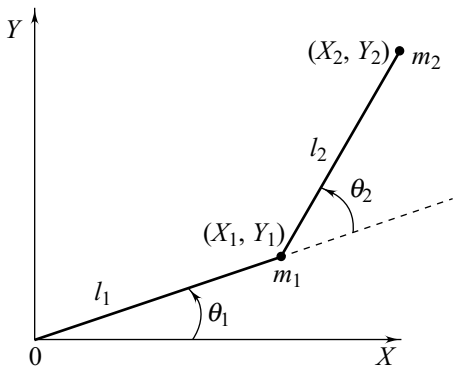


Fig. 10.1 A two-link robot

In its simplest form, feedback linearization amounts to cancelling the nonlinearities in a nonlinear system, so that the closed-loop dynamics is in a linear form. This very simple idea is demonstrated in the following example of control design for a two-link robot.

A two-link planar robot-arm manipulator, used extensively for simulations in the literature, is shown in Fig. 10.1. This arm is simple enough to simulate, *yet* has all the nonlinear effects common to general robot manipulators.

To determine the arm dynamics, we assume that the link masses m_1 and m_2 are concentrated at the ends of the links of lengths l_1 and l_2, respectively. We define the angle of the first link θ_1 with respect to an inertial frame, as depicted in Fig. 10.1. The angle of the second link θ_2 is defined with respect to the orientation of the first link. Torques τ_1 and τ_2 are applied by the actuators to control the angles θ_1 and θ_2, respectively.

Let us derive the dynamics of the two-link arm from first principles using *Lagrange's equation of motion* [38]:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{\theta}}} - \frac{\partial L}{\partial \boldsymbol{\theta}} = \boldsymbol{\tau}; \boldsymbol{\theta} = [\theta_1 \ \theta_2]^T; \boldsymbol{\tau} = [\tau_1 \ \tau_2]^T \tag{10.1}$$

with the *Lagrangian* L defined in terms of the kinetic energy K and potential energy P as

$$L = K(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) - P(\boldsymbol{\theta}) \tag{10.2}$$

For link 1, we have the positions and velocities:

$$\begin{aligned} X_1 &= l_1 \cos\theta_1; \ Y_1 = l_1 \sin\theta_1 \\ \dot{X}_1 &= -l_1 \dot{\theta}_1 \sin\theta_1; \ \dot{Y}_1 = l_1 \dot{\theta}_1 \cos\theta_1 \\ v_1^2 &= \dot{X}_1^2 + \dot{Y}_1^2 = l_1^2 \dot{\theta}_1^2 \end{aligned}$$

The kinetic and potential energies, for link 1, are

$$\begin{aligned} K_1 &= \tfrac{1}{2} m_1 v_1^2 = \tfrac{1}{2} m_1 l_1^2 \dot{\theta}_1^2 \\ P_1 &= m_1 g Y_1 = m_1 g l_1 \sin\theta_1 \end{aligned}$$

For link 2, we have the positions and velocities:

$$\begin{aligned} X_2 &= l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ Y_2 &= l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{aligned}$$

$$\dot{X}_2 = -l_1\dot{\theta}_1 \sin\theta_1 - l_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2)$$
$$\dot{Y}_2 = l_1\dot{\theta}_1 \cos\theta_1 + l_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2)$$
$$v_2^2 = \dot{X}_2^2 + \dot{Y}_2^2 = l_1^2\dot{\theta}_1^2 + l_2^2(\dot{\theta}_1 + \dot{\theta}_2)^2 + 2l_1l_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)\cos\theta_2$$

Therefore, kinetic energy for link 2 is

$$K_2 = \tfrac{1}{2}m_2v_2^2$$
$$= \tfrac{1}{2}m_2l_1^2\dot{\theta}_1^2 + \tfrac{1}{2}m_2l_2^2(\dot{\theta}_1 + \dot{\theta}_2)^2 + m_2l_1l_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)\cos\theta_2$$

The potential energy for link 2 is

$$P_2 = m_2gY_2 = m_2g\left(l_1\sin\theta_1 + l_2\sin\left(\theta_1 + \theta_2\right)\right)$$

The Legrangian for the entire arm is

$$L = K - P = K_1 + K_2 - P_1 - P_2$$
$$= \tfrac{1}{2}(m_1 + m_2)l_1^2\dot{\theta}_1^2 + \tfrac{1}{2}m_2l_2^2(\dot{\theta}_1 + \dot{\theta}_2)^2 + m_2l_1\,l_2(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)\cos\theta_2$$
$$-(m_1 + m_2)gl_1\sin\theta_1 - m_2gl_2\sin\left(\theta_1 + \theta_2\right)$$

Equation (10.1) is a vector equation comprised of two scalar equations. The individual terms needed to write down these two equations are

$$\frac{\partial L}{\partial\dot{\theta}_1} = (m_1 + m_2)l_1^2\dot{\theta}_1 + m_2l_2^2(\dot{\theta}_1 + \dot{\theta}_2) + m_2l_1\,l_2(2\dot{\theta}_1 + \dot{\theta}_2)\cos\theta_2$$
$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}_1} = (m_1 + m_2)l_1^2\ddot{\theta}_1 + m_2l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2l_1\,l_2(2\ddot{\theta}_1 + \ddot{\theta}_2)\cos\theta_2 - m_2l_1\,l_2(2\dot{\theta}_1\,\dot{\theta}_2 + \dot{\theta}_2^2)\sin\theta_2$$
$$\frac{\partial L}{\partial\theta_1} = -(m_1 + m_2)gl_1\cos\theta_1 - m_2gl_2\cos(\theta_1 + \theta_2)$$
$$\frac{\partial L}{\partial\dot{\theta}_2} = m_2l_2^2(\dot{\theta}_1 + \dot{\theta}_2) + m_2l_1\,l_2\dot{\theta}_1\cos\theta_2$$
$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}_2} = m_2l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2l_1\,l_2\ddot{\theta}_1\cos\theta_2 - m_2l_1\,l_2\dot{\theta}_1\,\dot{\theta}_2\sin\theta_2$$
$$\frac{\partial L}{\partial\theta_2} = -m_2l_1\,l_2(\dot{\theta}_1^2 + \dot{\theta}_1\,\dot{\theta}_2)\sin\theta_2 - m_2gl_2\cos\left(\theta_1 + \theta_2\right)$$

According to Lagrange's equation (10.1), the arm dynamics are given by the two coupled nonlinear differential equations

$$\tau_1 = [(m_1 + m_2)l_1^2 + m_2l_2^2 + 2m_2l_1\,l_2\cos\theta_2]\ddot{\theta}_1 + [m_2l_2^2 + m_2l_1\,l_2\cos\theta_2]\ddot{\theta}_2 - m_2l_1\,l_2(2\dot{\theta}_1\,\dot{\theta}_2 + \dot{\theta}_2^2)\sin\theta_2$$
$$+(m_1 + m_2)gl_1\cos\theta_1 + m_2gl_2\cos(\theta_1 + \theta_2)$$
$$\tau_2 = [m_2l_2^2 + m_2l_1\,l_2\cos\theta_2]\ddot{\theta}_1 + m_2l_2^2\ddot{\theta}_2 + m_2l_1\,l_2\dot{\theta}_1^2\sin\theta_2 + m_2gl_2\cos(\theta_1 + \theta_2)$$

Writing the arm dynamics in vector form, yields

$$\begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} -m_2 l_1 l_2 (2\dot{\theta}_1 \dot{\theta}_2 + \dot{\theta}_2^2)\sin\theta_2 \\ m_2 l_1 l_2 \dot{\theta}_1^2 \sin\theta_2 \end{bmatrix}$$
$$+ \begin{bmatrix} (m_1+m_2) g l_1 \cos\theta_1 + m_2 g l_2 \cos(\theta_1+\theta_2) \\ m_2 g l_2 \cos(\theta_1+\theta_2) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{10.3}$$

One may write the dynamics of the two link arms completely as

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) = \boldsymbol{\tau} \tag{10.4}$$

where the symmetric *inertia matrix*

$$\mathbf{M}(\boldsymbol{\theta}) = \begin{bmatrix} \alpha + \beta + 2\eta\cos\theta_2 & \beta + \eta\cos\theta_2 \\ \beta + \eta\cos\theta_2 & \beta \end{bmatrix},$$

and nonlinear terms

$$\mathbf{N}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta})$$

are given by $\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} -\eta(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2)\sin\theta_2 \\ \eta\dot{\theta}_1^2 \sin\theta_2 \end{bmatrix}; \mathbf{G}(\boldsymbol{\theta}) = \begin{bmatrix} \alpha e_1 \cos\theta_1 + \eta e_1 \cos(\theta_1+\theta_2) \\ \eta e_1 \cos(\theta_1+\theta_2) \end{bmatrix}$

$$\alpha = (m_1+m_2)l_1^2; \quad \beta = m_2 l_2^2; \quad \eta = m_2 l_1 l_2; \quad e_1 = g/l_1$$

This is a special form of state model called 'Brunovsky canonical form'. Many systems, like the robot arm, are naturally in the Brunovsky form. Moreover, it is often possible to transform general state models to Brunovsky form. This is accomplished by finding a suitable state-space transformation followed by an input transformation [126].

Defining the state vector as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{10.5}$$

we get the following state equations:

$$\dot{\mathbf{x}}_1 = \dot{\boldsymbol{\theta}} = \mathbf{x}_2 \tag{10.6a}$$

$$\begin{aligned} \dot{\mathbf{x}}_2 = \ddot{\boldsymbol{\theta}} &= -\mathbf{M}^{-1}(\boldsymbol{\theta})[\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta})] + \mathbf{M}^{-1}(\boldsymbol{\theta})\boldsymbol{\tau} \\ &= -\mathbf{M}^{-1}(\mathbf{x}_1)[\mathbf{V}(\mathbf{x}_1, \mathbf{x}_2) + \mathbf{G}(\mathbf{x}_1)] + \mathbf{M}^{-1}(\mathbf{x}_1)\boldsymbol{\tau} \\ &= \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\boldsymbol{\tau} \end{aligned} \tag{10.6b}$$

where $\mathbf{f}(\mathbf{x}) = -\mathbf{M}^{-1}(\mathbf{x}_1)[\mathbf{V}(\mathbf{x}_1, \mathbf{x}_2) + \mathbf{G}(\mathbf{x}_1)]; \ \mathbf{g}(\mathbf{x}) = \mathbf{M}^{-1}(\mathbf{x}_1)$

The control law

$$\boldsymbol{\tau} = \mathbf{g}^{-1}(\mathbf{x})[-\mathbf{f}(\mathbf{x}) + \mathbf{u}] \tag{10.7}$$

linearizes the system (10.6) to yield

$$\begin{aligned} \dot{\mathbf{x}}_1 &= \mathbf{x}_2 \\ \dot{\mathbf{x}}_2 &= \mathbf{u} \end{aligned} \tag{10.8}$$

or
$$\begin{bmatrix} \dot{x}_{11} \\ \dot{x}_{12} \\ \dot{x}_{21} \\ \dot{x}_{22} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{10.9}$$

which may be completely expressed as

$$\begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u} \tag{10.10}$$

A two-step design procedure follows.

Step 1 Use linear system design techniques to select a feedback control $\mathbf{u}(t)$ using the linear model (10.10).

Step 2 Compute the required arm torques from (10.7):

$$\boldsymbol{\tau} = \mathbf{g}^{-1}(\mathbf{x})[-\mathbf{f}(\mathbf{x}) + \mathbf{u}] \tag{10.11}$$

This is a nonlinear feedback control law that guarantees desired behavior. It relies on computing the torque $\boldsymbol{\tau}$ using Eqn. (10.7), that makes the nonlinear dynamics (10.6) equivalent to the linear dynamics (10.10); this is termed *feedback linearization*.

Let us consider a specific design problem: tracking the desired *motion trajectory* $\boldsymbol{\theta}_d(t)$.

Define the *tracking error* as

$$\mathbf{e}(t) = \boldsymbol{\theta}_d(t) - \boldsymbol{\theta}(t) \tag{10.12}$$

Therefore,

$$\dot{\mathbf{e}}(t) = \dot{\boldsymbol{\theta}}_d(t) - \dot{\boldsymbol{\theta}}(t); \quad \ddot{\mathbf{e}}(t) = \ddot{\boldsymbol{\theta}}_d(t) - \ddot{\boldsymbol{\theta}}(t) \tag{10.13}$$

Defining
$$\tilde{\mathbf{x}}_1 = \mathbf{e}, \text{ and } \tilde{\mathbf{x}}_2 = \dot{\mathbf{e}} \tag{10.14}$$

we can write robot-arm dynamics (10.10) in the form

$$\dot{\tilde{\mathbf{x}}}_1 = \tilde{\mathbf{x}}_2 \tag{10.15a}$$

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}_2 &= \ddot{\boldsymbol{\theta}}_d - \ddot{\boldsymbol{\theta}} \\ &= \ddot{\boldsymbol{\theta}}_d + \mathbf{M}^{-1}(\boldsymbol{\theta})[\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta})] - \mathbf{M}^{-1}(\boldsymbol{\theta})\boldsymbol{\tau} \\ &= \ddot{\boldsymbol{\theta}}_d + \mathbf{M}^{-1}(\mathbf{x}_1)[\mathbf{V}(\mathbf{x}_1, \mathbf{x}_2) + \mathbf{G}(\mathbf{x}_1)] - \mathbf{M}^{-1}(\mathbf{x}_1)\boldsymbol{\tau} \\ &= \ddot{\boldsymbol{\theta}}_d - \mathbf{f}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\boldsymbol{\tau} \end{aligned}$$

The control law

$$\boldsymbol{\tau} = \mathbf{g}^{-1}(\mathbf{x})[-\mathbf{f}(\mathbf{x}) - \mathbf{u} + \ddot{\boldsymbol{\theta}}_d] \tag{10.16}$$

(10.15b)

linearizes the system (10.15) to yield

$$\begin{bmatrix} \dot{\tilde{\mathbf{x}}}_1 \\ \dot{\tilde{\mathbf{x}}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u} \tag{10.17}$$

The following design procedure may be used.

Use linear quadratic regulator design to select a feedback control $\mathbf{u}(t)$, that stabilizes the tracking error system (10.17). The performance measure

$$J = \int_0^\infty (\tilde{\mathbf{x}}^T \mathbf{Q}\tilde{\mathbf{x}} + \mathbf{u}^T \mathbf{R}\mathbf{u})dt \tag{10.18}$$

with $$\mathbf{Q} = diag\{1, 0.1, 1, 0.1\}; \mathbf{R} = diag\{0.005, 0.005\}$$

penalizes the joint angles θ_1 and θ_2 much more strongly than the control inputs, to achieve high speed and accuracy. The feedback control law is of the form

$$\begin{aligned}\mathbf{u} &= -\mathbf{K}_1\tilde{\mathbf{x}}_1 - \mathbf{K}_2\tilde{\mathbf{x}}_2 \\ &= -\begin{bmatrix}K_{11} & K_{12}\end{bmatrix}\tilde{\mathbf{x}}_1 - \begin{bmatrix}K_{21} & K_{22}\end{bmatrix}\tilde{\mathbf{x}}_2\end{aligned} \tag{10.19}$$

The computed-torque controller then becomes (refer to (10.16))

$$\boldsymbol{\tau} = \mathbf{g}^{-1}(\mathbf{x})[-\mathbf{f}(\mathbf{x}) + \mathbf{K}_1\tilde{\mathbf{x}}_1 + \mathbf{K}_2\tilde{\mathbf{x}}_2 + \ddot{\boldsymbol{\theta}}_d] \tag{10.20}$$

The computed-torque controller is shown in Fig. 10.2, which has a multiloop structure; with a nonlinear inner feedback linearization loop and an outer tracking loop. Simulation of this controller for the two-link robot arm ($m_1 = 1$, $m_2 = 1$, $l_1 = 1$, $l_2 = 1$, $g = 9.8$; $\theta_{d1}(t) = \sin(\pi t)$, $\theta_{d2}(t) = \cos(\pi t)$) was done using MATLAB (refer to Problems A.18 and A.22 in Appendix A). Figures 10.3 show the tracking performance.

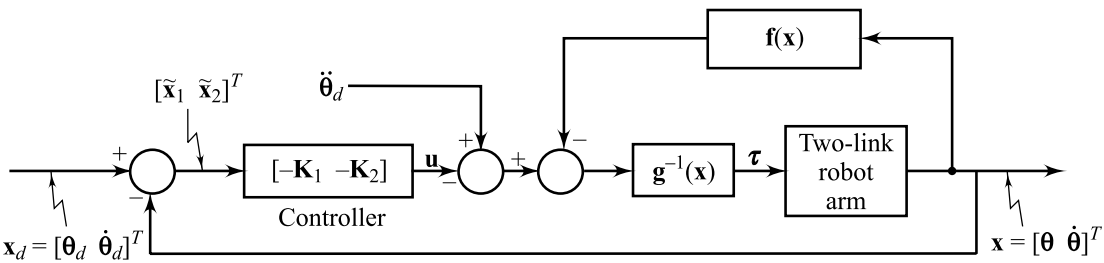


Fig. 10.2 Computed-torque controller

Unlike the linearization approach which ignores all higher-order terms of the plant, the feedback linearization approach utilizes the feedback, to render the given system a linear input-output dynamics. Then, on the basis of the linear system thus obtained, linear control techniques can be applied to address design issues. Finally, the resulting nonlinear control law is implemented to achieve the desired dynamical behavior.

The main drawback of this technique is that it relies on exact cancellation of nonlinear terms to get linear input-output behavior (This is equivalent to cancellation of the nonlinearity with its inverse). Consequently, if there are errors or uncertainty in the model of the nonlinear terms, the cancellation is no longer exact. Therefore, the applicability of such model-based approaches to feedback control of actual systems is quite limited, because they rely on the exact mathematical models of system nonlinearities.

If the functions $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}^{-1}(\mathbf{x})$ are not exactly known in the control scheme of Fig. 10.2, we may explore the option of constructing their estimates by two neural networks. We shall study this option of intelligent control in Chapter 11.

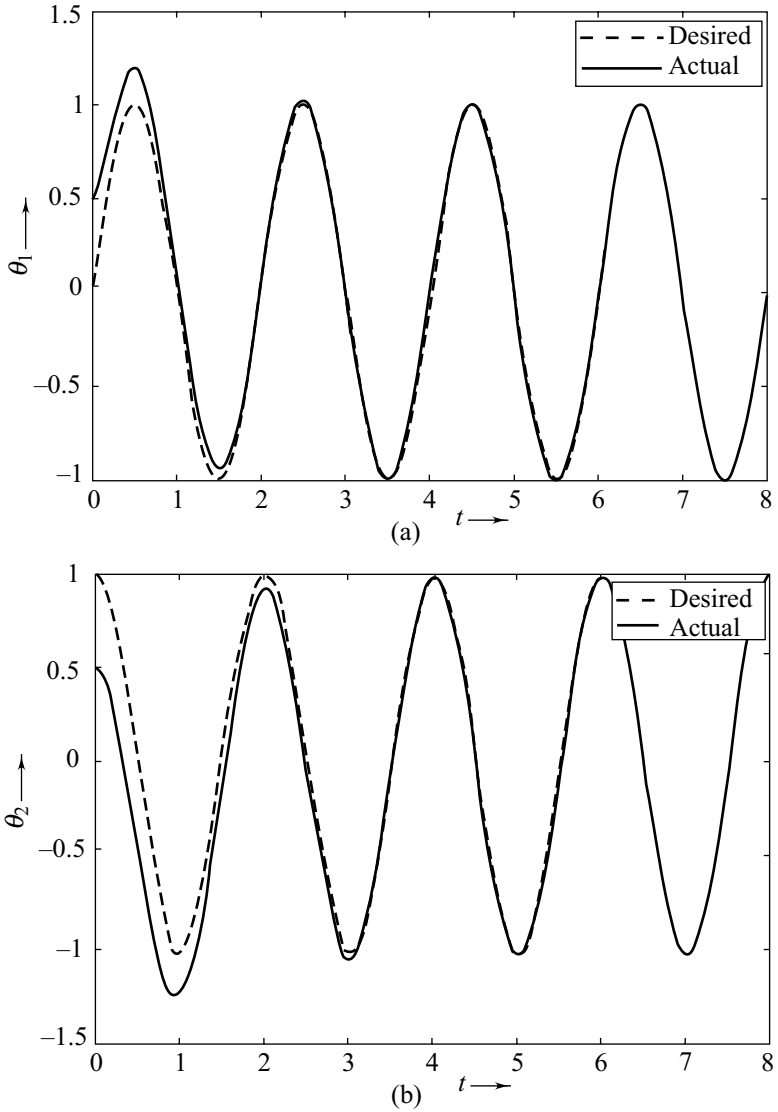


Fig. 10.3 Desired and actual trajectories

10.3 MODEL REFERENCE ADAPTIVE CONTROL

When we use a model of the plant as the basis of a control system design, we are tacitly assuming that this model is a reasonable representation of the plant. Although the design model almost always differs from the true plant in some details, we are confident that these details are not important enough to invalidate the design.

There are many applications, however, for which a design model cannot be developed with a reasonable degree of confidence. Moreover, most dynamic processes change with time. Parameters may vary because

of normal wear, aging, breakdown, and changes in the environment in which the process operates. The feedback mechanism provides some degree of immunity to discrepancies between the physical plant and the model that is used for the design of the control system. But sometimes, this is not enough. A control system designed on the basis of a nominal design model may not behave as well as expected, because the design model does not adequately represent the process in its operating environment.

How can we deal with processes that are prone to large changes, or for which adequate design models are not available? One approach is brute force, i.e., high loop-gain: as the loop-gain becomes infinite, the output of the process tracks the input with vanishing error. Brute force rarely works, however, for well-known reasons—dynamic instability, control saturation, and susceptibility to noise and other extraneous inputs.

In robust control design methods, model uncertainties are captured in a family of perturbed plant models, where each member of the family may represent the nominal plant, but which member does so, remains unknown. A robust controller satisfies the design requirements in connection with all the members of the family. Robust control design techniques are sophisticated and make it possible, for the control system design, to tolerate substantial variations in the model. But the price of achieving immunity to model uncertainties may be a sacrifice in performance. Moreover, robust control design techniques are not applicable to processes for which no (uncertainty) design model is available.

The adaptive control theory provides another approach to the design of uncertain systems. Unlike the fixed-parameter controller, adaptive controllers adjust their behavior on-line, in real-time, to the changing properties of the controlled processes. For example, in some control tasks, such as those in robot manipulations, the systems to be controlled have parameter uncertainty at the beginning of the control operation. Unless this uncertainty is gradually reduced on-line by an adaptation or estimation mechanism, it may cause inaccuracy or instability for the control systems. In many other tasks, such as those in power systems, the system dynamics may have well-known dynamics at the beginning, but experience unpredictable parameter variations as the control operation goes on. Without continuous 'redesign' of the controller, the initially appropriate controller design may not be able to control the changing plant well. Generally, the basic objective of adaptive control is to maintain consistent performance of a system in the presence of uncertainty or unknown variation in plant parameters. Since such parameter uncertainty, or variation occurs in many practical problems, adaptive control is useful in many industrial contexts. These include the following:

Robot Manipulation

Robots have to manipulate loads of various sizes, weights, and mass distributions. It is very restrictive to assume that the inertial parameters of the loads are well known before a robot picks them up and moves them away. If controllers with constant gains are used, and the load parameters are not accurately known, motion of the robot can be either inaccurate or unstable. Adaptive control, on the other hand, allows robots to move loads of unknown parameters with high speed and high accuracy.

Ship Steering

On long courses, ships are usually put under automatic steering. However, the dynamic characteristics of a ship strongly depend on many uncertain parameters, such as water depth, ship loading, and wind and wave conditions. Adaptive control can be used to achieve good control performance under varying operating conditions.

Aircraft Control

The dynamic behavior of an aircraft depends on its altitude, speed, and configuration. The ratio of variations of some parameters can lie between 10 to 50 in a given flight. Adaptive control can be used to achieve consistent aircraft performance over a large flight envelope.

Process Control

Models for metallurgical and chemical processes are usually complex and also hard to obtain. The parameters characterizing the processes vary from batch to batch. Furthermore, the working conditions are usually time-varying (e.g., reactor characteristics vary during the reactor's life, the raw materials entering the process are never exactly the same, atmospheric and climatic conditions also tend to change). In fact, process control is one of the most important application areas of adaptive control.

Adaptive control has also been applied to other areas, such as power systems.

The concept of controlling a process that is not well understood, or one in which the parameters are subject to wide variations, has a history that predates the beginning of modern control theory. The early theory was empirical, and was developed before digital computer techniques could be used for extensive performance simulations. Prototype testing was one of the few techniques available for testing adaptive control. At least one early experiment had disastrous consequences. As the more mathematically rigorous areas of control theory were developed starting in the 1960s, interest in adaptive control faded for a time, only to be reawakened in the late 1970s with the discovery of mathematically rigorous proofs of the convergence of some popular adaptive control algorithms. This interest continues unabated [130–136].

Many, apparently different, approaches to adaptive control have been proposed in the literature. Two schemes in particular have attracted much interest: 'Self-Tuning Regulator' (STR), and 'Model Reference Adaptive Control' (MRAC). These two approaches actually turn out to be special cases of a more general design philosophy.

In the following, we describe model reference adaptive control (MRAC); the discussion on self-tuning regulator is given in the next section.

Generally, a *model reference adaptive control* system can be schematically represented by Fig. 10.4. It is composed of four parts: a *plant*/*process* containing unknown parameters, a *reference model* for compactly specifying the desired output of the control system, a feedback *control law* containing adjustable parameters, and an *adaptation mechanism* for updating the adjustable parameters.

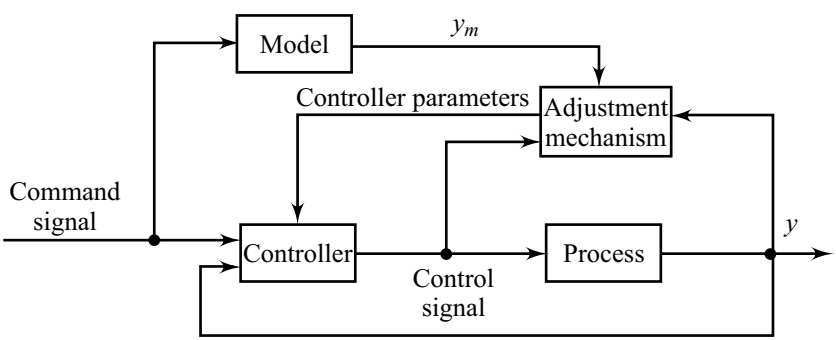


Fig. 10.4 Block diagram of a model-reference adaptive control system

The *plant* is assumed to have a known structure, although the parameters are unknown. For linear plants, this means that the number of poles and the number of zeros are assumed to be known, but that the locations of these poles and zeros are not.

A *reference model* is used to specify the ideal response of the adaptive control system to the external command. Intuitively, it provides the ideal plant response, which the adaptation mechanism should seek in adjusting the parameters. The choice of the reference model is part of the adaptive control system design. This choice should reflect the performance specifications in the control tasks, such as rise time, setting time, overshoot or frequency-domain characteristics.

The *controller* is usually parameterized by a number of adjustable parameters (implying that one may obtain a family of controllers by assigning various values to the adjustable parameters). The controller should have *perfect tracking* capacity in order to allow the possibility of tracking convergence. That is, when the plant parameters are exactly known, the corresponding controller parameters should make the plant output identical to that of the reference model. When the plant parameters are not known, the adaptation mechanism will adjust the controller parameters, so that perfect tracking is asymptotically achieved. If the control law is linear in terms of the adjustable parameters, it is said to be *linearly parameterized*. Existing adaptive control designs normally require linear parameterization of the controller—in order to obtain adaptation mechanisms with guaranteed stability and tracking convergence.

The *adaptation* mechanism is used to adjust the parameters in the control law. In MRAC systems, the adaptation law searches for parameters such that the response of the plant under adaptive control, becomes the same as that of the reference model, *i.e.*, the objective of the adaptation is to make the tracking error converge to zero. Clearly, the main difference from conventional control, lies in the existence of this mechanism. The main issue in adaptation design is to synthesize an adaptation mechanism which will guarantee that the control system remains stable and the tracking error converges to zero—even if the parameters are varied. Many formalisms in nonlinear control can be used to this end, such as Lyapunov theory, hyperstability theory, and passivity theory[126]. In this section, we shall use Lyapunov theory.

Thus, the desired performance in an MRAC (Fig. 10.4) is given in terms of a reference model, which, in turn, gives the desired response to the command signal. The inner control loop is an ordinary feedback loop composed of the plant and the controller; the parameters of the controller are adjusted by the outer loop in such a way that the error between the plant and model outputs becomes small. The key problem is to determine the adjustment mechanism so that a stable system, which brings the error to zero, is obtained.

From the block diagram of Fig. 10.4, one may jump to the false conclusion that MRAC has an answer to all control problems with uncertain plants. Before using such a scheme, important theoretical problems such as stability and convergence have to be considered. Since adaptive control schemes are both time-varying and nonlinear, stability and performance analysis becomes very difficult. Many advances have been made in proving stability under certain (sometimes limited) conditions. However, not much ground has been gained in proving performance bounds.

10.3.1 Lyapunov Stability of Non-Autonomous Systems

In Chapters 8 and 9, we studied Lyapunov analysis of autonomous systems. An adaptive control system is a non-autonomous system because the parameters involved in its dynamic equations vary with time. Although many of the ideas in Chapters 8 and 9 can be similarly applied to the non-autonomous case, the

conditions required in the treatment of non-autonomous systems are more restrictive. Scalar functions with explicit time-dependence, $V(\mathbf{x}, t)$, are required while in autonomous system analysis, time-invariant functions $V(\mathbf{x})$ suffice. Asymptotic stability analysis of non-autonomous systems is generally harder than that of autonomous systems since it is usually very difficult to find Lyapunov functions with a negative definite derivative. When $\dot{V}(\mathbf{x}, t)$ is only negative semidefinite, Lyapunov theorems on asymptotic stability are not applicable.

Lyapunov Theorem for Non-Autonomous Systems

If, in a certain neighborhood of the equilibrium point $\mathbf{0}$, there exists a scalar function $V(\mathbf{x}, t)$ with continuous partial derivatives such that

- $V(\mathbf{x}, t)$ is positive definite, and
- $\dot{V}(\mathbf{x}, t)$ is negative semidefinite,

then the equilibrium point $\mathbf{0}$, is stable in the sense of Lyapunov [126].

Similar to the case of autonomous systems, if in a certain neighborhood of the equilibrium point, $V(\mathbf{x}, t)$ is positive definite and $\dot{V}(\mathbf{x}, t)$, its derivative along the system trajectories, is negative semidefinite, then $V(\mathbf{x}, t)$ is called a Lyapunov function for the non-autonomous system.

The theorem stated above establishes stability in the sense of Lyapunov, and not asymptotic stability of the origin. An important and simple result which partially remedies this situation is *Barbalat's lemma*. Barbalat's lemma is a purely mathematical result concerning the asymptotic properties of functions and their derivatives. When properly used for non-autonomous systems, it may lead to satisfactory solution for many asymptotic stability problems.

Barbalat's Lemma

Before describing Barbalat's lemma itself, let us clarify a few points concerning the asymptotic properties of functions and their derivatives. Given a differentiable function $f(t)$, the following facts are important to keep in mind [126]:

- If f is lower bounded (for some $l > 0$, the region defined by $f(t) < l$ is bounded) and decreasing ($\dot{f} \leq 0$), then it converges to a limit. This is a standard result in calculus.
- The fact that $f(t)$ converges as $t \to \infty$, does not imply that $\dot{f}(t) \to 0$. For example, while the function, $f(t) = e^{-t} \sin(e^{2t}) \to 0$, its derivative $\dot{f}(t)$ is unbounded.

Given that a function tends towards a finite limit, what additional requirement can guarantee that its derivative actually converges to zero? Barbalat's lemma indicates that the derivative itself should have some smoothness properties; it should be *uniformly continuous*.

A function $\dot{f}(t)$ is uniformly continuous if one can always find a δ_R for a given $R > 0$, such that for any t_i and $\tau \in [0, \delta_R]$, we have

$$\left| \dot{f}(t_i + \tau) - \dot{f}(t_i) \right| < R$$

Uniform continuity of a function is often difficult to assert from the above definition. A more convenient approach is to examine the function's derivative. A very simple *sufficient condition for a differentiable function to be uniformly continuous, is that its derivative must be bounded*. Thus, if the function $f(t)$ is twice differentiable, then its derivative $\dot{f}(t)$ is uniformly continuous if its second derivative $\ddot{f}(t)$ is bounded. This can easily be seen from the finite difference theorem:

For all t_i and all $t_i + \tau$, there exists t' (between t_i and $t_i + \tau$), such that

$$\dot{f}(t_i + \tau) - \dot{f}(t_i) = \ddot{f}(t')[(t_i + \tau) - t_i]$$

If $R_1 > 0$ is an upper bound on the function $\left|\ddot{f}\right|$, then

$$\left|\dot{f}(t_i + \tau) - \dot{f}(t_i)\right| \leq R_1 \tau$$

For $\tau \in [0, \delta_R]$ and $\delta_R = R/R_1$, we have

$$\left|\dot{f}(t_i + \tau) - \dot{f}(t_i)\right| < R$$

which is the definition of uniform continuity.

We can now state the Barbalat's lemma (for proof, refer to [35, 126]):

If a differentiable function f (t) has a finite limit, and if $\dot{f}(t)$ *is uniformly continuous, then* $\dot{f}(t) \to 0$ *as* $t \to \infty$.

To apply Barbalat's lemma to the analysis of non-autonomous systems, one typically uses the following immediate corollary:

If a scalar function $V(\mathbf{x}, t)$ *satisfies the following conditions*:

- $V(\mathbf{x}, t)$ *is lower bounded,*
- $\dot{V}(\mathbf{x}, t)$ *is negative semidefinite, and*
- $\dot{V}(\mathbf{x}, t)$ *is uniformly continuous in time,*

then $\dot{V}(\mathbf{x}, t) \to 0$ as $t \to \infty$.

10.3.2 Application to a First-Order Control System

We illustrate the design methodology of MRAC through a simple example. Consider a system with the plant model of the form

$$\dot{y}_p = -a_p y_p + b_p u; y_p(0) \triangleq y_p^0 \tag{10.21}$$

where u is the control variable, and y_p is the measured state (output); a_p and b_p are unknown coefficients.

Assume that it is desired to obtain a closed-loop system described by

$$\dot{y}_m = -a_m y_m + b_m r \quad ; \quad y_m(0) \triangleq y_m^0 \tag{10.22}$$

a_m and b_m are known coefficients of the reference model.

When the parameters of the plant are known, the following control law gives perfect model following:

$$u(t) = br(t) - ay_p(t) \tag{10.23}$$

with the parameters

$$b = \frac{b_m}{b_p}, a = \frac{a_m - a_p}{b_p} \tag{10.24}$$

An MRAC which can find the appropriate gains $a(t)$ and $b(t)$ when parameters a_p and b_p are not known, may be obtained as follows (Fig. 10.5).

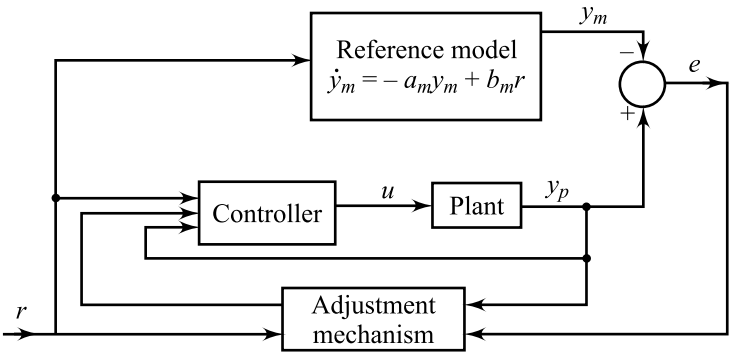


Fig. 10.5 A model reference adaptive system

Introduce the error variable

$$e(t) = y_p(t) - y_m(t) \tag{10.25a}$$

The rate of change of the error is given by

$$\begin{aligned}\frac{de(t)}{dt} &= [-a_p y_p(t) + b_p u(t)] - [-a_m y_m(t) + b_m r(t)] \\ &= -a_m e(t) + [a_m - a_p - b_p a(t)] y_p(t) + [b_p b(t) - b_m] r(t)\end{aligned} \tag{10.25b}$$

Notice that the error goes to zero if the parameters $a(t)$ and $b(t)$ are equal to the ones given by (10.24). We will now attempt to construct a parameter adjustment mechanism that will drive the parameters $a(t)$ and $b(t)$ to appropriate values, such that the resulting control law (10.23) forces the plant output $y_p(t)$ to follow the model output $y_m(t)$. For this purpose, we introduce the Lyapunov function,

$$V(e, a, b) = \tfrac{1}{2}\left[e^2(t) + \frac{1}{b_p\gamma}(b_p a(t) + a_p - a_m)^2 + \frac{1}{b_p\gamma}(b_p b(t) - b_m)^2\right]$$

where $\gamma > 0$.

This function is zero when $e(t)$ is zero and the controller parameters $a(t)$ and $b(t)$ are equal to the optimal values given by (10.24). The derivative of V is

$$\begin{aligned}\frac{dV}{dt} &= e(t)\frac{de(t)}{dt} + \frac{1}{\gamma}[b_p a(t) + a_p - a_m]\frac{da(t)}{dt} + \frac{1}{\gamma}[b_p b(t) - b_m]\frac{db(t)}{dt} \\ &= -a_m e^2(t) + \frac{1}{\gamma}[b_p a(t) + a_p - a_m]\left[\frac{da(t)}{dt} - \gamma y_p(t) e(t)\right] + \frac{1}{\gamma}[b_p b(t) - b_m]\left[\frac{db(t)}{dt} + \gamma r(t) e(t)\right]\end{aligned}$$

If the parameters are updated as

$$\begin{aligned}\frac{db(t)}{dt} &= -\gamma r(t) e(t) \\ \frac{da(t)}{dt} &= \gamma y_p(t) e(t),\end{aligned} \tag{10.26}$$

we get

$$\frac{dV}{dt} = -a_m e^2(t)$$

Thus, the adaptive control system is stable in the sense of Lyapunov, i.e., the signals e, a and b are bounded. To ensure that tracking error goes to zero, we compute second time derivative of Lyapunov function:

$$\frac{d^2V}{dt^2} = -2a_m e(t)\frac{de(t)}{dt}$$

From (10.25b), it follows that

$$\frac{d^2V}{dt^2} = -2a_m e(t)[-a_m e(t) + (a_m - a_p - b_p a(t))y_p(t) + (b_p b(t) - b_m)r(t)]$$
$$= f(e, a, b, y_p, r)$$

Since all the parameters are bounded, and $y_p(t) = e(t) + y_m(t)$ is bounded, $\ddot{V}$ is also bounded, which, in turn, implies that $\dot{V}$ is uniformly continuous. Therefore, the asymptotic convergence of the tracking error $e(t)$ is guaranteed by Barbalat's lemma.

Figure 10.6 shows the simulation (refer to Problem A.19 in Appendix A) of MRAC system with $a_p = 1$, $b_p = 0.5$, $a_m = 2$ and $b_m = 2$. The input signal is a square wave with amplitude 1. The adaptation gain $\gamma = 2$. The closed-loop system is close to the desired behavior, after only a few transients. The convergence rate depends critically, on the choice of the parameter γ.

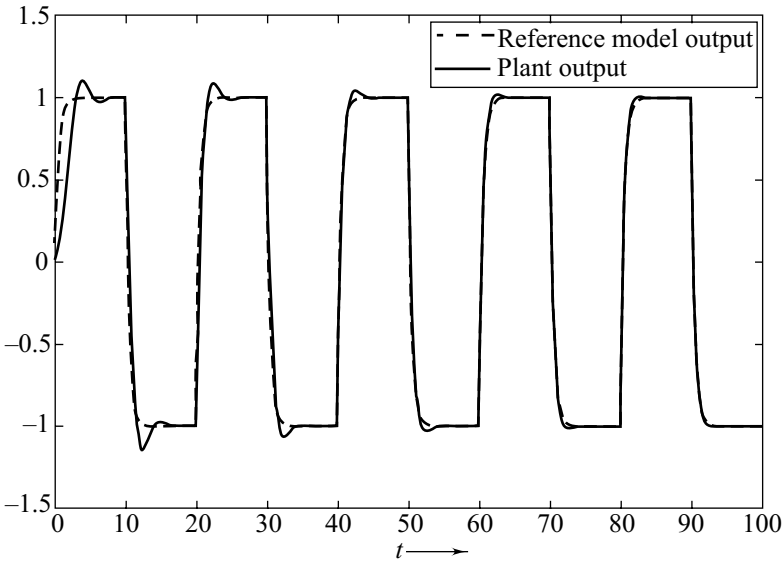


Fig. 10.6 Simulation of the MRAC system of Fig. 10.5

Plots in Fig. 10.6 were generated by simulating the following sets of equations:

(i) $\dot{y}_m(t) = -2y_m(t) + 2r(t); y_m(0) = 0$

This gives $y_m(t)$.

(ii) $\dot{y}_p(t) = -y_p(t) + 0.5u(t); y_p(0) = 0$

$u(t) = b(t)r(t) - a(t)y_p(t)$

$$\frac{db(t)}{dt} = -2r(t)e(t); b(0) = 0.5$$
$$\frac{da(t)}{dt} = 2y_p(t)e(t); a(0) = 1$$
$$e(t) = y_p(t) - y_m(t)$$

From this set of equations, we obtain $u(t)$ and $y_p(t)$.

10.4 SYSTEM IDENTIFICATION AND GENERALIZED PREDICTIVE CONTROL IN SELF-TUNING MODE

10.4.1 System Identification

The types of models that are needed for the design methods presented in this book, can be grouped into two categories: transfer function model and state variable model. If we have a transfer function description, we can obtain an equivalent state variable description and vice versa. These equivalent models are described by certain *parameters*—the elements of **F**, **g**, **c** matrices of the state model

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}u(k)$$
$$y(k) = \mathbf{c}\mathbf{x}(k)$$
$$\mathbf{x}(k) : n \times n \text{ state vector}$$
$$u(k) : \text{scalar input}$$
$$y(k) : \text{scalar output}$$

or the parameters α_i and β_j of the transfer function

$$G(z) = \frac{Y(z)}{U(z)} = \frac{\beta_1 z^m + \beta_2 z^{m-1} + \cdots + \beta_{m+1}}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n}$$

The category of such models gives us the *parameteric description* of the plant. The other category of models such as frequency-response curves (Bode plots, polar plots, *etc.*), time-response curves, etc., gives *nonparametric description* of the plant.

Plant models can be obtained from the first principles of physics. In many cases, however, it is not possible to make a complete model only from physical knowledge. In these circumstances, the designer may turn to the other source of information about plant dynamics—the data taken from experiments directly conducted to excite the plant, and to measure the response. The process of constructing models from experimental data is called *system identification*. In this section, we restrict our attention to identification of discrete parametric models, which includes the following steps:

(i) Experimental planning
(ii) Selection of model structure
(iii) Parameter estimation

Experimental Planning

The choice of experimental conditions for parameter estimation is of considerable importance. It is clear that the best experimental conditions are those that account for the final application of the model. This may occur naturally in some cases; e.g., in adaptive control (discussed later in this section) the model is

adjusted under normal operating conditions. Many 'classic' methods depend strongly on having specific input, e.g., sinusoid or impulse. There could be advantages in contriving such an artificial experiment if it subjects the system to a rich and informative set of conditions, in the shortest possible time. A requirement on the input signal is that it should sufficiently excite all the modes of the process.

One broad distinction in identification methods is between *on-line* and *off-line* experimentation. The on-line methods give estimates recursively, as the measurements are obtained, and are the only alternative if the identification is going to be used in an adaptive controller.

Selection of Model Structure

The model structures are derived from prior knowledge of the plant. In some cases the only *a priori* knowledge is that the plant can be described as a linear system in a particular range. It is, then, natural to use general representations of linear systems.

Consider a SISO dynamic system with input $\{u(t)\}$ and output $\{y(t)\}$. The sampled values of these signals can be related through the linear difference equation

$$y(k+n)+\alpha_1 y(k+n-1)+\cdots+\alpha_n y(k)=\beta_1 u(k+m)+\beta_2 u(k+m-1)+\cdots+\beta_{m+1}u(k);\boldsymbol{n}\geq m \qquad (10.27)$$

α_i and β_j are constant (unknown) parameters.

The number of parameters to be identified, depends on the order of the selected model, i.e., n in Eqn. (10.27). The calculations can be arranged so that it is possible to make a recursion in the number of parameters in the model. The methods of model-order selection are usually developed for the off-line solution.

The unknown process is not completely a black box. Some information about its dynamic behavior is known from basic principles and/or plant experience. Therefore, some estimate of the model's order, and some initial values for the unknown parameters, will be available. The more we know about the process, the more effective the postulated model will be. Consequently, we should use all available information for its development. The order of the postulated model is a very important factor. Remember, that complex models of high order will not necessarily produce better controller designs and may burden the computational effort, without tangible results.

Equation (10.27) may be expressed as

$$y(k)+\alpha_1 y(k-1)+\cdots+\alpha_n y(k-n)=\beta_1 u(k+m-n)+\beta_2 u(k+m-1-n)+\cdots+\beta_{m+1}u(k-n)$$

or

$$y(k)+\alpha_1 y(k-1)+\cdots+\alpha_n y(k-n)=\beta_1 u(k-d)+\beta_2 u(k-1-d)+\cdots+\beta_{n-d+1}u(k-n) \qquad (10.28)$$

$d = n - m \geq 0$ is the relative degree or control delay.

We shall use operator notation for conveniently writing linear difference equations. Let z^{-1} be the backward shift (or delay) operator:

$$z^{-1}y(k)=y(k-1) \qquad (10.29)$$

Then Eqn. (10.28) can be written as

$$A(z^{-1})y(k)=B(z^{-1})u(k)$$

where $A(z^{-1})$ and $B(z^{-1})$ are polynomials in the delay operator:

$$A(z^{-1})=1+\alpha_1 z^{-1}+\cdots+\alpha_n z^{-n}$$

$$B(z^{-1}) = z^{-d}(\beta_1 + \beta_2 z^{-1} + \cdots + \beta_{n-d+1} z^{-(n-d)})$$

We shall present the parameter-estimation algorithms for the case of $d = 1$ without any loss of generality; the results for any value of d easily follow.

For $d = 1$, we get the input-output model structure

$$A(z^{-1})y(k) = B(z^{-1})u(k) \tag{10.30}$$

where

$$A(z^{-1}) = 1 + \alpha_1 z^{-1} + \cdots + \alpha_n z^{-n}$$
$$B(z^{-1}) = \beta_1 z^{-1} + \beta_2 z^{-2} + \cdots + \beta_n z^{-n}$$

In the presence of the disturbance, model (10.30) takes the form

$$A(z^{-1})y(k) = B(z^{-1})u(k) + \varepsilon(k) \tag{10.31}$$

where $\varepsilon(k)$ is some disturbance of unspecified character.

The model (10.31) describes the dynamic relationship between the input and output signals, expressed in terms of the parameter vector

$$\boldsymbol{\theta} = [\alpha_1 \ldots \alpha_n \ \ \beta_1 \ldots \beta_n]^T \tag{10.32}$$

Introducing the vector of lagged input-output data,

$$\boldsymbol{\phi}(k) = [-y(k-1)\cdots - y(k-n) \ \ u(k-1)\cdots u(k-n)]; \tag{10.33}$$

Eqn. (10.31) can be rewritten as

$$y(k) = \boldsymbol{\phi}(k)\boldsymbol{\theta} + \varepsilon(k) \tag{10.34}$$

A model structure should be selected (i) that has a minimal set of parameters and is yet equivalent to the assumed plant description; (ii) whose parameters are uniquely determined by the observed data; and (iii) which will make subsequent control design simple.

Parameter Estimation

The dynamic relationship between the input and output of a scalar system is given by the model (10.34). Ignoring random effects $\varepsilon(k)$ on data collection, we have

$$y(k) = \boldsymbol{\phi}(k)\boldsymbol{\theta} \tag{10.35}$$

where $\boldsymbol{\phi}(k)$ is given by Eqn. (10.33) and $\boldsymbol{\theta}$ is given by Eqn. (10.32).

Using the observations

$$\{u(0), u(1), ..., u(N), y(0), y(1), ..., y(N)\}$$

we wish to compute the values of α_i and β_j in parameter vector $\boldsymbol{\theta}$, which will fit the observed data.

Thus, solving the parameter-estimation problem requires techniques for selecting a parameter estimate, which best represents the given data. For this, we require some idea of the goodness of the fit of a proposed value of $\boldsymbol{\theta}$ to the true $\boldsymbol{\theta}^\circ$. Because, by the very nature of the problem, $\boldsymbol{\theta}^\circ$ is unknown, it is unrealistic to define a direct parameter error between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^\circ$. We must define the error in a way that can be computed from $\{u(k)\}$ and $\{y(k)\}$.

Let $e(k,\boldsymbol{\theta})$ be the *equation error* comprising the extent to which the equations of motion (10.35) fail to be true for a specific value of $\boldsymbol{\theta}$ when used with the specific actual data:

$$e(k,\boldsymbol{\theta}) = y(k) - \boldsymbol{\phi}(k)\boldsymbol{\theta} \tag{10.36}$$

A simple criterion representing the goodness of fit, of a proposed value of $\boldsymbol{\theta}$, is given by

$$J(\boldsymbol{\theta}) = \sum_{k=1}^{N} e^2(k,\boldsymbol{\theta}) \tag{10.37}$$

The method called the *Least Squares Method*, based on minimizing the sum of the squares of the error, is a very simple and effective method of parameter estimation.

Since $y(k)$ depends on past data up to n earlier periods, the first error we can form is $e(n,\boldsymbol{\theta})$; the subsequent errors being $e(n+1, \boldsymbol{\theta})$, ..., $e(N, \boldsymbol{\theta})$:

$$\begin{aligned} e(n,\boldsymbol{\theta}) &= y(n) - \boldsymbol{\phi}(n)\boldsymbol{\theta} \\ e(n+1,\boldsymbol{\theta}) &= y(n+1) - \boldsymbol{\phi}(n+1)\boldsymbol{\theta} \\ &\vdots \\ e(N,\boldsymbol{\theta}) &= y(N) - \boldsymbol{\phi}(N)\boldsymbol{\theta} \end{aligned}$$

In vector-matrix notation,

$$\mathbf{e}(N,\boldsymbol{\theta}) = \mathbf{y}(N) - \boldsymbol{\Phi}(N)\boldsymbol{\theta} \tag{10.38}$$

where

$$\begin{aligned} \mathbf{e}(N,\boldsymbol{\theta}) &= [e(n,\boldsymbol{\theta}) \quad e(n+1,\boldsymbol{\theta}) \cdots e(N,\boldsymbol{\theta})]^T \\ \boldsymbol{\Phi}(N) &= [\boldsymbol{\phi}^T(n) \quad \boldsymbol{\phi}^T(n+1) \cdots \boldsymbol{\phi}^T(N)]^T \\ \mathbf{y}(N) &= [y(n) \quad y(n+1) \cdots y(N)]^T \end{aligned}$$

Note that $\mathbf{e}$ is $(N - n + 1) \times 1$ vector, $\mathbf{y}$ is $(N - n + 1) \times 1$ vector, $\boldsymbol{\Phi}$ is $(N - n + 1) \times 2n$ matrix and $\boldsymbol{\theta}$ is $2n \times 1$ vector.

The principle of least squares says that the parameters should be selected in such a way that the performance measure

$$J(\boldsymbol{\theta}) = \sum_{k=n}^{N} e^2(k,\boldsymbol{\theta}) = \mathbf{e}^T(N,\boldsymbol{\theta})\mathbf{e}(N,\boldsymbol{\theta}) \tag{10.39}$$

is minimized

The performance measure $J(\boldsymbol{\theta})$ can be written as

$$\begin{aligned} J(\boldsymbol{\theta}) &= [\mathbf{y}(N) - \boldsymbol{\Phi}(N)\boldsymbol{\theta}]^T [\mathbf{y}(N) - \boldsymbol{\Phi}(N)\boldsymbol{\theta}] \\ &= \mathbf{y}^T(N)\mathbf{y}(N) - \boldsymbol{\theta}^T\boldsymbol{\Phi}^T(N)\mathbf{y}(N) - \mathbf{y}^T(N)\boldsymbol{\Phi}(N)\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N)\boldsymbol{\theta} \\ &= \mathbf{y}^T(N)\mathbf{y}(N) - \boldsymbol{\theta}^T\boldsymbol{\Phi}^T(N)\mathbf{y}(N) - \mathbf{y}^T(N)\boldsymbol{\Phi}(N)\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N)\boldsymbol{\theta} \\ &\quad + \mathbf{y}^T(N)\boldsymbol{\Phi}(N)(\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N))^{-1}\boldsymbol{\Phi}^T(N)\mathbf{y}(N) - \mathbf{y}^T(N)\boldsymbol{\Phi}(N)(\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N))^{-1}\boldsymbol{\Phi}^T(N)\mathbf{y}(N) \end{aligned}$$

(Note that we have simply added and subtracted the same terms under the assumption that $[\boldsymbol{\Phi}^T(N)\,\boldsymbol{\Phi}(N)]$ is invertible).

Hence

$$\begin{aligned} J(\boldsymbol{\theta}) &= \mathbf{y}^T(N)[1 - (\boldsymbol{\Phi}(N)(\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N))^{-1}\boldsymbol{\Phi}^T(N)]\mathbf{y}(N) \\ &\quad + (\boldsymbol{\theta} - (\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N))^{-1}\boldsymbol{\Phi}^T(N)\mathbf{y}(N))^T\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N) \times (\boldsymbol{\theta} - (\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N))^{-1}\boldsymbol{\Phi}^T(N)\mathbf{y}(N) \end{aligned}$$

The first term in this equation is independent of $\boldsymbol{\theta}$, so we cannot reduce J *via* this term. Hence, to get the smallest value of J, we choose $\boldsymbol{\theta}$ so that the second term is zero. Denoting the value of $\boldsymbol{\theta}$ that achieves the minimization of J by $\hat{\boldsymbol{\theta}}$, we notice that

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N)]^{-1}\boldsymbol{\Phi}^T(N)\mathbf{y}(N) \tag{10.40a}$$

$$= \mathbf{P}(N)\boldsymbol{\Phi}^T(N)\mathbf{y}(N) \tag{10.40b}$$

where $$\mathbf{P}(N) = [\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N)]^{-1}$$

The least squares calculation for $\hat{\boldsymbol{\theta}}$ given by (10.40) is a 'batch' calculation since one has a batch of data from which the matrix $\boldsymbol{\Phi}$, and vector $\mathbf{y}$, are composed according to (10.38). In many cases, the observations are obtained sequentially. If the least squares problem has been solved for N observations, it seems to be a waste of computational resources to start from scratch when a new observation is obtained. Hence, it is desirable to arrange the computations in such a way that the results obtained for N observations can be used in order to get the estimates for $(N + 1)$ observations. The algorithm for calculating the least-squares estimate recursively is discussed below.

Let $\hat{\boldsymbol{\theta}}(N)$ denote the least-squares estimate based on N measurements. Then from (10.40)

$$\hat{\boldsymbol{\theta}}(N) = [\boldsymbol{\Phi}^T(N)\,\boldsymbol{\Phi}(N)]^{-1}\boldsymbol{\Phi}^T(N)\mathbf{y}(N)$$

It is assumed that the matrix $[\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N)]$ is nonsingular for all N. When an additional measurement is obtained, a row is added to the matrix $\boldsymbol{\Phi}$ and an element is added to the vector $\mathbf{y}$. Hence,

$$\boldsymbol{\Phi}(N+1) = \begin{bmatrix} \boldsymbol{\Phi}(N) \\ \boldsymbol{\phi}(N+1) \end{bmatrix}; \mathbf{y}(N+1) = \begin{bmatrix} \mathbf{y}(N) \\ y(N+1) \end{bmatrix}$$

The estimate $\hat{\boldsymbol{\theta}}(N+1)$ based on $N + 1$ measurements, can then be written as

$$\begin{aligned} \hat{\boldsymbol{\theta}}(N+1) &= [\boldsymbol{\Phi}^T(N+1)\boldsymbol{\Phi}(N+1)]^{-1}\boldsymbol{\Phi}^T(N+1)\mathbf{y}(N+1) \\ &= [\boldsymbol{\Phi}^T(N)\boldsymbol{\Phi}(N) + \boldsymbol{\phi}^T(N+1)\boldsymbol{\phi}(N+1)]^{-1}[\boldsymbol{\Phi}^T(N)\mathbf{y}(N) + \boldsymbol{\phi}^T(N+1)y(N+1)] \qquad (10.41) \\ \mathbf{P}(N+1) &= [\boldsymbol{\Phi}^T(N+1)\,\boldsymbol{\Phi}(N+1)]^{-1} \end{aligned}$$

Then from (10.41), we obtain

$$\mathbf{P}(N+1) = [\mathbf{P}^{-1}(N) + \boldsymbol{\phi}^T(N+1)\boldsymbol{\phi}(N+1)]^{-1} \tag{10.42}$$

We now need the inverse of a sum of two matrices. We will use the well-known *matrix inversion lemma*[1] for this purpose.

---

[1] Matrix inversion lemma is proved below.

$$[\mathbf{A} + \mathbf{BCD}]\left\{\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}\right]^{-1}\mathbf{DA}^{-1}\right\}$$

$$= \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{B}\left[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}\right]^{-1}\mathbf{DA}^{-1} - \mathbf{BCDA}^{-1}\mathbf{B}\left[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}\right]^{-1}\mathbf{DA}^{-1}$$

$$= \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{BC}\left[\mathbf{C}^{-1} - \mathbf{DA}^{-1}\mathbf{B}\right]\left[\mathbf{C}^{-1} - \mathbf{DA}^{-1}\mathbf{B}\right]^{-1}\mathbf{DA}^{-1}$$

$$= \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{BCDA}^{-1}$$

$$= \mathbf{I}$$

Let $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B}$ be nonsingular square matrices; then

$$[\mathbf{A} + \mathbf{B}\mathbf{C}\mathbf{D}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B}]^{-1}\mathbf{D}\mathbf{A}^{-1} \tag{10.43}$$

To apply (10.43) to (10.42), we make the associations

$$\begin{aligned}\mathbf{A} &= \mathbf{P}^{-1}(N)\\ \mathbf{B} &= \boldsymbol{\phi}^T(N+1)\\ \mathbf{C} &= 1\\ \mathbf{D} &= \boldsymbol{\phi}(N+1)\end{aligned}$$

Now, the following result can easily be established.

$$\mathbf{P}(N+1) = \mathbf{P}(N) - \mathbf{P}(N)\boldsymbol{\phi}^T(N+1)[1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N) \times \boldsymbol{\phi}^T(N+1)]^{-1}\boldsymbol{\phi}(N+1)\mathbf{P}(N)$$

Substituting the expression for $\mathbf{P}(N+1)$ into (10.41), we obtain

$$\begin{aligned}\hat{\boldsymbol{\theta}}(N+1) &= \{\mathbf{P}(N) - \mathbf{P}(N)\boldsymbol{\phi}^T(N+1)[1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]^{-1} \times \\ &\quad \boldsymbol{\phi}(N+1)\mathbf{P}(N)\}[\boldsymbol{\Phi}^T(N)\mathbf{y}(N) + \boldsymbol{\phi}^T(N+1)y(N+1)] \\ &= \mathbf{P}(N)\boldsymbol{\Phi}^T(N)\mathbf{y}(N) + \mathbf{P}(N)\boldsymbol{\phi}^T(N+1)y(N+1) - \mathbf{P}(N)\boldsymbol{\phi}^T(N+1) \times \\ &\quad [1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]^{-1}\boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\Phi}^T(N)\mathbf{y}(N) - \mathbf{P}(N) \times \\ &\quad \boldsymbol{\phi}^T(N+1)[1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]^{-1}\boldsymbol{\phi}(N+1)\mathbf{P}(N) \times \boldsymbol{\phi}^T(N+1)y(N+1) \\ &= \hat{\boldsymbol{\theta}}(N) + \mathbf{P}(N)\boldsymbol{\phi}^T(N+1)[1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]^{-1} \times \\ &\quad [1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]y(N+1) - \mathbf{P}(N)\boldsymbol{\phi}^T(N+1) \times \\ &\quad [1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]^{-1}\boldsymbol{\phi}(N+1)\hat{\boldsymbol{\theta}}(N) - \mathbf{P}(N) \times \\ &\quad \boldsymbol{\phi}^T(N+1)[1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]^{-1}\boldsymbol{\phi}(N+1)\mathbf{P}(N)\ \boldsymbol{\phi}^T(N+1)y(N+1)\end{aligned}$$

This gives

$$\hat{\boldsymbol{\theta}}(N+1) = \hat{\boldsymbol{\theta}}(N) + \mathbf{K}(N)[y(N+1) - \boldsymbol{\phi}(N+1)\hat{\boldsymbol{\theta}}(N)] \tag{10.44a}$$

$$\mathbf{K}(N) = \mathbf{P}(N)\boldsymbol{\phi}^T(N+1)[1 + \boldsymbol{\phi}(N+1)\mathbf{P}(N)\boldsymbol{\phi}^T(N+1)]^{-1} \tag{10.44b}$$

$$\mathbf{P}(N+1) = [1 - \mathbf{K}(N)\boldsymbol{\phi}(N+1)]\mathbf{P}(N) \tag{10.44c}$$

Equations (10.44) give the recursive least squares algorithm. Notice that for a single-output system, no matrix inversion is required.

The Eqns (10.44) have a strong intuitive appeal. The estimate $\hat{\boldsymbol{\theta}}(N+1)$ is obtained by adding a correction to the previous estimate $\hat{\boldsymbol{\theta}}(N)$. The correction is proportional to $y(N+1) - \boldsymbol{\phi}(N+1)\hat{\boldsymbol{\theta}}(N)$ where the term $\boldsymbol{\phi}(N+1)\hat{\boldsymbol{\theta}}(N)$ is the expected output at the time $N+1$, based on the previous data $\boldsymbol{\phi}(N+1)$ and the previous estimate $\hat{\boldsymbol{\theta}}(N)$. Thus, the next estimate of $\boldsymbol{\theta}$ is given by the old estimate corrected by a term, linear in error, between the observed output $y(N+1)$ and the predicted output $\boldsymbol{\phi}(N+1)\hat{\boldsymbol{\theta}}(N)$. The components of the vector $\mathbf{K}(N)$ are weighting factors that show how the correction and the previous estimate should be combined.

Replacing N by recursive parameter k in Eqns (10.44), we rewrite the recursive least squares (RLS) algorithm as

$$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) + \mathbf{K}(k)[y(k+1) - \boldsymbol{\phi}(k+1)\hat{\boldsymbol{\theta}}(k)] \tag{10.45a}$$

$$\mathbf{K}(k) = \mathbf{P}(k)\boldsymbol{\phi}^T(k+1)[1 + \boldsymbol{\phi}(k+1)\mathbf{P}(k)\boldsymbol{\phi}^T(k+1)]^{-1} \tag{10.45b}$$

$$\mathbf{P}(k+1) = [1 - \mathbf{K}(k)\boldsymbol{\phi}(k+1)]\mathbf{P}(k) \tag{10.45c}$$

Any recursive algorithm requires some initial value to be started up. In (10.44), we require $\hat{\boldsymbol{\theta}}(N)$ and $\mathbf{P}(N)$ (equivalently, in (10.45) we require $\hat{\boldsymbol{\theta}}(k)$ and $\mathbf{P}(k)$). We may collect a batch of $N > 2n$ data values, and solve the batch formula once for $\mathbf{P}(N)$ and $\hat{\boldsymbol{\theta}}(N)$.

However, it is more common to start the recursion at $k = 0$ with $\mathbf{P}(0) = \alpha\mathbf{I}$ and $\hat{\boldsymbol{\theta}}(0) = \mathbf{0}$, where α is some large constant. You may pick $\mathbf{P}(0) = \alpha\mathbf{I}$, but choose $\hat{\boldsymbol{\theta}}(0)$ to be the best guess that you have, at what the parameter values are.

We have presented the least squares method ignoring random effects on data collection, i.e., $\varepsilon(k)$ in Eqn. (10.34) has been neglected. If $\varepsilon(k)$ is white noise, the least squares estimate given by (10.45) converges to the desired value. However, if $\varepsilon(k)$ is colored noise, the least squares estimation usually gives a *biased* (wrong mean value) estimate. This can be overcome by using various extensions of the least squares estimation.

We have seen that parameter estimation can be done either on-line or off-line. Off-line estimation may be preferable if the parameters are constant, and there is sufficient time for estimation before control. However, for parameters which vary (even though slowly) during operation, on-line parameter estimation is necessary to keep track of the parameter values. Since problems in the adaptive control context usually involve slowly time-varying parameters, on-line estimation methods are, thus, more relevant.

The main purpose of the on-line estimators is to provide parameter estimates for self-tuning control.

10.4.2 Self-Tuning Regulator

If the plant is imperfectly known, perhaps because of random time-varying parameters or because of the effects of environmental changes on the plant's dynamic characteristics, then the *initial plant model* and the resulting control design will not be sufficient to obtain an acceptable performance for all time. It then becomes necessary to carry out plant-identification and control-design procedures continuously, or at intervals of time, depending on how fast the plant parameters change. This 'self-design' property of the system, to compensate for unpredictable changes in the plant, is the aspect of performance that is usually considered in defining an adaptive control system.

The identification of the dynamic characteristics of the plant should be accomplished without affecting the normal operation of the system. To identify the plant model, we must impose a control signal on the plant and analyze the system response. Identification may be made from normal operating data of the plant or by use of test (additional) signals, such as sinusoidal ones of small amplitude. The plant should be in normal operation during the test; the test signals imposed should not unduly disturb normal outputs. Furthermore, inputs and system noise should not confuse the test. Normal inputs are ideal as test signals since no difficulties with undesired outputs, or confusing inputs, will arise. However, identification with normal inputs is only possible when they have adequate signal characteristics (bandwidth, amplitude, and so on) for proper identification.

Once the plant has been identified, a decision must be made about how the adjustable parameters (controller characteristics) should be varied to maintain acceptable performance. The control signals are then modified according to the plant identification and control decision. The three functions:

(i) plant identification,
(ii) control design based on the identification results, and
(iii) actuation based on the control design

can easily by implemented using a digital computer. Figure 10.7a shows a block diagram representation of the adaptive control scheme. The system obtained is called a *Self-Tuning Regulator* (STR) because it has facilities for tuning its own parameters. The regulator can be thought of as being composed of the following two loops:

(i) The inner loop is the conventional feedback control loop consisting of the plant and the regulator.
(ii) The parameters of the regulator are adjusted on-line by the outer loop, which is composed of the recursive-parameter estimator and design calculations.

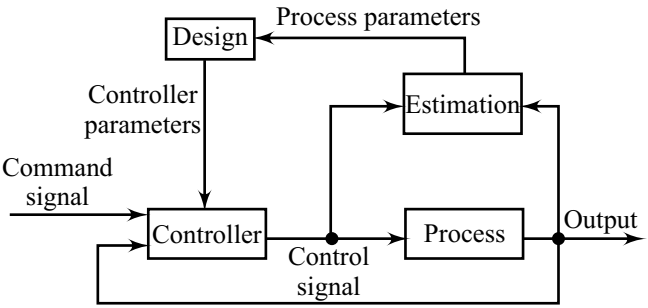


Fig. 10.7 (a) Block diagram of a self-tuning regulator

A self-tuning regulator, therefore, consists of a recursive parameter estimator (plant identifier) coupled with a control-design procedure, such that the currently estimated parameter values are used to provide feedback-controller coefficients. At each sampling, an updated parameter estimate is generated and a controller is designed, assuming that the current parameter estimate is actually the true value. The approach of using the estimates as if they were the true parameters for the purpose of design, is called *certainty equivalence adaptive control*.

From the block diagram of Fig. 10.7a, one may jump to the false conclusion that such regulators can be switched on and used blindly without any *a priori* considerations; the only requirement being a recursive parameter estimation scheme and a design procedure. We have, no doubt, an array of parameter-estimation schemes and an array of controller-design methods for plants with known parameters. However, all the possible combinations may not have a self-tuning property, which requires that the performance of the regulator coincides with the performance that would be obtained, if the system parameters were known exactly. Before using a combination, important theoretical problems, such as stability and convergence, have to be tackled. There are cases wherein self-tuning regulators have been used profitably, though some of their properties have not been fully understood theoretically; on the other hand, bad choices have been disastrous in some other cases.

So far, only a small number of available combinations have been explored from the stability, convergence, and performance points of view. It is a striking fact, uncovered by Astrom and Wittenmark [132], that in some circumstances a combination of simple least-squares estimation and minimum-variance control has a self-tuning property. The same is true for some classes of pole-shifting regulators. Computer-based controllers incorporating these concepts are now commercially available.

10.4.3 Generalized Predictive Control

Clarke, Mothadi and Tuffs [120,121] proposed an alternative—Generalized Predictive Control (GPC), to pole placement and minimum variance designs used in self-tuning regulators. The argument for

introducing GPC in a self-tuning context was that it was based on a more flexible criterion than minimum variance controllers without requiring an excessive amount of computations. Although it originated in an adaptive control context, GPC has many attractive features which definitely makes it worthwhile considering even for non-adaptive control structures. We first consider the GPC approach for a non-adaptive control structure.

The generalized predictive control (GPC) differs in at least three ways from the control design methods considered so far in this book.

(i) In linear quadratic control (Chapter 8), the cost function is defined over the time interval $[0, \infty)$:

$$J = \frac{1}{2}\sum_{k=0}^{\infty}[e^2(k) + \rho u^2(k)];\ e(k) = y - y_r$$

where $y(k)$ is the *actual* output, y_r is the *reference*/*command* value, $u(k)$ is the *control* signal, and ρ is a *weighting factor*.

We call this control problem an *infinite-horizon* problem. Note that 'infinite horizon' does not necessarily mean that infinite amount of time is required for the control $u(k)$ to achieve the desired performance; it just means that there is no fixed deadline on time yielding the desired performance. In fact, as we know, a good design yields stability and steady-state accuracy in seconds.

The other design methods considered in this book so far (e.g., PID, Pole-Placement) are also based on infinite-horizon assumption, though this is not explicitly visible as in linear quadratic control.

(ii) The control design methods considered so far are all *off-line* design methods; the design calculations are carried out in one shot before implementation (unless the design is a part of the MRAC/Self-Tuning loops).

(iii) The design in these methods is based on a *fixed* model of the plant. A model, however, is always an approximation of the system under consideration. With time, the parameters of the model become more and more inaccurate because of internal/external disturbances. Therefore, the control law $u(k)$ calculated at $k = 0$ would become more and more inaccurate when considered further in the future, if adequate provision is not built-in to account for the changes in the model. This, in fact, is the essence of feedback control; the error signal is a measure of the internal/external disturbances, and hence changes in the model; the control law $u(k)$ is forced to be a function of the error signal.

In GPC, we use the concepts of finite horizon, sequential design (on-line design), and the control strategy has open-loop structure. The GPC approach can be described as follows:

(1) Assume the measured (actual) value of the current system output is $y(k)$. With the data known up to time k, the value of the output $y(k + j)$ is *predicted* over a certain time horizon, called the *prediction horizon* N; $j = 1, 2, \ldots, N$. This 'output prediction' is based on the explicit use of the fixed plant model, and depends on the future values of the control variable $u(k + j)$ within a *control horizon* N_u; $j = 1, 2, \ldots, N_u$; $N_u \le N$. If $N_u < N$, then $u(k + j) = u(k + N_u)$; $j = N_u + 1, \ldots, N$.

(2) A reference trajectory $r(k + j)$; $j = 1, \ldots, N$, is defined which describes the desired system trajectory over the prediction horizon.

(3) The vector of future controls $u(k + j)$ is computed such that a cost function of the following form is minimized:

$$J = \sum_{j=1}^{N}[\hat{y}(k + j) - r(k + j)]^2 + \rho\sum_{j=1}^{N_u}\left[\Delta u\left(k + j - 1\right)\right]^2 \tag{10.46}$$

where $\hat{y}(k+j)$ is the predicted output sequence obtained with the data known up to time k, $\Delta u(k+j-1)$ is a future control increment ($\Delta u(k) = u(k) - u(k-1)$) obtained from minimization of cost function J, and ρ is a weighting factor. The horizons (N, N_u) and the weighting factor (ρ) are design parameters. The reference trajectory $r(k+j)$ can be a known constant, or known future variations.

(4) Once the minimization is achieved, the first optimized control action $u(k)$ is applied to the plant, and the resulting plant output is measured. This measurement of the plant output is used as initial state of the model to perform the next iteration.

Steps 1 to 4 are repeated at each sampling instant. The block diagram of the GPC scheme is shown in Fig.10.7b.

The following prime characteristics distinguish GPC approach from other design methods:

- At each sample, the control signal is determined to achieve a desired behavior in the following N steps. This is called a *receding horizon* strategy.
  As per the *principle of optimality* (Chapter 14), the first element $u(k)$ of the sequence of controls is *optimal* if the sequence, at every sample instant, had been determined to optimize the cost function (10.46) with $N = \infty$. This is because our control problem is an infinite-horizon problem. Therefore, using finite-horizon structure in GPC is an approximation, necessitated by the requirement of reducing the computational time for calculating $u(k+j)$ on-line.
- The GPC scheme of Fig. 10.7b is apparently an open-loop structure; therefore, one may doubt the robustness features of the scheme. The robustness is, in fact, built in the receding horizon and on-line properties of the scheme: at every decision step, the generalized predictive controller observes the state of the true system, synchronizes the estimate that it has with this, and tries to find the best sequence of actions given the updated state.

For the problem formulation with cost function (10.46), predictions $\hat{y}(k+j)$ are based on the measured values of $y(k)$, $y(k-1)$,…, and not on the predicted values $\hat{y}(k)$, $\hat{y}(k-1)$,…. This virtually amounts to a feedback loop, a source of measure of the internal/external disturbances.

Most real-world dynamical systems are inherently nonlinear. It provides motivation for the application of GPC strategies, given nonlinear model of the plant. However, in many situations, on-line nonlinear optimization problem makes implementation of GPC scheme impractical. For many nonlinear systems, a linearized model is acceptable when the system is working around the operating point. The GPC scheme with a linear predictive model is a powerful design method in the toolkit of control practitioners.

As the control variables in a GPC scheme are calculated based on the predicted output, the model needs to be able to reflect the dynamic behavior of the system as accurately as possible. The non-adaptive control scheme of Fig. 10.7b, when inserted in an adaptive loop such as self-tuning mode of Fig. 10.7a, will yield an improved performance.

The derivation that follows, employs a linear predictive model.

The Predictor

When considering regulation about a particular operating point, even a nonlinear plant generally admits a locally-linearized model (refer to Eqns (10.30)–(10.32)):

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \varepsilon(k) \tag{10.47a}$$

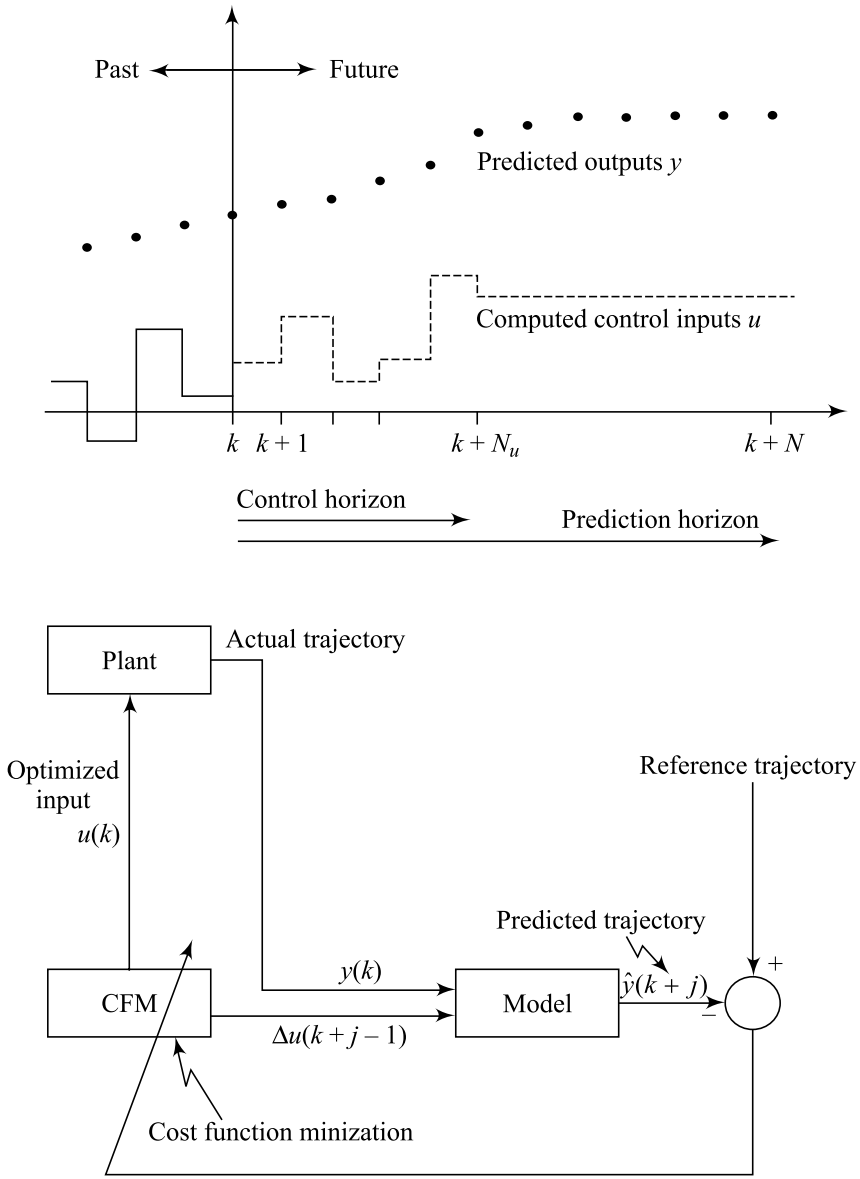


Fig. 10.7 (b) The GPC Structure

where A and B are polynomials in the backward shift operator z^{-1}:

$$A(z^{-1}) = 1 + \alpha_1 z^{-1} + \cdots + \alpha_n z^{-n} \tag{10.47b}$$

$$B(z^{-1}) = \beta_1 + \beta_2 z^{-1} + \cdots + \beta_n z^{-n} \tag{10.47c}$$

$\varepsilon(k)$ = some disturbance of unspecified character.

The leading elements β_1, β_2, ..., of the polynomial B are set to zero to account for the dead-time of the plant; and the trailing elements β_n, β_{n-1},…, are set to zero if the degree of polynomial B is less than n.

Principal disturbances encountered in industrial applications are accommodated by modeling $\varepsilon(k)$ as a white noise sequence independent of past control inputs. To obtain a controller with integral action, it is further assumed that the term $\varepsilon(k)$ is modeled as integrated white noise:

$$\varepsilon(k) = \frac{\xi(k)}{1 - z^{-1}} \tag{10.48}$$

when $\xi(k)$ is an uncorrelated random sequence. Combining with (10.47a), we obtain

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \xi(k)/\Delta \tag{10.49}$$

where Δ is the differencing operator $(1 - z^{-1})$.

Considering the time instant $k + j$, model (10.47) equivalently reads

$$A(z^{-1})\Delta y(k+j) = B(z^{-1})\Delta u(k+j-1) + \xi(k+j) \tag{10.49a}$$

To systematically derive a j-step ahead predictor of output, the model is reorganized by introduction of the following identity (*Diophantine equation*).

$$1 = \Delta A(z^{-1})E_j(z^{-1}) + z^{-j}F_j(z^{-1}) \tag{10.50a}$$

where E_j and F_j are polynomials uniquely defined, given the polynomial A and the prediction interval j;

$$\deg(E_j) = j - 1 \quad \text{and} \quad \deg(F_j) = \deg(A) = n:$$

$$E_j = e_0^{(j)} + e_1^{(j)}z^{-1} + \cdots + e_{j-1}^{(j)}z^{-(j-1)} \tag{10.50b}$$

$$F_j = f_0^{(j)} + f_1^{(j)}z^{-1} + \cdots + f_n^{(j)}z^{-n} \tag{10.50c}$$

Multiplying E_j to both sides of (10.49) and using the identity (10.50a) gives

$$(1 - z^{-j}F_j)y(k+j) = E_jB\Delta u(k+j-1) + E_j\xi(k+j)$$

or

$$y(k+j) = E_j\ B\Delta u(k+j-1) + F_jy(k) + E_j\xi(k+j) \tag{10.51}$$

Given that the sequence of future control inputs (i.e., $u(k+i)$ for $i > 1$) is known, and measured output data up to time k is available, the optimal predictor for $y(k+j)$ is the expectation conditioned on the information gathered up to time k (since E_j is of degree $j-1$, the noise components are all in the future):

$$\hat{y}(k+j\,|\,k) = G_j\Delta u(k+j-1) + F_jy(k) \tag{10.52a}$$

where $G_j = E_jB$ is a polynomial of order $n + j - 1$

$$= g_0^{(j)} + g_1^{(j)}z^{-1} + \cdots + g_{n+j-1}^{(j)}\ z^{-(n+j-1)} \tag{10.52b}$$

By multiplying $B/A\Delta$ to both sides of identity (10.50a), we obtain

$$G_j = \frac{B}{A\Delta} - z^{-j}F_j\frac{B}{A\Delta}$$

$$= \frac{B}{A\Delta} - z^{-j}\left[f_0^{(j)} + f_1^{(j)}z^{-1} + \cdots + f_n^{(j)}z^{-n}\right]\frac{B}{A\Delta} \tag{10.52c}$$

The polynomial $\frac{B}{A\Delta} = \frac{B(z^{-1})}{A(z^{-1})(1-z^{-1})}$ represents the z-transform of the response $y(k)$ of the process to unit-step input.

$$\text{Step response} = g_0 + g_1 z^{-1} + \cdots + g_{j-1} z^{-(j-1)} + g_j z^{-j} + \cdots \tag{10.52d}$$

It is obvious that the first j terms in G_j are same as the j coefficients of the step response of the process. This gives us one way of computing G_j for the prediction equation (10.52a). Both G_j and F_j may be computed by recursion of the Diophantine equation (10.50a), as is explained below.

With $\tilde{A}$ defined as ΔA, we have from (10.50a)

$$1 = E_j \tilde{A} + z^{-j} F_j \tag{10.53a}$$

Since $\tilde{A}$ is monic, the solution to

$$1 = E_1 \tilde{A} + z^{-1} F_1$$

is obviously

$$E_1 = 1; F_1 = z[1 - \tilde{A}]$$

Assume now that the solution to (10.53a) for some j exists, and consider the equation for $j + 1$:

$$1 = E_{j+1} \tilde{A} + z^{-(j+1)} F_{j+1} \tag{10.53b}$$

Subtracting the two gives

$$0 = \tilde{A}[E_{j+1} - E_j] + z^{-j}[z^{-1} F_{j+1} - F_j] \tag{10.54a}$$

Since $\deg(E_{j+1} - E_j) = \deg(E_{j+1}) = j$, it turns out to be good idea to define

$$E_{j+1} - E_j = \bar{E}_j + e_j^{(j+1)} z^{-1} \tag{10.54b}$$

where $e_j^{(j+1)}$ specifies the coefficient to z^{-j} in the polynomial E_{j+1}. Using this, (10.54a) can be rewritten as

$$0 = \tilde{A}\bar{E}_j + z^{-j}[z^{-1} F_{j+1} - F_j + \tilde{A} e_j^{(j+1)}] \tag{10.54c}$$

By again exploiting that $\tilde{A}$ is monic, it is evident that $\bar{E}_j = 0$, leading to

$$E_{j+1} = E_j + e_j^{(j+1)} z^{-j}$$

Consequently,

$$z^{-1} F_{j+1} - F_j + \tilde{A} e_j^{(j+1)} = 0$$

or

$$F_{j+1} = z[F_j - \tilde{A} e_j^{(j+1)}]$$

Comparing the coefficients, we obtain

$$\begin{aligned} e_j^{(j+1)} &= f_0^{(j)} \\ f_i^{(j+1)} &= f_{i+1}^{(j)} - \tilde{\alpha}_{i+1} f_0^{(j)}, i = 0, 1, \ldots, n \\ \tilde{A} &= 1 + \tilde{\alpha}_1 z^{-1} + \cdots + \tilde{\alpha}_n z^{-n} \end{aligned}$$

The following equations provide a summary of the recursions of Diophantine equation:

$$E_{j+1} = E_j + f_0^{(j)} z^{-j} \tag{10.55a}$$

$$f_i^{(j+1)} = f_{i+1}^{(j)} - \tilde{\alpha}_{i+1} f_0^{(j)}, i = 0, 1, \ldots, n \tag{10.55b}$$

with

$$f_{n+1}^{(j)} = 0$$

$$E_1 = 1, \text{ and } F_1 = z\ (1 - \tilde{A}) \tag{10.55c}$$

The only unknown quantities in the prediction equation (10.52a) are now the future control inputs. In order to derive the control law, it is necessary to separate these from the part of the expression containing known (past) data.

$$\hat{y}(k+1) = (G_1 - g_0)\Delta u(k) + F_1 y(k) + g_0 \Delta u(k) \tag{10.56a}$$

$$\hat{y}(k+j) = z^{j-1}[G_j - \bar{G}_j]\Delta u(k) + F_j y(k) + \bar{G}\Delta u(k+j-1); 1 < j \le N \tag{10.56b}$$

where $\bar{G}_j = g_0 + g_1 z^{-1} + \cdots + g_{j-1} z^{-(j-1)}$ (10.56c)

Deriving the Control Law

First the predictions derived above are expressed in the following vector notation.

$$\hat{\mathbf{y}} = \mathbf{\Gamma}\, \tilde{\mathbf{u}} + \boldsymbol{\varphi} \tag{10.57}$$

where

$$\hat{\mathbf{y}} = [\hat{y}(k+1) \quad \hat{y}(k+2) \cdots \hat{y}(k+N)]^T$$

$$\tilde{\mathbf{u}} = [\Delta u(k) \quad \Delta u(k+1) \cdots \Delta u(k+N_u-1)]^T$$

$$\boldsymbol{\varphi} = [\varphi(k+1) \quad \varphi(k+2) \cdots \varphi(k+N)]^T$$

with $\varphi(k+j) = z^{j-1}[G_j - \bar{G}_j]\Delta u(k) + F_j y(k)$.

$\mathbf{\Gamma}$ is a matrix of dimension $N \times N_u$:

$$\mathbf{\Gamma} = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ \bullet & \bullet & \cdots & g_0 \\ \vdots & \vdots & & \vdots \\ g_{N-1} & g_{N-2} & \cdots & g_{N-N_u} \end{bmatrix} \tag{10.58}$$

With the reference trajectory expressed in the form of a vector

$$\mathbf{r} = [r(k+1) \quad r(k+2) \cdots r(k+N)]^T,$$

the expression of the cost function of (10.46) can be written as

$$J = (\mathbf{\Gamma}\tilde{\mathbf{u}} + \boldsymbol{\varphi} - \mathbf{r})^T (\mathbf{\Gamma}\tilde{\mathbf{u}} + \boldsymbol{\varphi} - \mathbf{r}) + \rho \tilde{\mathbf{u}}^T \tilde{\mathbf{u}} \tag{10.59}$$

The sequence of future controls is determined by setting the derivative of the cost function to zero:

$$\frac{\partial J}{\partial \tilde{\mathbf{u}}} = 2\boldsymbol{\Gamma}^T(\boldsymbol{\Gamma}\tilde{\mathbf{u}} + \boldsymbol{\varphi} - \mathbf{r}) + 2\rho\tilde{\mathbf{u}})$$

$$= 2\boldsymbol{\Gamma}^T\boldsymbol{\Gamma}\tilde{\mathbf{u}} + 2\boldsymbol{\Gamma}^T(\boldsymbol{\varphi} - \mathbf{r}) + 2\rho\tilde{\mathbf{u}} = \mathbf{0}$$

or

$$\tilde{\mathbf{u}} = [\boldsymbol{\Gamma}^T\boldsymbol{\Gamma} + \rho\mathbf{I}]^{-1}\boldsymbol{\Gamma}^T(\mathbf{r} - \boldsymbol{\varphi}) \tag{10.60}$$

The matrix involved in the inversion is of the much reduced dimension $N_u \times N_u$. In particular, if $N_u = 1$ (as is usefully chosen for a 'simple' plant), this reduces to a scalar computation.

10.4.4 Application to a First-Order Control System

Consider a first-order system with a plant model of the form

$$y(k+1) = f\,y(k) + gu(k); y(0) \triangleq y^0 \tag{10.61}$$

where u is the control variable and y is the measured state (output); f and g are unknown coefficients.

An especially simple adaptive controller results by combining the least squares method of parameter estimation with the generalized predictive controller. The least squares parameter-estimation algorithm requires relatively small computational effort and has a reliable convergence, but is applicable only for small noise-signal ratios. Several applications have shown that the combination of least squares parameter estimation with generalized predictive control gives good results.

Let us assume that for the system given by Eqn. (10.61), the desired steady-state value for the controlled variable $y(k)$ is a constant reference input r. We select the generalized predictive control parameters: $\rho = 0.1$, $N = 4$, $N_u = 1$.

If the system parameters were known, the feedback controller should take the form (10.60). Since the parameters are assumed to be unknown, the least squares error estimates will be used in place of the true values of $f°$ and $g°$ of the parameters f and g. The parameter estimates $\hat{f}$ and $\hat{g}$ are derived from the input-output measurements.

To simulate the system (refer to Problem A.20 in Appendix A), the data values were obtained from Eqn. (10.61) assuming the true parameters

$$f° = 1.1052;\ g° = 0.0526$$

and sampling interval $T = 0.1$ sec.

With the initial estimate $\hat{\boldsymbol{\theta}}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{P}(0) = \alpha\,\mathbf{I}$ with large value of α, we use Equations (10.45) to generate the new parameter estimate, and implement the generalized predictive control law. The plot of Fig. 10.8 was generated using this procedure. The input signal is a square wave with amplitude 10. The closed-loop system is close to the desired behavior after a few transients.

10.4.5 Relations between MRAC and STR

As described above, MRAC control and STR arise from different perspectives; with the parameters in MRAC systems being updated so as to minimize the tracking errors between the plant output and

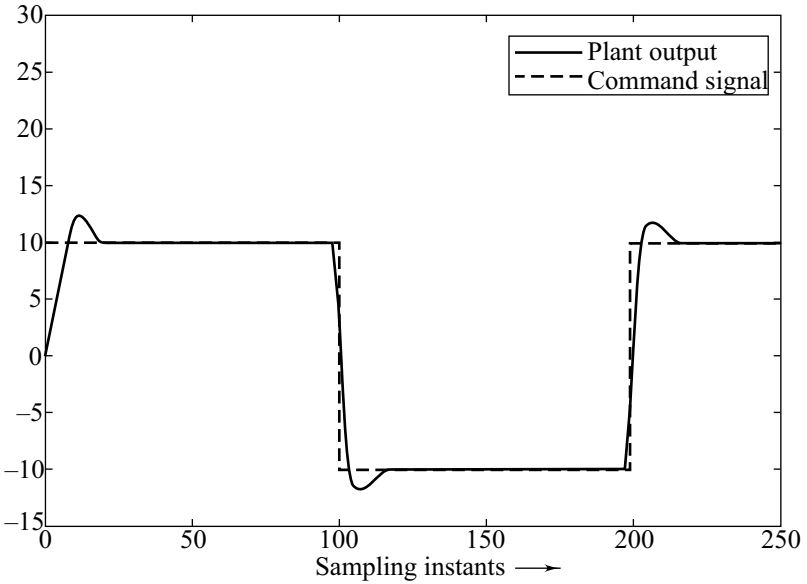


Fig. 10.8 Simulation results

reference model output, and the parameters in STR systems being updated so as to minimize the data-fitting error in input-output measurements. However, there are strong relations between the two design methodologies. Comparing Figs 10.4 and 10.7a, we note that the two kinds of systems have both an inner loop for control and an outer loop for parameters estimation. From a theoretical point of view, it can actually be shown that MRAC and STR systems can be put under a *unified* framework.

The two methods can be quite different in terms of analysis and implementation. Compared with MRAC systems, STR systems are more flexible because of the possibility of coupling various controllers with various estimators (i.e., the separation of control and estimation). However, the stability and convergence of self-tuning regulators are generally quite difficult to guarantee, often requiring the signals in the system to be sufficiently rich so that the estimated parameters converge to the true parameters. If the signals are not very rich (for example, if the reference signal is zero or a constant), the estimated parameters may not be close to the true parameters, and the stability and convergence of the resulting control system may not be guaranteed. In this situation, one must either introduce perturbation signals in the input, or somehow modify the control law. In MRAC systems, however, the stability and tracking error convergence are usually guaranteed—regardless of the richness of the signals.

10.5 SLIDING MODE CONTROL

As discussed earlier, modeling inaccuracies can have strong adverse effects on nonlinear control systems. Therefore, any practical design must address them explicitly. A major approach to dealing with model uncertainly is *adaptive control*, which we have discussed earlier in this chapter. Another major approach is the *variable structure sliding mode control*, which is the subject of this section.

For the class of problems to which it applies, sliding mode controller design provides a systematic approach to the problem of maintaining stability and consistent performance in the face of modeling

imprecisions. Sliding mode control has been successfully applied to robot manipulators, underwater vehicles, automotive engines, high-performance electric motors, and power systems.

We have, informally, already introduced the reader to variable structure sliding mode control systems in Section 9.10 (revisiting this section will be helpful). In the following, a formal introduction is presented. For a detailed account of the subject, refer to Slotine and Li [126], and Zak [35].

Consider a simple pendulum of mass M, suspended by a string of length l having negligible mass. Let θ be the angular displacement as shown in Fig. 10.9. τ represents the torque applied at the point of suspension, which will be considered to be the control input to the system. Ignoring the effects of friction, the system can be represented mathematically as

$$J\ddot{\theta}(t) + Mgl \sin\theta(t) = \tau(t)$$

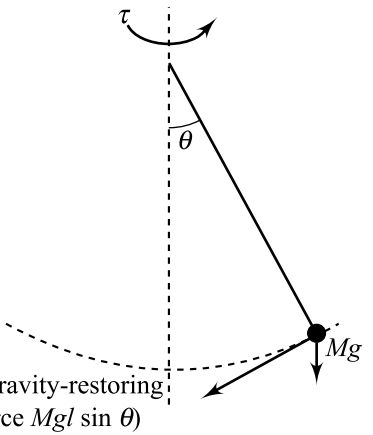


Fig. 10.9 Schematic of a pendulum

where g is the acceleration due to gravity, and $J = Ml^2$ is the moment of inertia.

By appropriate scaling, the essential dynamics of the system are captured by

$$\ddot{y}(t) = -\alpha \sin y(t) + u(t) \tag{10.62}$$

where α is a positive scalar

Ignoring the nonlinear sine term, we get the following linear approximation of the pendulum equation:

$$\ddot{y}(t) = u(t) \tag{10.63a}$$

Choosing $x_1 = y$ and $x_2 = \dot{y}$ as state variables, we have

$$\begin{aligned} \mathbf{x} &= [x_1 \quad x_2]^T \\ \dot{x}_1(t) &= x_2(t); \\ \dot{x}_2(t) &= u(t) \end{aligned} \tag{10.63b}$$

The linear control methodologies (pole placement, optimal control) attempt to *minimize*, in some sense, the transfer functions relating the disturbances to the outputs of interest. Here we explore discontinuous control (variable structure control, refer to Section 9.10) methodology for disturbance rejection. The system with two control structures, corresponding to $u = +1$ and $u = -1$, is considered. The variable structure law governing the dynamics is given by

$$u(t) = \begin{cases} -1 & \text{if} \quad \sigma(x_1, x_2) > 0 \\ +1 & \text{if} \quad \sigma(x_1, x_2) < 0 \end{cases} \tag{10.64a}$$

where the *switching function* is defined by

$$\sigma(x_1, x_2) = \lambda x_1 + x_2 \tag{10.64b}$$

λ is a positive design scalar. The reason for the use of the term 'switching function' is clear, since the function in Eqns (10.64) is used to decide which control structure is in use at any point (x_1, x_2) in the phase plane. The expression in Eqn. (10.64a) is usually written more concisely as

$$u(t) = -\text{sgn}(\sigma(t)) \tag{10.64c}$$

where sgn $(\cdot)$ is the sign function. This function exhibits the property that

$$\sigma \operatorname{sgn}(\sigma) = |\sigma|$$

Figure 10.10a shows typical trajectories (parabolas) for $u = \pm k$, and a typical switching line. Close to the origin, on either side of the switching line, the trajectories point towards the line; an instant after the control structure changes, the system trajectory will recross the switching line and the control structure must switch back. Intuitively, high-frequency switching between the two control structures will take place as the system trajectories repeatedly cross the line. This high frequency motion is described as *chattering*. If infinite-frequency switching were possible, the motion would be trapped or constrained to remain on the line. The motion when confined to the switching line satisfies the differential equation obtained from rearranging $\sigma(x_1, x_2) = 0$, namely

$$x_2 = -\lambda x_1 \tag{10.65a}$$

or

$$\dot{y}(t) = -\lambda y(t) \tag{10.65b}$$

This represents a first-order decay and the trajectories will 'slide' along the switching line to the origin. Such a dynamical behavior is described as *sliding mode* and the switching line is termed the *sliding surface*. During sliding motion, the system behaves as a reduced-order system which is apparently independent of the control. The choice of the sliding surface, represented in our example by the parameter λ, governs the performance response whilst the control law itself, is designed to guarantee that trajectories are driven to the 'region' of the sliding surface where the sliding motion takes place. To achieve this objective, the control action is required to satisfy certain conditions, called *reachability conditions*.

To develop the reachability conditions and the region of sliding motion, we consider the system (10.63b) with u given by (10.64), i.e., $u = \pm 1$ corresponding to the two control structures. Figure 10.10b shows typical trajectories of the control system with these control structures, and a typical switching line.

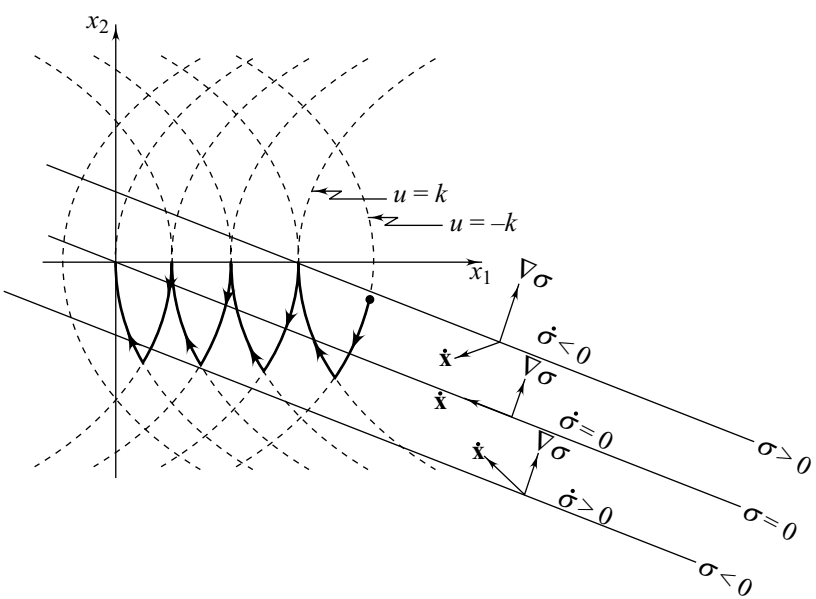


Fig. 10.10 (a) An illustration of the reachability conditions

Assume that the system under consideration starts with initial conditions corresponding to point A in Fig. 10.10b. The control switches when the representative point reaches B. By geometry of the situation, we see that the trajectory, resulting from the reversal of the control at point B, will bring the representative point on a parabola much closer to the origin. This will continue until the trajectory intersects the switching line at a point closer to the origin than the points A_1 and A_2 which are points of intersection of the switching line $\sigma(x_1, x_2)$, with parabolas passing through the origin: $x_1 = -\frac{1}{2}x_2|x_2|$. The coordinates of the points A_1 and A_2 are obtained as $\left(\frac{-2}{\lambda^2}, \frac{2}{\lambda}\right)$ and $\left(\frac{2}{\lambda^2}, \frac{-2}{\lambda}\right)$, respectively. The region where the sliding motion takes place, is a part of the switching line between the points A_1 and A_2 as is seen below.

$$\begin{aligned}\sigma(t)\dot{\sigma}(t) &= \sigma(\lambda\dot{x}_1 + \dot{x}_2)\\ &= \sigma(\lambda x_2 + u)\\ &= \sigma(\lambda x_2 - \operatorname{sgn}(\sigma))\\ &= \lambda\sigma x_2 - |\sigma|\end{aligned}$$

Since

$$\lambda\sigma x_2 \le \lambda|\sigma||x_2|$$

we have

$$\begin{aligned}\sigma(t)\dot{\sigma}(t) &\le \lambda|\sigma||x_2| - |\sigma|\\ &\le |\sigma|(\lambda|x_2| - 1)\end{aligned}$$

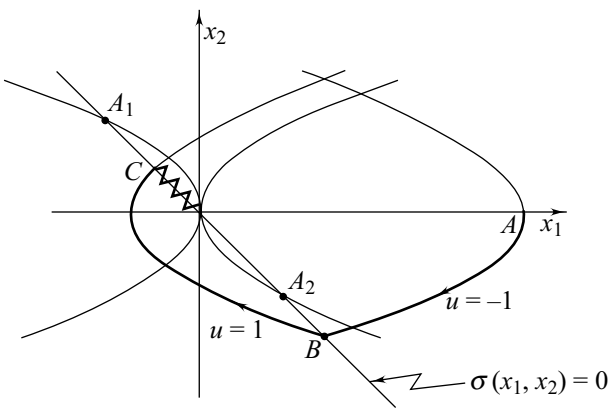


Fig. 10.10 (b) Typical trajectories of the closed-loop system (10.53)–(10.54)

For values of x_2 satisfying the inequality

$$\lambda|x_2| < 1, \tag{10.66a}$$

we have

$$\begin{aligned}&\sigma(t)\dot{\sigma}(t) < 0;\\ &\dot{\sigma}(t) = \left(\frac{\partial\sigma}{\partial\mathbf{x}}\right)^T \frac{d\mathbf{x}}{dt} = (\nabla\sigma)^T\,\dot{\mathbf{x}}\end{aligned} \tag{10.66b}$$

Inequality (10.66b) is equivalently expressed by the following two conditions:

$$\lim_{\sigma \to 0^+} \dot{\sigma} = \lim_{\sigma \to 0^+} (\nabla \sigma)^T \dot{\mathbf{x}} < 0$$

and

$$\lim_{\sigma \to 0^-} \dot{\sigma} = \lim_{\sigma \to 0^-} (\nabla \sigma)^T \dot{\mathbf{x}} > 0 \tag{10.67}$$

These conditions, called *reachability conditions*, ensure that when $\lambda |x_2| < 1$, the system trajectories on either side of the line $\sigma(x_1, x_2) = 0$ point towards the line. This is illustrated in Fig. 10.10a.

The control action, rather than prescribing the dynamic performance, ensures, instead, that the reachability conditions are satisfied. The choice of the sliding surface governs the system performance. It should be noted that the control action required to satisfy reachability conditions is discontinuous in nature.

The double-integrator system of Eqns (10.63) is a linear approximation of the pendulum dynamics given in Eqn. (10.62). An alternative interpretation is that the nonlinear term $\alpha \sin y(t)$ is a *disturbance* or uncertainly in the nominal double-integrator system. The key result is that, in finite time, the phase portrait intercepts the sliding surface and is forced to remain there. The significance of this is that, once sliding is established, the double-integrator system and the pendulum behave in an identical fashion, namely,

$$\dot{y}(t) = -\lambda\, y(t)$$

The effect of disturbance or uncertainty in the nominal double-integrator system has been completely rejected. As such, the closed-loop system is *robust*, i.e., it is insensitive to mismatches between the model used for control law design, and the plant on which it will be implemented. The control action applied to the plant *does not* utilize any knowledge of the uncertainty.

The concepts of sliding mode control, developed through an example, are summarized below.

10.5.1 Sliding Mode Control Algorithm

The problem is to regulate a dynamic system subject to parameter uncertainties and nonlinearities. A controller is sought to force the system to reach, and subsequently remain on, a predefined surface (called the *sliding surface*) within the state space. The dynamical behavior of the system, when confined to the surface, is called the *sliding motion*. The advantages of obtaining such a motion are two fold: firstly, there is a reduction in order; and, secondly, the sliding motion is insensitive to parameter variations. The latter property of invariance towards *uncertainty* makes the methodology an attractive one for designing *robust control* for uncertain systems.

The design approach comprises the following two components:

- The design of a sliding surface in the state space, so that the reduced-order sliding motion satisfies the specifications imposed on the design.
- The synthesis of a control law, discontinuous about the sliding surface, such that the trajectories of the closed-loop motion are directed towards the surface.

The closed-loop dynamical behavior obtained for using a variable structure control law, comprises two distinct types of motion. The initial phase, often referred to as the *reaching phase*, occurs whilst the states are being driven towards the sliding surface. This motion is, in general, affected by the disturbances present. Only when the states reach the surface, and the sliding motion takes place, does the system become insensitive to uncertainty.

A sliding mode will exist if, in the vicinity of the sliding surface, the state velocity vectors are directed towards the surface. In such a case, the sliding surface attracts trajectories when they are in its vicinity; and, once a trajectory intersects the sliding surface, it will stay on it thereafter.

A hyper surface

$$S : \sigma(x_1, x_2, \ldots, x_n) = \sigma(\mathbf{x}) = 0 \tag{10.68a}$$

is attractive if

(i) any trajectory starting on the surface remains there; and
(ii) any trajectory starting outside the surface tends to it at least asymptotically.

The following conditions (called *reachability conditions*) ensure that the motion of the state trajectory $\mathbf{x}(t)$ of the single-input dynamical system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, t) \tag{10.68b}$$

on either side of the sliding surface $\sigma(\mathbf{x}) = 0$, is towards the surface.

$$\lim_{\sigma \to 0^+} \dot{\sigma} < 0; \text{ and } \lim_{\sigma \to 0^-} \dot{\sigma} > 0$$

in some domain Ω of the state space

The two conditions may be combined to give

$$\sigma \dot{\sigma} < 0$$

in the neighborhood of the sliding surface, i.e.,

$$\lim_{\sigma \to 0} \sigma \frac{d\sigma}{dt} < 0 \tag{10.68c}$$

In the sliding mode, the trajectory remains on the hyper surface S for all times after hitting S, and so in the sliding mode we require

$$\frac{d\sigma}{dt} = \left(\frac{\partial \sigma}{\partial \mathbf{x}} \right)^T \dot{\mathbf{x}} = 0$$

$$\sigma(\mathbf{x}) = 0 \tag{10.69}$$

In general, if the reachability conditions are satisfied globally, i.e., Ω is the entire state space, then, since

$$\tfrac{1}{2} \frac{d}{dt} \sigma^2 = \sigma \dot{\sigma} < 0,$$

it follows that

$$V(\sigma) = \tfrac{1}{2} \sigma^2 \tag{10.70}$$

is a Lyapunov function for $\sigma(t)$.

Extension of these concepts to the multi-input situations is straightforward.

We illustrate these concepts through the design of a sliding mode controller for a two-link robot.

10.5.2 Sliding Mode Controller for a Two-Link Robot

The plant model, derived earlier in Section 10.2, is given by Eqns (10.6):

$$\begin{aligned}\dot{\mathbf{x}}_1 &= \dot{\boldsymbol{\theta}} = \mathbf{x}_2 \\ \dot{\mathbf{x}}_2 &= \ddot{\boldsymbol{\theta}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\boldsymbol{\tau}\end{aligned} \tag{10.71}$$

where

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} = \begin{bmatrix}x_{11} \\ x_{12} \\ x_{21} \\ x_{22}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}}\end{bmatrix} = \begin{bmatrix}\theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix}; \boldsymbol{\tau} = \begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix}$$

θ_1 and θ_2 are the angles of the two links, defined in Fig. 10.1, and τ_1 and τ_2 are the torques applied by the actuators to control the angles θ_1 and θ_2, respectively.

The design of a variable structure sliding mode controller consists of the following two phases:

- Sliding (switching) surface design so as to achieve the desired system behavior, when restricted to the surface.
- Selecting feedback gains of the controller, so that the closed-loop system is stable to the sliding surface.

Let us consider a specific design problem for the two-link robot under study: tracking the desired motion trajectory $\boldsymbol{\theta}_d(t)$.

Define the *tracking error* as

$$\mathbf{e}(t) = \boldsymbol{\theta}_d(t) - \boldsymbol{\theta}(t) \tag{10.72a}$$

Therefore,

$$\dot{\mathbf{e}}(t) = \dot{\boldsymbol{\theta}}_d(t) - \dot{\boldsymbol{\theta}}(t);\ \ddot{\mathbf{e}}(t) = \ddot{\boldsymbol{\theta}}_d(t) - \ddot{\boldsymbol{\theta}}(t) \tag{10.72b}$$

Defining $\tilde{\mathbf{x}}_1 = \mathrm{e}$ and $\tilde{\mathbf{x}}_2 = \dot{\mathrm{e}}$, we can write robot dynamics (10.71) in the form (refer to Eqns (10.15))

$$\begin{aligned}\dot{\tilde{\mathbf{x}}}_1 &= \tilde{\mathbf{x}}_2 \\ \dot{\tilde{\mathbf{x}}}_2 &= \ddot{\boldsymbol{\theta}}_d - \mathbf{f}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\boldsymbol{\tau}\end{aligned} \tag{10.73}$$

Here, we use linear sliding surface (although we can use nonlinear sliding surface as well) defined by the equation

$$\boldsymbol{\sigma}(\tilde{\mathbf{x}}) = \boldsymbol{\lambda}\ \tilde{\mathbf{x}}_1 + \mathbf{I}\,\tilde{\mathbf{x}}_2 = \mathbf{0} \tag{10.74a}$$

or

$$\begin{bmatrix}\sigma_1(\tilde{\mathbf{x}}) \\ \sigma_2(\tilde{\mathbf{x}})\end{bmatrix} = \begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}\begin{bmatrix}\tilde{x}_{11} \\ \tilde{x}_{12}\end{bmatrix} + \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\tilde{x}_{21} \\ \tilde{x}_{22}\end{bmatrix} = \begin{bmatrix}0 \\ 0\end{bmatrix} \tag{10.74b}$$

Note that

$$\begin{aligned}\sigma_1(\tilde{\mathbf{x}}) &= \lambda_1\tilde{x}_{11} + \tilde{x}_{21} = \lambda_1(\theta_{1d} - \theta_1) + (\dot{\theta}_{1d} - \dot{\theta}_1) \\ \sigma_2(\tilde{\mathbf{x}}) &= \lambda_2\tilde{x}_{12} + \tilde{x}_{22} = \lambda_2(\theta_{2d} - \theta_2) + (\dot{\theta}_{2d} - \dot{\theta}_2)\end{aligned} \tag{10.75}$$

We have assumed the coefficient of $\tilde{x}_{21}$ to be unity, without loss of generality. If this were not the case, we could divide both sides of $\sigma_1(\tilde{\mathbf{x}}) = 0$ by the coefficient of $\tilde{x}_{21}$, to ensure that coefficient of $\tilde{x}_{21}$ is 1. The same argument applies to the selection of unity coefficient of $\tilde{x}_{22}$.

We now combine equations of the plant and that of the sliding surface (Eqns (10.73) and (10.74)).

$$\tilde{\mathbf{x}}_2 = -\boldsymbol{\lambda}\tilde{\mathbf{x}}_1$$

Therefore,

$$\dot{\tilde{\mathbf{x}}}_1 = -\boldsymbol{\lambda}\,\tilde{\mathbf{x}}_1 \tag{10.76}$$

The above equation describes the system dynamics in sliding (observe the order-reduction of system dynamics in sliding). The response of the system in sliding is completely specified by an appropriate choice of the parameters λ_1 and λ_2 of the switching surface. While in sliding, the system is not affected by model uncertainties.

After designing a sliding surface, we construct a feedback controller. The controller objective is to drive the plant state to the sliding surface, and maintain it on the surface for all subsequent time. We use a generalized Lyapunov approach in constructing the controller. Specifically, we use a distance measure, $V = \frac{1}{2}\boldsymbol{\sigma}^T\boldsymbol{\sigma} = \frac{1}{2}(\sigma_1^2 + \sigma_2^2)$, from the sliding surface $\boldsymbol{\sigma}$ as a Lyapunov function candidate. Then, we select the controller so that the time derivative of the chosen Lyapunov function candidate, evaluated on the solution of the controlled system, is negative-definite with respect to the switching surface; thus, ensuring the motion of the state trajectory to the surface, as it is illustrated in Fig. 10.10a. Our goal is to find $\boldsymbol{\tau}$ so that

$$\frac{d}{dt}\tfrac{1}{2}\boldsymbol{\sigma}^T\boldsymbol{\sigma} = \boldsymbol{\sigma}^T\dot{\boldsymbol{\sigma}} < 0 \tag{10.77}$$

$$\boldsymbol{\sigma}^T\dot{\boldsymbol{\sigma}} = \boldsymbol{\sigma}^T[\boldsymbol{\lambda}\dot{\tilde{\mathbf{x}}}_1 + \mathbf{I}\dot{\tilde{\mathbf{x}}}_2] = \boldsymbol{\sigma}^T\left[\boldsymbol{\lambda}(\dot{\mathbf{x}}_{1d} - \dot{\mathbf{x}}_1) + (\dot{\mathbf{x}}_{2d} - \mathbf{f}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\boldsymbol{\tau})\right]$$

We consider the controller structure of the form

$$\boldsymbol{\tau} = \mathbf{g}^{-1}(\mathbf{x})\left[-\mathbf{f}(\mathbf{x}) + \dot{\mathbf{x}}_{2d} + \boldsymbol{\lambda}(\dot{\mathbf{x}}_{1d} - \dot{\mathbf{x}}_1) + \begin{Bmatrix} k_1\,\mathrm{sgn}(\sigma_1) \\ k_2\,\mathrm{sgn}(\sigma_2) \end{Bmatrix}\right] \tag{10.78}$$

where $k_1 > 0$ and $k_2 > 0$ are the gains to be determined so that the condition $\boldsymbol{\sigma}^T\dot{\boldsymbol{\sigma}} < 0$ is satisfied. To determine these gains, we substitute $\boldsymbol{\tau}$, given by (10.78), into the expression $\boldsymbol{\sigma}^T\dot{\boldsymbol{\sigma}}$.

$$\boldsymbol{\sigma}^T\dot{\boldsymbol{\sigma}} = -[\sigma_1 \quad \sigma_2]\begin{bmatrix} k_1\,\mathrm{sgn}(\sigma_1) \\ k_2\,\mathrm{sgn}(\sigma_2) \end{bmatrix}$$

$$= -\sigma_1 k_1\,\mathrm{sgn}(\sigma_1) - \sigma_2 k_2\,\mathrm{sgn}(\sigma_2) = -k_1|\sigma_1| - k_2|\sigma_2| < 0$$

Thus, the sliding surface $\boldsymbol{\sigma}(\tilde{\mathbf{x}}) = \mathbf{0}$ is asymptotically attractive. The larger the values of gains, the faster the trajectory converges to the sliding surface. Note that tolerance of sliding mode control to model imprecision and disturbances, is high; satisfying asymptotic stability requirement, despite the presence of uncertainties, ensures asymptotic tracking.

Simulation of this controller for the two-link robot arm ($m_1 = 1$, $m_2 = 1$, $l_1 = 1$, $l_2 = 1$, $g = 9.8$; $\theta_{d1}(t) = \sin(\pi t)$, $\theta_{d2}(t) = \cos(\pi t)$) was done using MATLAB (refer to Problem A.21 in Appendix A). Figures 10.11 show the tracking performance.

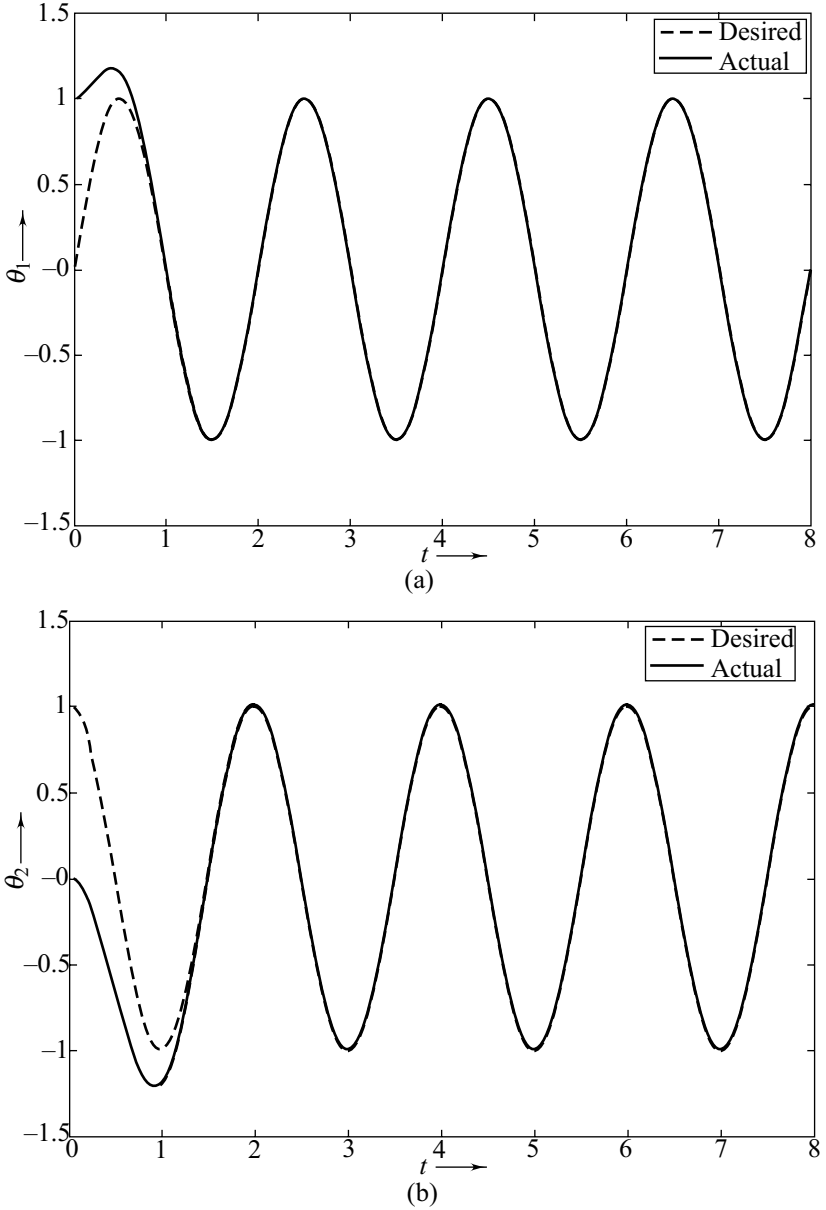


Fig. 10.11 Desired and actual trajectories

PROBLEMS

10.1 The system

$$\begin{aligned}\dot{x}_1 &= x_1x_2 + x_3\\ \dot{x}_2 &= -2x_2 + x_1u\\ \dot{x}_3 &= \sin x_1 + 2x_1x_2 + u\\ y &= x_1\end{aligned}$$

can be transformed to Brunovsky form by differentiating $y(t)$ repeatedly, and substituting state derivatives from the given system equations, until the control input $u(t)$ appears:

$$\begin{aligned}\dot{y} &= \dot{x}_1 = x_1x_2 + x_3\\ \ddot{y} &= x_1\dot{x}_2 + \dot{x}_1x_2 + \dot{x}_3\\ &= \left[\sin x_1 + x_2x_3 + x_1x_2^2\right] + \left[1 + x_1^2\right]u\\ &\equiv f(\mathbf{x}) + g(\mathbf{x})u; \mathbf{x} = [x_1 \quad x_2 \quad x_3]^T\end{aligned}$$

Defining variables as $z_1 \equiv y, z_2 \equiv \dot{y},$ we obtain

$$\begin{aligned}\dot{z}_1 &= z_2\\ \dot{z}_2 &= f(\mathbf{x}) + g(\mathbf{x})u\end{aligned}$$

This may be converted to a linear system by redefinition of the input as

$$v(t) \equiv f(\mathbf{x}) + g(\mathbf{x})u(t)$$

so that

$$u(t) \equiv \frac{1}{g(\mathbf{x})}(-f(\mathbf{x}) + v(t))$$

for then one obtains

$$\dot{z}_1 = z_2; \dot{z}_2 = v$$

which is equivalent to

$$\ddot{y} = v$$

With a PD tracking control

$$v = \ddot{y}_d + K_D\dot{e} + K_Pe$$

where tracking error is defined as

$$e(t) \equiv y_d(t) - y(t); y_d(t) \text{ is the desired trajectory;}$$

the closed-loop system becomes

$$\ddot{e} + K_D\dot{e} + K_Pe = 0$$

The complete controller implied by this feedback linearization technique, is given by

$$u(t) = \frac{1}{g(\mathbf{x})}(-f(\mathbf{x}) + \ddot{y}_d + K_D\dot{e} + K_Pe)$$

(a) Draw the structure of the feedback linearization controller showing PD outer loop and nonlinear inner loop.

(b) Select gains K_P and K_D so that the closed-loop system has a natural frequency of ω_n = 10 rad/sec, and a damping ratio of $\zeta = 0.707$.

(c) Suppose that it is desired for the plant output $y(t)$ to follow the trajectory $y_d = \sin(2\pi t)$. Simulate the system and plot actual output $y(t)$, desired output $y_d(t)$, and the tracking error $e(t)$; given $\mathbf{x}(0) = [1\ 1\ 1]^T$.

10.2 One useful method for specifying system performance is by means of a model that will produce the desired output for a given input. The model need not be actual hardware. It can only be a mathematical model simulated on a computer. In a model reference control system, the outputs of the model and that of the plant are compared and the difference is used to generate the control signals.

Consider a nonlinear, time-varying plant described by

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -b & -a(t)x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = \mathbf{f}(\mathbf{x}, \boldsymbol{u}, t)$$

where $a(t)$ is time-varying and b is a positive constant.

Assume the reference model equation to be

$$\dot{\mathbf{x}}_m = \begin{bmatrix} \dot{x}_{m1} \\ \dot{x}_{m2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_{m1} \\ x_{m2} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} v$$

$$= \mathbf{A}_m \mathbf{x}_m + \mathbf{b}_m v$$

The error vector

$$\mathbf{e} = \mathbf{x}_m - \mathbf{x}$$

Define a function

$$V(\mathbf{e}) = \mathbf{e}^T \mathbf{P} \mathbf{e}$$

where $\mathbf{P}$ is positive-definite, real, symmetric matrix.

Then $\dot{V}(\mathbf{e})$ is defined as

$$\dot{V}(\mathbf{e}) = \mathbf{e}^T (\mathbf{A}_m^T \mathbf{P} + \mathbf{P}\mathbf{A}_m)\mathbf{e} + 2M$$

where

$$M = \mathbf{e}^T \mathbf{P} \left[\mathbf{A}_m \mathbf{x} - \mathbf{f}(\mathbf{x}, u, t) + \mathbf{b}_m v \right]$$

The assumed $V(\mathbf{e})$ function is a Lyapunov function if

(i) ${\mathbf{A}_m}^T \mathbf{P} + \mathbf{P}\mathbf{A}_m = -\mathbf{Q}$ is a negative-definite matrix; and

(ii) the control u can be chosen to make the scalar quatity M nonpositive.

Choosing the matrix $\mathbf{Q}$ to be

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix} = \text{positive definite,}$$

we obtain

$$\dot{V}(\mathbf{e}) = -\left(q_{11} e_1^2 + q_{22} e_2^2 \right) + 2M$$

where

$$M = [e_1 \quad e_2]\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}\left\{\begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -b & -2\zeta\omega_n \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 0 \\ u \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 v \end{bmatrix}\right\}$$

$$= (e_1 p_{12} + e_2 p_{22})\left[-(\omega_n^2 - b)x_1 - 2\zeta\omega_n x_2 + a(t)x_2^2 + \omega_n^2 v - u\right]$$

If we choose u so that

$$u(t) = -(\omega_n^2 - b)x_1 - 2\zeta\omega_n x_2 + \omega_n^2 v + a_m x_2^2 \operatorname{sign}\left(e_1 p_{12} + e_2 p_{22}\right);\ a_m = \max |\, a(t) \,|$$

then

$$M = (e_1 p_{12} + e_2 p_{22})[a(t) - a_m \operatorname{sign}(e_1 p_{12} + e_2 p_{22})]x_2^2$$
$$= \text{nonpositive}$$

(a) Draw a block diagram representing the structure of the model reference adaptive control system.

(b) For the parameters: $a(t) = 0.2 \sin t$, $b = 8$, $\zeta = 0.7$, $\omega_n = 4$; simulate the closed-loop system and plot $\mathbf{x}(t)$, $\mathbf{x}_m(t)$, and $\mathbf{e}(t)$.

10.3 Consider the adaptive control design problem for a plant, approximately represented by a first-order differential equation

$$\dot{y} = -a_p y + b_p u$$

where $y(t)$ is the plant output, $u(t)$ is its input, and a_p and b_p are constant plant parameters (unknown to the adaptive controller). The desired performance of the control system is specified by a first-order reference model

$$\dot{y}_m = -a_m y_m + b_m r$$

where a_m and b_m are known constant parameters, and $r(t)$ is a bounded external reference signal. Using Lyapunov synthesis approach, formulate a control law, and an adaptation law, such that the resulting model-following error $y(t) - y_m(t)$, asymptotically converges to zero.

Simulate the MRAC system with $a_p = 1$, $b_p = 2$, $a_m = 3$ and $b_m = 3$, adaptation gain $\gamma = 1.5$; initial values of both parameters of the controller are chosen to be 0, indicating no *a priori* knowledge, and the initial conditions of the plant and the model are both zero. Use two different reference signals in the simulation: $r(t) = 2$, and $r(t) = 2 \sin(3t)$.

10.4 The following data were collected from a cell concentration sensor, measuring absorbance in a biochemical stream. The input u is the flow rate deviation (in dimensionless units) and the sensor output y is given in volts. The flow rate (input) is piecewise constant between sampling instants. The process is not at steady-state initially; so y can change even though $u = 0$. Fit a first-order model

$$y(k) = a_1 y(k-1) + b_1 u(k-1)$$

to the data using the least-squares approach. Plot the model response and the actual data.

Time(sec)	*u*	*y*
0	0	3.000
1	3	2.456
2	2	5.274
3	1	6.493
4	0	6.404
5	0	5.243
6	0	4.293
7	0	3.514
8	0	2.877
9	0	2.356
10	0	1.929

10.5 Step test data have been obtained for the off-gas CO_2 concentration response—obtained by changing the feed rate to a bioreactor. At $k = 0$, a unit-step change in input u occurs, but the output change at the first sample ($k = 1$) is not observed until the next sampling instant. The data is given in the table below.

Estimate the model parameters in the second-order difference equation

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2)$$

from the input-output data using the least-squares approach. Plot the model response and the actual data.

k	0	1	2	3	4	5	6	7	8	9	10
$y(k)$	0	0.058	0.217	0.360	0.488	0.6	0.692	0.772	0.833	0.888	0.925

10.6 The following data were collected for a process:

Time(sec)	*Input u*	*Output y*
0	1	4.0000
1	1	–2.0000
2	0	–1.0000
3	1	8.5000
4	1	–9.7500
5	1	1.6250
6	1	21.0625
7	0	–30.8438
8	1	7.1406
9	0	51.9766
10	1	–89.2461

Fit a second-order model

$$y(k) + a_1 y(k-1) + a_2 y(k-2) = b_1 u(k-1) + b_2 u(k-2)$$

to the data using the *least-squares* approach. Plot the model response and the actual data.

Simulate a self-tuner based on least squares estimation of the parameters of the model, and the pole-placement design given in Chapter 7. The system is required to track a constant input of amplitude 10; the dynamics specified as $\zeta = 0.5$, $\omega_n = 1$.

10.7 Consider a nonlinear system described by the equation

$$\ddot{x} + a(t)\dot{x}^2 \cos 3x = u$$

where $a(t)$ in unknown but satisfies

$$1 \le a(t) \le 2$$

With the nominal value of $a(t) = 1.5$, we have

$$\ddot{x} = f + u; f = -1.5\dot{x}^2 \cos 3x$$

In order to have the system track $x(t) = x_d(t) = \sin(\pi t/2)$, we define a sliding surface

$$\sigma = \dot{\tilde{x}} + \lambda \tilde{x}; \tilde{x} = x - x_d$$

We then have

$$\dot{\sigma} = \ddot{x} - \ddot{x}_d + \lambda \dot{\tilde{x}} = f + u - \ddot{x}_d + \lambda \dot{\tilde{x}}$$

Selecting

$$u = -f + \ddot{x}_d - \lambda \dot{\tilde{x}} - k \ \text{sgn}(\sigma); k > 0,$$

we get

$$\dot{\sigma} = -k \, \text{sgn}(\sigma)$$

We consider the function $V = \frac{1}{2}\sigma^2$. Note that V is positive-definite with respect to the sliding surface ($\sigma = 0$). The time derivative of V, evaluated on the trajectories of the closed-loop system, is

$$\begin{aligned} \dot{V} &= \sigma\dot{\sigma} \\ &= -\sigma k \, \text{sgn}(\sigma) = -k \mid \sigma \mid < 0 \end{aligned}$$

Thus, the sliding surface is asymptotically attractive, and the system, restricted to the sliding surface, can be made asymptotically stable by an appropriate choice of the parameter λ of the sliding surface.

Simulate the system, and plot tracking error and control law with

$$\lambda = 2, k = 0.1, \ a(t) = \mid \sin t \mid + 1$$

Chapter 11

Intelligent Control with Neural Networks/Support Vector Machines

11.1 TOWARDS INTELLIGENT SYSTEMS

Man has always dreamed of creating machines with human-like attributes. In this technological world, there are machines that have emulated several human functions with tremendous capacity and capabilities. Robots in manufacturing, mining, agriculture, space, ocean exploration, and health sciences, are just a few examples.

At present, these machines are more or less 'slaves' to the 'commands'. One of the tenets of recent research in robotics and systems science is that intelligence can be cast into a machine. This is, perhaps, an ultimate challenge to science—to create *intelligent systems* that emulate human *intelligence*.

Human intelligence possesses robust attributes with complex sensory, control, affective (emotional processes), and cognitive (thought processes) aspects of information processing and decision making. Biological neurons, over one hundred billion in number, in our central nervous system (CNS), play a key role in these functions. Essentially, CNS acquires information from the external environment through various natural sensory mechanisms such as vision, hearing, touch, taste, and smell. It integrates the information and provides appropriate interpretation through the cognitive computing. The cognitive process then advances further towards some attributes such as learning, recollection, and reasoning, which results in appropriate actions through muscular control.

Recent progress in information-based technology has significantly broadened the capabilities and application of computers. Today's computers, however, are merely being used for the storage and processing of numerical data. If we wish to emulate in a machine (computer), some of the cognitive functions (learning, remembering, reasoning, perceiving, etc.) of humans, we have to generalize the definition of information and develop new mathematical tools and hardware that must deal with the simulation and processing of cognitive information. Mathematics, as we know it today, was developed for the understanding of physical processes, whereas the process of cognition does not necessarily follow these mathematical laws. Then what is *cognitive mathematics*? This is a difficult and challenging question to answer. However, scientists have realized that if we re-examine some of the 'mathematical aspects' of our thinking process and 'hardware aspects' of 'the neurons'—the principle element of the brain—we may succeed to some extent in the emulation process.

Biological neuronal processes are enormously complex, and the progress made in the understanding of the field through experimental observations is limited and crude. Nevertheless, it is true that this limited understanding of the biological processes has provided a tremendous impetus to the emulation of certain human learning behaviors, through the fields of mathematics and systems science. In neuronal information processing, there are a variety of complex mathematical operations and mapping functions involved, that, in synergism, act as a parallel-cascade computing structure. As system scientists, our objective is that, based upon this limited understanding of the brain, we create an intelligent cognitive system that can aid humans in various decision-making tasks. New computing theories under the category of *neural networks*, have been evolving. Hopefully, these new computing methods with the neural network architecture as the basis, will be able to provide a *thinking machine*—a low-level cognitive machine for which the scientists have been striving for so long.

The cognitive functions of the brain, unlike the computation functions of the computer, are based upon *relative grades* of information acquired by the neural sensory systems. The conventional mathematical tools, whether deterministic or probabilistic, are based upon some absolute measure of information. Our natural sensors acquire information in the form of relative grades rather than in absolute numbers. The 'perceptions' and 'actions' of the cognitive process also appear in the form of relative grades. The *theory of fuzzy logic,* which is based upon the notion of graded membership, provides mathematical power for the emulation of the higher-order cognitive functions—the thought and perception process. A marriage between the two evolving disciplines—neural networks and fuzzy logic—may provide a tremendous impetus to the theory for the important field of cognitive information.

The subject of intelligent systems is in an exciting state of research and we believe that we are slowly progressing towards the development of truly intelligent systems. The present-day versions of intelligent systems are not truly intelligent; however, the loose usage of the term 'intelligent' acts as a reminder that we have a long way to go.

11.2 INTRODUCTION TO SOFT COMPUTING AND INTELLIGENT CONTROL SYSTEMS

Complex dynamic systems have been integral and critical components of modern society. The unprecedented rate at which computers, networks, and other technologies are being developed, ensures that our dependence on such systems will continue to increase. While advances in science and technology have enabled us to design and build complex systems, comprehensive understanding of how to control and optimize them is clearly lacking. The mere existence of complex systems does not necessarily mean that they are operating under the most desirable conditions with enough robustness to withstand the limits of disturbances that inevitably arise.

Many problem-solving nonlinear control structures (refer to Chapter 10) have been developed over the past forty years—Feedback Linearization, Model-Reference Adaptive Control, Self-Tuning Control, Generalized Model Predictive Control, Sliding Mode Control, etc. These structures fall short of the requirements of modern complex systems. While extensions and modifications to these *conventional* architectures continues to be popular, other approaches are being explored as well.

The conventional methods of control design use mathematical models derived by the application of physical laws. The goal of mathematical modeling is to provide a set of equations that purports to describe interrelations between the system quantities as well as the relations between these quantities and external inputs. We can use different types of equations and their combinations, like algebraic, differential (ordinary/partial), difference, integral, or functional equations. A mathematical model can be viewed as a mathematical representation of the significant relevant aspects of a physical system (significance and relevance being in relation to an application where the model is to be used).

Whenever devising algebraic, differential, difference equations (or any other model from application of physical laws) is feasible, using a reasonable number of equations that can solve the given problem in a reasonable time, at a reasonable cost, and with reasonable accuracy, there is no need to look for an alternative. Today, however, there are a large number of instances in diverse fields, including control systems, wherein at least one of these criteria is not satisfied; one, therefore seeks other avenues to solve the given problem.

Since the inception of the notion of fuzzy logic in 1965, we started thinking about the *quantitative* and *qualitative* aspects of control mechanisms, and introduced the notion of *intelligent control systems*. This logic is capable of emulating certain functional elements of human intelligence. In partnership with other mathematical tools such as neural networks, the field of fuzzy logic is responsible for creation of a new field—the field of *soft computing*. In this decade, the field of soft computing has become a new emerging discipline in providing solutions to complex industrial and management problems—problems that are deeply surrounded by both qualitative and quantitative uncertainties. The elements of this emerging field provide some mathematical strength in the emulation of human-like intelligence and in the creation of systems that we call *intelligent systems*.

The conventional field of control is based on the traditional mathematical concepts. The mathematics through which we develop scientific and engineering techniques, is based upon some precise, quantitative aspects and rigorous concepts. Such quantitative aspects and rigorous concepts are beautiful, but they fail to formulate the imprecise and qualitative nature of our cognitive behavior—*the intelligence*.

What is the character of human intelligence? Is it precise, quantitative, rigorous, and computational? The answer is negative. We are very bad at calculations or any kind of computing. A negligible percentage of human beings can multiply two three-digit numbers in their heads. The basic function of human intelligence is to ensure survival in nature, not to perform precise calculations. The human brain can process millions of visual, acoustic, olfactory (concerned with smelling), tactile (the sense of touch), and motor data, and it shows astonishing abilities to learn from experience, generalize from learned rules, recognize patterns and make decisions. We want to transfer some of the human mental faculties of learning, generalizing, memorizing, and predicting into our models, algorithms, smart machines and intelligent artificial systems, in order to enable them to survive in highly technological environment, that is, to solve given tasks based on previous experience with reasonable accuracy, at reasonable cost, and in a reasonable amount of time.

The basic premises of soft computing are as follows:

- The real world is pervasively imprecise and uncertain.
- The precision and certainty carry a cost.

The guiding principle of soft computing, which follows from these premises is as follows:

- Exploit tolerance for imprecision, uncertainty, and partial truth, to achieve tractability, robustness, and low solution costs.

The guiding principle of soft computing differs strongly from that of classical hard computing which requires precision, certainty, and rigor. Many contemporary problems do not lend themselves to precise solutions within the framework of classical hard computing; for instance, recognition problems (handwriting, speech, objects, and images), computer graphics, mobile robot coordination, and data compression. To be able to deal with such problems, there is often no choice but to accept solutions that are suboptimal and inexact. In addition, even when precise solutions can be obtained, their cost is generally much higher than that of solutions which are imprecise and yet yield results within the range of acceptability.

Soft computing is not a single methodology, it is an evolving collection of methodologies for the representation of ambiguity in human thinking. The core methodologies of soft computing are: *fuzzy logic*, *neural networks*, and *evolutionary computation*. These methodologies have their strengths and weaknesses. For example, fuzzy logic is most effective when human solution is available. In this context, fuzzy logic is employed as a programming language that serves to translate a human solution into the language of fuzzy IF-THEN rules. Neural networks do not require the availability of a human solution, but can be trained by exemplification. The primary contribution of evolutionary computation, which is inspired by genetic evolution in humans and animals, is algorithms for systematized random search for obtaining the best possible solution in a huge solution space. Evolutionary algorithms are a class of global optimization techniques.

As real-life problems become more varied and more complex, we find that no single soft-computing methodology suffices to deal with them. To conceive, design, analyze, and use intelligent systems, we frequently have to employ the totality of soft computing tools that are available. The constituent methodologies in soft computing are, for the most part, complementary and synergistic rather than competitive. What this means is that in many applications, it is advantageous to employ the constituent methodologies in combination rather than in a stand-alone mode. In Chapters 11–14, we will employ soft computing methodologies—in stand-alone and hybrid modes—to obtain solutions to control problems, called intelligent control systems.

Some other general terms used in the literature with reference to intelligent systems are as follows. Soft computing is serving as the foundation for the emerging field of *computational intelligence* (the field is sometimes referred to as *machine intelligence*). When a machine (which almost always means a computer system) improves its performance at a given task over time without reprogramming, it can be said to have learned something. *Machine learning* is the key to machine intelligence, just as human learning is the key to human intelligence.

There is a significant overlap in the fields of soft computing, computational intelligence, machine learning, and machine intelligence. The meaning of various terms can change quickly and unpredictably depending on the context in which they are used. However, the loose definitions given here will serve our purpose in this book.

11.3 BASICS OF MACHINE LEARNING

When a computer program improves its performance at a given task over time, it can be said to have learned something. We will accept *automatic performance improvement with experience at a given task* as a rough-and-ready definition of machine learning, without delving too deeply into the philosophical implications. In general, to have a well-defined learning problem, we must identify the following three components [149]:

- the sources of experience;
- the class of tasks; and
- the measures of performance to be improved.

11.3.1 The Training Experience

Attempting to incorporate human-like abilities into software solutions is not an easy task. Only recently, after an attempt to analyze an ocean of data obtained from various sensors, it became clear how complex are the problems our senses routinely solve, and how difficult it is to replicate in software even the simplest aspects of human information processing. How, for example, can one make machines 'see', where 'see' means to recognize different objects and classify them into classes. For smart machines to recognize or to make decisions, they must be *trained* first on a set of training examples. Each new smart machine (software) should be able to learn the problem in its areas of operation.

In machine learning applications, there are two major sources of training experience:

- Experimental data (examples, samples, measurements, patterns, observations—expressed in the form of numerical data).
- Structured human knowledge (experience, expertise, heuristics).

Structured human knowledge is a form of training experience that is based on the existence of a human solution to the problem. However, the mere existence of human solution in some linguistic form is not sufficient. One must be able to articulate to structure the human solution in the language of a learning machine, for example, in the form of IF-THEN rules. The key idea is that the structured human knowledge describes the operation of the process of interest from the standpoint of some (human expert) operator of the process, and captures the empirical knowledge of operation of that process that has been acquired through direct experience with the actual operation of the process.

When the experience is available directly in a raw form (numerical data), i.e., no expert is available to help the learning machine, the machine by itself is required to extract the knowledge from the numerical data. The numerical training examples typically consist of observed values of system states: $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, …, $\mathbf{x}^{(P)}$, and the response to each state: $y^{(1)}$, $y^{(2)}$, …, $y^{(P)}$. Each state $\mathbf{x}^{(p)}$ is characterized by n state variables of the system: $\mathbf{x}^{(p)}$: $\{x_1^{(p)},\ldots, x_n^{(p)}\}$; $p = 1, 2,\ldots, P$. We have assumed a scalar response.

In general, learning is most reliable when the training examples represent the distribution of examples over which the final system performance must be measured. If training experience consists of data that lies in region S of state space, then S must be fully representative of situations over which the algorithm will later be used. Current theory of machine learning rests on the crucial assumption that the distribution of training examples is identical to the distribution of unseen examples—the data the machine has not

seen during its training phase. Despite our need to make this assumption in order to obtain theoretical results, it is important to keep in mind that the assumption is often violated in practice.

11.3.2 The Class of Tasks

In the system-science framework, a set $\{\mathbf{x}^{(p)}, y^{(p)}\}$; $p = 1, ..., P$ of training data pairs typically contains the inputs $\mathbf{x}^{(p)}$ and the desired output $y^{(p)}$. The design of a particular type of learning machine depends on the type of the outcome. In control applications, for example, y is a *continuous-variable*, and in pattern recognition applications, y is a *categorical variable* (class label).

Pattern Recognition Tasks Patterns are represented by feature vectors, $\mathbf{x}^{(p)}$, in feature space (state space). The main goal is to divide the feature space into regions assigned to the classes of patterns. If a feature vector falls into a certain region, the associated pattern is assigned to the corresponding class.

Function Approximation Tasks It is a problem of interpolation: we fit a mathematical function describing a curve, so that the curve passes, as close as possible, through all of the data points.

The learning task is *inferring input-output functional dependencies from a set of training examples, in order to predict future outcomes from observed data*. The prediction of continuous variables (function approximation tasks) is known as *regression*, and the prediction of categorical variables (pattern recognition tasks) is known as *classification*. Of fundamental importance in closed-loop control applications is the regression property of learning machines. Our focus in this chapter will be on this property. It is a *supervised learning* task.

Supervised Learning

Function approximation is a *supervised learning* problem where there is an input $\mathbf{x}$, an output y, and the task is to learn the mapping from the input to the output. The approach in machine learning is that we assume a parametric model of the form: $\hat{y} = g(\mathbf{x}|\boldsymbol{\theta})$, where $g(\cdot)$ is the model and $\boldsymbol{\theta}$ are its parameters. The machine learning program optimizes the parameters, $\boldsymbol{\theta}$, such that the error is minimized, that is, our estimates, $\hat{y}$, are as close as possible to the correct values, y, given in the training set. The name 'supervised learning', refers to the dependence of the 'learner' on the 'supervisor' to select informative states, and to provide actual/observed output for each state.

Note that the supervised-learning task is to learn function $g(\mathbf{x}|\boldsymbol{\theta})$, called the *target function*; the only information available is a training data set $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1, 2, ..., P\}$. A learning algorithm that at best guarantees that the learned target function $g(\cdot)$ fits the training data well, is not our design objective. Our aim is to use the machine for predicting output values for the data beyond the training data; for the data that the machine has not seen during its training phase. The actual/observed output for the unseen data is not known, and we aim to use the prediction of the machine for decision making.

Traditional mathematical models (differential/difference equations) are based on the application of physical laws, and employ hard computing. In machine learning, on the other hand, analytical models (target functions $g(\cdot)$) are based on direct empirical experience, and employ soft computing. Naturally, if the physics of the problem is well understood and a traditional mathematical model is feasible, one need not resort to machine learning methods.

Lacking information on the physics of the problem, our assumption is that the best 'model' for prediction is the model that is *induced* by the observed training data. *Inductive learning* methods formulate a model based on soft-computing methodologies by finding empirical regularities over the training examples, and these regularities induce the approximation of the target function well over other unseen examples. The inductive learning hypothesis is as follows:

Any model found to approximate the target function well over a sufficiently large set of training examples, will also approximate the target function well over other unobserved examples. It generalizes from the specific training examples, hypothesizing a general function that covers these examples and other cases beyond the training examples.

Soft-computing methodologies provide many alternative structures for realizing the target function $g(\cdot)$. The two most commonly used structures are neural networks and fuzzy logic.

Unsupervised Learning

Another machine learning application is concerned with *unsupervised learning*. In supervised learning, the aim is to learn a mapping from the input to an output whose correct values are provided by a supervisor. In unsupervised learning, there is no such supervisor and we only have input data. The goal is to unravel the underlying similarities, and *cluster* 'similar' input vectors together. A major issue in unsupervised learning is that of defining 'similarity' between two input vectors and choosing an appropriate measure for it.

Reinforcement Learning

In some applications, the output of the system is a *sequence of actions*. In such cases, a single action is not important; what is important is the *policy* —the sequence of correct actions to reach the goal. There is no such thing as the best action in any intermediate state; an action is good if it is part of a good policy. In such a case, the machine learning program should be able to assess the goodness of policies and learn from past good action sequences to be able to generate a policy. Such learning methods are called *reinforcement learning* algorithms.

Reinforcement learning is an on-line learning procedure that *rewards* an action for its *good* output result and *punishes* it for a *bad* output result. The evaluation of an output as *good* or *bad* depends on the specific problem and the environment. For a control system, if the system continues to be in the desired region in state space after an action, the output is judged as *good*, otherwise it is considered as *bad*. The reward/penalty of an action is the *reinforcement signal*.

Evolution versus Learning

The adaptation of creatures to their environments results from the interaction of two processes, namely, evolution and learning. Evolution is a slow stochastic process at the population level that determines the basic structures of a species. Evolution operates on biological entities, rather than on individuals themselves. At the other end, learning is a process of gradually improving an individual's adaptation capability to its environment by tuning the structure of the individual.

Evolution is based on the Darwinian model, also called the *principle of natural selection*, or *survival of the fittest*, while learning is based on the human cognitive faculties. Evolutionary algorithms are stochastic search methods that employ a search technique based on the Darwinian model, whereas neural networks and fuzzy systems are learning methods based on human learning model.

Combinations of learning and evolution, embodied by evolving neural networks and evolving fuzzy systems, have better adaptability to a dynamic environment.

11.3.3 The Performance Measures

The design of a learning machine for optimal performance requires careful consideration of several factors that influence the machine's performance. Performance is not just measured as the *accuracy* achieved by the machine, but aspects such as *computational complexity* and *convergence characteristics* are just as important.

In the following we present performance measures for a learning machine under three subheadings: accuracy, computational complexity, and convergence.

Accuracy

The accuracy of a learning machine is dependent on its generalization capability.

Generalization The aim of machine learning is rarely to replicate the training data but the prediction for new cases. That is, we would like to be able to generate the right output for an input outside the training set; one for which the correct output is not known but is to be predicted for decision making. How well a model trained on the training set predicts the right output for unseen examples in operational situation, is called *generalization*.

We assume that all the data (training data + new data in operational situation) are generated independently from some unknown (but fixed) probability distribution $\Omega(\mathbf{x}, y)$. This is a standard assumption in learning theory; data generated this way is commonly referred to as *iid* (independent and identically distributed). Our goal is to find a function $g(\cdot)$ that will generalize *well* to unseen examples, that is, $g(\mathbf{x}) = y$ for examples $(\mathbf{x}, y)$ other than the training examples, generated from $\Omega(\mathbf{x}, y)$.

Generalization is a very important aspect of machine learning. Since it is a measure of how well the machine interpolates to points not used during training, the ultimate objective of machine learning is to produce a learner with low *generalization error*, that is to minimize the *true risk* function

$$E_G(\Omega, \boldsymbol{\theta}) = \int (g(\mathbf{x}|\boldsymbol{\theta}) - y)^2 \, d\Omega(\mathbf{x}, y) \tag{11.1}$$

where $\boldsymbol{\theta}$ are adjustable parameters of the learning machine model.

Since $\Omega(\mathbf{x}, y)$ is generally not known, $\boldsymbol{\theta}$ are found through minimization of the *empirical risk* function

$$E_T(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{P} \sum_{p=1}^{P} (g(\mathbf{x}^{(p)}|\boldsymbol{\theta}) - y^{(p)})^2 \tag{11.2}$$

over a finite training data set

$$\mathcal{D} : \{\mathbf{x}^{(p)}, y^{(p)}; p = 1, 2, ..., P\} \sim \Omega(\mathbf{x}, y) \tag{11.3}$$

When $P \to \infty$, then *empirical error* $E_T \to$ generalization error E_G.

The aim of machine learning is, therefore, to learn the examples in the training set well, while providing good generalization to examples not included in the training set. It is hoped that a small empirical (training) error will also give a small true (generalization) error.

Validation We can measure the generalization ability of a machine if we have access to data outside the training set. We simulate this by dividing, often randomly, the training set we have into two parts. We use one part for training (i.e., for building a learning machine) and the other part, called the *test set/validation set*, is used for testing the generalization ability. In the validation set, for each input $\mathbf{x}$, the output y is known, but these pairs of data are unknown to the machine, since they have not been used during training. The inputs from the validation set are given to the trained machine. The machine outputs predictions , $\hat{y}$, which are then compared with the actual values, y ; the empirical error is then calculated as a measure of generalization capability of the machine. Assuming large enough training and test sets, the machine that is the most accurate on the test set is the best. After training and testing, the machine is ready for use with the learned parameters 'frozen'. The machine with low empirical error is expected to give reasonable outputs for the data it has not seen before. Research shows a dependence of generalization error on the size of the training set, the machine architecture, and the number of free parameters in the machine model.

Overfitting The learning machine design aims at 100% accuracy in predicting the training examples. While this is sometimes a reasonable design strategy, in fact it can lead to difficulties when there is noise in the training data, or the number of training examples is too small to produce a representative sample of $\Omega(\mathbf{x}, y)$. In either of these cases, this design approach can produce a machine that *overfits* the training examples. We will say that *a machine overfits the training examples if some other machine that fits the training examples less well actually performs better on the test data.*

Overfitting of a training set means that the machine memorizes the training examples, and consequently loses the ability to generalize. That is, machines that overfit cannot predict correct output for data patterns not seen during training. Overfitting occurs when machine architecture is too complex (a neural network with large number of weights, a fuzzy logic model with large number of rules, etc.), compared to the complexity of the function underlying the data. If we have a machine model that is too complex, the data is insufficient to constrain it and we may end up with bad prediction function. Or if there is noise in the data, an over complex model may learn not only the underlying function but also the noise in the data and may make it a bad fit. This is called overfitting. In such a case, having more training data helps but only up to certain point.

Figure 11.1 illustrates the impact of overfitting in a typical application of machine learning. The horizontal axis of the plot indicates the complexity of the machine. The vertical axis indicates the accuracy of predictions made by the machine. The solid line shows the accuracy of the machine over the training examples, where the broken line indicates the accuracy measured over the test examples. Predictably, the accuracy of the machine over the training examples increases monotonically as the machine grows in complexity. However, the accuracy over the test examples first increases then decreases.

If the machine is trained for too long, the excess free parameters start to memorize all the training patterns, including the noise contained in the training set. Figure 11.2 presents an illustration of training and generalization errors as a function of training time. From the start of the training, both the training and generalization errors decrease— usually exponentially. In the case of oversized machines, there is a

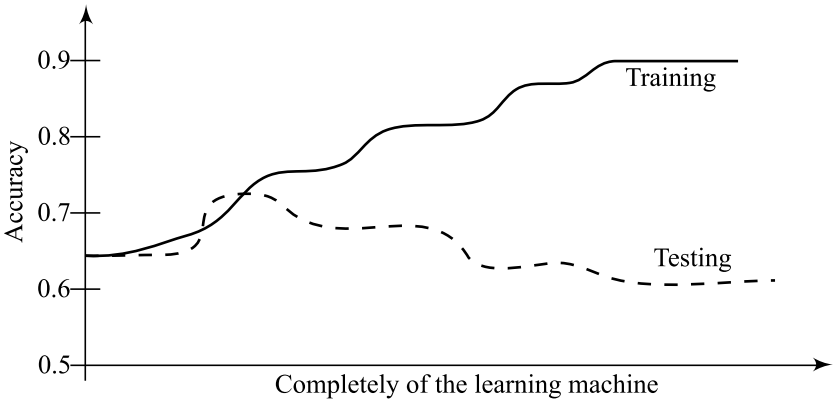


Fig. 11.1 Impact of machine complexity

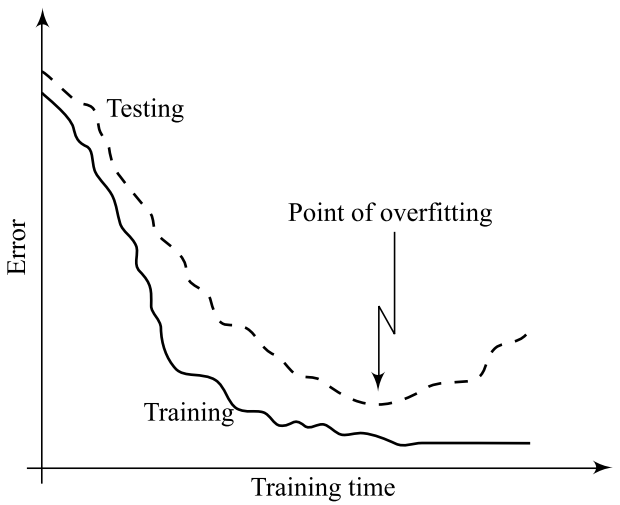


Fig. 11.2 Impact of training time

point at which the training error continuous to decrease, while the generalization error starts to increase. This is the point of overfitting. The training should stop as soon as an increase in generalization error is observed.

Machine learning tasks based on real-world data are unlikely to find the noise-free data assumption tenable. Also $\Omega(\mathbf{x}, y)$ is generally unknown; empirical evidence shows that the available finite amount of data is insufficient to represent the distribution of total data in operational situations. Therefore, whenever the prediction comes from inductive learning, it will not, in general, be provably correct. The question is how to improve the generalization performance. A great deal of research has gone into clever engineering tricks and heuristics to aid in the design of learning machines which will not overfit on a given data set, consequently giving a better generalization performance.

Computational Complexity

The computational complexity of a learning machine is directly influenced by the following design choices:

1. *The machine architecture*

The learning machine architecture (soft computing methodologies) is an evolving collection of representations of the target function in the learning task. Some of the important architectures we will be exploring in the book are

- neural networks;
- fuzzy logic models; and
- kernel functions (support vector machines).

These architectures are all competitive for a given learning task. Computational complexity of these models are varied, but we have to balance complexity with accuracy. The more complex models of these architectures usually yield better accuracy, but only up to a point; a trade-off is thus required in the selection of architecture.

2. *The size of free parameters*

The larger the size of the parameter vector $\boldsymbol{\theta}$ of a model, the more calculations are needed to predict outputs after training, and the more learning calculations are needed for training-patterns presentation.

3. *The training set size*

The larger the training set size, the more patterns are presented for training. Therefore, the total number of learning calculations increases.

4. *Complexity of optimization method*

As will be discussed later in this book, sophisticated optimization algorithms have been developed to obtain optimum values of machine model parameters. Optimization improves accuracy. This sophistication comes, however, at the cost of increased computational complexity.

An acceptable trade-off between computational complexity and accuracy is a very important issue in the design of learning systems.

Convergence

Convergence characteristics of a learning machine can be described by the ability of the machine to converge to specified error levels (usually considering the generalization error). While convergence analysis is an empirical approach, rigorous theoretical analysis has been done for some learning machine architectures.

11.4 A BRIEF HISTORY OF NEURAL NETWORKS

Historically, research in artificial neural networks was inspired by the desire to produce artificial systems capable of sophisticated 'intelligent' processing similar to the human brain. The science of artificial

neural networks made its first significant appearance in 1943 when Warren McCulloch and Walter Pitts published their study in this field. They suggested a simple neuron model (known today as *MP artificial neural model*) and implemented it as an electrical circuit. In 1949, Donald Hebb highlighted the connection between psychology and physiology, pointing out that a neural pathway is reinforced each time it is used. *Hebb's learning rule*, as it is sometimes known, is still used and quoted today. Improvements in hardware and software in the 1950s ushered in the age of computer simulation. It became possible to test theories about nervous system functions. Research expanded; neural network terminology came into its own.

The *perceptron* is the earliest of the neural network paradigms. Frank Rosenblatt built this learning machine device in hardware in 1958 and caused quite a stir.

The perceptron has been a fundamental building block for more powerful models, such as the ADALINE (ADAptive LINear Elements) and MEDALINE (Multiple ADALINEs in parallel), developed by Bernard Widrow and Marcian Hoff in 1959. Their learning rule, sometimes known as *Widrow–Hoff rule*, was simple yet elegant.

Affected by the predominately rosy outlook of the time, some people exaggerated the potential of neural networks. Biological comparisons were blown out of proportion. In 1969, in the midst of many outrageous claims, Marvin Minsky and Seymour Papert published 'Perceptrons', an influential book condemning Rosenblatt's perceptron. The limitations of the perceptron were significant; the charge was that it could not solve any 'interesting'problems. It brought to a halt, much of the activity in neural network research.

Nevertheless, a few dedicated scientists such as Teuvo Kohonen and Stephen Grossberg, continued their efforts. In 1982, John Hopfield introduced a recurrent-type neural network that was based on the interaction of neurons through a feedback mechanism. His approach was based on Hebb's learning rule. The back-propagation learning rule arrived on the neural-network scene at approximately the same time from several independent sources (Werbos; Parker; Rumelhart, Hinton and Williams). Essentially, a refinement of the Widrow–Hoff learning rule, the backpropagation learning rule provided a systematic means for training multilayer networks, thereby overcoming the limitations presented by Minsky. Minsky's appraisal has proven excessively pessimistic; networks now routinely solve many of the problems that he posed in his book.

Research in the 1980s triggered the present boom in the scientific community. New and better models are being proposed, and the limitations of some of the 'old' models are being chipped away. A number of today's technological problems are in areas where neural-network technology has demonstrated potential: speech processing, image processing and pattern recognition, time-series prediction, real-time control and others.

As the research on neural networks is evolving, more and more types of networks are being introduced, while still less emphasis is being placed on the connection to the biological neural network. In fact, the neural networks that are most popular today have very little resemblance to the brain, and one might argue that it would be more fair to regard them simply as a discipline under statistics.

The application of artificial neural networks in closed-loop control, has recently been rigorously studied. One property of these networks, central to most control applications, is that of function approximation. Such networks can generate input/output maps which can approximate any continuous function with the required degree of accuracy. This emerging technology has given us control design techniques that do

not depend on parametrized mathematical models. Neural networks are used to estimate the unknown nonlinear functions; the controller formulation uses these estimated results.

When neural networks are used for control of systems, it is important that results and claims are based on firm analytical foundations. This is especially important when these control systems are to be used in areas where the cost of failure is very high. For example, when human life is threatened, as in aircrafts, nuclear plants, etc. It is also true that without a good theoretical framework, it is unlikely that the research in the discipline will progress very far, as intuitive invention and tricks cannot be counted on to provide good solutions to controlling complex systems under a high degree of uncertainty. Strong theoretical results guaranteeing control system properties such as stability are still to come, although promising results have been reported recently of progress in special cases. The potential of neural networks in control systems clearly needs to be further explored and both, theory and applications, need to be further developed.

The rest of the chapter gives a gentle introduction to the application of neural networks in control systems. A single chapter can in no way do justice to the multitude of interesting neural network results, that have appeared in literature. Not only would space be required, but in the time required to detail current results, new results would certainly arise. Instead of trying to cover a large spectrum of such a vast field, we will focus on what is generally regarded as the core of the subject. This chapter is meant to be a stepping-stone that could lead interested readers on to other books for additional information on the current status, and future trends of the subject.

11.5 NEURON MODELS

A discussion of anthropomorphism to introduce neural network technology may be worthwhile—as it helps explain the terminology of neural networks. However, anthropomorphism can lead to misunderstanding when the metaphor is carried too far. We give here a brief description of how the brain works; a lot of details of the complex electrical and chemical processes that go on in the brain, have been ignored. A pragmatic justification for such a simplification is that by starting with a simple model of the brain, scientists have been able to achieve very useful results.

11.5.1 Biological Neuron

To the extent a human brain is understood today, it seems to operate as follows: bundles of neurons, or nerve fibers, form nerve structures. There are many different types of neurons in the nerve structure, each having a particular shape, size and length depending upon its function and utility in the nervous system. While each type of neuron has its own unique features needed for specific purposes, all neurons have two important structural components in common. These may be seen in the typical biological neuron shown in Fig. 11.3. At one end of the neuron are a multitude of tiny, filament-like appendages called *dendrites*, which come together to form larger branches and trunks where they attach to *soma*, the body of the nerve cell. At the other end of the neuron is a single filament leading out of the soma, called an *axon*, which has extensive branching on its far end. These two structures have special electrophysiological properties which are basic to the function of neurons as *information processors*, as we shall see next.

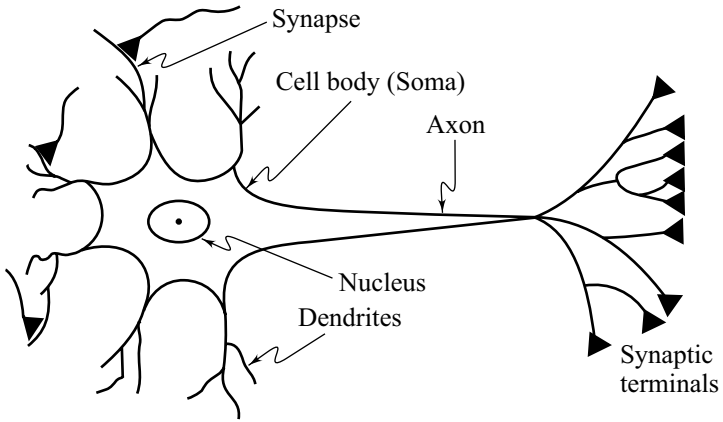


Fig. 11.3 A typical biological neuron

Neurons are connected to each other via their axons and dendrites. Signals are sent through the axon of one neuron to the dendrites of other neurons. Hence dendrites may be represented as the inputs to the neuron, and the axon as its output. Note that each neuron has many inputs through its multiple dendrites, whereas it has only one output through its single axon. The axon of each neuron forms connections with the dendrites of many other neurons, with each branch of the axon meeting exactly one dendrite of another cell at what is called a *synapse*. Actually, the axon terminals do not quite touch the dendrites of the other neurons, but are separated by a very small distance of between 50 and 200 angstroms. This separation is called the *synaptic gap*.

A conventional computer is typically a single processor acting on explicitly programmed instructions. Programmers break tasks into tiny components, to be performed in sequence rapidly. On the other hand, the brain is composed of ten billion or so neurons. Each nerve cell can interact directly with up to 200,000 other neurons (though 1000 to 10,000 is typical). In place of explicit rules that are used by a conventional computer, it is the pattern of connections between the neurons, in the human brain, that seems to embody the 'knowledge' required for carrying out various information-processing tasks. In human brain, there is no equivalent of a CPU that is in overall control of the actions of all the neurons.

The brain is organized into different regions, each responsible for different functions. The largest parts of the brain are the cerebral hemispheres, which occupy most of the interior of the skull. They are layered structures; the most complex being the outer layer, known as the *cerebral cortex*, where the nerve cells are extremely densely packed to allow greater interconnectivity. Interaction with the environment is through the visual, auditory and motion control (muscles and glands) parts of the cortex.

In essence, neurons are tiny electrophysiological information-processing units which communicate with each other through electrical signals. The synaptic activity produces a voltage pulse on the dendrite which is then conducted into the soma. Each dendrite may have many synapses acting on it, allowing massive interconnectivity to be achieved. In the soma, the dendrite potentials are added. Note that neurons are able to perform more complex functions than simple addition on the inputs they receive, but considering a simple summation is a reasonable approximation.

When the soma potential rises above a critical threshold, the axon will fire an electrical signal. This sudden burst of electrical energy along the axon is called axon potential and has the form of an electrical

impulse or spike that lasts about 1 msec. The magnitude of the axon potential is constant and is not related to the electrical stimulus (soma potential). However, neurons typically respond to a stimulus by firing not just one but a barrage of successive axon potentials. What varies is the frequency of axonal activity. Neurons can fire between 0 to 1500 times per second. Thus, information is encoded in the nerve signals as the instantaneous frequency of axon potentials and the mean frequency of the signal.

A synapse couples the axon with the dendrite of another cell. The synapse releases chemicals called *neurotransmitters,* when its potential is raised sufficiently by the axon potential. It may take the arrival of more than one spike before the synapse is triggered. The neurotransmitters that are released by the synapse diffuse across the gap and chemically activate gates on the dendrites, which, when open, allow charged ions to flow. It is this flow of ions that alters the dendritic potential and provides voltage pulse on the dendrite, which is then conducted into the neighboring neuron body. At the synaptic junction, the number of gates that open on the dendrite depends upon the number of neurotransmitters released. It also appears that some synapses excite the dendrites they affect, whilst others serve to inhibit it. This corresponds to altering the local potential of the dendrite in a positive or negative direction.

Synaptic junctions alter the effectiveness with which the signal is transmitted; some synapses are good junctions and pass a large signal across, whilst others are very poor, and allow very little through.

Essentially, each neuron receives signals from a large number of other neurons. These are the inputs to the neuron which are 'weighted'. That is, some signals are stronger than others. Some signals excite (are positive), and others inhibit (are negative). The effects of all weighted inputs are summed. If the sum is equal to or greater than the *threshold* for the neuron, the neuron *fires* (gives output). This is an 'all-or-nothing' situation. Because the neuron either fires or doesn't fire, the rate of firing, not the amplitude, conveys the magnitude of information.

The ease of transmission of signals is altered by activity in the nervous system. The neural pathway between two neurons is susceptible to fatigue, oxygen deficiency, and agents like anesthetics. These events create a resistance to the passage of impulses. Other events may increase the rate of firing. This ability to adjust signals is a mechanism for *learning*.

After carrying a pulse, an axon fiber is in a state of complete non-excitability for a certain time called the refractory period. For this time interval, the nerve does not conduct any signals, regardless of the intensity of excitation. Thus, we may divide the time scale into consecutive intervals, each equal to the length of the refractory period. This will enable a discrete-time description of the neurons' performance in terms of their states at discrete-time instances.

11.5.2 Artificial Neuron

Artificial neurons bear only a modest resemblance to real things. They model approximately three of the processes that biological neurons perform (there are at least 150 processes performed by neurons in the human brain).

An artificial neuron

(i) evaluates the input signals, determining the strength of each one;
(ii) calculates a total for the combined input signals and compares that total to some threshold level; and
(iii) determines what the output should be.

Inputs and Outputs

Just as there are many inputs (stimulation levels) to a biological neuron, there should be many input signals to our artificial neuron (AN). All of them should come to our AN simultaneously. In response, a biological neuron either 'fires' or 'doesn't fire' depending upon some *threshold* level. Our AN will be allowed a single output signal, just as is present in a biological neuron: many inputs, one output (Fig. 11.4).

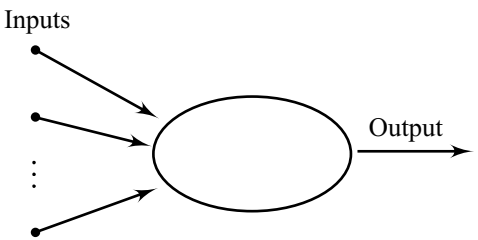


Fig. 11.4 Many inputs, one output model of a neuron

Weighting Factors

Each input will be given a relative weighting, which will affect the impact of that input (Fig. 11.5). This is something like varying synaptic strengths of the biological neurons—some inputs are more important than others in the way they combine to produce an impulse. Weights are adaptive coefficients within the network, that determine the intensity of the input signal. In fact, this adaptability of connection strength is precisely what provides neural networks their ability to learn and store information, and, consequently, is an essential element of all neuron models.

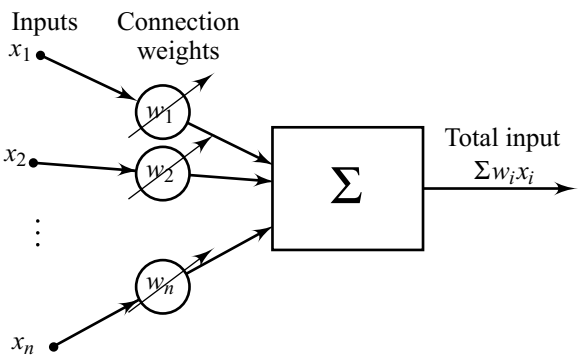


Fig. 11.5 A neuron with weighted inputs

Excitatory and inhibitory inputs are represented simply by positive or negative connection weights, respectively. Positive inputs promote the firing of the neuron, while negative inputs tend to keep the neuron from firing.

Mathematically, we could look at the inputs and the weights on the inputs as vectors.

The *input vector*

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{11.4a}$$

and the connection *weight vector*

$$\mathbf{w}^T = [w_1 \quad w_2 \cdots w_n] \tag{11.4b}$$

The total input signal is the product of these vectors. The result is a scalar

$$\sum_{i=1}^{n} w_i x_i = \mathbf{w}^T \mathbf{x} \tag{11.4c}$$

Activation Functions

Although most neuron models sum their input signals in basically the same manner, as described above, they are not all identical in terms of how they produce an output response from this input. Artificial neurons use an *activation function*, often called a *transfer function*, to compute their activation as a function of total input stimulus. Several different functions may be used as activation functions, and, in fact, the most distinguishing feature between existing neuron models is precisely which transfer function they employ.

We will, shortly, take a closer look at the activation functions. We first build a neuron model, assuming that the transfer function has a threshold behavior, which is, in fact, the type of response exhibited by biological neurons: when the total stimulus exceeds a certain threshold value θ, a constant output is produced, while no output is generated for input levels below the threshold. Figure 11.6a shows this neuron model. In this diagram, the neuron has been represented in such a way that the correspondence of each element with its biological counterpart may be easily seen.

Equivalently, the threshold value can be subtracted from the weighted sum and the resulting value compared to zero; if the result is positive, then output a 1, else output a 0. This is shown in Fig. 11.6b; note that the shape of the function is the same but now the jump occurs at zero. The threshold effectively adds an offset to the weighted sum.

An alternative way of achieving the same effect is to take the threshold out of the body of the model neuron, and connect it to an extra input value that is fixed to be 'on' all the time. In this case, rather than subtracting the threshold value from the weighted sum, the extra input of +1 is multiplied by a weight and added in a manner similar to other inputs—this is known as *biasing* the neuron. Figure 11.6c shows a neuron model with a bias term. Note that we have taken constant input '1' with an adaptive weight 'b' in our model.

The first formal definition of a synthetic neuron model, based on the highly simplified considerations of the biological neuron, was formulated by McCulloch and Pitts (1943). The two-port model (inputs-activation value-output mapping) of Fig. 11.6 is essentially the MP neuron model. It is important to look at the features of this unit—which is an important and popular neural network building block.

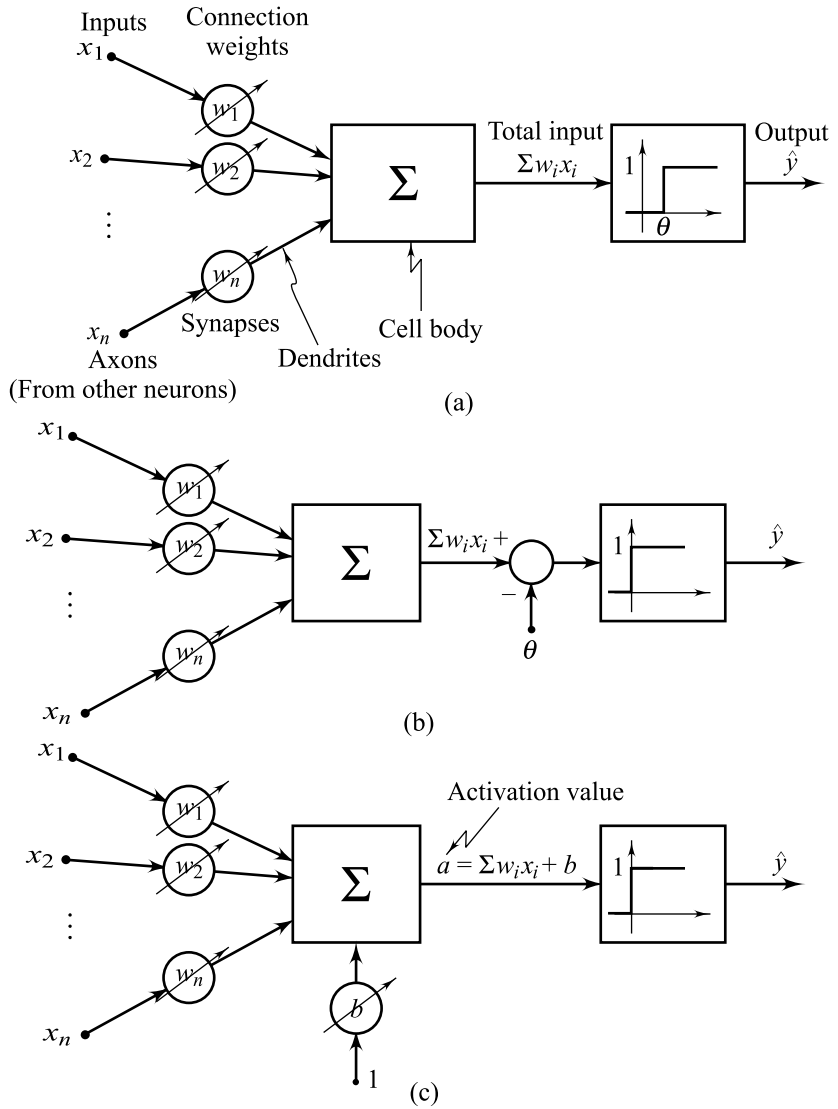


Fig. 11.6 The MP neuron model

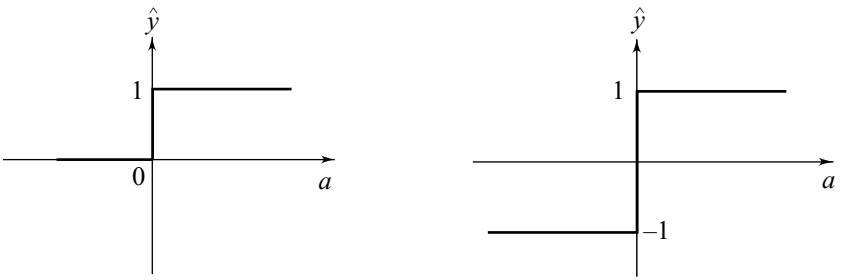
It is a simple unit, thresholding a weighted sum of its inputs to get an output. It specifically does not take any account of the complex patterns and timings of the actual nervous activity in real neural systems, nor does it have any of the complicated features found in the body of biological neurons. This ensures its status as a *model*, and not a *copy* of a real neuron.

The MP artificial neuron model involves two important processes:

(i) Forming net activation by combining inputs. The input values are amalgamated by a weighted additive process to achieve the neuron activation value a (refer to Fig. 11.6c).

(ii) Mapping this activation value a into the neuron output $\hat{y}$. This mapping from activation to output may be characterized by an 'activation' or 'squashing' function.

For the activation functions that implement input-to-output compression or squashing, the range of the function is less than that of the domain. There is some physical basis for this desirable characteristic. Recall that in a biological neuron, there is a limited range of output (spiking frequencies). In the MP model, where DC levels replace frequencies, the squashing function serves to limit the output range. The squashing function shown in Fig. 11.7a limits the output values to {0, 1}, while that in Fig. 11.7b limits the output value to {–1, 1}. The activation function of Fig. 11.7a is called *unipolar*, while that in Fig. 11.7b is called *bipolar* (both positive and negative responses of neurons are produced).



(a) Unipolar squashing function (b) Bipolar squashing function

Fig. 11.7

11.5.3 Mathematical Model

From the above discussion, it is evident that the artificial neuron is really nothing more than a simple mathematical equation, for calculating an output value from a set of input values. From now onwards, we will be more on a mathematical footing; the reference to biological similarities will be reduced. Therefore, names like a *processing element*, a *unit*, a *node*, a *cell*, etc., may be used for the neuron. A neuron model (a processing element/a unit/a node/a cell of our neural network), will be represented as follows:

The input vector

$$\mathbf{x} = [x_1 \quad x_2 \cdots x_n]^T;$$

the connection weight vector

$$\mathbf{w}^T = [w_1 \quad w_2 \cdots w_n];$$

the unity-input weight b (bias term), and the output $\hat{y}$ of the neuron are related by the following equation:

$$\hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{11.5}$$

where $\sigma(\cdot)$ is the activation function (transfer function) of the neuron.

The weights are always adaptive. We can simplify our diagram as in Fig. 11.8a; adaptation need not be specifically shown in the diagram.

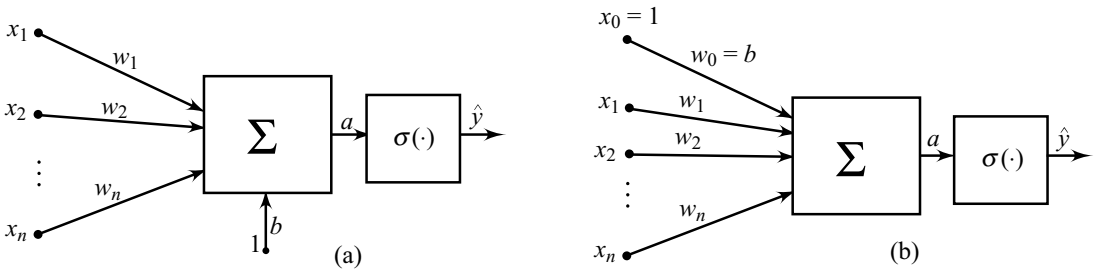


Fig. 11.8 Mathematical model of a neuron (perceptron)

The bias term may be absorbed in the input vector itself as shown in Fig. 11.8b.

$$\hat{y} = \sigma(a)$$

$$= \sigma\left(\sum_{i=0}^{n} w_i x_i\right); \; w_0 = b, \; x_0 = 1 \tag{11.6a}$$

$$= \sigma\left(\sum_{i=1}^{n} w_i x_i + w_0\right) = \sigma\left(\mathbf{w}^T \mathbf{x} + w_0\right) \tag{11.6b}$$

In the literature, this model of an artificial neuron is also referred to as a *perceptron* (the name was given by Rosenblatt in 1958).

The expressions for the neuron output $\hat{y}$ are referred to as the *cell recall mechanism*. They describe how the output is reconstructed from the input signals and the values of the cell parameters.

The artificial neural systems under investigation and experimentation today, employ a variety of activation functions that have more diversified features than the one presented in Fig. 11.7. Below, we introduce the main activation functions that will be used later in this chapter.

The MP neuron model shown in Fig. 11.6 used the *hard-limiting activation function*. When artificial neurons are cascaded together in layers (discussed in the next section), it is more common to use a *soft-limiting activation function*. Figure 11.9a shows a possible bipolar soft-limiting semilinear activation

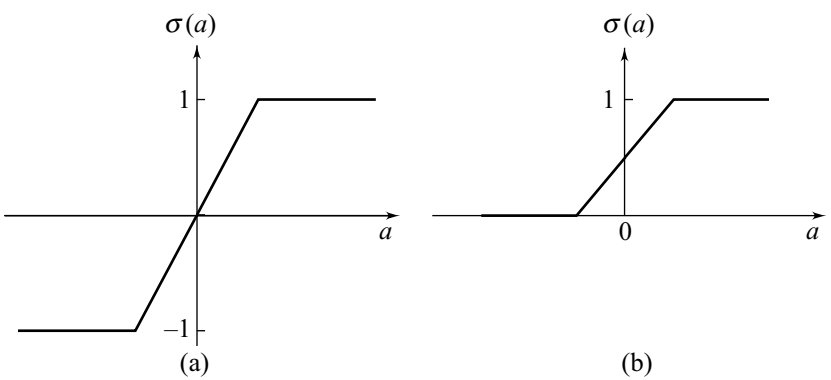


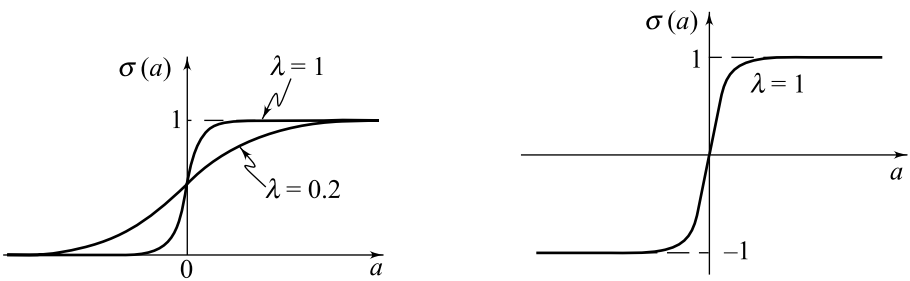
Fig. 11.9 Soft-limiting activation functions

function. This function is, more or less, the ON-OFF type, as before, but has a sloping region in the middle. With this smooth thresholding function, the value of the output will be practically 1 if the weighted sum exceeds the threshold by a huge margin and, conversely, it will be practically –1 if the weighted sum is much less than the threshold value. However, if the threshold and the weighted sum are almost the same, the output from the neuron will have a value somewhere between the two extremes. This means that the output from the neuron can be related to its inputs in a more useful and informative way. Figure 11.9b shows a unipolar soft-limiting function.

For many training algorithms (discussed in later sections), the derivative of the activation function is needed; therefore, the activation function selected must be differentiable. The *logistic* or *sigmoid* function, which satisfies this requirement, is the most commonly used soft-limiting activation function. The sigmoid function (Fig. 11.10a):

$$\sigma(a) = \frac{1}{1 + e^{-\lambda a}} \tag{11.7}$$

is continuous and varies monotonically from 0 to 1 as a varies from $-\infty$ to ∞. The gain of the sigmoid, λ, determines the steepness of the transition region. Note that as the gain approaches infinity, the sigmoid approaches a hard-limiting nonlinearity. One of the advantages of the sigmoid is that it is *differentiable*. This property had a significant impact historically, because it made it possible to derive a gradient search learning algorithm for networks with multiple layers (discussed in later sections).



(a) Sigmoid functions (b) Hyperbolic tangent function

Fig. 11.10

The sigmoid function is unipolar. A bipolar function with similar characteristics is a *hyperbolic tangent* (Fig. 11.10b):

$$\sigma(a) = \frac{1 - e^{-\lambda a}}{1 + e^{-\lambda a}} = \tanh\left(\tfrac{1}{2}\lambda a\right) \tag{11.8}$$

The biological basis of these activation functions can easily be established. It is known that neurons located in different parts of the nervous system have different characteristics. The neurons of the ocular motor system have a sigmoid characteristic, while those located in the visual area have a Gaussian characteristic. As we said earlier, anthropomorphism can lead to misunderstanding when the metaphor is carried too far. It is now a well-known result in neural network theory that a two-layer neural network is capable of solving any classification problem. It has also been shown that a two-layer network is capable of solving any nonlinear function approximation problem [138, 141]. This result does not require the use of sigmoid nonlinearity. The proof assumes only that nonlinearity is a continuous, smooth, monotonically

increasing function that is bounded above and below. Thus, numerous alternatives to sigmoid could be used, without a biological justification. In addition, the above result does not require that the nonlinearity be present in the second (output) layer. It is quite common to use linear output nodes since this tends to make learning easier. In other words,

$$\sigma(a) = \lambda a;\ \lambda > 0 \tag{11.9}$$

is used as an activation function in the output layer. Note that this function does not 'squash' (compress) the range of output.

Our focus in this chapter will be on two-layer perceptron networks with the first (hidden) layer having *log-sigmoid*

$$\sigma(a) = \frac{1}{1+e^{-a}} \tag{11.10a}$$

or *tan-sigmoid*

$$\sigma(a) = \frac{1-e^{-a}}{1+e^{-a}} \tag{11.10b}$$

activation function, and the second (output) layer having *linear* activation function

$$\sigma(a) = a \tag{11.11}$$

The log-sigmoid function has historically been a very popular choice, but since it is related to the tan-sigmoid by the simple transformation

$$\sigma_{\text{log-sigmoid}} = (\sigma_{\text{tan-sigmoid}} + 1)/2 \tag{11.12}$$

both of these functions are in use in neural network models.

We have so far described two classical neuron models:

- perceptron—a neuron with sigmoidal activation function (sigmoidal function is a softer version of the original perceptron's hard limiting or threshold activation function); and
- linear neuron—a neuron with linear activation function.

11.6 NETWORK ARCHITECTURES

In the biological brain, a huge number of neurons are interconnected to form the network and perform advanced intelligent activities. The artificial neural network is built by neuron models. Many different types of artificial neural networks have been proposed, just as there are many theories on how biological neural processing works. We may classify the organization of the neural networks largely into two types: a feedforward net and a recurrent net. The feedforward net has a hierarchical structure that consists of several layers, without interconnection between neurons in each layer, and signals flow from input to output layer in one direction. In the recurrent net, multiple neurons in a layer are interconnected to organize the network. In the following, we give typical characteristics of the feedforward net and the recurrent net, respectively.

11.6.1 Feedforward Networks

A feedforward network consists of a set of *input terminals* which feed the input patterns to a layer or subgroup of neurons. The layer of neurons makes independent computations on data that it receives, and passes the results to another layer. The next layer may, in turn, make its independent computations and pass on the results to yet another layer. Finally, a subgroup of one or more neurons determines the output

from the network. This last layer of the network is the *output layer*. The layers that are placed between the input terminals and the output layer are called *hidden layers*.

Some authors refer to the input terminals as the input layer of the network. We do not use that convention since we wish to avoid ambiguity. Note that each neuron in a network makes its computation based on the weighted sum of its inputs. There is one exception to this rule: the role of the 'input layer' is somewhat different as units in this layer are used only to hold input data, and to distribute the data to units in the next layer. Thus, the 'input layer' units perform no function—other than serving as a buffer, fanning out the inputs to the next layer. These units do not perform any computation on the input data, and their weights, strictly speaking, do not exist.

The network outputs are generated from the output layer units. The output layer makes the network information available to the outside world. The hidden layers are internal to the network and have no direct contact with the external environment. There may be from zero to several hidden layers. The network is said to be *fully connected* if every output from a single node is channeled to every node in the next layer.

The number of input and output nodes needed for a network will depend on the nature of the data presented to the network, and the type of the output desired from it, respectively. The number of neurons to use in a hidden layer, and the number of hidden layers required for processing a task, is less obvious. Further comments on this question will appear in a later section.

A Layer of Neurons

A one-layer network with n inputs and q neurons is shown in Fig. 11.11. In the network, each input x_i; $i = 1, 2, ..., n$ is connected to the jth neuron input through the weight w_{ji}; $j = 1, 2, ..., q$. The jth neuron has a *summer* that gathers its weighted inputs to form its own scalar output

$$\sum_{i=1}^{n} w_{ji} x_i + w_{j0}; j = 1, 2, ..., q$$

Finally, the jth neuron outputs $\hat{y}_j$ through its activation function $\sigma(\cdot)$:

$$\hat{y}_j = \sigma\left(\sum_{i=1}^{n} w_{ji}\, x_i + w_{j0}\right); j = 1, 2, ..., q \tag{11.13a}$$

$$= \sigma(\mathbf{w}_j^T \mathbf{x} + w_{j0}); j = 1, 2, ..., q \tag{11.13b}$$

where weight vector $\mathbf{w}_j$ is defined as

$$\mathbf{w}_j^T = [w_{j1} \quad w_{j2} \cdots w_{jn}] \tag{11.13c}$$

Note that it is common for the number of inputs to be different from the number of neurons (i.e., $n \neq q$). A layer is not constrained to have the number of its inputs equal to the number of its neurons.

In vector-matrix notation, the layer shown in Fig. 11.11 has $q \times 1$ output vector

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_q \end{bmatrix}, \tag{11.14a}$$

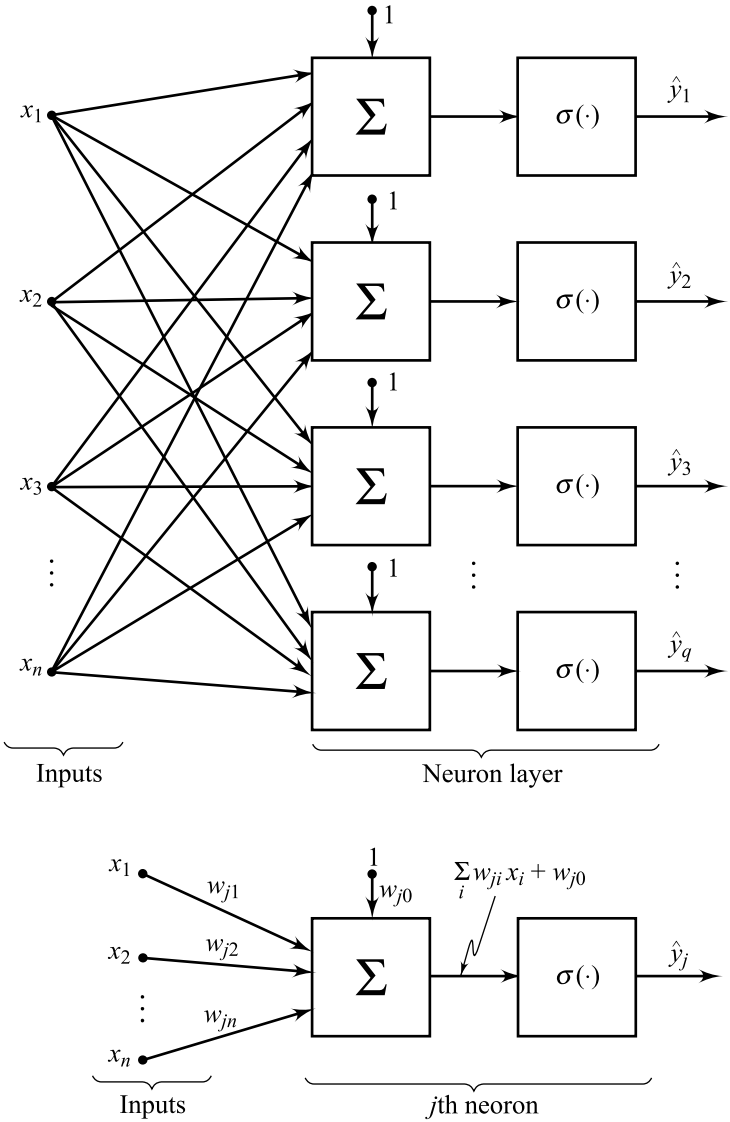


Fig. 11.11 A one-layer network

$q \times n$ weight matrix

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & & \vdots \\ w_{q1} & w_{q2} & \cdots & w_{qn} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \mathbf{w}_3^T \\ \vdots \\ \mathbf{w}_q^T \end{bmatrix} \tag{11.14b}$$

and $q \times 1$ bias vector

$$\mathbf{w}_0 = \begin{bmatrix} w_{10} \\ w_{20} \\ \vdots \\ w_{q0} \end{bmatrix} \tag{11.14c}$$

Note that the row indices on the elements of matrix $\mathbf{W}$ indicate the destination neuron for the weight, and the column indices indicate which source is the input for that weight. Thus, the indices in w_{ji} say that the signal from the ith input is connected to the jth neuron.

The activation vector is

$$\mathbf{W}\mathbf{x} + \mathbf{w}_0 = \begin{bmatrix} \mathbf{w}_1^T \mathbf{x} + w_{10} \\ \mathbf{w}_2^T \mathbf{x} + w_{20} \\ \vdots \\ \mathbf{w}_q^T \mathbf{x} + w_{q0} \end{bmatrix}$$

The outputs are

$$\begin{aligned} \hat{y}_1 &= \sigma(\mathbf{w}_1^T \mathbf{x} + w_{10}) \\ \hat{y}_2 &= \sigma(\mathbf{w}_2^T \mathbf{x} + w_{20}) \\ &\vdots \\ \hat{y}_q &= \sigma(\mathbf{w}_q^T \mathbf{x} + w_{q0}) \end{aligned}$$

Introducing the nonlinear matrix operator $\mathbf{\Gamma}$, the mapping of the input space $\mathbf{x}$ to output space $\hat{\mathbf{y}}$, implemented by the network, can be expressed as (Fig. 11.12)

$$\hat{\mathbf{y}} = \mathbf{\Gamma}(\mathbf{W}\mathbf{x} + \mathbf{w}_0) \tag{11.15a}$$

where

$$\mathbf{\Gamma}(\cdot) \triangleq \begin{bmatrix} \sigma(\cdot) & 0 & \cdots & 0 \\ 0 & \sigma(\cdot) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \sigma(\cdot) \end{bmatrix} \tag{11.15b}$$

Note that the nonlinear activation functions $\sigma(\cdot)$ on the diagonal of the matrix operator $\mathbf{\Gamma}$, operate componentwise on the activation vector of each neuron.

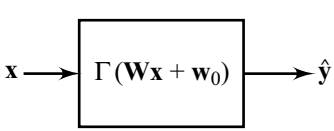


Fig. 11.12 Input-output map of a one-layer network

The input and output vectors $\mathbf{x}$ and $\hat{\mathbf{y}}$ are often called *input* and *output patterns*, respectively. The mapping of the input pattern to an output pattern as given by (11.15), is of the feedforward and instantaneous type since it involves no time delay between input $\mathbf{x}$ and the output $\hat{\mathbf{y}}$.

Neural networks normally have at least two layers of neurons, with the first layer neurons having nonlinear and differentiable activation functions. Such networks, as we will see shortly, can approximate

any continuous function. In real life, we are faced with nonlinear problems, and multilayer neural network structures have the capability of providing solutions to these problems.

If the relationship between the input and output signals is linear, or can be treated as such, a single layer neural network having linear neurons is the best solution. "Adaptive Linear Element" (*Adaline*) is the name given to a neuron with linear activation function and a learning rule for adapting the weights. Single-layer adaline networks have a capacity for a wide range of applications, whenever the problem at hand can be treated as linear.

Multi-layer Perceptrons

A two-layer NN, depicted in Fig. 11.13, has n inputs and two layers of neurons, with the first layer having m neurons that feed into the second layer having q neurons. The first layer is known as the *hidden layer*, with m the number of *hidden-layer neurons*; the second layer is known as the output layer, with q the number of *output-layer neurons*. It is common for different layers to have different numbers of neurons. Note that the outputs of the hidden layer are inputs to the following layer (output layer); and the network is fully connected. Neural networks with multiple layers are called *Multi-layer Perceptrons* (MLP); their computing power is significantly enhanced over the one-layer NN.

All continuous functions (exhibiting certain smoothness) can be approximated to any desired accuracy with a network of one hidden layer of sigmoidal hidden units, and a layer of linear output units [141]. Does it mean that there is no need to use more than one hidden layer and/or mix different types of activation functions? This is not quite true. It may be that the accuracy can be improved using a more sophisticated network architecture. In particular, when the complexity of the mapping to be learned is high, it is likely that the performance can be improved. However, since implementation and training of the network become more complicated, it is customary to apply only a single hidden layer of similar activation functions, and an output layer of linear units. Our focus is on two-layer feedforward neural networks with

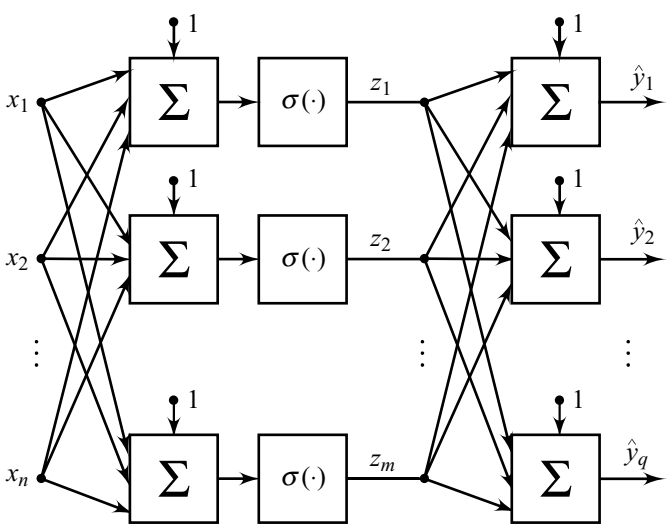


Fig. 11.13 A two-layer network

sigmoidal/hyperbolic tangent hidden units and linear output units. This is probably the most commonly used network architecture, as it works quite well in many practical applications.

Defining the hidden-layer outputs z_l allows one to write

$$z_l = \sigma\left(\sum_{i=1}^{n} w_{li}x_i + w_{l0}\right); \; l = 1, 2, \ldots, m \tag{11.16}$$
$$= \sigma(\mathbf{w}_l^T\mathbf{x} + w_{l0})$$

where

$$\mathbf{w}_l^T \triangleq [w_{l1} \quad w_{l2} \cdots w_{ln}]$$

In vector-matrix notation, the hidden layer in Fig. 11.13 has $m \times 1$ output vector

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}, \tag{11.17a}$$

$m \times n$ weight matrix

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & & \vdots \\ w_{ml} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \tag{11.17b}$$

and $m \times 1$ bias vector

$$\mathbf{w}_0 = \begin{bmatrix} w_{10} \\ w_{20} \\ \vdots \\ w_{m0} \end{bmatrix} \tag{11.17c}$$

The output

$$\mathbf{z} = \boldsymbol{\Gamma}(\mathbf{W}\mathbf{x} + \mathbf{w}_0) \tag{11.18a}$$

where

$$\boldsymbol{\Gamma}(\cdot) \triangleq \begin{bmatrix} \sigma(\cdot) & 0 & \cdots & 0 \\ 0 & \sigma(\cdot) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma(\cdot) \end{bmatrix} \tag{11.18b}$$

Defining the second-layer weight matrix as

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ v_{q1} & v_{q2} & \cdots & v_{qm} \end{bmatrix} \tag{11.19a}$$

and bias vector as

$$\mathbf{v}_0 = \begin{bmatrix} v_{10} \\ v_{20} \\ \vdots \\ v_{q0} \end{bmatrix}, \tag{11.19b}$$

one may write the NN output as

$$\hat{y}_j = \left(\sum_{l=1}^{m} v_{jl} z_l + v_{j0} \right); j = 1, 2, ..., q \tag{11.20}$$

$$= \mathbf{v}_j^T \mathbf{z} + v_{j0}$$

where $\mathbf{v}_j^T \triangleq [v_{j1} \quad v_{j2} \cdots v_{jm}]$

The output vector

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_q \end{bmatrix} \tag{11.21a}$$

is given by the expression

$$\hat{\mathbf{y}} = \mathbf{V}\mathbf{z} + \mathbf{v}_0$$
$$= \mathbf{V}(\mathbf{\Gamma}(\mathbf{W}\mathbf{x} + \mathbf{w}_0) + \mathbf{v}_0) \tag{11.21b}$$

Figure 11.14 shows the input–output map.

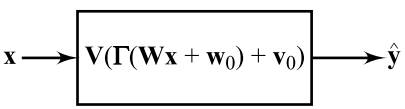


Fig. 11.14 Input-Output map of a two-layer network

11.6.2 Recurrent Networks

The feedforward networks (Figs 11.11–11.14) implement fixed-weight mappings from the input space to the output space. Because the networks have fixed weights, the *state* of any neuron is solely determined by the input to the unit, and not the initial and past states of the neurons. This independence of initial and past states of the network neurons limits the use of such networks because no *dynamics* are involved. The maps implemented by the feedforward networks of the type shown in Figs 11.11–11.14, are *static* maps.

To allow initial and past state involvement along with serial processing, *recurrent neural networks* utilize *feedback.* Recurrent neural networks are also characterized by use of nonlinear processing units; thus, such networks are nonlinear dynamic systems (Networks of the form shown in Figs 11.11–11.14 are nonlinear static systems).

The architectural layout of a recurrent network takes many different forms. We may have feedback from the output neurons of a feedforward network to the input terminals. Yet another possible form is feedback from the hidden neurons of the network to the input terminals. When the feedforward network has two or more hidden layers, the possible forms of feedback expand even further. Recurrent networks have a rich repertoire of architectural layouts.

It should be noted that many real-world problems, which one might think would require recurrent architectures for their solution, turn out to be solvable with feedforward architectures as well. A multilayer feedforward network, which realizes a static map can represent the input/output behavior of a dynamic system. For this to be possible, one must provide the neural network with information about the history of the system—typically, delayed inputs and outputs. How much history is needed, depends on the desired accuracy. There is a trade-off between accuracy and computational complexity of training, since the number of inputs used, affects the number of weights in the neural network—and subsequently, the training time (Section 11.11 will give more details). One sometimes starts with as many delayed signals as the order of the system, and then modifies the network accordingly. It also appears that using a two hidden-layer network—instead of one hidden layer—has certain computational advantages. The number of neurons in the hidden layer(s) is typically chosen based on empirical criteria, and one may iterate over a number of networks to determine a neural network that has a reasonable number of neurons and accomplishes the desired degree of approximation.

From numerous practical applications published over the past decade, there seems to be substantial evidence that multilayer feedforward networks possess an impressive ability to perform reasonably well in most cases of practical interest. Lately, there have also been some theoretical results that attempt to explain the reasons for the success [138].

Our focus is on two-layer feedforward neural networks with sigmoidal/hyperbolic tangent hidden units and linear output units. This is probably the most commonly used network architecture as it works quite well in many practical applications.

11.7 FUNCTION APPROXIMATION WITH NEURAL NETWORKS

Of fundamental importance in NN closed-loop control applications is the *universal function approximation* property of NNs having at least two layers (one-layer NNs do not generally have a universal approximation capability).

The basic universal approximation result says [141] that any smooth function $\mathbf{f}(\mathbf{x})$ can be approximated arbitrarily closely on a compact set using a two-layer NN with appropriate weights. This result has been shown using sigmoid activations, RBF activations, and others. Specifically, let $\mathbf{f}(\mathbf{x})$ be a smooth function; $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^T$, $\mathbf{f}(\cdot) = [f_1(\cdot)\ f_2(\cdot)\ \cdots\ f_q(\cdot)]^T$, $\mathbf{S}$ be a compact set in n-dimensional state space and ε_N be a positive number. There exists a two-layer NN (Eqn. (11.21b)) such that

$$\mathbf{f}(\mathbf{x}) = \mathbf{V}(\mathbf{\Gamma}(\mathbf{W}\mathbf{x} + \mathbf{w}_0) + \mathbf{v}_0) + \boldsymbol{\varepsilon}$$

with $\|\boldsymbol{\varepsilon}\| < \varepsilon_N$ for all $\mathbf{x} \in \mathbf{S}$, for some (sufficiently large) number m of hidden-layer neurons. The value $\boldsymbol{\varepsilon}$ (generally a function of $\mathbf{x}$), is called the *NN function approximation error*, and it decreases as the hidden-layer size m increases. We say that on the compact set $\mathbf{S}$, $\mathbf{f}(\mathbf{x})$ is 'within ε_N of the NN functional range'. Approximation results have also been shown for smooth functions with a *finite* number of discontinuities.

Note that in this result, the activation functions are not needed on the NN output layer (i.e., the output layer activation functions are linear). It also happens that the bias terms v_{j0} on the output layers are not needed, though the hidden layer bias terms $w_{\ell 0}$ are required.

Note further that, though the result says 'there exists an NN that approximates $\mathbf{f}(\mathbf{x})$', it does not show how to determine the required number of units in the hidden layer. The issue of finding the required number of units in the hidden layer such that an NN does indeed approximate a given function $\mathbf{f}(\mathbf{x})$ closely enough, is not an easy one (If the function approximation is to be carried out in the context of a dynamic closed-loop feedback control scheme, the issue is thornier and is discussed in subsequent sections). This issue has been addressed in the literature [138, 141], and a significant result has been derived about the approximation capabilities of two-layer networks when the function to be approximated exhibits a certain smoothness. Unfortunately, the result is difficult to apply for selecting the number of hidden units. The guidelines to select the appropriate number of hidden neurons are rather empirical at the moment. To avoid large number of neurons and the corresponding inhibitively large training times, the smaller number of hidden layer neurons are often used in the first trial. One increases accuracy by adding more hidden neurons. Excessively large number of hidden units may lead to poor generalization, a key feature of the performance of NN.

Because of the above-mentioned results, one might think that there is no need for using more than one hidden layer, and/or different types of activation functions. This is not quite true: it may be that accuracy can be improved using a more sophisticated network architecture. In particular, when the complexity of the mapping to be learned is high (e.g., functions with discontinuities), it is likely that the performance can be improved. Experimental evidence tends to show that using a two hidden-layer network for continuous functions has sometimes advantages over a one hidden-layer network, as the former requires shorter training times.

11.7.1 The Basic Learning Mechanism

Each processing element (neuron) in a neural network has a number of inputs (x_i), each of which must store a connection weight (w_{ji}). The element sums up the weighted input ($w_{ji}x_i$) and computes one, and only one, activation signal (a_j). The output signal is a function ($\sigma(\cdot)$) of the weighted sum. Figure 11.15 summarizes how a processing element works.

The function $\sigma(\cdot)$ remains fixed for the life of the processing element. It is generally decided upon as part of the design, and it cannot be changed dynamically. In other words, the transfer function currently cannot be adjusted or modified during the operation or running of the network.

However, the weights (w_{ji}) are variables. They can be dynamically adjusted to produce a given output (y_j). This dynamic modification of the variable weights is the very essence of *learning*. At the level of a single processing element, this self-adjustment is very simple. When many processing elements do it collectively, we say it resembles 'intelligence'. The meaningful information is in the modified weights. The ability of an entire neural network to adapt itself (change the w_{ji} values) to achieve a given output (y_j), is its uniqueness.

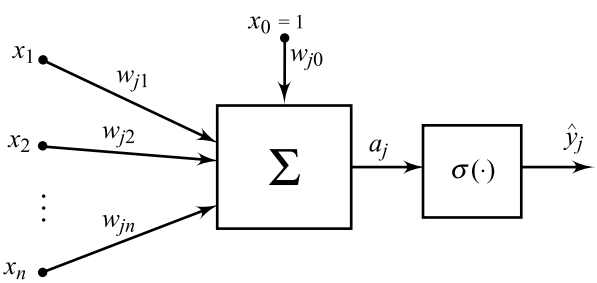


Fig. 11.15 A processing element

Pairs of inputs and outputs are applied to the neural network. These pairs of data are used to *teach* or *train* the network, and as such are referred to as the *training set*. Knowing what output is expected from each

input, the network automatically adjusts or adapts the strengths of the connections between processing elements. The method used for the adjusting process is called the *learning rule*.

Neural networks deal only with numeric input data. Therefore, we must convert or encode information from the external environment to numeric data form. Additionally, it is often necessary to scale data. Inhibitory inputs are just as important as excitatory inputs. The input scheme should adequately allow for both the types (allow positive and negative weights). A provision is also usually made for constant-source input to serve as an offset or bias term for the transfer or activation function.

The numeric output data of a neural network will, likewise, require decoding and scaling to make it compatible with the external environment.

Important characteristics of the network depend on:

(i) the transfer or activation functions of the processing elements;
(ii) the structure of the network (number of neurons, layers and interconnections); and
(iii) the learning rules of the network.

11.7.2 Supervised Learning Rules

These rules compute the necessary change in the connection weights by presenting the network given input pattern, comparing the obtained response with a desired response known *a priori* and then changing the weights in the direction of decreasing error. More clearly, in the supervised learning mode, a neural network is supplied with a sequence of examples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)})$, $(\mathbf{x}^{(2)}, \mathbf{y}^{(2)})$, ..., $(\mathbf{x}^{(p)}, \mathbf{y}^{(p)})$, ..., of desired input-output pairs. When each input $\mathbf{x}^{(p)}$ is fed into the neural network, the corresponding desired output $\mathbf{y}^{(p)}$ is also supplied to the neural network. As shown in Fig. 11.16, the difference between the actual neural network (NN) output $\hat{\mathbf{y}}^{(p)}$ and the desired output $\mathbf{y}^{(p)}$ is measured in the *error-signal generator*, which then produces error signals for the NN to correct its weights in such a way that the actual output will move closer to the desired output. In the subsequent sections, commonly used learning rules (algorithms) are presented.

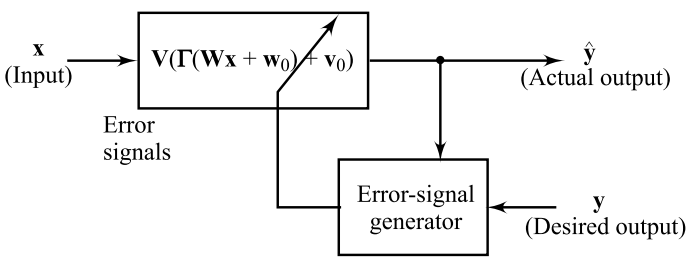


Fig. 11.16 Supervised learning

11.8 LINEAR LEARNING MACHINES

Linear mathematical functions are the best understood, and the simplest for neural network (NN) learning. The classical NN literature has developed methods for linear function learning; we will refer to the associated NN structures as *linear learning machines*. These techniques, which include both efficient iterative procedures and theoretical analysis of their generalization properties, provide the framework within which the construction of more complex (nonlinear) functions will be developed in the subsequent

sections. In this section, we present algorithms for training linear machines. These algorithms will be relevant to the study of multilayer neural networks and support vector machines in later sections.

For learning problems with a scalar output, only one neuron (perceptron) constitutes the linear learning machine. For multiple outputs (represented by vector **y**), single layer of perceptrons gives us the required structure. We present the algorithms for the scalar-output case; extension to vector-output case is straightforward.

11.8.1 Least Squares Algorithm

Consider the simple case of a single neuron with linear activation function. Figure 11.17 is a schematic diagram of such a network. An input signal $\mathbf{x} = [x_1, x_2, ..., x_n]^T$, comprising features and augmented by a constant input component (bias), is applied to the neuron; weighted and summed to give an output signal $\hat{y}$:

$$\begin{aligned}\hat{y} &= \sum_{i=1}^{n} w_i x_i + w_0 \\ &= \mathbf{w}^T + w_0\end{aligned} \tag{11.22a}$$

where $\mathbf{w}^T = \begin{bmatrix} w_1 & w_2 \cdots w_n \end{bmatrix}$

Defining $(n + 1) \times 1$ vector

$$\bar{\mathbf{x}} = \begin{bmatrix} 1 & x_1 & x_2 \cdots x_n \end{bmatrix}^T$$

and $1 \times (n + 1)$ vector

$$\bar{\mathbf{w}}^T = \begin{bmatrix} w_0 & w_1 & w_2 \cdots w_n \end{bmatrix},$$

we can express Eqn. (11.22a) as

$$\hat{y} = \bar{\mathbf{w}}^T \bar{\mathbf{x}} \tag{11.22b}$$

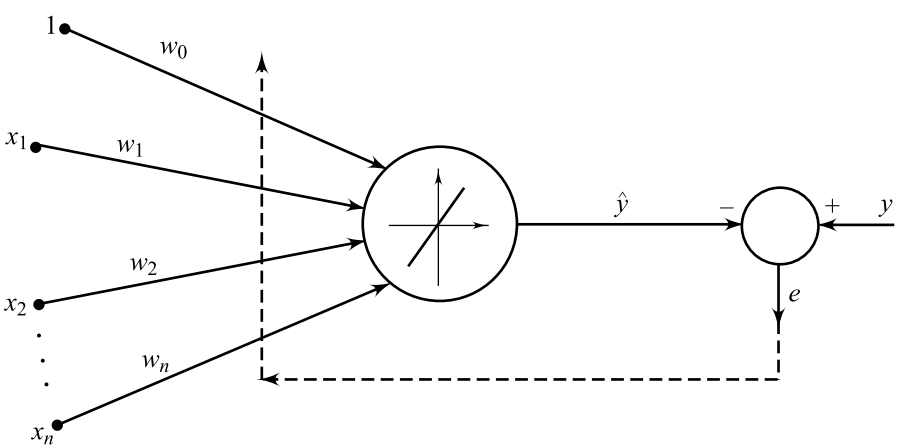


Fig. 11.17 Learning scheme for a linear neuron
(Dashed arrows indicate that weights adaptation depends on the output error)

The learning task is to find the weights of the neuron (estimate the parameters of the proposed linear model (11.22a)) using a *finite* number of measurements, observations, or patterns. The learning environment, thus, comprises a training set of measured data (patterns):

$$\{\mathbf{x}^{(p)}, y^{(p)}; p = 1, 2, ..., P\}$$

consisting of an input vector **x** and output or system response y, and the orresponding learning rule for the adaptation of the weights (In the following discussion, learning algorithm is given for the case of one neuron only, and the desired output is a scalar variable. The extension of the algorithm for **y**, a vector, is straightforward). The choice of a performance criterion, or the measure of goodness of the estimation, depends primarily on the data, and on the desired simplicity of the learning algorithm. In the neural network field, the most widely used performance criterion (cost function) is the sum of error squares:

$$E = \tfrac{1}{2}\sum_{p=1}^{P}\left(e^{(p)}\right)^2 = \tfrac{1}{2}\sum_{p=1}^{P}\left(y^{(p)} - \hat{y}^{(p)}\right)^2 \tag{11.23}$$

(The constant ½ is used for computational convenience only. It gets cancelled out by the differentiation required in the error minimization process).

It is obvious that network equation (11.22b) is exactly a linear model with $(n + 1)$ linear parameters. So we can employ the least-squares methods, discussed in Chapter 10, to minimize the error in the sense of least squares.

A matrix of input vectors $\mathbf{x}^{(p)}$; $p = 1, 2, ..., P$ (called the *data matrix* **X**) and vector **y** of the desired outputs $y^{(p)}$; $p = 1, 2, ..., P$, are introduced as follows:

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(P)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(P)} \\ \vdots & \vdots & & \vdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(P)} \end{pmatrix} = \left[\bar{\mathbf{x}}^{(1)}\, \bar{\mathbf{x}}^{(2)} \cdots \bar{\mathbf{x}}^{(P)}\right] \tag{11.24a}$$

$$\mathbf{y} = \left[y^{(1)}\, y^{(2)} \cdots y^{(P)}\right]^T$$

The weights $\bar{\mathbf{w}}$ are required to satisfy the following equations (refer to Eqn. (11.22b)):

$$\begin{aligned} y^{(1)} &= \bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(1)} \\ y^{(2)} &= \bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(2)} \\ &\vdots \\ y^{(P)} &= \bar{\mathbf{w}}^T \bar{\mathbf{x}}^{(P)} \end{aligned}$$

Therefore,

$$[y^{(1)}\, y^{(2)} \,....\, y^{(P)}] = \bar{\mathbf{w}}^T [\bar{\mathbf{x}}^{(1)}\, \bar{\mathbf{x}}^{(2)} \cdots \bar{\mathbf{x}}^{(P)}]$$

$$\mathbf{y}^T = \bar{\mathbf{w}}^T \mathbf{X}$$

or

$$\underset{P\times 1}{\mathbf{y}} = \underset{P\times(n+1)}{\mathbf{X}^T}\ \underset{(n+1)\times 1}{\bar{\mathbf{w}}}$$

In the least squares sense, the best or optimal $\bar{\mathbf{w}}$ that minimizes E results from the equation (refer to Eqns (10.38–10.40))

$$\bar{\mathbf{w}} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y} = [w_0 \;\; w_1 \;\; w_2 \cdots w_n]^T \tag{11.24b}$$

From a computational point of view, the calculation of optimal weights requires the pseudo-inverse of the $P \times (n + 1)$ matrix $\mathbf{X}$.

An alternative solution to this type of problem is the 'Recursive Least Squares' (RLS) algorithm (refer to Eqns (10.45)). For the learning problem in hand, the steps given in Table 11.1 implement the algorithm.

Table 11.1 Recursive Least Squares Algorithm

Given is a set of P measured data points that are used for training:

$$\{\mathbf{x}^{(p)}, y^{(p)}; p = 1, 2, \ldots, P\}$$

consisting of the input pattern vector $\mathbf{x}$ and the desired response y;

$$\bar{\mathbf{x}} = [1 \;\; x_0 \;\; x_1 \;\; x_2 \cdots x_n]^T$$

The weight vector

$$\bar{\mathbf{w}}^T = [w_0 \;\; w_1 \;\; w_2 \cdots w_n]$$

is to be constructed.

Perform the following training steps for $p = 1, 2, \ldots, P$.

Step 1 Set the iteration index $k = 0$. Initialize the weight vector $\bar{\mathbf{w}}(0) = \mathbf{0}$ and the matrix $\mathbf{P}(0) = \alpha\mathbf{I}_{(n+1)}$, where α should be very large number, say, of the order of 10^8 to 10^{15}.

Step 2 Apply the next ($p = 1$ for the first one) training pair $\left\{\bar{\mathbf{x}}^{(p)}, y^{(p)}\right\}$ to the linear neuron.

Step 3 Calculate the error for the applied data pair: $e(k+1) = y(k+1) - \bar{\mathbf{x}}^T(k+1)\bar{\mathbf{w}}(k)$.

Step 4 Calculate the vector $\mathbf{K}(k)$.

$$\mathbf{K}(k) = \mathbf{P}(k)\bar{\mathbf{x}}(k+1)[1 + \bar{\mathbf{x}}^T(k+1)\mathbf{P}(k)\bar{\mathbf{x}}(k+1)]^{-1}$$

Step 5 Calculate the updated weight vector.

$$\bar{\mathbf{w}}(k+1) = \bar{\mathbf{w}}(k) + \mathbf{K}(k)e(k+1)$$

Step 6 Find the matrix $\mathbf{P}(k + 1)$

$$\mathbf{P}(k+1) = \mathbf{P}(k) - \mathbf{K}(k)\bar{\mathbf{x}}^T(k+1)\mathbf{P}(k)$$

Step 7 Stop the adaptation of the weights if the error is smaller than the predefined value. Otherwise set $k \to k + 1$ and go back to step 2.

11.8.2 Gradient Descent Algorithm

We have shown how, for a linear neuron, the weight values which minimize the sum-of-squares error function can be found explicitly in terms of the pseudo-inverse of a matrix. It is important to note that this result is possible only for the case of a linear neural network, with a sum-of-squares error function as the performance criterion. If a nonlinear activation function, such as a sigmoid, is used, or if a different error function is considered, then a closed-form solution is no longer possible. However, if the activation function is differentiable, as is the case of the sigmoid, the derivatives of the error function with respect to

the weight parameters can easily be evaluated. These derivatives can then be used in a variety of gradient-based optimization algorithms for finding the minimum of the error function. Here we consider one of the simplest of such algorithms, *the steepest descent algorithm*, for a single linear neuron. We will later extend the algorithm to multilayer neural networks with sigmoidal/linear units.

For a linear neuron of Fig.11.17, the training set comprises the pairs

$$\{\mathbf{x}^{(p)}, y^{(p)}; p = 1, 2, ..., P\}$$

The performance criterion (refer to Eqn.(11.23)) is

$$E = \frac{1}{2}\sum_{p=1}^{P}(y^{(p)} - \hat{y}^{(p)})^2 = \frac{1}{2}\sum_{p=1}^{P}(e^{(p)})^2 \tag{11.25}$$

where (refer to Eqn.(11.22a))

$$\hat{y}^{(p)} = \sum_{i=1}^{n} w_i x_i^{(p)} + w_0 = \mathbf{w}^T \mathbf{x}^{(p)} + w_0 \tag{11.26}$$

To understand the gradient descent algorithm, it is helpful to visualize the error space of possible weight vectors and the associated values of the performance criterion (*cost function*). For linear neuron, the error surface is parabolic with a single global minimum.

Gradient descent search determines a weight vector that minimizes the cost function by starting with an arbitrary initial weight vector, then repeatedly modifying it in small steps. At each step, the weight vector is altered in the direction that produces the steepest descent along the error surface. This process continues until the global minimum error is reached.

Let $w_i(k)$ be the weights on the iteration index k, and the associated cost function is $E(k)$. The search direction given by $-(\partial E(k)/\partial w_i(k))$, takes us iteratively towards the minimum point according to the rule

$$w_i(k+1) = w_i(k) - \eta \frac{\partial E(k)}{\partial w_i(k)} \tag{11.27}$$

where η, the positive step-size parameter, is taken as less than 1, and is called the *learning rate*.

The two most useful training protocols are batch and incremental. In batch training, all patterns are presented to the network before learning takes place. The cost function $E(k)$ for the batch training is given by Eqn.(11.25). A variation of this approach is to update weights for each of the training pairs. This is known as *incremental* mode of training. We first consider the incremental mode.

Incremental Training

For the incremental training, the cost function at iteration k, is

$$E(k) = \frac{1}{2}(y^{(p)} - \hat{y}(k))^2 = \frac{1}{2}[e^{(p)}(k)]^2 \tag{11.28a}$$

for the training pair $(\mathbf{x}^{(p)}, y^{(p)},)$ with

$$\hat{y}^{(p)}(k) = \sum_{i=1}^{n} w_i(k) x_i^{(p)} + w_0(k) \tag{11.28b}$$

Note that the components $x_i^{(p)}$ of the input vector $\mathbf{x}^{(p)}$, and the desired output $y^{(p)}$ are not functions of the iteration index k.

The gradients with respect to weights and bias are computed as follows:

$$\frac{\partial E(k)}{\partial w_i(k)} = e^{(p)}(k)\frac{\partial e^{(p)}(k)}{\partial w_i(k)} = -e^{(p)}(k)\frac{\partial \hat{y}^{(p)}(k)}{\partial w_i(k)}$$

$$= -e^{(p)}(k)\frac{\partial}{\partial w_i(k)}\left[\sum_{i=1}^{n} w_i(k)x_i^{(p)} + w_0(k)\right]$$

$$= -e^{(p)}(k)x_i^{(p)}$$

$$\frac{\partial E(k)}{\partial w_0(k)} = -e^{(p)}(k)$$

The gradient descent algorithm becomes

$$w_i(k+1) = w_i(k) + \eta\, e^{(p)}(k)\, x_i^{(p)};\ p = 1, 2, ..., P;\ i = 1, 2, ..., n \tag{11.29a}$$

$$w_0(k+1) = w_0(k) + \eta\, e^{(p)}(k) \tag{11.29b}$$

In terms of vectors, this algorithm may be expressed as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta\, e^{(p)}(k)\, \mathbf{x}^{(p)};\ p = 1, 2, ..., P \tag{11.30a}$$

$$w_0(k+1) = w_0(k) + \eta\, e^{(p)}(k) \tag{11.30b}$$

Incremental training algorithm iterates over the training examples $p = 1, 2, \ldots, P$; at each iteration, altering the weights as per Eqns (11.30). The sequence of these weight updates iterated over all the P training examples, provides a reasonable approximation to the gradient descent with respect to the batch of data. By making the value of η reasonably small, incremental gradient descent can be made to approximate true gradient descent arbitrarily closely.

At each presentation of the data $(\mathbf{x}^{(p)}, y^{(p)})$, one step of training algorithm is performed which updates both the weights and the bias. An *epoch* is defined as one complete run through all the P associated pairs. When an epoch has been completed, the pair $(\mathbf{x}^{(1)}, y^{(1)})$ is presented again and another run through all the P pairs is performed. It is hoped that after many epochs, the output error will be small enough.

Note that the approach of teaching the network one fact at a time from one data pair does not work. All the weights set so meticulously for one fact, could be drastically altered in learning the next fact. The network has to learn everything together, finding the best weight settings for the total set of facts. Therefore, with incremental training, the training should stop only after an epoch has been completed.

Batch Training

Our true interest lies in learning to minimize the total error over the entire batch of training examples. All P pairs are presented to the linear unit (one at a time) and a cumulative error is computed, after all pairs have been presented. At the end of this procedure, the neuron weights and bias are updated once. The result is as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta\left[\sum_{p=1}^{P} e^{(p)}(k)\right]\mathbf{x}^{(p)} \tag{11.31a}$$

$$w_0(k+1) = w_0(k) + \eta \sum_{p=1}^{P} e^{(p)}(k) \tag{11.31b}$$

In batch training, the iteration index corresponds to the number of times the set of P pairs is presented and the cumulative error is compounded. That is, k corresponds to epoch number.

Compared with the incremental mode, the batch mode is an inherent averaging process. This leads to a better estimate of the gradients; thus to more well-behaved convergence. Both the incremental and batch training modes are commonly used in practice; the error surface in the MLP network case, as we will see shortly, may contain multiple local minima, and incremental training can sometimes avoid falling into these local minima.

Error Function for Stopping Criterion

The sum of error squares over all the training pairs is accumulated in the incremental mode of learning. After the learning epoch (the sweep through all the training patterns) is completed ($p = P$), the total error E_P is compared with the acceptable (desired) value E_{des}; learning is terminated if $E_P < E_{des}$. Otherwise a new learning epoch is started. In the batch mode, weight updating is performed *after* the presentation of *all* the training examples that constitute an epoch. The error E_P is compared with E_{des} after each iteration of a learning epoch.

The sum of error squares is not good as stopping criterion because E_P increases with the increase of the number of data pairs. The more data, the larger is E_P. Scaling of the error function gives a better stopping criterion. The *root mean square error* (RMSE) is a widely used scaled error function:

$$E_{RMS} = \sqrt{\frac{1}{P}\sum_{p=1}^{P}(e^{(p)})^2}$$

There will be no need to change the learning algorithm derived earlier. Training is performed using *sum-of-error squares* as the cost function (performance criterion), and RMSE is used as a stopping criterion at training. However, if desired, for the batch mode the learning algorithm with *average square error*:

$$E_{av} = \frac{1}{2P}\sum_{p=1}^{P}(e^{(p)})^2$$

may be used as the cost function for training the network.

11.9 TRAINING THE MULTILAYER PERCEPTRON NETWORK–BACKPROPAGATION ALGORITHM

In the previous section, we dealt with the training of a linear neuron using *least squares algorithm* and *gradient descent algorithm*. Both of these algorithms can easily be extended to a network with a layer of linear neurons.

For real-world problems, one has no previous knowledge of what kind of dependency function between input **x** and output **y** is most suitable; a linear function may not lead to satisfactory performance. Trial-

and-error design of a nonlinear function is a difficult task, but inescapable necessity. What we seek is a clever choice of the nonlinearity. This is the approach of Multi-Layer Perceptron (MLP) networks. MLP networks can at least in principle, provide the optimal solution to an arbitrary function approximation problem.

Consider the two-layer network shown in Fig. 11.13. There is nothing magical about this network; it implements linear functions in a space where the inputs have been mapped nonlinearly using sigmoidal transformation. It is natural to ask whether *every* nonlinear function can be implemented by a network of this form. The answer is 'yes'—any continuous function from input to output can be implemented by a network of the form of Fig. 11.13, given sufficient number of hidden units. If **x** is fed to the input terminals (including the bias), the 'activation' propagates in the feedforward direction, and the output values of the hidden units are calculated. Each hidden unit is a perceptron by itself and applies the nonlinear sigmoid to its weighted sum. If the hidden units' outputs were linear, the hidden layer would be of no use for function approximation; linear combination of linear combinations is another linear combination.

One is not limited to MLP networks of the form of Fig. 11.13. More hidden layers with their own weights can be placed after the first layer with sigmoid hidden units, thus calculating nonlinear transformations of the first layer of hidden units and implementing more complex functions of the inputs. In practice, we rarely go beyond one hidden layer since analyzing a network with many hidden layers is quite complicated; but sometimes when the hidden layer contains too many hidden units, it may be sensible to go to multiple hidden layers, preferring 'long and narrow' networks to 'short and flat' networks.

The key power provided by MLP networks is that they admit simple gradient-based training algorithms. This is made possible because sigmoid is a continuous and differentiable function, *with a useful property that its derivative is easily expressed in terms if its output*.

Consider a sigmoidal neuron of Fig. 11.8. The activation

$$a = \sum_{i=1}^{n} w_i x_i + w_0 \tag{11.32a}$$

and the output

$$\hat{y} = \sigma(a) = \frac{1}{1+e^{-a}} \tag{11.32b}$$

The derivative

$$\begin{aligned}\frac{d}{da}\sigma(a) &= \frac{d}{da}\left[\frac{1}{1+e^{-a}}\right] = -\frac{1}{(1+e^{-a})^2}(-e^{-a}) \\ &= \frac{1}{1+e^{-a}}\left[\frac{e^{-a}}{1+e^{-a}}\right] = -\frac{1}{1+e^{-a}}\left[1-\frac{1}{1+e^{-a}}\right] \\ &= \sigma(a)[1-\sigma(a)] = \hat{y}(1-\hat{y})\end{aligned} \tag{11.32c}$$

11.9.1 Backpropagation Algorithm

We aim to derive the backpropagation algorithm for setting the weights based on training patterns, for the two-layer perceptron network of Fig. 11.13, which is frequently used in practice. Extension of the results derived in this section to more general perceptron networks is straightforward.

Backpropagation is one of the simplest and most general method for supervised training of MLP networks. It is a natural extension of the gradient descent algorithm derived in the previous section for a linear neuron. The gradient descent algorithm worked for the linear unit because the error, proportional to the square of the difference between the actual output and the desired output, could be evaluated in terms of input terminals-to-output layer weights. Similarly, in a two-layer network, it is a straightforward matter to find out how the error depends on hidden-to-output layer weights. In fact, this dependency is analogous to the linear unit case.

But how should the input terminals-to-hidden layer weights be learned; the ones governing the nonlinear transformation of the input vectors? If the 'proper' outputs for hidden units were known for any input, the input terminals-to-hidden layer weights could be adjusted to approximate it. However, there is no explicit 'supervisor' to state what the hidden units' output should be. The power of back propagation is that it allows us to calculate an 'effective' error for each hidden unit, and thus derive a learning rule for input terminals-to- hidden layer weights.

We begin by defining the cost function for incremental training (iterating through the training examples one at a time):

$$E(k) = \frac{1}{2}\sum_{j=1}^{q}[e_j^{(p)}(k)]^2; e_j^{(p)}(k) = y_j^{(p)} - \hat{y}_j^{(p)}(k) \tag{11.33a}$$

where $\hat{y}_j^{(p)}$ is evaluated using the equations

$$\hat{y}_j^{(p)}(k) = \sum_{\ell=1}^{m} v_{j\ell}(k)\, z_\ell^{(p)}(k) + v_{j0}(k);\ j = 1, .., q \tag{11.33b}$$

$$z_\ell^{(p)}(k) = \sigma\left(\sum_{i=1}^{n} w_{\ell i}(k)\, x_i^{(p)} + w_{\ell 0}(k)\right);\ \ell = 1, .., m \tag{11.33c}$$

These equations directly follow from Eqns ((11.28), (11.20), (11.16)), with the difference that now we have a layer of q linear units, rather than a single linear unit.

For each training example p, every weight $v_{j\ell}$; $j = 1, ..., q$; $\ell = 1, ..., m$, is updated by adding to it $\Delta v_{j\ell}$:

$$\Delta v_{j\ell} = -\eta \frac{\partial E(k)}{\partial v_{j\ell}(k)} \tag{11.34a}$$

$$v_{j\ell}(k+1) = v_{j\ell}(k) - \eta \frac{\partial E(k)}{\partial v_{j\ell}(k)} \tag{11.34b}$$

With linear activation function in the output layer, the update rule becomes

$$v_{j\ell}(k+1) = v_{j\ell}(k) + \eta\, e_j^{(p)}(k)\, z_\ell^{(p)}(k) \tag{11.34c}$$

$$v_{j0}(k+1) = v_{j0}(k) + \eta\, e_j^{(p)}(k) \tag{11.34d}$$

We now consider the hidden layer of the network. Unlike the output nodes, the desired outputs of the hidden nodes are unknown. The backpropagation algorithm for a given input-output pair ($\mathbf{x}^{(p)}$, $y^{(p)}$) performs two phases of data flow. First the input pattern $\mathbf{x}^{(p)}$ is propagated from the input terminals to the output layer; and as a result of the forward flow of the data, it produces an output $\hat{\mathbf{y}}^{(p)}$. Then the error

signals $\mathbf{e}^{(p)}$ resulting from the difference between $\mathbf{y}^{(p)}$ and $\hat{\mathbf{y}}^{(p)}$ are *backpropagated* from the output layer to the hidden layer, to update the weights $w_{\ell i}$. Error backpropagation may be computed by expanding the error derivative using the chain rule, as follows:

$$w_{\ell i}(k+1) = w_{\ell i}(k) - \eta \frac{\partial E(k)}{\partial w_{\ell i}(k)} \tag{11.35a}$$

$$\frac{\partial E(k)}{\partial w_{\ell i}(k)} = -e_j^{(p)}(k) \frac{\partial \hat{y}_j^{(p)}(k)}{\partial w_{\ell i}(k)} \tag{11.35b}$$

$\hat{y}_j^{(p)}$ is a function of $z_\ell^{(p)}$; which, in turn, is a function of $x_i^{(p)}$:

$$\hat{y}_j^{(p)} = \sum_{\ell=1}^{m} v_{j\ell}\, z_\ell^{(p)} + v_{j0}$$

$$z_\ell^{(p)} = \sigma(a); a = \sum_{i=1}^{n} w_{\ell i}\, x_i^{(p)} + w_{\ell 0}$$

Therefore,

$$\frac{\partial E(k)}{\partial w_{\ell i}(k)} = -e_j^{(p)}(k) \frac{\partial \hat{y}_j^{(p)}}{\partial z_\ell^{(p)}(k)} \frac{\partial z_\ell^{(p)}(k)}{\partial a(k)} \frac{\partial a(k)}{\partial w_{\ell i}(k)} \tag{11.35c}$$

$$\frac{\partial \hat{y}_j^{(p)}(k)}{\partial z_\ell^{(p)}(k)} = v_{j\ell}(k)$$

$$\frac{\partial z_\ell^{(p)}(k)}{\partial a(k)} = \frac{\partial \sigma(a(k))}{\partial a(k)} = \sigma(a(k))[1-\sigma(a(k))]$$

$$= z_\ell^{(p)}(k)[1 - z_\ell^{(p)}(k)]$$

$$\frac{\partial a(k)}{\partial w_{\ell i}(k)} = x_i^{(p)}$$

Therefore,

$$\frac{\partial E(k)}{\partial w_{\ell i}(k)} = -x_i^{(p)}[z_\ell^{(p)}(k)]\,[1 - z_\ell^{(p)}(k)] \sum_{j=1}^{m} v_{j\ell}(k) e_j^{(p)}(k) \tag{11.35d}$$

and the update rule becomes

$$w_{\ell i}(k+1) = w_{\ell i}(k) + \eta \delta_\ell^{(p)}(k) x_i^{(p)} \tag{11.35e}$$

where *backpropagated error*

$$\delta_\ell^{(p)} = z_\ell^{(p)}(k)[1 - z_\ell^{(p)}(k] \sum_{j=1}^{m} v_{j\ell}(k) e_j^{(p)}(k) \tag{11.35f}$$

$$w_{\ell 0}(k+1) = w_{\ell 0}(k) + \eta \delta_\ell^{(p)}(k) \tag{11.35g}$$

The backpropagation algorithm consists of repeating the following iterative procedure until the neural network output error has become sufficiently small.

Forward Recursion to Compute MLP Output

Present input vector $\mathbf{x}^{(p)}$ to the MLP, and compute the MLP output using

$$z_\ell^{(p)}(k) = \sigma\left(\sum_{i=1}^{n} w_{\ell i}(k)\, x_i^{(p)} + w_{\ell 0}(k)\right);\ \ell = 1, ..., m \tag{11.36a}$$

$$\hat{y}_j^{(p)}(k) = \sum_{\ell=1}^{m} v_{j\ell}(k)\, z_\ell^{(p)}(k) + v_{j0}(k);\ j = 1, ..., q \tag{11.36b}$$

with initial weights $w_{\ell 0}^{(0)}$, $w_{\ell i}^{(0)}$, $v_{j0}^{(0)}$, $v_{j\ell}^{(0)}$, randomly chosen.

Backward Recursion for Backpropagated Errors

$$e_j^{(p)}(k) = y_j^{(p)} - \hat{y}_j^{(p)}(k);\ j = 1, ..., q \tag{11.36c}$$

$$\delta_\ell^{(p)}(k) = z_\ell^{(p)}(k)\,[1 - z_\ell^{(p)}(k)] \sum_{j=1}^{q} v_{j\ell}\, e_j^{(p)}(k);\ \ell = 1, ..., m \tag{11.36d}$$

Computation of MLP Weights and Bias Updates

$$v_{j\ell}(k+1) = v_{j\ell}(k) + \eta\, e_j^{(p)}(k)\, z_\ell^{(p)}(k) \tag{11.36e}$$

$$v_{j0}(k+1) = v_{j0}(k) + \eta\, e_j^{(p)}(k) \tag{11.36f}$$

$$w_{\ell i}(k+1) = w_{\ell i}(k) + \eta\, \delta_\ell^{(p)}(k)\, x_i^{(p)}(k) \tag{11.36g}$$

$$w_{\ell 0}(k+1) = w_{\ell 0}(k) + \eta\, \delta_\ell^{(p)}(k) \tag{11.36h}$$

Batch training algorithm follows on the similar lines.

11.9.2 Improvements on Gradient Descent

There are many sorts of training algorithms for NN; the basic type we have discussed in the previous subsection is the backpropagation training algorithm. Though the backpropagation algorithm enjoys great success, one must remember that it is a gradient-based technique, so that the usual caveats associated with step sizes, local minima and so on, must be kept in mind while using it.

The NN weights and biases are typically initialized to small random (positive and negative) values. A typical error surface graph in 1-D is shown in Fig. 11.18, which shows a local minimum and a global minimum. If the weight is initialized as shown in Case 1, there is a possibility that the gradient descent might find the local minimum. Several authors have determined better techniques to initialize the weights than the random selection, particularly for the multilayer NN. Among these are Nguyan and Widrow, whose techniques are used, for instance, in MATLAB. Such improved initialization techniques can also significantly speed up convergence of the weights to their final values.

An improved version of gradient descent is given by *Momentum Gradient Algorithm*. Momentum allows a network to respond, not only to the local gradient, but also to recent trends in error surface. The learning rule with the inclusion of a momentum term can be written as (refer to Eqns (11.34a – 11.35a))

$$\Delta w_{\ell i}(k) = -\eta \frac{\partial E(k)}{\partial w_{\ell i}(k)} + \alpha \Delta w_{\ell i}(k-1);\ 0 \le |\alpha| \le 1 \tag{11.37}$$

Without momentum, a network may get stuck in a shallow local minimum; adding mementum can help the NN 'ride through' local minima. (Case 1 in Fig. 11.18 may not get stuck in local minimum while learning with momentum). In the MATLAB Neural Network Toolbox are some examples which show that learning with momentum can significantly speed up and improve the performance of backpropagation.

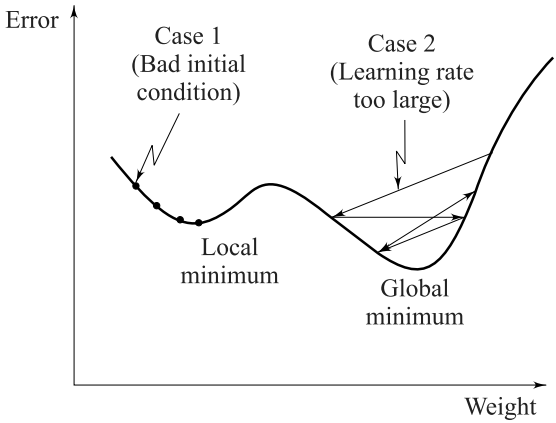


Fig. 11.18 Typical 1-D NN error surface

Only small learning constants η guarantee a true gradient descent. The price of this guarantee is an increased total number of learning steps that need to be made to reach a satisfactory solution. It is desirable to monitor the progress of learning so that η can be increased at appropriate stages of training to speed up the minimum seeking.

When broad minima yield small gradient values, then a larger value of η will result in a more rapid convergence. However, for problems with steep and narrow minima, if the learning rate η is too large, then the NN can overshoot the minimum cost value, jumping back and forth over the minimum, and failing to converge, as shown in Fig. 11.18, Case 2. Adapting the learning rates can significantly speed up the convergence of the weights.

All the refinements: selecting better initial conditions, using learning with momentum, and using an adaptive learning rate, are available in the MATLAB NN Toolbox.

In practice, the gradient method is quite slow. Other methods are available which converge much faster. In most applications, it is therefore difficult to justify using the gradient method. Nevertheless, the method has gained a remarkable popularity in the neural network community. The primary properties in favor of the method are the simplicity at which it is implemented, and the modest requirement of data storage. In most situations, the drawback associated with slow convergence motivates the use of more sophisticated methods.

The category of fast algorithms uses standard numerical optimization techniques. Three types of numerical optimization techniques for neural network training have been incorporated in MATLAB: Conjugate gradient; quasi-Newton; and Levenberg–Marquardt.

The reader is advised to refer to the literature [137–143] for details on improvements suggested above.

11.10 RADIAL BASIS FUNCTION NETWORKS

Radial basis function (RBF) networks have gained considerable attention as an alternative to Multi-Layer Perceptrons (MLP) trained by the backpropagation algorithm. Both MLP networks and RBF networks are the basic constituents of feedforward neural networks.

An RBF neuron uses radially symmetric activation function $\phi(\|\mathbf{x} - \mathbf{c}\|)$, i.e., the argument of the function is the Euclidean distance of the input vector $\mathbf{x}$ from a center $\mathbf{c}$, which justifies the name *radial basis function* (RBF). Function $\phi(\cdot)$ can take various forms; the *Gaussian* form is more widely used.

A Gaussion basis function is typically parameterized by two parameters: the *center* which defines its position, and a *spread* parameter that determines its shape. The spread parameter is equal to the standard deviation σ in case of a one-dimensional Gaussian function (do not confuse the standard deviation parameter σ with the sigmoidal activation function $\sigma(\cdot)$). In the case of a multivariate input vector $\mathbf{x}$, the parameters that define the shape of the hyper-Gaussian function are elements of a covariance matrix $\mathbf{\Sigma}$. With the selection of the same spread parameter σ for all components of the input vector, the covariance matrix $\mathbf{\Sigma} = \text{diag}(\sigma^2)$.

The input vector

$$\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^T$$

and the output $\phi(\cdot)$ of an RBF (Gaussian) neuron are related by the following equation.

$$\phi(\mathbf{x},\mathbf{c},\sigma) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2}\right) \tag{11.38}$$

where $\mathbf{c}$ is the center and σ is the spread parameter of the Gaussian function.

Unlike sigmoidal neuron, there are no connection weights between the input terminals and the RBF unit (refer to Fig. 11.19); the center $\mathbf{c}$ and the spread parameter σ represent the weights.

RBF networks are structurally equivalent to the two-layer perceptron network shown in Fig. 11.13. Both have one hidden layer with a nonlinear activation function, and an output layer containing one or more neurons with linear activation functions. In an RBF network, one does not augment, both the n-dimensional input vector $\mathbf{x}$ and the hidden layer output vector with the bias term +1.

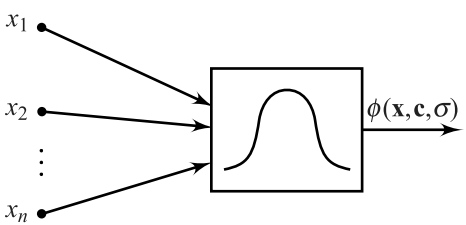


Fig. 11.19 Gaussian function in RBF neuron model

The architecture of an RBF network is presented in Fig. 11.20. The network consists of n inputs $x_1, x_2, ..., x_n$; and a hidden layer of m nonlinear processing units (refer to Eqn. (11.38)):

$$\phi_\ell(\mathbf{x},\mathbf{c}_\ell,\sigma_\ell) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_\ell\|^2}{2\sigma_\ell^2}\right);\ \ell = 1, 2, ..., m \tag{11.39a}$$

The output of the network is computed as a weighted sum of the outputs of the RBF units:

$$\hat{y}_j = \sum_{\ell=1}^{m} w_{j\ell}\phi_\ell(\mathbf{x},\mathbf{c}_\ell,\sigma_\ell);\ j = 1, 2, ..., q \tag{11.39b}$$

where $w_{j\ell}$ is the connection weight between the RBF unit ℓ and the j^{th} component of the output vector.

It follows from equations (11.39) that the parameters ($\mathbf{c}_\ell$, σ_ℓ, $w_{j\ell}$) govern the mapping properties of the RBF neural network. It has been shown [141] that the RBF network can fit any arbitrary function with just one hidden layer.

In the RBF network, the output of each RBF node is the same for all input points $\mathbf{x}$ having the same Euclidean distance from the respective centers $\mathbf{c}_i$, and decreases exponentially with the distance. In contrast, the output of each sigmoidal node in a multilayer perceptron network saturates to the same

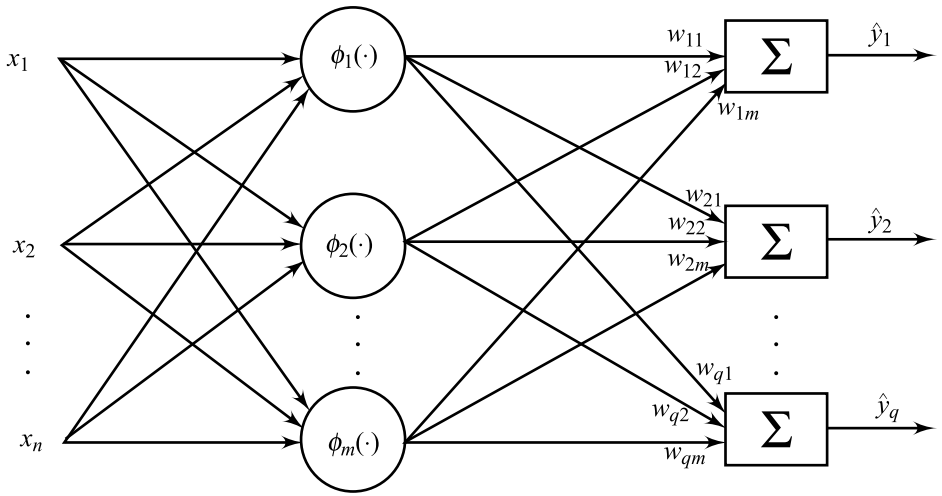


Fig. 11.20 RBF network architecture

value with increasing $\sum_i w_i x_i$. In other words, *the activation responses of the nodes are of a local nature in the RFB and of a global nature in the multilayer perceptron networks*.

This intrinsic difference has important repercussions for both the convergence speed and the generalization performance. In general, multilayer perceptrons learn slower than their RBF counterparts. In contrast, multilayer perceptrons exhibit improved generalization properties, especially for the regions that are not represented sufficiently in the training set.

Training RBF Networks

There are two sets of parameters governing the mapping properties of RBF networks: the weights $w_{j\ell}$; $j = 1, 2, ..., q$; $\ell = 1, 2, ..., m$, in the output layer, and the parameters $\{\mathbf{c}_\ell, \sigma_\ell\}$ of the radial basis functions. We select an appropriate cost function:

$$E = \frac{1}{2}\sum_{j=1}^{q}(y_j - \hat{y}_j)^2 ; \hat{y}_j = \sum_{\ell=1}^{m} w_{j\ell}\,\phi_\ell(\mathbf{x}, \mathbf{c}_\ell, \sigma_\ell) \tag{11.40}$$

The estimation of the weights $w_{j\ell}$, the centers $\mathbf{c}_\ell$, and the variances σ_ℓ^2 becomes a typical task of nonlinear optimization process:

$$w_{j\ell}(k+1) = w_{j\ell}(k) - \eta_1 \frac{\partial E(k)}{\partial w_{j\ell}(k)} \tag{11.41a}$$

$$\mathbf{c}_\ell(k+1) = \mathbf{c}_\ell(k) - \eta_2 \frac{\partial E(k)}{\partial \mathbf{c}_\ell(k)} \tag{11.41b}$$

$$\sigma_\ell(k+1) = \sigma_\ell(k) - \eta_3 \frac{\partial E(k)}{\partial \sigma_\ell(k)} \tag{11.41c}$$

k is the iteration index.

The computational complexity of such a scheme is prohibitive for a number of practical situations. When obtaining gradient information is difficult or expensive, we may use *genetic algorithm* for the nonlinear optimization problem (discussed later in Chapter 13).

Several alternative schemes have been proposed for training RBF networks. Some of these schemes learn only the centers $\mathbf{c}_\ell$ of the RBF units, and therefrom determine the spread parameters σ_ℓ. The basic idea is to ensure suitable overlapping of the basis functions. A rule of thumb is to take σ_ℓ equal to, or a multiple of, the average distance to the several nearest neighbors of $\mathbf{c}_\ell$ ($\|\mathbf{c}_{\ell+1} - \mathbf{c}_\ell\|$).

Once the centers and the spread parameters are chosen, the weights $w_{j\ell}$ in the output layer of the network in Fig. 11.20, can be determined as follows. Output neuron j is driven by the signals $\phi_\ell(\mathbf{x}, \mathbf{c}_\ell, \sigma_\ell)$ produced by the layer of RBF neurons, which are themselves driven by the input vector (stimulus) $\mathbf{x}$ applied to the input terminals. Supervised learning may be visualized as learning with the help of a 'supervisor' having knowledge in the form of input-output examples $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1, 2, ..., P\}$. For known RBF centres and spread parameters, this knowledge may be translated (refer to Eqns (11.39a)) in the form : $\{\boldsymbol{\phi}^{(p)}, \mathbf{y}^{(p)}; p = 1, ..., P\}$. Neuron j in the output layer is driven by the vector $\boldsymbol{\phi}^{(p)}$. By virtue of the built-in knowledge, the supervisor is able to provide the neural network with a desired response $y_j^{(p)}$ from $\boldsymbol{\phi}^{(p)}$. The network parameters $w_{j\ell}$ are adjusted under the combined influence of the training vector $\boldsymbol{\phi}^{(p)}$ and the error $e_j^{(p)} = y_j^{(p)} - \hat{y}_j^{(p)}$, which is the difference between the desired response $y_j^{(p)}$ and the actual response $\hat{y}_j^{(p)}$ (refer to Eqns (11.39b)) of the network. The least squares estimation or the gradient descent algorithm (refer to Section 11.7) may be used for learning the weights $w_{j\ell}$.

Fixed Centers Although there exist some cases in which the nature of the problem suggests a specific choice for the centers, in the general case, these centers may be selected randomly from the training set. Provided that the training set is distributed in a representative manner over the space of all the patterns (input vectors), this seems to be a reasonable way to choose the m centers.

Training the Centers If the centers are not preselected, they have to be estimated during the training phase along with the weights $w_{j\ell}$. This can be achieved by unraveling the *clustering* (unsupervised learning) properties of the data, and choosing a representative of each cluster as the corresponding center. The *Self-Organizing Map* (SOM), developed by Kohonen, is an unsupervised, clustering network. Proper clusters are formed by discovering the similarities and dissimilarities among the input data [141].

These techniques of RBF network training are used in MATLAB.

11.11 SYSTEM IDENTIFICATION WITH NEURAL NETWORKS

The main goal of the present chapter is to describe approaches to neural-network-based control that are found to be practically applicable to a reasonably wide class of unknown nonlinear systems. Systems identification is an integral part of such a control system design and, consequently, it calls for considerable attention as well. The system identification is necessary to establish a model based on which the controller can be designed, and it is useful for tuning and simulation before applying the controller to the real system. In this section, attention is drawn to identification of neural network models for nonlinear dynamic systems from a series of measurements on the systems.

We give here a generic working procedure for system identification with neural networks. Time-invariant nonlinear dynamic systems with scalar input and scalar output are considered here. Extension to the case of vector input and vector output is straight forward.

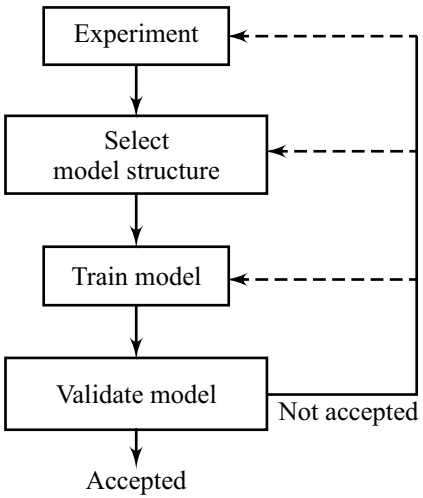


Fig. 11.21 System identification procedure

The multilayer feedforward network is straightforward to employ for the discrete-time modeling of dynamic systems for which there is a nonlinear relationship between the system's input and output. Let k count the multiple sampling periods so that $y(k)$ specifies the present output while $y(k-1)$ signifies the output observed at the previous sampling instant, etc. It is assumed that the output of the dynamic system at discrete-time instances can be described as a function of number of past inputs and outputs:

$$y(k) = f(y(k-1), ..., y(k-n), u(k-1), ..., u(k-m)); \; n \geq m \tag{11.42}$$

A multilayer network can be used for approximating $f(\cdot)$ if the inputs to the network are chosen as the n past outputs and m past inputs of the dynamic system.

When attempting to identify a model of a dynamic system, it is a common practice to follow the procedure depicted in Fig. 11.21.

11.11.1 Experiment

The primary purpose of an experiment is to produce a set of examples of how the dynamic system to be identified responds to various control inputs (These examples can later be used to train neural network to model the system). The experiment is particularly important in relation to nonlinear modeling; one must be extremely careful to collect a set of data that describes how the system behaves over its entire range of operation. The following issues must be considered in relation to acquisition of data (For detailed information, refer to [131]).

Sampling Frequency

The sampling frequency should be chosen in accordance with the desired dynamics of the closed-loop system consisting of controller and the system. A high sampling frequency permits a rapid reference tracking and a smoother control signal, but the problems with numerical ill-conditioning will become more pronounced. Consequently, the sampling frequency should be selected as a sensible compromise.

Input Signals

While for identification of linear systems, it is sufficient to apply a signal containing a finite number of frequencies, a nonlinear system demands, roughly speaking, that all combinations of frequencies and amplitudes in the system's operating range are represented in the signal. As a consequence, the necessary

size of the data set increases dramatically with the number of inputs and outputs. Unfortunately, there is no obvious remedy to this curse of dimensionality.

Before an input signal is selected, it is important to identify the operating range of the system. Special care must be taken not to excite dynamics that one does not intend to incorporate in the model (e.g., mechanical resonances).

Processing the Data

Intelligent processing of the data is often much more important than trying a large number of different model structures and training schemes. Many different types of processing can be considered for extracting the most valuable information from the measured data, and to make it suitable for neural-network modeling. Some suggestions are given in the following paragraphs.

Filtering is widely used for removing from the measured signals, noise, periodic disturbances, offsets, and the effects of 'uninteresting' dynamics. When high-frequency noise/disturbances cause problems, it is recommended to remove them by using an analog presampling filter to avoid an aliasing phenomenon. Offset, drift, and low-frequency disturbances can be removed by filtering the data after sampling.

Sometimes, a large number of input-output pairs from a small regime of entire operating range, dominates the data set. When training on such a data set, it is likely that the model obtained will be very accurate in the regime that was over-represented at the expense of poor performance outside the regime. A little 'surgery' on the data set might be necessary here to eliminate redundant information. Apart from obtaining a more equal weighting of the information, a reduction of the data set size also has the benefit that training times will be reduced.

It is also recommended to remove outliers from the data set (or, alternatively, insert interpolated values of the output signal). Outliers will often have a fatal impact on the training model.

Before training, it is often useful to scale all the signals so that they always fall within a specified range, say [–1, 1]. Another approach for scaling is to normalize the mean and standard deviation of the training set, e.g., to zero mean and unity standard deviation. The signals are likely to be measured in different physical units, and without scaling there is a tendency that the signal of largest magnitude will be too dominating. Moreover, scaling makes the training algorithm numerically robust and leads to faster convergence.

11.11.2 Model Structure Selection

The model structure selection is basically concerned with the following two issues:

- Selecting an internal network architecture
- Selecting the inputs to the network

An often-used approach is to let the internal architecture be feedforward multilayer network. Probably the most commonly used network architecture is a two-layer feedforward network with sigmoidal/hyperbolic tangent hidden units and linear output units. This architecture works quite well in many practical applications. In our presentation, we use this architecture. However, the reader is referred to more fundamental textbooks/research papers for a treatment of other types of neural networks in the control loop.

The input structure we use here consists of a number of past inputs and outputs (refer to Fig. 11.22):

$$\hat{y}(k \mid \boldsymbol{\theta}) = \sum_{\ell=1}^{M} v_\ell \sigma\left(\sum_{i=1}^{N} w_{\ell i}\phi_i(k) + w_{\ell 0} \right) + v_0 \tag{11.43a}$$

where $\hat{y}$ is the predicted value of the output y at sampling instant $t = kT$ (T = sampling interval), $\boldsymbol{\theta} = \{v_\ell \quad w_{\ell i}\}$ is the vector containing the adjustable parameters in the neural network (*weights*), $\boldsymbol{\phi}$ is the *regression vector* which contains past outputs and past inputs (regressors's dependency on the weights is ignored):

$$\begin{aligned} \boldsymbol{\phi}(k) &= [y(k-1) \cdots y(k-n)\ u(k-1) \cdots u(k-m)]^T \\ &= [\phi_1(k) \quad \phi_2(k) \cdots \phi_N(k)]^T \end{aligned} \tag{11.43b}$$

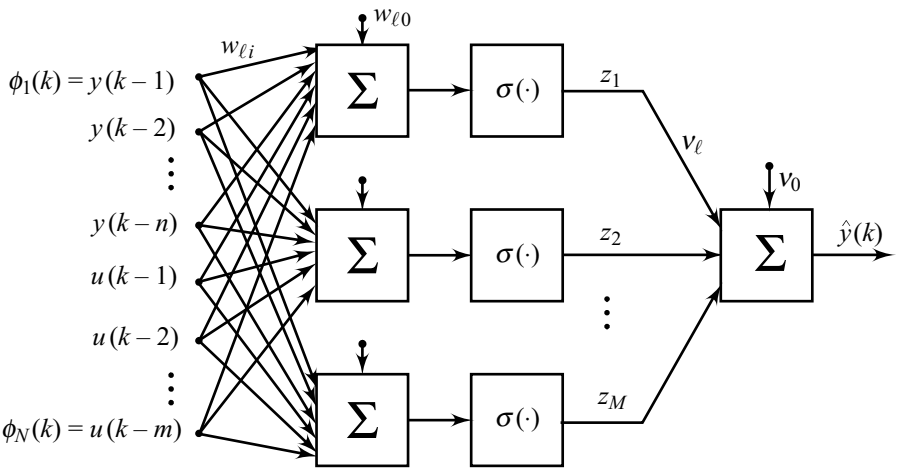


Fig. 11.22 Input structure

Often, it is of little importance that the network architecture has selected vector $\boldsymbol{\theta}$ too small or too large. However, a wrong choice of *lag space*, i.e., the number of delayed signals used as regressors, may have a disastrous impact on some control applications. Too small obviously implies that essential dynamics will not be modeled, but too large can also be a problem. From the theory of linear systems, it is known that too large a lag space may manifest itself as common factors in the identified transfer function. An equivalent behavior must be expected in the nonlinear case. Although it is not always a problem, common factors (corresponding to hidden modes) may lead to difficulties in some of the controller designs.

It is necessary to determine both, a sufficiently large lag space and an adequate number of hidden units. While it is difficult to apply physical insight towards the determination of number of hidden units, it can often guide the proper lag space. If the lag space is properly determined, the model structure selection problem is substantially reduced. If one has no idea regarding the lag space, it is sometimes possible to determine it empirically.

11.11.3 Training

Assume now, that a data set has been acquired and that some model structure has been selected. According to the identification procedure in Fig. 11.21, the next step is to apply the data set to pick 'the best' model among the candidates contained in the model structure. This is the *training stage*. The training can be computationally intensive, but it is generally one of the easiest stages in the identification. It is not very difficult to implement a training algorithm in a computer, but one might as well resort to one of the many available software packages, e.g., MATLAB.

The training procedure can be rephrased in more formal terms. An experiment is performed on the time-invariant nonlinear dynamic system to collect a set of data, that describes how the system behaves over its entire range of operation:

$$\text{Experimental data: } \{[u(t), y(t)];\ t = 1,2,3,...\} \tag{11.44a}$$

From the experimental data, we generate the training data. Since the system is assumed to be time-invariant, the experimental data could be equivalently represented as

$$\{[u(t), y(t)];\ t = -n+1, -n+2,...,0,1,2,...\}$$

The following P pairs $\{\boldsymbol{\phi}(k), y(k);\ k = 1,2,...,P\}$, are used for training the neural network (refer to Eqns (11.43)):

$$\begin{aligned}
\boldsymbol{\phi}(1) &= [y(0)\, y(-1)\cdots y(1-n)\, u(0)\, \cdots\, u(1-m)]^T;\ y(1)\\
\boldsymbol{\phi}(2) &= [y(1)\, y(0)\cdots y(2-n)\, u(1)\cdots u(2-m)]^T;\ y(2)\\
&\vdots\\
\boldsymbol{\phi}(P) &= [y(P-1)\, \cdots\, y(P-n)\, \cdots\, u(P-1)\, \cdots\, u(P-m)]^T;\ y(P)
\end{aligned} \tag{11.44b}$$

The purpose of the training is to determine a mapping from the data set to the set of candidate models

$$\hat{y}(k\,|\,\boldsymbol{\theta}) = g[\boldsymbol{\phi}(k), \boldsymbol{\theta}] \tag{11.45}$$

so that a model is obtained which provides predictions that are in some sense close to the true outputs of the system. The most commonly used measure of closeness for this type of problems is specified in terms of a mean square error criterion

$$J(\boldsymbol{\theta}) = \frac{1}{2P}\sum_{k=1}^{P}[y(k) - \hat{y}(k\,|\,\boldsymbol{\theta})]^2 \tag{11.46}$$

The most appealing feature of mean square error criterion is the simplicity with which a weight update rule can be derived. The principle of the gradient (descent) iterative search method, is that at each iteration, the weights are modified along the opposite direction of the gradient. That is, the search direction is selected as $-\dfrac{\partial J}{\partial \boldsymbol{\theta}}$.

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta^{(i)}\frac{\partial J}{\partial \boldsymbol{\theta}^{(i)}} \tag{11.47}$$

When applying the gradient method to the training of multilayer feedforward networks, it is useful to order the computations in a fashion that utilizes the particular structure of the network. The method, called the *backpropagation algorithm*, was discussed in Section 11.9. Batch method of the backpropagation algorithm refers to the fact that each iteration on the parameter vector requires an evaluation of the entire data set.

It is sometimes useful to identify a system online, simultaneously, with the acquirement of measurements. Adaptive control is an example of such an application. In this case, a model must be identified and a control system designed online, because the dynamics of the system to be controlled vary with time. Obviously *batch methods* are unsuitable in such applications as the amount of computation, required in each iteration, might exceed the time available within one sampling interval. Moreover, old data will be obsolete when the system to be identified is time-dependent.

In a *recursive algorithm*, one input-output pair from the training set, $[\phi(k), y(k)]$, is evaluated at a time and used for updating the weights. In the neural network community, this is frequently referred to as *incremental* or *online* backpropagation (refer to Section 11.9).

11.11.4 Validation

In the validation stage, the trained model is evaluated to clarify if it represents the underlying system adequately. Ideally, the validation should be performed in accordance with the intended use of the model. As it turns out, this is often rather difficult. For instance, if the intention is to use the model for designing a control system, the validation ought to imply that a controller was designed and its performance tested in practice. For most applications, this level of ambition is somewhat high, and it is common to apply a series of simple 'standard' tests instead of concentrating on investigating particular properties of the model. Although this is less than ideal, it is good as a preliminary validation to quickly exclude really poor models.

Most of the tests require a set of data that was not used during training. Such a data set is commonly known as *test* or *validation* set. It is desirable that the test set satisfies the same demands as the *training set*, regarding representation of the entire operating range.

A very important part of the validation is to simply inspect the plot, comparing observed outputs to predictions. Unless the signal-to-noise ratio is very poor, it can show the extent of *overfitting* as well as possible systematic errors.

If the sampling frequency is high, compared to the dynamics of the system, a visual inspection of the predictions will not reveal possible problems. Some scalar quantities (correlation functions) to measure the *accuracy* of the predictions, have been suggested. Reliable estimates of the *average generalization error* are also useful for validation purposes, but their primary application is for model structure selection. The estimates are good for rapidly comparing different model structures to decide which one is likely to be the best.

11.12 CONTROL WITH NEURAL NETWORKS

Neural-network-based control constitutes a very large research field, and it is difficult to give a clear overview of the entire field. Here, in this section, an attempt has been made to outline a feasible path through the 'jungle' of neural network solutions. A completely automatic procedure for control system design is not realistic; the emphasis is on the guidelines for working solutions.

It is believed that one of the most important lessons to be learnt from the numerous automatic control applications developed over the past half century, is that *simple solutions* actually solve most problems

quite well. Regardless of the fact that all systems, to some extent, exhibit a nonlinear behavior, it turns out that they can often be controlled satisfactorily with simple *linear* controllers. When neural networks are introduced as a tool for improving the performance of control systems for a general class of unknown *nonlinear* systems, it should be done in the same spirit. A consequence of this philosophy is that our focus is on simple control structures that yield good performance in practice.

11.12.1 Inverse Model of the System used as Controller

When neural networks were originally proposed for controlling unknown nonlinear systems, one of the first methods being reported was on training a network to act as the inverse of the system, and use this as a controller. This, in fact, amounts to linearization (the input-output transfer function unity) of the nonlinear system by properly compensating for the nonlinearity involved (refer to Section 10.2). Explained in brief, the basic principle is as follows:

Assume that the system to be controlled can be described by

$$y(k) = f_1[y(k-1), ..., y(k-n), u(k-1),..., u(k-m)] \tag{11.48}$$

Perform an experiment on the system to collect a set of data, that describes how the system behaves over its entire range of operation:

$$\{[u(k), y(k)], k = 1, 2, 3, ... \} \tag{11.49}$$

Using identification procedures described in the earlier section, we can infer a neural network model of the system using this data set.

An inverse model of the system can be inferred from the data set

$$\{[y(k), u(k)], k = 1, 2, 3, ... \} \tag{11.50}$$

The output of the inverse model is $u(k)$:

$$u(k) = f_2[(y(k+1), y(k), ..., y(k-n+1), u(k-1),..., u(k-m+1)] \tag{11.51}$$

The inverse model can be used as controller for the system. Let the 'desired' closed-loop system behave as

$$\frac{Y(z)}{R(z)} = M(z) = z^{-1}; y(k+1) = r(k) \tag{11.52}$$

Substitute in Eqn. (11.51), the output $y(k+1)$ by the desired output—the reference, $r(k)$. If the network represents the exact inverse, the control input produced by it will drive the system output at time $k+1$ to $r(k)$. The principle is illustrated in Fig. 11.23a.

The most straightforward way of training a network as the inverse of a system, is to approach the problem as a system-identification problem analogous to the one considered in the previous section—an experiment is performed, a network architecture is selected, and the network is trained off-line. The difference from system identification lies in the choice of regressors and network output. They are now selected as shown in a functional relation (11.51). The network is then trained to minimize the criterion

$$J = \frac{1}{2P} \sum_{k=1}^{P} [u(k) - \hat{u}(k|\boldsymbol{\theta})]^2 \tag{11.53}$$

We will call this procedure, the *general training* procedure for an inverse model.

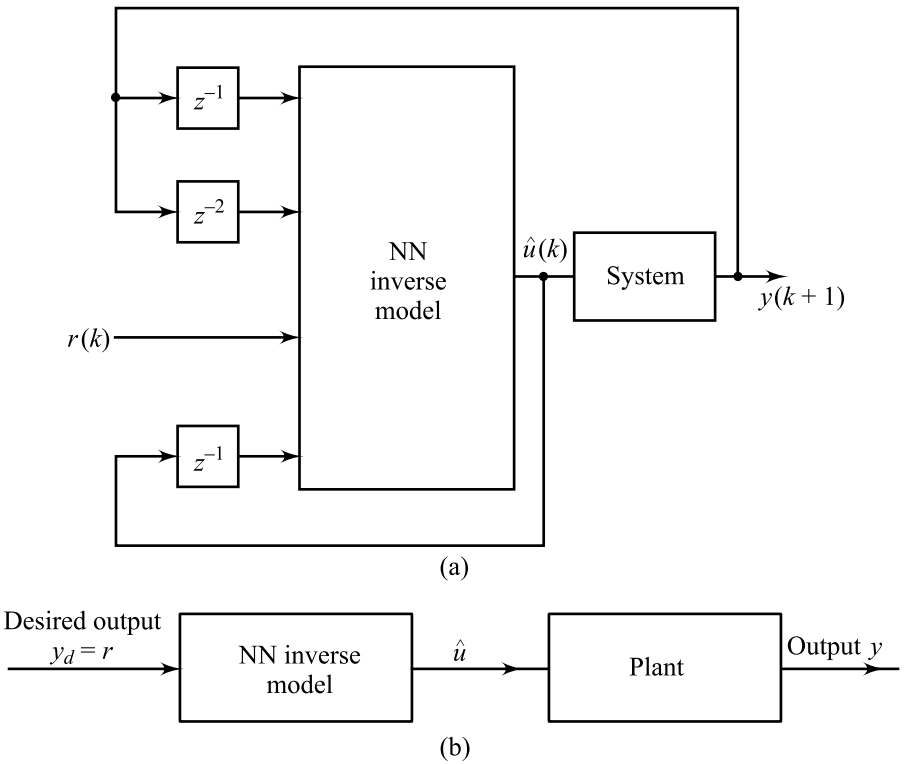


Fig. 11.23 Inverse model of the system as a controller

This is basically an off-line procedure. After this training phase, the structure for an on-line operation looks like the one shown in Fig. 11.23b, that is, the neural network representing the inverse of the plant precedes the plant. The trained neural network should be able to take the desired output value $y_d = r$ as its input, and produce appropriate $\hat{u}$ as an input to the plant.

The practical relevance of using an inverse model of the system as a controller is limited, due to a number of serious inconveniences. The control scheme will typically result in a poor robustness with a high sensitivity to noise and high-frequency disturbances (corresponding to unity forward-path transfer function in the linear case). In addition, one will often encounter a very active control signal, which may adversely affect the system/actuators. If the system is linear, this occurs when its zeros are situated close to the unit circle. In the nonlinear case, there is no unique set of zeros, but, of course, a similar phenomenon exists.

If the inverse model is unstable (corresponding to zeros of the system outside the unit circle in the linear case), one must anticipate that the closed-loop system becomes unstable. Unfortunately, this situation occurs quite frequently in practice. Discretization of linear continuous-time models under quite common circumstances, can result in zeros outside the unit circle—regardless that the continuous-time model has no zeros, or all zeros are in the left half of the plane. In fact, for a model with a pole excess of at least two, one or more zeros in the discretized model will converge to the unit circle, or even outside, as the sampling frequency is increased. It must be expected that a similar behavior can also be found in discrete models of nonlinear systems.

Another problem with the design arises when the system to be controlled is not one-to-one, since then a unique inverse model does not exist. If this non-uniqueness is not reflected in the training set, one can, in principle, yield a particular inverse which might be adequate for controlling the system. Most often, however, one will end up with a useless, incorrect inverse model.

Despite these drawbacks, in a number of domains (stable systems, and one-to-one mapping plants), this general training architecture is a viable alternative.

11.12.2 Feedforward-Feedback Control

Many of the problems mentioned in the previous subsection can be taken care of by employing a control structure of the form shown in Fig. 11.24. The feedforward control is used for improving the reference tracking, while feedback is used for stabilizing the system and for suppressing disturbances.

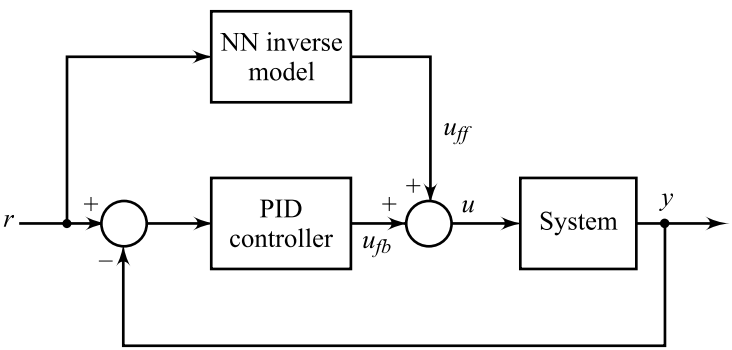


Fig. 11.24 Feedforward-feedback control structure

An inverse model is trained as discussed earlier (refer to Eqn. (11.51)):

$$u(k) = f[y(k+1), y(k), ..., y(k-n+1), u(k-1), ..., u(k-m+1)] \tag{11.54}$$

The feedforward component of the control input is then composed by substituting all system outputs by corresponding reference values:

$$u_{ff}(k) = f[r(k+1), ..., r(k-n+1), u_{ff}(k-1), ..., u_{ff}(k-m+1)] \tag{11.55}$$

If the complete reference trajectory is known in advance, implementation of the scheme is particularly easy. It is then possible to compute the contribution from the feedforward controller beforehand, and store the entire sequence of control inputs $\{u_{ff}\}$ for use in the computer program implementing the control system.

Although a neural network feedforward can be useful for optimizing many control systems, one must be careful not to use it uncritically. An inaccurate feedforward control may actually harm, rather than enhance, performance.

11.12.3 Model Reference Adaptive System

In the context of training inverse models, which are to be used as controllers, the trained inverse model, somehow, ought to be validated in terms of performance of the final closed-loop system. This points out

a serious weakness associated with the general training procedure for an inverse model: the criterion (11.53) expresses the objective to minimize the discrepancy between the network output and a sequence of 'true' control inputs. This is not really a relevant objective. In practice, it is not possible to achieve zero generalization error and consequently, the trained network will have certain inaccuracies. Although these are reasonably small in terms of the network output being close to the ideal control signal, there may be large deviations between the reference and the output of the system when the network is applied as controller for the system. The weakness lies in the fact that the training procedure is not *goal directed*. The goal is that, in some sense, the system output should follow the reference signal closely. It would be more desirable to minimize a criterion of the following type:

$$J = \frac{1}{2P}\sum_{k=1}^{P}[r(k) - y(k)]^2 \tag{11.56a}$$

which clearly is goal directed. Unfortunately, the minimization of this criterion is not easily carried out off-line, considering that the system output, $y(k)$, depends on the output of the inverse model, $u(k-1)$.

Inspired by the recursive training algorithms, the network might alternatively be trained to minimize

$$J_k = J_{k-1} + [r(k) - y(k)]^2 \tag{11.56b}$$

This is an on-line approach and, therefore, the scheme constitutes an *adaptive controller*.

Assuming that J_{k-1} has already been minimized, the weights at time k are adjusted according to

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) - \eta\,\frac{de^2(k)}{d\boldsymbol{\theta}} \tag{11.57a}$$

where

$$e(k) = r(k) - y(k) \tag{11.57b}$$

and

$$\frac{de^2(k)}{d\boldsymbol{\theta}} = -\frac{dy(k)}{d\boldsymbol{\theta}}\,e(k) \tag{11.57c}$$

By application of the chain rule, the gradient $\dfrac{dy(k)}{d\boldsymbol{\theta}}$ can be calculated:

$$\frac{dy(k)}{d\boldsymbol{\theta}} = \frac{\partial y(k)}{\partial u(k-1)}\,\frac{du(k-1)}{d\boldsymbol{\theta}} \tag{11.58a}$$

Jacobians of the system, $\dfrac{\partial y(k)}{\partial u(k-1)}$, are required. These are generally unknown since the system is unknown. To overcome this problem, a *forward model* of the system is identified to provide estimates of the Jacobians:

$$\frac{\partial y(k)}{\partial u(k-1)} \approx \frac{\partial \hat{y}(k)}{\partial u(k-1)} \tag{11.58b}$$

The forward model is obtained by the system identification procedure described in the earlier section.

Fortunately, inaccuracies in the forward model need not have a harmful impact on the training. The Jacobian is a scalar factor and, in the simplified algorithm (11.57), will only change the step-size of the algorithm. Thus, as long as the Jacobians have the correct sign, the algorithm will converge if the step-size parameter is sufficiently small. We will call this procedure the *specialized training* procedure for the inverse model.

The deadbeat character, appearing when inverse models are used directly as controllers, will often result in an unnecessarily fast response to reference changes. An active control signal may even harm the system or the actuators. Consequently, it might be desirable to train the network to achieve some prescribed low-pass behavior of the closed-loop system. Say, have the closed-loop system following the model:

$$y_m(k) = \frac{B_m(z^{-1})}{A_m(z^{-1})} r(k) \tag{11.59}$$

The polynomials A_m and B_m are selected arbitrarily, by the designer.

The control design is, in this case, related to 'Model Reference Adaptive System' (MRAS); a popular type of adaptive controller (discussed earlier in Section 10.3).

Since this *specialized training* is an on-line approach, the combination of having many weights to adjust and having only the slow convergence of a gradient method, will often be disastrous. Before the weights are properly adjusted, the system may have been driven outside the operating range with possibly serious consequences. Often *general training* can be used to provide a decent initialization of the network so that the specialized training is only used for 'fine tuning' of the controller.

The simplified specialized training is quite easily implemented with the backpropagation algorithm (refer to Fig. 11.25). This algorithm is applied on the inverse model NN2:

$$u(k-1) = f[y(k+1), y(k), ..., y(k-n+1), u(k-2),..., u(k-m)]$$

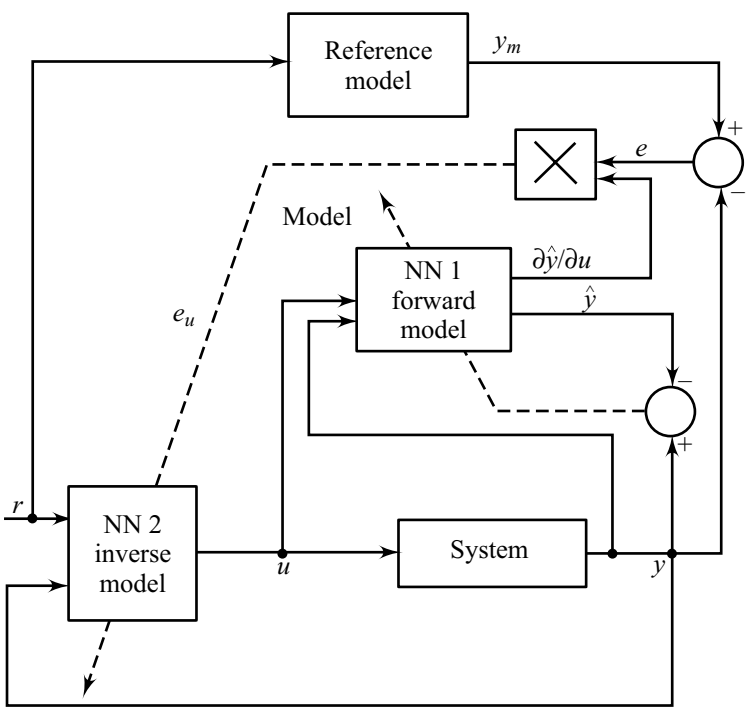


Fig. 11.25 Specialized training

by assuming the following 'virtual' error $e_u(k)$ on the output of the controller:

$$\frac{de^2(k)}{d\boldsymbol{\theta}} = -\frac{dy(k)}{d\boldsymbol{\theta}}e(k) = -\frac{\partial\hat{y}(k)}{\partial u(k-1)}\frac{du(k-1)}{d\boldsymbol{\theta}}e(k) = -\frac{du(k-1)}{d\boldsymbol{\theta}}\boldsymbol{e}_u(k) \simeq -\frac{\partial u(k-1)}{\partial\boldsymbol{\theta}}e_u(k)$$

where
$$e_u(k) = \frac{\partial\hat{y}(k)}{\partial u(k-1)}\,e(k) \tag{11.60}$$

$$\frac{\partial\hat{y}(k)}{\partial u(k-1)} \simeq \frac{NN1(u(k-1)+\varepsilon) - NN1(u(k-1))}{\varepsilon}$$

A better estimate of the derivative is obtained as follows:

For a multilayer feedforward network (NN 1) with one hidden layer of sigmoid units and a linear output (Fig. 11.22),

$$\hat{y}(k) = \sum_{\ell=1}^{M} v_\ell \sigma\left(\sum_{i=1}^{N} w_{\ell i}\phi_i(k) + w_{\ell 0}\right) + v_0 \tag{11.61a}$$

$$\boldsymbol{\phi}(k) = [y(k-1), ..., y(k-n), u(k-1), ..., u(k-m)] \tag{11.61b}$$

The derivative of the output with respect to the regressor $\phi_i(k)$, is given by

$$\frac{\partial\hat{y}(k)}{\partial\phi_i(k)} = \sum_{\ell=1}^{M} v_\ell w_{\ell i}\sigma'(a) \tag{11.62a}$$

$$= \sum_{\ell=1}^{M} v_\ell w_{\ell i}\sigma(a)\,[1-\sigma(a)]$$

where
$$a = \sum_{i=1}^{N} w_{\ell i}\phi_i(k) + w_{\ell 0} \tag{11.62b}$$

We can obtain $\partial\hat{y}(k)/\partial u(k-1)$ from this relation.

11.13 SUPPORT VECTOR MACHINES

Support vector machine (SVM) theory provides the most principled approach to the design of neural networks. Statistical learning theory provides a sound mathematical basis for the formulation of support vector machines. SVM theory applies to pattern classification and nonlinear regression, using any one of the following network architectures: RBF networks, MLPs with a single hidden layer, and polynomial machines. For each of these feedforward networks, we may use the support vector learning algorithm to implement the learning process using a given set of training data; automatically determining the required number of hidden units.

Our interest in control problems is more on nonlinear regression. To explain how a support vector machine works for regression problems, it is perhaps easiest to start with the case of linearly separable patterns that could arise in the context of binary pattern classification. In this context, the main idea of a support vector machine is to construct a hyperplane as the decision surface in such a way that the

margin of separation between Class 1 and Class 2 examples is maximized. We will then take up the more difficult case of linearly nonseparable patterns. With the material on how to find the optimal hypersurface for linearly nonseparable patterns at hand, we will formally describe the construction of a support vector machine for real-life (nonlinear) pattern recognition task. As we shall see shortly, basically the idea of a support vector machine hinges on the following two mathematical operations:

(i) Nonlinear mapping of input patterns into high-dimensional feature space.
(ii) Construction of optimal hyperplane for linearly separating the features discovered in Step (i).

The final stage of our presentation will be to extend these results for application to nonlinear regression problems.

11.13.1 Hard-Margin Linear SVM

Our presentation on SVM begins with the easiest classification problem: binary classification of linearly separable data (separating functions will be hyperplanes). The presentation will gradually increase in complexity.

Let the set of training (data) examples $\mathcal{D}$ be

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_P, y_P)\} \tag{11.63}$$

where $\mathbf{x}_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{in} \end{bmatrix}^T$ is an n-dimensional *input vector* (pattern with n-features) for the ith example in a real-valued space $X \subseteq \Re^n$; y_i is its *class label* (output value), and $y_i \in \{+1, -1\}$. +1 denotes Class 1 and –1 denotes Class 2.

To build a classifier, SVM finds a linear function of the form

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{11.64}$$

so that the input vector $\mathbf{x}_i$ is assigned to Class 1 if $f(\mathbf{x}_i) \geq 0$, and to Class 2 if $f(\mathbf{x}_i) < 0$, i.e.,

$$y_i = \begin{cases} +1 & \text{If} \quad \mathbf{w}^T\mathbf{x}_i + b \geq 0 \\ -1 & \text{If} \quad \mathbf{w}^T\mathbf{x}_i + b < 0 \end{cases} \tag{11.65}$$

Hence $f(\mathbf{x})$ is a real-valued function $f : X \subseteq \Re^n \rightarrow \Re$.

$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}^T \in \Re^n$ is called the *weight vector* and $b \in \Re$ is called the *bias*.

In essence, SVM finds a hyperplane

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{11.66}$$

that separates Class 1 and Class 2 training examples. This hyperplane is called the *decision boundary or decision surface*. Geometrically, the hyperplane (11.66) divides the input space into two half spaces: one half for Class 1 examples and the other half for Class 2 examples. Note that hyperplane (11.66) is a line in a two-dimensional space and a plane in a three-dimensional space.

For linearly separable data, there are many hyperplanes (lines in two-dimensional feature space; Fig. 11.26) that can perform separation. How can one find the best one? The SVM framework provides good answer to this question. Among all the hyperplanes that minimize the training error, find the one

with the largest *margin*—the gap between the data points of the two classes. This is an intuitively acceptable approach: select the decision boundary that is far away from both the classes (Fig. 11.27). Large-margin separation is expected to yield good classification on previously unseen data, i.e., good generalization.

From linear algebra, we know that in $\mathbf{w}^T\mathbf{x} + b = 0$, $\mathbf{w}$ defines a direction perpendicular to the hyperplane. $\mathbf{w}$ is called the *normal vector* (or simply *normal*) of the hyperplane. Without changing the normal vector $\mathbf{w}$, varying b moves the hyperplane parallel to itself. Note also that $\mathbf{w}^T\mathbf{x} + b = 0$ has an inherent degree of freedom. We can rescale the hyperplane to $k\mathbf{w}^T\mathbf{x} + kb = 0$ for $k \in \Re^+$(positive real number), without changing the hyperplane.

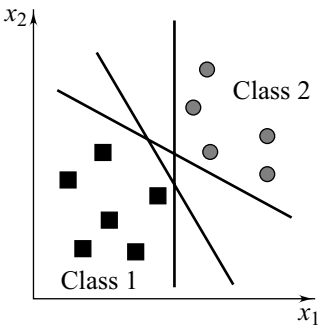


Fig. 11.26 Possible decision boundaries

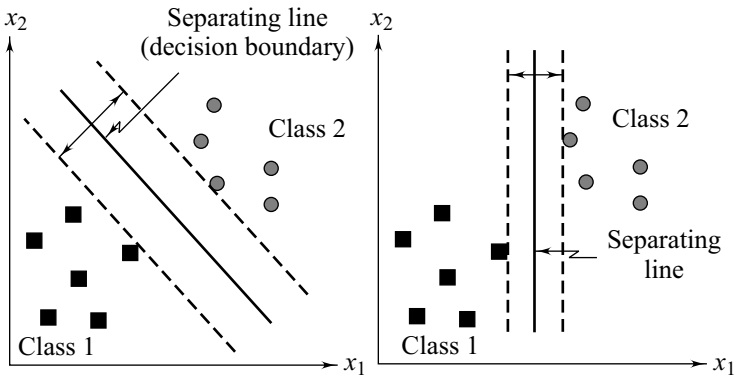


Fig. 11.27

Since SVM maximizes the margin between Class 1 and Class 2 data points, let us find the margin. The linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ gives an algebraic measure of the distance from $\mathbf{x}$ to the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$. We can express $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{x}_N + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{11.67}$$

where $\mathbf{x}_N$ is the normal projection of $\mathbf{x}$ onto the hyperplane and r is the desired algebraic distance (Fig. 11.28). Since by definition, $f(\mathbf{x}_N) = \mathbf{w}^T\mathbf{x}_N + b = 0$, it follows that

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \mathbf{w}^T\left(\mathbf{x}_N + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + b$$

$$= r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} = r\frac{(\|\mathbf{w}\|)^2}{\|\mathbf{w}\|} = r\|\mathbf{w}\|$$

or

$$r = \frac{f(\mathbf{x})}{\|\mathbf{w}\|} \tag{11.68}$$

Now consider a Class 1 data point $\left(\mathbf{x}^{(1)}, +1\right)$ that is closest to the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ (Fig. 11.28). The distance $d^{(1)}$ of this data point from the hyperplane is

$$d^{(1)} = \frac{f(\mathbf{x}^{(1)})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{x}^{(1)} + b}{\|\mathbf{w}\|} \tag{11.69a}$$

Similarly

$$d^{(2)} = \frac{f(\mathbf{x}^{(2)})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{x}^{(2)} + b}{\|\mathbf{w}\|} \tag{11.69b}$$

where $(\mathbf{x}^{(2)}, -1)$ is a Class 2 data point closest to the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$.

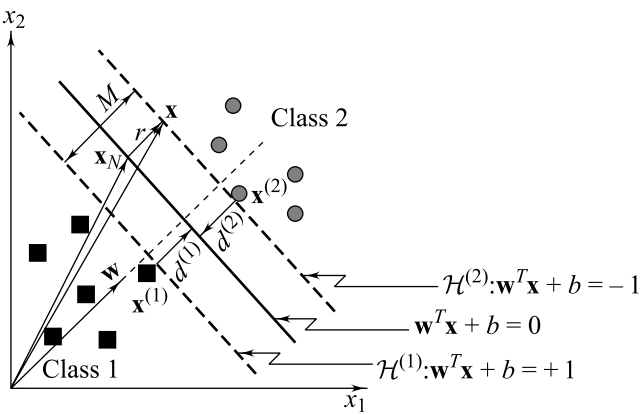


Fig. 11.28 Geometric interpretation of algebraic distances of points to a hyperplane for two-dimensional case

We define two parallel hyperplanes $\mathcal{H}^{(1)}$ and $\mathcal{H}^{(2)}$ that pass through $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, respectively. $\mathcal{H}^{(1)}$ and $\mathcal{H}^{(2)}$ are also parallel to the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$. We can rescale $\mathbf{w}$ and b to obtain (this rescaling, as we shall see later, simplifies the quest for significant patterns, called *support vectors*)

$$\begin{aligned} \mathcal{H}^{(1)} &: \mathbf{w}^T\mathbf{x}^{(1)} + b = +1 \\ \mathcal{H}^{(2)} &: \mathbf{w}^T\mathbf{x}^{(2)} + b = -1 \end{aligned} \tag{11.70}$$

such that

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \geq 1 \quad &\text{if} \quad y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad &\text{if} \quad y_i = -1 \end{aligned} \tag{11.71a}$$

or equivalently

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \tag{11.71b}$$

which indicates that no training data fall between hyperplanes $\mathcal{H}^{(1)}$ and $\mathcal{H}^{(2)}$. The distance between the two hyperplanes is the margin M. In the light of rescaling given by (11.70),

$$d^{(1)} = \frac{1}{\|\mathbf{w}\|};\ d^{(2)} = \frac{-1}{\|\mathbf{w}\|} \tag{11.72}$$

where the '–' sign indicates that $\mathbf{x}^{(2)}$ lies on the side of the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ opposite to that where $\mathbf{x}^{(1)}$ lies. From Fig. 11.28, it follows that

$$M = \frac{2}{\| \mathbf{w} \|} \tag{11.73}$$

Equation (11.73) states that maximizing the margin of separation between classes is equivalent to minimizing the Euclidean norm of the weight vector $\mathbf{w}$.

Since SVM looks for the separating hyperplane that minimizes the Euclidean norm of the weight vector, this gives us an optimization problem. A full description of the solution method requires a significant amount of optimization theory, which is beyond the scope of this book. We will only use relevant results from optimization theory, without giving formal definitions, theorems or proofs (refer to [29] for details).

Our interest here is in the following nonlinear optimization problem with inequality constraints:

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) \\ &\text{subject to} \quad g_i(\mathbf{x}) \geq 0;\ i = 1, \ldots, m \end{aligned} \tag{11.74}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$, and the functions f and g_i are continuously differentiable.

The optimality conditions are expressed in terms of the *Lagrangian function*

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \tag{11.75}$$

where $\boldsymbol{\lambda} = [\lambda_1 \cdots \lambda_m]^T$ is a vector of Lagrange multipliers.

An optimal solution to the problem (11.74) must satisfy the following necessary conditions, called *Karush–Kuhn–Tucker* (KKT) *conditions*:

$$\begin{aligned} &\text{(i)} \quad \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_j} = 0;\ j = 1, \ldots, n \\ &\text{(ii)} \quad g_i(\mathbf{x}) \geq 0;\ i = 1, \ldots, m \\ &\text{(iii)} \quad \lambda_i \geq 0;\ i = 1, \ldots, m \\ &\text{(iv)} \quad \lambda_i g_i(\mathbf{x}) = 0;\ i = 1, \ldots, m \end{aligned} \tag{11.76}$$

In view of condition (iii), the vector of Lagrange multipliers belongs to the set $\{\boldsymbol{\lambda} \in \Re^m,\ \boldsymbol{\lambda} \geq \mathbf{0}\}$. Also note that condition (ii) is the original set of constraints.

Our interest, as we will see shortly, is in convex functions f and linear functions g_i. For this class of optimization problems, when there exist vectors $\mathbf{x}^0$ and $\boldsymbol{\lambda}^0$ such that the point $(\mathbf{x}^0, \boldsymbol{\lambda}^0)$ satisfies the KKT conditions (11.76), then $\mathbf{x}^0$ gives the global minimum of the function $f(\mathbf{x})$, with the constraint given in (11.74).

Let

$$L^*(\mathbf{x}) = \max_{\boldsymbol{\lambda} \in \Re^m} L(\mathbf{x}, \boldsymbol{\lambda}), \text{ and } L_*(\boldsymbol{\lambda}) = \min_{\mathbf{x} \in \Re^n} L(\mathbf{x}, \boldsymbol{\lambda})$$

It is clear from these equations that for any $\mathbf{x} \in \Re^n$ and $\boldsymbol{\lambda} \in \Re^m$,

$$L_*(\boldsymbol{\lambda}) \leq L(\mathbf{x}, \boldsymbol{\lambda}) \leq L^*(\mathbf{x})$$

and thus, in particular

$$L_*(\boldsymbol{\lambda}) \leq L^*(\mathbf{x})$$

This holds for any $\mathbf{x} \in \Re^n$and $\boldsymbol{\lambda} \in \Re^m$; so it holds for the $\boldsymbol{\lambda}$ that maximizes the left-hand side, and the $\mathbf{x}$ that minimizes the right-hand side. Thus

$$\max_{\boldsymbol{\lambda} \in \Re^m} \min_{\mathbf{x} \in \Re^n} L(\mathbf{x}, \boldsymbol{\lambda}) \leq \min_{\mathbf{x} \in \Re^n} \max_{\boldsymbol{\lambda} \in \Re^m} L(\mathbf{x}, \boldsymbol{\lambda})$$

The two problems, min-max and max-min, are said to be *dual* to each other. We refer to the min-max problem as the *primal problem*. The objective to be minimized, $L^*(\mathbf{x})$, is referred to as the *primal function*. The max-min problem is referred to as the *dual problem*, and $L_*(\boldsymbol{\lambda})$ as the *dual function*. The optimal primal and dual function values are equal when f is a convex function and g_i are linear functions. The concept of duality is widely used in the optimization literature. The aim is to provide an alternative formulation of the problem which is more convenient to solve computationally and/or has some theoretical significance. In the context of SVM, the dual problem is not only easy to solve computationally, but also crucial for using *kernel functions* to deal with nonlinear decision boundaries. This will be clear later in this section.

The nonlinear optimization problem defined in (11.74) can be represented as min-max problem, as is seen below.

For the Lagrangian (11.75), we have

$$L^*(\mathbf{x}) = \max_{\boldsymbol{\lambda} \in \Re^m} \left[f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \right]$$

Since $g_i(\mathbf{x}) \geq 0$ for all i, $\lambda_i = 0$ ($i = 1, \ldots, m$) would maximize the Lagrangian. Thus

$$L^*(\mathbf{x}) = f(\mathbf{x})$$

Therefore, our original constrained problem (11.74) becomes the min-max primal problem:

$$\begin{aligned} &\underset{\mathbf{x} \in \Re^n}{\text{minimize}} \quad L^*(\mathbf{x}) \\ &\text{subject to} \quad g_i(\mathbf{x}) \geq 0; \; i = 1, \ldots, m \end{aligned}$$

The concept of duality gives the following formulation for max-min dual problem:

$$\underset{\boldsymbol{\lambda} \in \Re^m, \boldsymbol{\lambda} \geq \mathbf{0}}{\text{maximize}} \; L_*(\boldsymbol{\lambda})$$

More explicitly, this nonlinear optimization problem with *dual variables* $\boldsymbol{\lambda}$, can be written in the form:

$$\underset{\boldsymbol{\lambda} \geq \mathbf{0}}{\text{maximize}} \quad \min_{\mathbf{x} \in \Re^n} \left[f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \right] \tag{11.77}$$

Let us now state the learning problem in SVM.

Given a set of linearly separable training examples,

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_P, y_P)\},$$

learning is to solve the following constrained minimization problem:

$$\begin{aligned} &\text{minimize} \quad && f(\mathbf{w}) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{subject to} \quad && y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1;\ i = 1, \ldots, P \end{aligned} \tag{11.78}$$

This formulation is called the *hard-margin* SVM. Solving this problem will produce the solutions for $\mathbf{w}$ and b, which in turn, give us the maximal margin hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ with the margin $2/\|\mathbf{w}\|$.

The objective function is quadratic and convex in parameters $\mathbf{w}$, and the constraints are linear in parameters $\mathbf{w}$ and b. The dual formulation of this constrained optimization problem is obtained as follows.

First we construct the Lagrangian:

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{P} \lambda_i [y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] \tag{11.79}$$

The KKT conditions are as follows:

$$\begin{aligned} &\text{(i)} \quad \frac{\partial L}{\partial \mathbf{w}} = \mathbf{0}; \text{ which gives } \mathbf{w} = \sum_{i=1}^{P} \lambda_i y_i \mathbf{x}_i \\ &\qquad \frac{\partial L}{\partial b} = 0; \text{ which gives } \sum_{i=1}^{P} \lambda_i y_i = 0 \\ &\text{(ii)} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0;\ i = 1, \ldots, P \\ &\text{(iii)} \quad \lambda_i \geq 0;\ i = 1, \ldots, P \\ &\text{(iv)} \quad \lambda_i [y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] = 0;\ i = 1, \ldots, P \end{aligned} \tag{11.80}$$

From condition (i) of KKT conditions (11.80), we observe that the solution vector has an expansion in terms of training examples. Note that although the solution $\mathbf{w}$ is unique (due to the strict convexity of the function $f(\mathbf{w})$), the dual variables λ_i need not be. There is a dual variable λ_i for each training data point. Condition (iv) of KKT conditions (11.80) shows that for data points not on the margin hyperplanes (i.e., $\mathcal{H}^{(1)}$ and $\mathcal{H}^{(2)}$), $\lambda_i = 0$:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 > 0 \quad \Rightarrow \quad \lambda_i = 0$$

For data points on the margin hyperplanes, $\lambda_i \geq 0$:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 = 0 \quad \Rightarrow \quad \lambda_i \geq 0$$

However, the data points on the margin hyperplanes with $\lambda_i = 0$ do not contribute to the solution $\mathbf{w}$, as is seen from condition (i) of KKT conditions (11.80). The data points on the margin hyperplanes with associated dual variables $\lambda_i > 0$ are called *support vectors*, which give the name to the algorithm, *support vector machines*.

To postulate the dual problem, we first expand Eqn. (11.79), term by term, as follows:

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{P} \lambda_i y_i\ \mathbf{w}^T\mathbf{x}_i - b\sum_{i=1}^{P} \lambda_i y_i + \sum_{i=1}^{P} \lambda_i \tag{11.81}$$

Transforming from the primal to its corresponding dual can be done by setting to zero the partial derivatives of the Lagrangian (11.81) with respect to the *primal variables* (i.e., $\mathbf{w}$ and b), and substituting

the resulting relations back into the Lagrangian. This is simply to substitute condition (i) of KKT conditions (11.80) into the Lagrangian (11.81) to eliminate the primal variables; which gives us the dual objective function.

The third term on the right-hand side of Eqn. (11.81) is zero by virtue of condition (i) of KKT conditions (11.80). Furthermore, from this condition we have

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{P} \lambda_i y_i \; \mathbf{w}^T\mathbf{x}_i = \sum_{i=1}^{P} \sum_{j=1}^{P} \lambda_i \lambda_j \, y_i y_j \, \mathbf{w}_i^T \mathbf{x}_j$$

Accordingly, minimization of function L in Eqn. (11.81) with respect to primal variables $\mathbf{w}$ and b, gives us the following dual objective function:

$$L_*(\boldsymbol{\lambda}) = \sum_{i=1}^{P} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} \lambda_i \lambda_j \, y_i y_j \, \mathbf{w}_i^T \mathbf{x}_j \tag{11.82}$$

We may now state the dual optimization problem.

Given a set of linearly separable training examples$\{(\mathbf{x}_i, y_i)\}_{i=1}^{P}$, find the dual variables $\{\lambda_i\}_{i=1}^{P}$, that maximize the objective function (11.82) subject to the constraints

- $\sum_{i=1}^{P} \lambda_i y_i = 0$ (11.83)
- $\lambda_i \geq 0$; $i = 1,..., P$

This formulation is dual formulation of the *hard-margin SVM*.

Having solved the dual problem numerically (using MATLAB's **quadprog** function, for example), the resulting optimum λ_i values are then used to compute $\mathbf{w}$ and b. $\mathbf{w}$ is computed using condition (i) of KKT conditions (11.80):

$$\mathbf{w} = \sum_{i=1}^{P} \lambda_i y_i \mathbf{x}_i \tag{11.84a}$$

and b is computed using condition (iv) of KKT conditions (11.80). For support vectors $\{\mathbf{x}_s, y_s\}$, this condition becomes $\lambda_i > 0$, and

$$y_s(\mathbf{w}^T\mathbf{x}_s + b) = 1$$

Instead of depending on one support vector to compute b, in practice all support vectors are used to compute b, and then their average is taken on the final value for b. This is because the values of λ_i are computed numerically and can have numerical errors.

$$b = \frac{1}{N_{SV}} \sum_{s=1}^{N_{SV}} \left[\frac{1}{y_s} - \mathbf{w}^T\mathbf{x}_s \right]; \; N_{SV} = \text{total number of support vectors} \tag{11.84b}$$

11.13.2 Soft-Margin Linear SVM

The linear separable case is the ideal situation. In practice, however, the training data is almost always noisy, i.e., containing errors due to various reasons. For example, some examples may be labeled

incorrectly. Furthermore, practical problems may have some degree of randomness. Even for two identical input vectors, their labels may be different.

For SVM to be useful, it must allow noise in the training data. However, with noisy data, the linear SVM algorithm presented earlier, will not find a solution because the constraints cannot be satisfied. For example, in Fig. 11.29, there is a Class 2 point (circle) in the Class 1 region, and a Class 1 point (square) in the Class 2 region. However, in spite of the couple of mistakes, the decision boundary seems to be good. But the hard margin classifier presented previously cannot be used, because all the constraints.

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1; \;\; i = 1, \ldots, P$$

cannot be satisfied.

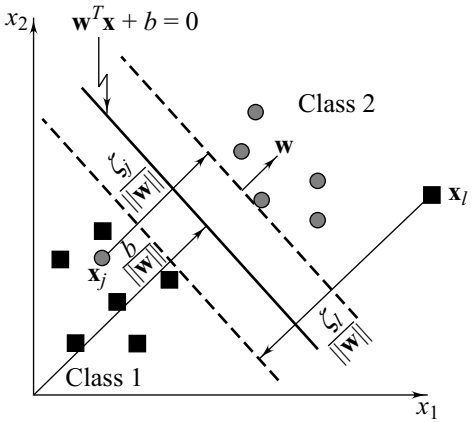


Fig. 11.29 Soft decision boundary: $\mathbf{x}_j$ and $\mathbf{x}_l$ are error data points

So the constraints have to be modified to permit mistakes. To allow errors in data, we can relax the margin constraints by introducing *slack* variables, $\zeta_i (\geq 0)$, as follows:

$$\mathbf{w}^T\mathbf{x}_i + b \geq 1 - \zeta_i \quad \text{for} \quad y_i = +1$$
$$\mathbf{w}^T\mathbf{x}_i + b \leq -1 + \zeta_i \quad \text{for} \quad y_i = -1$$

Thus, we have the new constraints

$$\begin{gathered} y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \zeta_i; \;\; i = 1, \ldots, P \\ \zeta_i \geq 0 \end{gathered} \tag{11.85}$$

The geometric interpretation is shown in Fig. 11.29.

We also need to penalize the errors in the objective function. A natural way is to assign an extra cost for errors, to change the objective function to

$$\tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\sum_{i=1}^{P}\zeta_i\right); C \geq 0$$

where C is a user specified penalty parameter. This parameter is a trade-off parameter between margin and mistakes.

The new optimization problem becomes

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{P}\zeta_i \\ &\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \zeta_i;\ i = 1,\ldots,P \\ &\qquad\qquad \zeta_i \geq 0;\ i = 1,\ldots,P \end{aligned} \tag{11.86}$$

This formulation is called the *soft-margin SVM*.

Proceeding in the manner similar to that described earlier for separable case, we may formulate the dual problem for nonseparable patterns as follows.

The Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\zeta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{P}\zeta_i - \sum_{i=1}^{P}\lambda_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \zeta_i] - \sum_{i=1}^{P}\mu_i\zeta_i \tag{11.87}$$

where $\lambda_i, \mu_i \geq 0$ are the dual variables.

The KKT conditions for optimality are as follows:

$$\begin{aligned} &\text{(i)} \quad \frac{\partial L}{\partial \mathbf{w}} = \mathrm{w} - \sum_{i=1}^{P}\lambda_i y_i \mathbf{x}_i = 0 \\ &\qquad \frac{\partial L}{\partial b} = -\sum_{i=1}^{P}\lambda_i y_i = 0 \\ &\qquad \frac{\partial L}{\partial \zeta_i} = C - \lambda_i - \mu_i = 0;\ i = 1\ ,\ldots,\ P \\ &\text{(ii)} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \zeta_i \geq 0;\ i = 1,\ldots,\ P \\ &\qquad \zeta_i \geq 0;\ i = 1,\ldots,\ P \\ &\text{(iii)} \quad \lambda_i \geq 0;\ i = 1,\ldots,\ P \\ &\qquad \mu_i \geq 0;\ i = 1,\ldots,\ P \\ &\text{(iv)} \quad \lambda_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \zeta_i) = 0;\ i = 1,\ldots,\ P \\ &\qquad \mu_i\zeta_i = 0;\ i = 1,\ldots,\ P \end{aligned} \tag{11.88}$$

We substitute the relations in condition (i) of KKT conditions (11.88) into the Lagrangian (11.87) to obtain dual objective function. From the relation $C - \lambda_i - \mu_i = 0$, we can deduce that $\lambda_i \leq C$ because $\mu_i \geq 0$. Thus the dual formulation of the *soft-margin SVM* is

$$\begin{aligned} &\text{maximize } L_*(\boldsymbol{\lambda}) = \sum_{i=1}^{P}\lambda_i - \tfrac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}\lambda_i\lambda_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ &\text{subject to} \quad \sum_{i=1}^{P}\lambda_i y_i = 0 \\ &\qquad\qquad 0 \leq \lambda_i \leq C;\ i = 1,\ldots,P \end{aligned} \tag{11.89}$$

Interestingly, ζ_i and μ_i are not in the dual objective function; the objective function is identical to that for the separable case. The only difference is the constraint $\lambda_i \le C$ (inferred from $C - \lambda_i - \mu_i = 0$ and $\mu_i \ge 0$).

The dual problem (11.89) can also be solved numerically, and the resulting λ_i values are then used to compute **w** and b. The weight vector **w** is computed using Eqn. (11.84a).

The bias parameter b is computed using condition (iv) of KKT conditions (11.88):

$$\lambda_i (y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \zeta_i) = 0 \tag{11.90a}$$

$$\mu_i \zeta_i = 0 \tag{11.90b}$$

Since we do not have values for ζ_i, we have to get around it. λ_i can have values in the interval $0 \le \lambda_i \le C$. We will separate it into the following three cases:

Case 1: $\lambda_i = 0$

We know that $C - \lambda_i - \mu_i = 0$. With $\lambda_i = 0$, we get $\mu_i = C$. Since $\mu_i\zeta_i = 0$ (Eqn. (11.90b)), this implies that $\zeta_i = 0$; which means that the corresponding ith pattern is correctly classified without any error (as it would have been with hard-margin SVM). Such patterns may lie on margin hyperplanes or outside the margin. However, they don't contribute to the optimum value of **w**, as is seen from Eqn. (11.84a).

Case 2: $0 < \lambda_i < C$

We know that $C - \lambda_i - \mu_i = 0$. Therefore, $\mu_i = C - \lambda_i$, which means $\mu_i > 0$. Since $\mu_i\zeta_i = 0$ (Eqn. (11.90b)), this implies that $\zeta_i = 0$. Again the corresponding ith pattern is correctly classified. Also from Eqn. (11.90a), we see that for $\zeta_i = 0$ and $0 < \lambda_i < C$, $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$; so the corresponding patterns are support vectors.

Case 3: $\lambda_i = C$

It can easily be seen that $\zeta_i \ne 0$ in this case. But $\zeta_i \ge 0$ is a constraint of the problem. So $\zeta_i > 0$; which means that the corresponding pattern is mis-classified or lies inside the margin.

We can use support vectors, as in Eqn. (11.84b), to compute the value of b.

The following points need attention of the reader:

- One of the most important properties of SVM is that the solution is sparse in λ_i. Most training data points are outside the margin area and their λ_i's in the solution are 0. The data points on the margin having $\lambda_i = 0$, also do not contribute to solution. Only those data points that are on the margin hyperplanes with $0 < \lambda_i < C$ (support vectors) and inside the margin (errors; $\lambda_i = C$) contribute to solution. Without this sparsity property, SVM would not be practical for large data sets.
- The final decision boundary is

  $$\mathbf{w}^T\mathbf{x} + b = 0$$

  Substituting for **w** and b from Eqns (11.84), we obtain

  $$\left(\sum_{i=1}^{P} \lambda_i y_i \mathbf{x}_i\right)^T \mathbf{x} + b = \sum_{i=1}^{P}\sum_{j=1}^{P} \lambda_i y_i \mathbf{x}_i^T \mathbf{x}_j + \frac{1}{N_{SV}} \sum_{s=1}^{N_{SV}} \left[\frac{1}{y_s} - \sum_{s=1}^{P} \lambda_s y_s \mathbf{x}_s^T \mathbf{x}_s\right] = 0 \tag{11.91}$$

  We notice that **w** and b do not need to be explicitly computed. As we will shortly see, this is crucial for using kernel functions to handle nonlinear decision boundaries.

- Finally, we still have the problem of determining the parameter C. The value of C is usually chosen by trying a range of values on the training set to build multiple classifiers and then testing them on validation set before selecting the one that gives the best classification result on the validation set.

11.13.3 Nonlinear SVM

The SVM formulations discussed so far, require that Class 1 and Class 2 examples can be linearly represented, i.e., the decision boundary must be a hyperplane. However, for many real-life data sets, the decision boundaries are nonlinear. To deal with nonlinearly separable data, the same formulation and solution techniques as far the linear case are still used. We only transform the input data from its original space into another space (usually, a much higher dimensional space) so that a linear decision boundary can separate Class 1 and Class 2 examples in the transformed space, which is called the *feature space*. The original data space is called the *input space*.

Through a nonlinear mapping $\boldsymbol{\phi}$, the original data set $\{(\mathbf{x}_1, y_1),...,(\mathbf{x}_P, y_P)\}$becomes

$$\{\boldsymbol{\phi}(\mathbf{x}_1, y_1),...,\boldsymbol{\phi}(\mathbf{x}_P, y_P)\} \tag{11.92}$$

Figure 11.30 illustrates the process. In the input space, the training examples cannot be linearly separated; in the feature space, they can be separated linearly.

With the transformation, the optimization problem in (11.86) becomes

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{P}\zeta_i \\ &\text{subject to} \quad y_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \zeta_i; \quad i = 1,\ldots,P \\ &\qquad\qquad \zeta_i \geq 0; \quad i = 1,\ldots,P \end{aligned} \tag{11.93}$$

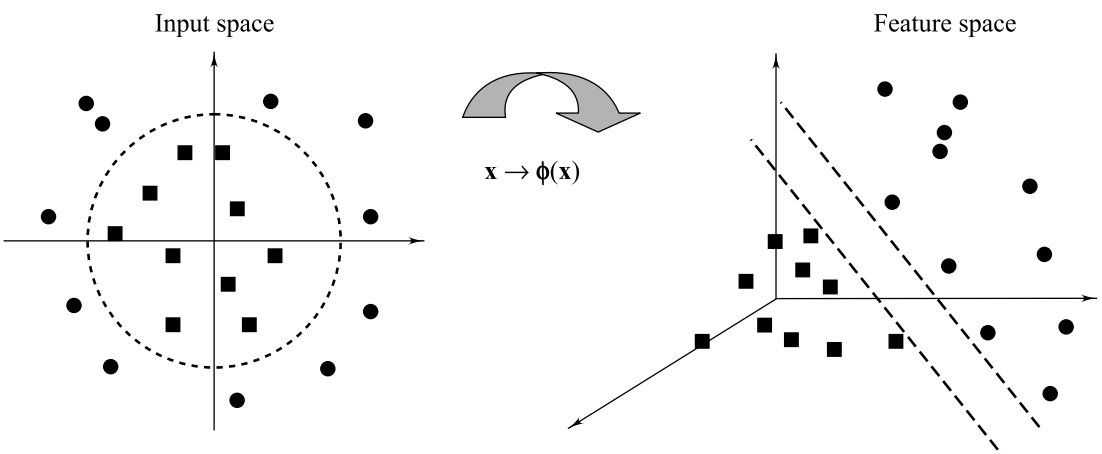


Fig. 11.30 Transformation from the input space to feature space

The corresponding dual is

$$\begin{aligned} &\text{minimize } L_*(\boldsymbol{\lambda}) = \sum_{i=1}^{P} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} \lambda_i \lambda_j y_i y_j (\boldsymbol{\phi}^T(\mathbf{x}_i)\, \boldsymbol{\phi}(\mathbf{x}_j)) \\ &\text{subject to } \sum_{i=1}^{P} \lambda_i y_i = 0 \\ &\qquad 0 \le \lambda_i \le C;\ i = 1, \ldots, P \end{aligned} \tag{11.94}$$

The potential problem with this approach is that it may suffer from the curse of dimensionality. The number of dimensions in the feature space can be huge with some useful transformations, even with reasonable number of attributes in the input space. Fortunately, explicit transformations can be avoided if we notice that for the dual problem (11.94), the construction of the decision boundary only requires the evaluation of $\boldsymbol{\phi}^T(\mathbf{x}_i)\, \boldsymbol{\phi}(\mathbf{x}_j)$ in the feature space. With reference to (11.91), we have the following decision boundary in feature space:

$$\sum_{i=1}^{P} \sum_{j=1}^{P} \lambda_i y_j \boldsymbol{\phi}^T(\mathbf{x}_i)\, \boldsymbol{\phi}(\mathbf{x}_j) + b = 0 \tag{11.95}$$

Thus, if we have to compute $\boldsymbol{\phi}^T(\mathbf{x}_i)\, \boldsymbol{\phi}(\mathbf{x}_j)$ in the feature space using the input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ directly, then we would not need to know the feature vector $\boldsymbol{\phi}(\mathbf{x})$ or even the mapping $\boldsymbol{\phi}$ itself. In SVM, this is done through the use of *kernel functions*, denoted by K:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j) \tag{11.96}$$

Commonly used kernels include the following:

$$\begin{aligned} &\text{Polynomial of degree } d:\ K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^d \\ &\text{Gaussian RBF}:\ K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \| \mathbf{x}_i - \mathbf{x}_j \|^2 \right) \end{aligned} \tag{11.97}$$

We replace $\boldsymbol{\phi}^T(\mathbf{x}_i)\, \boldsymbol{\phi}(\mathbf{x}_j)$ in (11.94) and (11.95) with kernel. We would never need to explicitly know what $\boldsymbol{\phi}$ is.

However, how do we know that a kernel function is indeed a dot product in some feature space? This question is answered by a theorem, called the *Mercer's theorem*, which we will not discuss here. The kernels in (11.97) satisfy this theorem. Refer to [138,141] for details.

11.13.4 Function Approximation using SVM

Suppose we are given training data

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_P, y_P)\};\ \ \mathbf{x} \in \Re^n,\ y \in \Re$$

where $\mathbf{x}_i \in \Re^n$ are the input patterns, as in classification problems; and $y_i \in \Re$ now has continuous values. Our goal is to find a function $f(\mathbf{x})$ that has at most ε deviation (where ε is a prescribed parameter) from the actually obtained targets y_i for all the training data, and at the same time, is as flat as possible. In other words, we do not care about errors as long as they are less than ε, but will not accept any deviation larger than this.

For pedagogical reasons, we begin by describing the case of linear functions f, taking the form

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b;\ \mathbf{w} \in \Re^n, b \in \Re \tag{11.98}$$

Flatness in the case of (11.98) means that one seeks small $\mathbf{w}$. One way to ensure this is to minimize the Euclidean norm, i.e., $\|\mathbf{w}\|^2$, This additional requirement on performance (in addition to the constraint on maximum allowable error in estimate of y_i) improves generalization.

Formally we can write this problem as a constrained optimization problem:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{subject to} && y_i - \mathbf{w}^T\mathbf{x}_i - b \le \varepsilon;\ \ i = 1,\ldots,P \\ & && \mathbf{w}^T\mathbf{x}_i + b - y_i \le \varepsilon;\ i = 1,\ldots,P \end{aligned} \tag{11.99}$$

The tacit assumption in (11.99) is that a function f given by (11.98) actually exists that approximates all pairs $(\mathbf{x}_i, y_i)$ with ε precision, or in other words, that the constrained optimization problem is *feasible*. It should be noted that the optimization problem cannot accommodate data points with errors larger than ε; constraints cannot be satisfied for such data points. For SVM to be useful, it must allow noise in the training data. Analogously to the 'soft margin' classifier described earlier, one can introduce slack variables ζ_i, ζ_i^* to cope with otherwise infeasible constraints of the optimization problem (11.99). Hence we arrive at the following formulation:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{P}\left(\zeta_i + \zeta_i^*\right) \\ &\text{subject to} && y_i - \mathbf{w}^T\mathbf{x}_i - b \le \boldsymbol{\varepsilon} + \zeta_i;\ \ i = 1,\ldots,P \\ & && \mathbf{w}^T\mathbf{x}_i + b - y_i \le \boldsymbol{\varepsilon} + \zeta_i^*;\ i = 1,\ldots,\boldsymbol{P} \\ & && \zeta_i, \zeta_i^* \ge 0;\ i = 1,...,P \end{aligned} \tag{11.100}$$

The constant $C > 0$ determines the trade-off between the flatness of f given by (11.98), and the amount by which deviations larger than ε are tolerated.

The formulation (11.100) corresponds to dealing with a so-called ε-*insensitive loss* (*error*) *function*, described below as

$$\left|y_i - \hat{y}_i\right|_\varepsilon \triangleq \begin{cases} 0 & \text{if } |\, y_i - \hat{y}_i\,| \le \varepsilon; \hat{y}_i \triangleq f(\mathbf{x}_i) \\ |\, y_i - \hat{y}_i\,| - \varepsilon & \text{otherwise} \end{cases} \tag{11.101}$$

This loss function defines an ε-insensitive tube (Fig. 11.31); the loss (error) is equal to zero for training data points inside the tube $(|y_i - \hat{y}_i| \le \varepsilon)$, the loss is ζ_i for data 'above' the tube $\left(y_i - \hat{y}_i - \varepsilon = \zeta_i\right)$ and ζ_i^* for data 'below' the tube $(\hat{y}_i - y_i - \varepsilon = \zeta_i^*)$. Only the data points outside the tube contribute to the loss (error), with deviations penalized in a linear fashion.

As with procedures applied to SVM classifiers, the constrained optimization problem (11.100) is solved by forming the Lagrangian:

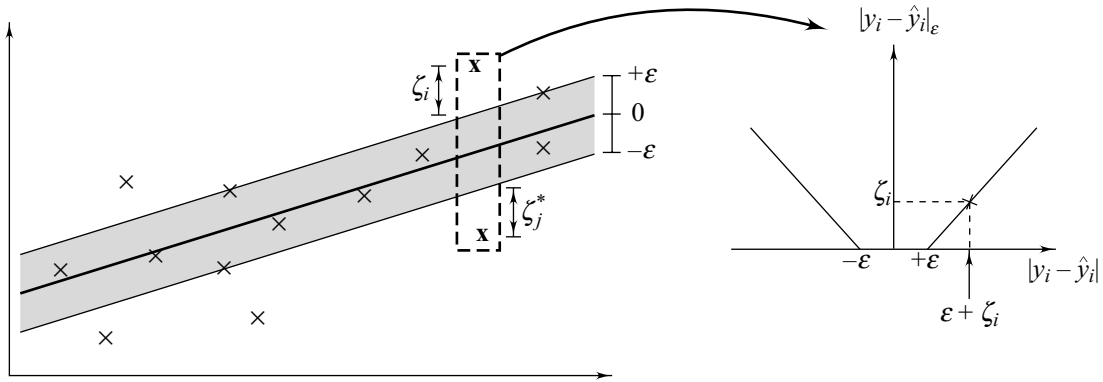


Fig. 11.31 The soft-margin loss setting

$$
\begin{aligned}
&L(\mathbf{w}, b, \boldsymbol{\zeta}, \boldsymbol{\zeta}^*, \boldsymbol{\lambda}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*) \\
&\quad = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{P}\left(\zeta_i + \zeta_i^*\right) - \sum_{i=1}^{P}\lambda_i(\varepsilon + \zeta_i - y_i + \mathbf{w}^T\mathbf{x}_i + b) \\
&\qquad - \sum_{i=1}^{P}\lambda_i^*\left(\varepsilon + \zeta_i^* + y_i - \mathbf{w}^T\mathbf{x}_i - b\right) - \sum_{i=1}^{P}\left(\mu_i\zeta_i + \mu_i^*\zeta_i^*\right)
\end{aligned} \tag{11.102}
$$

where $\mathbf{w}$, b, ζ_i and ζ_i^* are the primal variables, and λ_i, λ_i^*, μ_i, $\mu_i^* \geq 0$ are the dual variables.

The KKT conditions are as follows:

$$
\begin{aligned}
&\text{(i)} \quad \frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{P}\left(\lambda_i - \lambda_i^*\right)\mathbf{x}_i = \mathbf{0} \\
&\qquad \frac{\partial L}{\partial b} = \sum_{i=1}^{P}\left(\lambda_i^* - \lambda_i\right) = 0 \\
&\qquad \frac{\partial L}{\partial \zeta_i} = C - \lambda_i - \mu_i = \mathbf{0}; \quad i = 1, \ldots, P \\
&\qquad \frac{\partial L}{\partial \zeta_{\mathrm{i}}^*} = C - \lambda_i^* - \mu_i^* = 0; \quad i = 1, \ldots, P \\
&\text{(ii)} \quad \varepsilon + \zeta_i - y_i + \mathbf{w}^T\mathbf{x}_i + b \geq 0; \quad i = 1, \ldots, P \\
&\qquad \varepsilon + \zeta_i^* + y_i - \mathbf{w}^T\mathbf{x}_i - b \geq 0; \quad i = 1, \ldots, P \\
&\qquad \zeta_i, \zeta_i^* \geq 0; \quad i = 1, ..., P \\
&\text{(iii)} \quad \lambda_i, \lambda_i^*, \mu_i, \mu_i^* \geq 0; \quad i = 1, \ldots, P
\end{aligned} \tag{11.103}
$$

(iv) $\lambda_i \left(\varepsilon + \zeta_i - y_i + \mathbf{w}^T \mathbf{x}_i + b\right) = 0; \; i = 1, \ldots, P$

$\lambda_i^* \left(\varepsilon + \zeta_i^* + y_i - \mathbf{w}^T \mathbf{x}_i - b\right) = 0; \; i = 1, ..., P$

$\mu_i \zeta_i = 0; \; i = 1, \ldots, P$

$\mu_i^* \zeta_i^* = 0; \; i = 1, \ldots, P$

Substituting the relations in condition (i) of KKT conditions (11.103) yields the dual objective function. The procedure is parallel to what has been followed earlier. The resulting dual optimization problem is

$$\text{maximize } L_*(\boldsymbol{\lambda}, \boldsymbol{\lambda}^*) = -\varepsilon \sum_{i=1}^{P} (\lambda_i + \lambda_i^*) + \sum_{i=1}^{P} (\lambda_i - \lambda_i^*) y_i - \tfrac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} (\lambda_i - \lambda_i^*)(\lambda_j - \lambda_j^*) \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \sum_{i=1}^{P} (\lambda_i - \lambda_i^*) = 0 \tag{11.104}$$

$$\lambda_i, \lambda_i^* \in [0, C]$$

From condition (i) of KKT conditions (11.103), we have

$$\mathbf{w} = \sum_{i=1}^{P} \left(\lambda_i - \lambda_i^*\right) \mathbf{x}_i \tag{11.105}$$

Thus the weight vector $\mathbf{w}$ is completely described as a linear combination of the training patterns $\mathbf{x}_i$. One of the most important properties of SVM is that the solution is sparse in λ_i, λ_i^*. For $|\hat{y}_i - y_i| < \varepsilon$, the second factor in the following KKT conditions:

$$\begin{aligned} \lambda_i (\varepsilon + \zeta_i - y_i + \mathbf{w}^T \mathbf{x}_i + b) &= \lambda_i (\varepsilon + \zeta_i - y_i + \hat{y}_i) = 0 \\ \lambda_i^* (\varepsilon + \zeta_i^* + y_i - \mathbf{w}^T \mathbf{x}_i - b) &= \lambda_i^* (\varepsilon + \zeta_i^* + y_i - \hat{y}_i) = 0 \end{aligned} \tag{11.106}$$

are nonzero; hence λ_i, λ_i^* have to be zero. This equivalently means that all the data points inside the ε-insensitive tube (a large number of training examples belong to this category) have corresponding λ_i, λ_i^* equal to zero. Further, from (11.106) it follows that only for $|\hat{y}_i - y_i| \geq \varepsilon$, the dual variables λ_i, λ_i^* may be nonzero. Since there can never be a set of dual variables λ_i, λ_i^* which are both simultaneously nonzero, as this would require slacks in both directions ('above' the tube and 'below' the tube), we have $\lambda_i \times \lambda_i^* = 0$.

From KKT conditions (11.103), it follows that

$$\begin{aligned} \left(C - \lambda_i\right) \zeta_i &= 0 \\ (C - \lambda_i^*) \zeta_i^* &= 0 \end{aligned} \tag{11.107}$$

Thus the only samples ($\mathbf{x}_i$, y_i) with corresponding $\lambda_i, \lambda_i^* = C$ lie outside the ε-insensitive tube around f. For $\lambda_i, \lambda_i^* \in (0, C)$, we have ζ_i, $\zeta_i^* = 0$ and moreover the second factor in (11.106) has to vanish. Hence b can be computed as follows:

$$\begin{aligned} b &= y_i - \mathbf{w}^T \mathbf{x}_i - \varepsilon \quad \text{for} \quad \lambda_i \in (0, C) \\ b &= y_i - \mathbf{w}^T \mathbf{x}_i + \varepsilon \quad \text{for} \quad \lambda_i^* \in (0, C) \end{aligned} \tag{11.108}$$

All data points with $\lambda_i, \lambda_i^* \in (0, C)$ are used to compute b, and then their average is taken as the final value for b.

The examples that come with nonvanishing dual variables λ_i, λ_i^* are called *support vectors*.

The next step is to make SVM algorithm nonlinear. This would be achieved by simply preprocessing the training patterns $\mathbf{x}_i$ by a map $\boldsymbol{\phi}$ into some feature space, and then applying the standard SVM algorithm.

All pattern-recognition/function approximation (classification/regression) problems when solved using SVM algorithms presented in this section, are basically *quadratic optimization problems*. Attempting MATLAB functions for SVM algorithms discussed in this section, will be a rich learning experience for the reader.

REVIEW EXAMPLES

Review Example 11.1

A high performance drive system consists of a motor and a controller integrated to perform a precise mechanical maneuver. This requires the shaft speed, and/or position of the motor to clearly follow a specified trajectory, regardless of unknown load variations and other parameter uncertainties.

A backpropagation neural network can be trained to emulate the unknown nonlinear plant dynamics by presenting a suitable set of input/output patterns generated by the plant. Once system dynamics have been identified using a neural network, many conventional control techniques can be applied to achieve the desired objective of trajectory tracking.

In this example, we study a neural-network-based identification and control strategy for trajectory control of a dc motor.

DC Motor Model

Although it is not mandatory to obtain a motor model if a neural network (NN) is used in the motor-control system, it may be worth doing so, from the analytical perspective, in order to establish the foundation of the NN structure. We will use input/output patterns generated by simulation of this model for training of NN (In a real life situation, experimentally generated input/output patterns will be used for training).

The dc motor dynamics are given by the following equations (refer to Fig. 11.32):

$$v_a(t) = R_a i_a(t) + L_a \frac{di_a}{dt} + e_b(t) \tag{11.109}$$

$$e_b(t) = K_b \omega(t) \tag{11.110}$$

$$T_M(t) = K_T i_a(t) \tag{11.111}$$

$$= J \frac{d\omega(t)}{dt} + B\omega(t) + T_L(t) + T_F \tag{11.112}$$

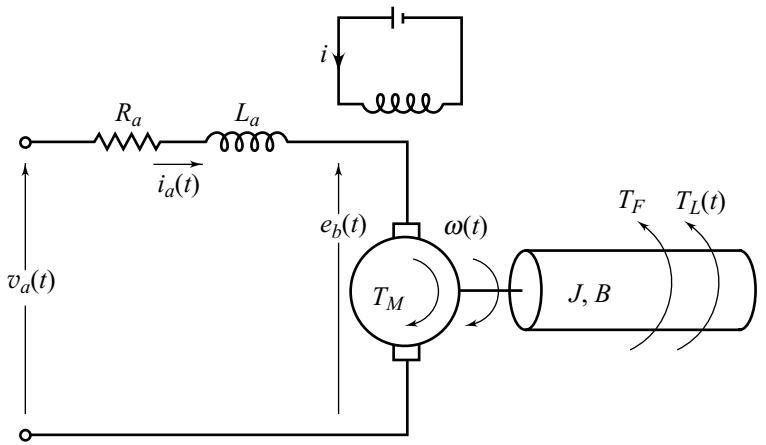


Fig. 11.32 Armature-controlled dc motor with load

where

$v_a(t)$ = applied armature voltage (volts);
$e_b(t)$ = back emf (volts);
$i_a(t)$ = armature current (amps);
R_a = armature winding resistance (ohms);
L_a = armature winding inductance (henrys);
$\omega(t)$ = angular velocity of the motor rotor (rad/sec);
$T_M(t)$ = torque developed by the motor (newton-m);
K_T = torque constant (newton-m/amp);
K_b = back emf constant (volts/(rad/sec));
J = moment of inertia of the motor rotor with attached mechanical load (kg-m^2);
B = viscous-friction coefficient of the motor rotor with attached mechanical load ((newton-m)/(rad/sec));
$T_L(t)$ = disturbance load torque (newton-m); and
T_F = frictional torque (newton-m).

The load torque $T_L(t)$ can be expressed as

$$T_L(t) = \psi(\omega) \tag{11.113}$$

where the function $\psi(\cdot)$ depends on the nature of the load.

For most propeller driven or fan type loads, the function $\psi(\cdot)$ takes the following form:

$$T_L(t) = \mu\omega^2(t)[\text{sgn}\omega(t)] \tag{11.114}$$

where μ is a constant.

DC motor drive system can be expressed as single-input, single-output system by combining Eqns (11.109)–(11.110):

$$L_aJ\frac{d^2\omega(t)}{dt^2} + (R_aJ + L_aB)\frac{d\omega(t)}{dt} + (R_aB + K_bK_T)\omega(t) + L_a\frac{dT_L(t)}{dt} + R_a[T_L(t) + T_F] + K_T v_a(t) = 0 \tag{11.115}$$

The discrete-time model is derived by replacing all continuous differentials with finite differences.

$$L_aJ\left[\frac{\omega(k+1) - 2\omega(k) + \omega(k-1)}{T^2}\right] + (R_aJ + L_aB)\left[\frac{\omega(k+1) - \omega(k)}{T}\right] + (R_aB + K_bK_T)\omega(k) + L_a\left[\frac{T_L(k) - T_L(k-1)}{T}\right] + R_aT_L(k) + R_aT_F + K_Tv_a(k) = 0 \tag{11.116}$$

$$T_L(k) = \mu\omega^2(k)[\text{sgn}\omega(k)] \tag{11.117}$$

$$T_L(k-1) = \mu\omega^2(k-1)[\text{sgn}\omega(k)] \tag{11.118}$$

T = sampling period

$\omega(k) \triangleq \omega(t = kT)$; $k = 0, 1, 2, \ldots$

Manipulation of Eqns (11.116)–(11.118) yields

$$\omega(k+1) = K_1\omega(k) + K_2\omega(k-1) + K_3[\text{sgn}\omega(k)]\omega^2(k) + K_4[\text{sgn}\omega(k)]\omega^2(k-1) + K_5v_a(k) + K_6 \tag{11.119}$$

where

$$K_1 = \frac{2L_aJ + T(R_aJ + L_aB) - T^2(R_aB + K_bK_T)}{L_aJ + T(R_aJ + L_aB)}$$

$$K_2 = -\frac{L_aJ}{L_aJ + T(R_aJ + L_aB)}$$

$$K_3 = -\frac{T(\mu L_a + \mu R_aT)}{L_aJ + T(R_aJ + L_aB)} \tag{11.120}$$

$$K_4 = \frac{T\mu L_a}{L_aJ + T(R_aJ + L_aB)}$$

$$K_5 = \frac{K_TT^2}{L_aJ + T(R_aJ + L_aB)}$$

$$K_6 = -\frac{T_FR_aT^2}{L_aJ + T(R_aJ + L_aB)}$$

The following parameter values are associated with the dc motor

$$\begin{aligned} J &= 0.068 \text{ kg-m}^2 \\ B &= 0.03475 \text{ newton-m/(rad/sec)} \\ R_a &= 7.56\ \Omega \\ L_a &= 0.055 \text{ H} \\ K_T &= 3.475 \text{ newton-m/amp} \\ K_b &= 3.475 \text{ volts/(rad/sec)} \end{aligned} \tag{11.121}$$

$\mu = 0.0039$ newton-m/(rad/sec)^2
$T_F = 0.212$ newton-m
$T = 40$ msec $= 0.04$ sec

With these motor parameters, the constants K_1, K_2, K_3, K_4, K_5 and K_6 become

$$\begin{aligned} K_1 &= 0.34366 \\ K_2 &= -0.1534069 \\ K_3 &= -2.286928 \times 10^{-3} \\ K_4 &= 3.5193358 \times 10^{-4} \\ K_5 &= 0.2280595 \\ K_6 &= -0.105184 \end{aligned} \tag{11.122}$$

Identification of Inverse Dynamics

Equation (11.119) can be manipulated to obtain the inverse dynamic model of the drive system as

$$v_a(k) = f[\omega(k+1), \omega(k), \omega(k-1)] \tag{11.123}$$

The right-hand side of Eqn. (11.123) is a nonlinear function of the speed ω and is given by

$$f(\omega(k+1), \omega(k), \omega(k-1)) = \frac{1}{K_5}[\omega(k+1) - K_1\omega(k) - K_2\omega(k-1) - K_3\{\text{sgn}\omega(k)\}\omega^2(k) - K_4\{\text{sgn}\omega(k)\}\omega^2(k-1) - K_6] \tag{11.124}$$

which is assumed to be unknown (It is assumed that the only available qualitative *a priori* knowledge about the plant is a rough estimate of the order of the plant). A neural network is trained to emulate the unknown function $f(\cdot)$. The values $\omega(k+1)$, $\omega(k)$ and $\omega(k-1)$, which are the independent variables of $f(\cdot)$, are selected as the inputs to the NN. The corresponding target $f(\omega(k+1), \omega(k), \omega(k-1))$ is given by Eqn. (11.124). This quantity is also equal to the armature voltage $v_a(k)$, as seen from Eqn. (11.123). Randomly generated input patterns of $[\omega(k+1), \omega(k), \omega(k-1)]$ and the corresponding target $v_a(k)$, are used for off-line training. The training data is generated within the constrained operating space. In conforming with the mechanical and electrical hardware limitations of the motor, and with a hypothetical operating scenario in mind, the following constrained operating space is defined:

$$\begin{aligned} &-30 < \omega(k) < 30 \text{ rad/sec} \\ &|\omega(k-1) - \omega(k)| < 1.0 \text{ rad/sec} \\ &|v_a(k)| < 100 \text{ volts} \end{aligned} \tag{11.125}$$

The estimated motor armature voltage given by the NN identifier is

$$\hat{v}_a(k-1) = N(\omega(k), \omega(k-1), \omega(k-2)) \tag{11.126}$$

Trajectory Control of DC Motor using Trained NN

The objective of the control system is to drive the motor so that its speed $\omega(k)$ follows a reference (prespecified) trajectory $\omega_r(k)$. A controller topology is presented in Fig. 11.33. The NN trained to emulate inverse dynamics of the dc motor, is used to estimate the motor armature voltage $\hat{v}_a(k)$, which

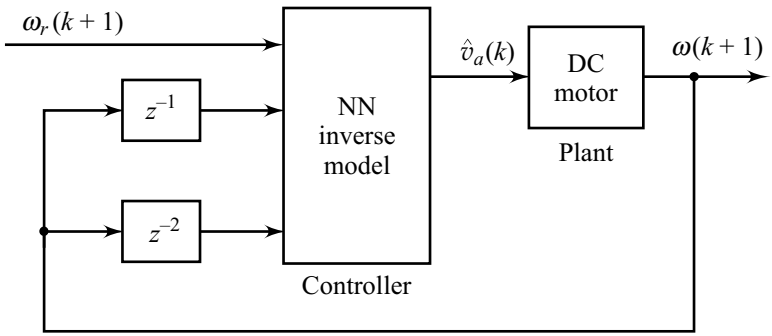


Fig. 11.33 A structure for NN-based speed control

enables accurate trajectory control of the shaft speed $\omega(k)$. Refer to Appendix B for realization of the controller.

Review Example 11.2

In this example, we study a neural-network–based identification and control strategy for temperature control of a water bath.

The plant consists of a laboratory 7-liter water bath as depicted in Fig. 11.34. A personal computer reads the temperature of the water bath through a link consisting of a diode-based temperature sensor module (SM) and an 8-bit A/D converter. The plant input, produced by the computer, is limited between 0 and 5 volts, and controls the duty cycle for a 1.3 kW heater via a pulse-width-modulation (PWM) scheme.

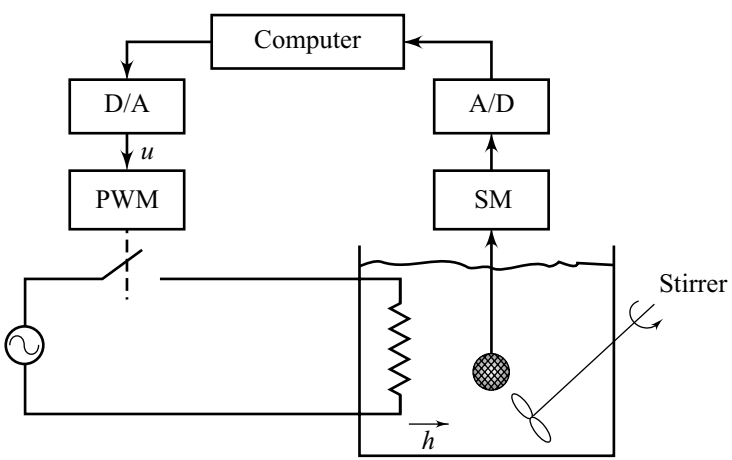


Fig. 11.34 Water bath control system

The temperature of water in a stirred tank is described by the equation

$$C\frac{dy(t)}{dt} = \frac{Y_0 - y(t)}{R} + h(t) \tag{11.127}$$

where $y(t)$ is the temperature of water in the tank (ºC), $h(t)$ is the heat flowing into the tank through the base heater (watts), Y_0 is the temperature of the surroundings (assumed constant, for simplicity), C denotes the tank thermal capacity (Joules/ºC), and R is the thermal resistance between tank borders and surroundings. Assuming R and C as essentially constant, we can obtain discrete-time description of the thermal system as follows:

$$\begin{aligned} x(t) &\triangleq y(t) - Y_0 \\ \dot{x}(t) &= -\frac{1}{RC}x(t) + \frac{1}{C}h(t) = -\alpha\, x(t) + \beta\, h(t) \end{aligned} \tag{11.128}$$

The discrete-time state equation (sampling period = T):

$$x(k+1) = F\,x(k) + g\,h(k) \tag{11.129}$$

where

$$F = e^{-\alpha T};\ g = \beta \int_0^T e^{-\alpha\tau}\, d\tau = \frac{\beta}{\alpha}\left[1 - e^{-\alpha T}\right]$$

We modify this model to include a saturating nonlinearity, so that the water temperature cannot exceed some limitation. The nonlinear plant model then becomes (obtained from real plant by experimentation)

$$y(k+1) = F\,y(k) + \frac{g}{1+\exp\left[0.5y(k)-40\right]}u(k) + (1-F)Y_0 \tag{11.130}$$

$$\begin{aligned} \alpha &= 1.00151 \times 10^{-4} \\ \beta &= 8.67973 \times 10^{-3} \\ Y_0 &= 25^{\circ}\text{C} \\ T &= 30 \text{ sec} \\ u &= \text{input to the PWM, limited between 0 and 5 volts.} \end{aligned} \tag{11.131}$$

With these parameters, the simulated system is equivalent to a SISO temperature control system of a water bath, that exhibits linear behavior up to about 70ºC and then becomes nonlinear and saturates at about 80ºC.

The task is to learn how to control the plant described in Eqn. (11.130), in order to follow a specified reference $y_r(k)$, minimizing some norm of error $e(k) = y_r(k) - y(k)$ through time. It is assumed that the model in Eqn. (11.130) is unknown; the only available qualitative *a priori* knowledge about the plant is a rough estimate of the order of the plant.

A neural network is trained to emulate the inverse dynamics of the plant. Assume that at instant $k + 1$, the current output $y(k + 1)$, the $P - 1$ previous values of y, and P previous values of u are all stored in memory. Then the P pairs $(\mathbf{x}^T(k - i), u(k - i))$; $i = 0, 1, ..., P - 1$, $\mathbf{x}^T(k) = [y(k + 1), y(k)]$, can be used as patterns for training the NN at time $k + 1$. A train of pulses is applied to the plant and the corresponding input/output pairs are recorded. The NN is then trained with reasonably large sets of data, chosen from the experimentally obtained data bank, in order to span a considerable region of the control space (We will use input/output patterns generated by simulation of the plant model for training the NN).

A controller topology is presented in Fig. 11.35. It is assumed that the complete reference trajectory $y_r(k)$ is known in advance. The feedforward component of the control input is then composed by substituting all system outputs by corresponding reference values. Refer to Appendix B for realization of the controller.

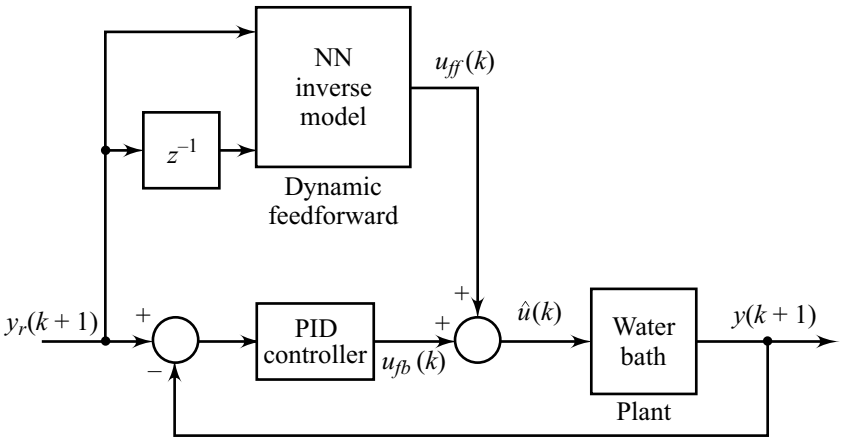


Fig. 11.35 A structure for NN-based temperature control

PROBLEMS

11.1 It is believed that the output y of a plant is linearly related to the input u; that is,

$$\hat{y} = w_1 u + w_2$$

(a) What are the values of w_1 and w_2 if the following measurements are obtained:
$u = 2, y = 5, u = -2, y = 1$.

(b) One more measurement is taken: $u = 5$, $y = 7$. Find a least-squares estimate of w_1 and w_2 using all the three measurements.

(c) Find the unique minimal sum of error squares in this linear fit to the three points.

11.2 Consider the network in Fig. P11.2. An input signal $\mathbf{x}$ comprising features and augmented by a constant input component (bias) is applied to the network with linear activation function. The network gives the output $\hat{\mathbf{y}}$.

(a) Organize the weights as row vectors:

$$\mathbf{w}_j^T = [w_{j1}\ w_{j2} \cdots w_{jn}\ w_{j0}];$$
$$j = 1, 2, \ldots, q$$

and write the equations (model) that this network represents.

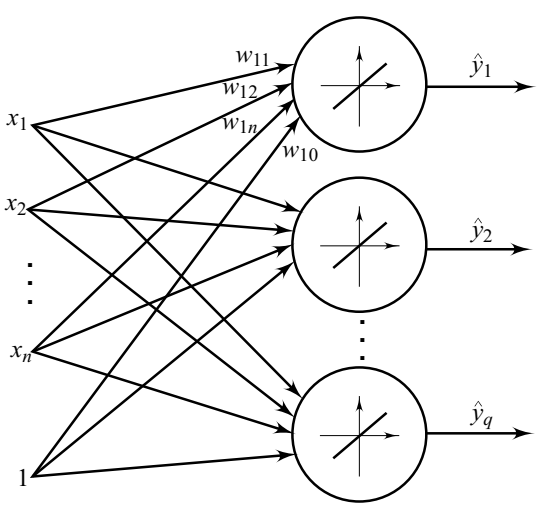


Fig. P11.2

(b) The learning environment comprises a training set of P data pairs $\{\mathbf{x}^{(p)}, \mathbf{y}^{(p)}; p = 1, 2, ..., P\}$ consisting of the input vector $\mathbf{x}$ and output vector $\mathbf{y}$.

Prove that the gradient descent learning rule for the network is

$$w_j(k+1) = w_j(k) + \eta e_j(k)\,\mathbf{x}$$

where k is the iteration index, η is the learning rate parameter, and $e_j = y_j - \hat{y}_j$

11.3 Consider the RBF network shown in Fig. 11.20. There are two sets of parameters governing the mapping properties of this network: the weights w_{ji}; $i = 1,2,...m$; $j = 1,2,...,q$, in the output layer and the center $\mathbf{c}_i$ of the radial basis functions. The simplest form of RBF network training is with fixed centers. In particular, they are commonly chosen, in a random manner, as a subset of the input data set. A sufficient number of centres randomly selected from the input data set, would distribute according to the probability density function of the training data, thus providing an adequate sampling of the input space. Because the centers are fixed, the mapping performed by the hidden layer is fixed as well. Derive gradient descent training algorithm to determine the appropriate settings of the weights in the network output layer so that the performance of the network mapping is optimized.

11.4 It is desired to design a one-layer NN with one input x and one output $\hat{y}$ that associates input $x^{(1)} = -3$ with the target output $y^{(1)} = 0.4$, and input $x^{(2)} = 2$ with the target output $y^{(2)} = 0.8$. Determine the parameters w and w_0 of the network

$$\hat{y} = \sigma(wx + w_0)$$

with unipolar sigmoidal (log-sigmoid) activation function, that minimize the error

$$E = \left[\left(y^{(1)} - \hat{y}^{(1)}\right)^2 + \left(y^{(2)} - \hat{y}^{(2)}\right)^2\right]$$

11.5 Streamline the notation in Chapter 11 for a three-layer NN. For instance, define $\mathbf{W}^{h1}$ as weights of Hidden layer 1 with m nodes; $\mathbf{W}^{h2}$ as weights of Hidden layer 2 with p nodes; and $\mathbf{V}$ as weights of output layer with q nodes.

Input variables	: x_i; $i = 1, ..., n$
Outputs of hidden layer 1	: z_ℓ; $\ell = 1, ..., m$
Outputs of hidden layer 2	: t_r; $r = 1, ..., p$
Outputs of output layer	: $\hat{y}_j$; $j = 1, ..., q$
Desired outputs	: y_j
Learning constant	: η

Derive the backpropagation algorithm for the three-layer network, assuming the output layer has linear activation and the two hidden layers have unipolar sigmoidal activations.

11.6 Consider a four-input single-node perceptron with a bipolar sigmoidal function (tan-sigmoid)

$$\sigma(a) = \frac{2}{1+e^{-a}} - 1$$

where 'a' is the activation value for the node.

(a) Derive the weight update rule for $\{w_i\}$ for all i. The learning rate $\eta = 0.1$. Input variables: x_i; $i = 1, 2, 3, 4$. Desired output is y.

(b) Use the rule in part (a) to update the perceptron weights incrementally for one epoch. The set of input and desired output patterns is as follows:

$$\mathbf{x}^{(1)} = [1 \quad -2 \quad 0 \quad -1]^T, \qquad y^{(1)} = -1$$
$$\mathbf{x}^{(2)} = [0 \quad 1.5 \quad -0.5 \quad -1]^T, \quad y^{(2)} = -1$$
$$\mathbf{x}^{(3)} = [-1 \quad 1 \quad 0.5 \quad -1]^T, \qquad y^{(3)} = 1$$

The initial weight vector is chosen as

$$\mathbf{w}_0^T = [1 \quad -1 \quad 0 \quad 0.5]$$

The perceptron does not possess bias term.

(c) Use the training data and initial weights given in part (b) and update the perceptron weights for one epoch in batch mode.

11.7 We are given the two-layer backpropagation network shown in Fig. P11.7.

(a) Derive the weight update rules for $\{v_\ell\}$ and $\{w_{\ell i}\}$ for all i and ℓ. Assume that activation function for all the nodes is a unipolar sigmoid function

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

where 'a' represents the activation value for the node. The learning constant $\eta = 0.1$. The desired output is y.

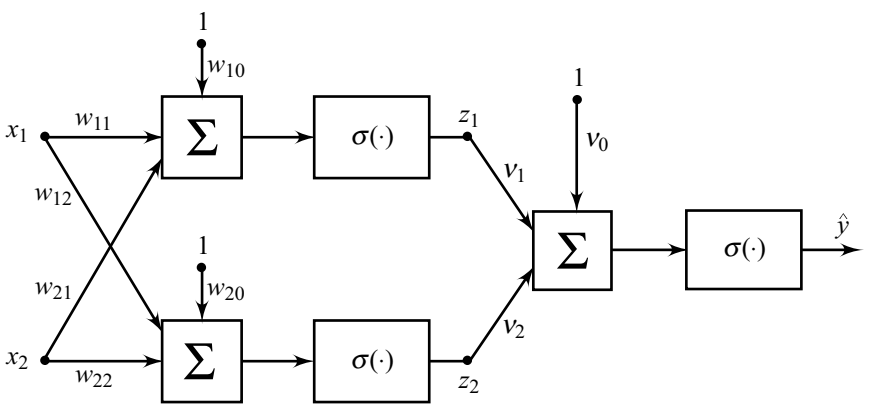


Fig. P11.7

(b) Use the equations derived in part (a) to update the weights in the network for one step with input vector $\mathbf{x} = [1\ 0]^T$, desired output $y = 1$, and the initial weights:

$$w_{10} = 1,\ w_{11} = 3,\ w_{12} = 4,\ w_{20} = -6,\ w_{21} = 6,\ w_{22} = 5$$
$$v_0 = -3.92,\ v_1 = 2, \text{ and } v_2 = 4$$

(c) As a check, compute the error with the same input for initial weights and updated weights and verify that the error has decreased.

11.8 We are given the two-layer backpropagation network in Fig. P11.8.

Derive the weight update rules for $\{v_\ell\}$ and $\{w_\ell\}$ for all ℓ. Assume that activation function for all the nodes is a bipolar sigmoid function

$$\sigma(a) = \frac{2}{1+e^{-a}} - 1$$

where 'a' is the activation value for the node. The learning constant is $\eta = 0.4$. The desired output is y.

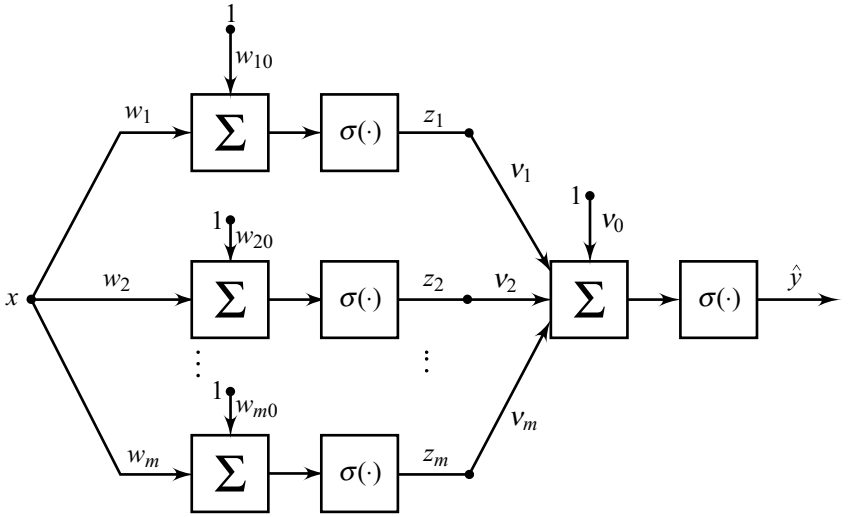


Fig. P11.8

11.9 We are given the two-layer back propagation network shown in Fig.P11.9.

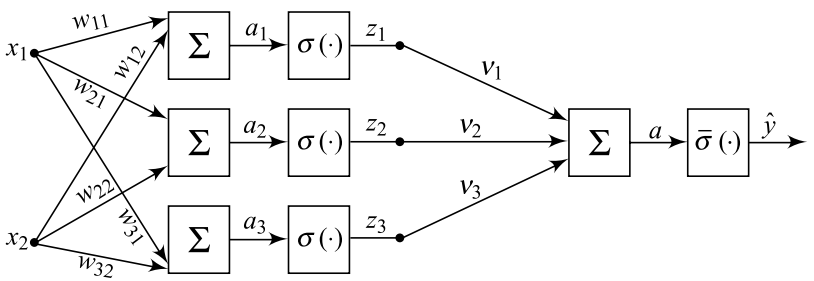


Fig. P11.9

(a) Derive the weight update rules in incremental mode for $\{v_\ell\}$ and $\{w_{\ell i}\}$ for all i and ℓ; the iteration index is k. Assume that the activation function for all nodes in the hidden layer is

$$\sigma(a) = \frac{1}{1+e^{-a_\ell}}$$

and the activation function for the node in the output layer is

$$\bar{\sigma}(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

The learning constant $\eta = 0.2$. The desired output is y.

(b) Use the equations derived in part (a) to update the weights in the network for one step with input vector $\mathbf{x} = [0.5 \ \ -0.4]^T$, desired output $y = 0.15$, and the initial weights: $w_{11} = 0.2$, $w_{12} = 0.1$, $w_{21} = 0.4$, $w_{22} = 0.6$, $w_{31} = 0.3$, $w_{32} = 0.5$; $v_1 = 0.1$, $v_2 = 0.2$ and $v_3 = 0.1$.

Chapter 12

Fuzzy Logic and Neuro-Fuzzy Systems

12.1 INTRODUCTION

In the previous chapter, we were mostly concerned with learning from experimental data (examples, samples, measurements, patterns, or observations). Our emphasis was on the following machine learning problem setting:

There is some unknown dependency (mapping, function) $y = f(\mathbf{x})$ between some high-dimensional input vector $\mathbf{x}$ and a scalar output y (or vector output $\mathbf{y}$). The only information available about the underlying dependency is a training data set $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1,2,\ldots,P\}$. We employed neural networks to learn this dependency. The number of neurons, their link structure, and the corresponding weights were the subjects of learning procedure.

It may be noted that depending upon the problem, the neural-network weights have different physical meanings, and sometimes it is hard to find any physical meaning at all. Neural network learning is, thus, a 'block box' design situation (Fig. 12.1a) in which the process is entirely unknown but there are known examples $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1,2,\ldots,P\}$. The *knowledge* (information) is available only in the form of data pairs; the neural network is required to be trained using this knowledge before the machine could be used for prediction.

A large amount of data can constitute a proportionally large amount of information. But this comes with a level of uncertainty. As we come to know more, we also know how much we do not know, and our awareness of the concept of complexity seems to increase. We tend to forego some precise data and allow uncertainty to creep into our perception. This is when we start describing things in a slightly *vague* and *fuzzy* manner.

Consider, for example, a real-life situation in process industry. Control of large and complex processes is facilitated by using distributed computer control systems (DCCS). Acquisition of process data, i.e., collection of instantaneous values of process variables, and status messages of plant control facilities (valves, pumps, motors, etc.) needed for efficient direct digital control; processing of collected data; plant hardware monitoring, system check and diagnosis; closed-loop control and logic control functions; etc., are the routine tasks of DCCS. Enormous amount of data (constituting a proportionally large amount of information) is, thus, generated. Closed-loop control design using machine learning paradigm

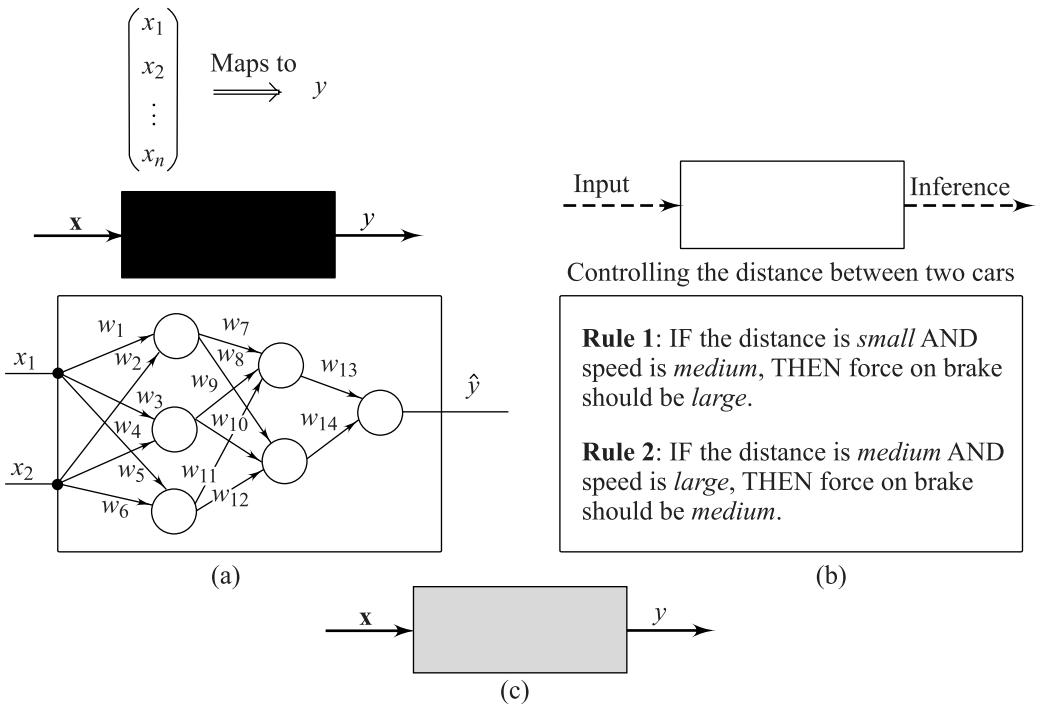


Fig. 12.1 Neural networks and fuzzy logic models as examples of 'black box' 'white box', and 'grey box' modeling approaches [138]

based on the knowledge (information) embedded in the data, is feasible. This approach is, however, seldom used in process industry.

In a man–machine control system, an experienced process operator employs, consciously or subconsciously, a set of IF-THEN rules to control a process. The operator estimates the *important* process variables (such as *error*, *rate of change of error*) at discrete time instants, and based on this information, s/he manipulates the control signal. The estimation of the process variables is *not* done in numerical form; it is rather done in linguistic form. For example, s/he may categorize the variable 'error' into the following labels:

'error is negative'

'error is near zero'

'error is positive'

Analogously, s/he defines the categories of 'rate of change of error' and 'change of control'.

The categories (linguistic labels) of the process variables are, in general, vague and qualitative. Their purpose is to describe in a qualitative way control strategies based on human experience and understanding. A commonly used way of expressing the knowledge (information) based on human experience and understanding is through IF-THEN rules. A typical rule in this kind of knowledge base will be of the form:

IF *error* is *near zero* AND *rate of change of error* is *near zero* THEN change of control is *zero*

Process operators have no difficulties with understanding and interpreting this kind of rules because they have the background to hearing problems and solutions described like this. However, providing a computer with the same level of understanding is a difficult task. How can we represent 'expert knowledge' that uses vague and ambiguous terms, in a computer? Can it be done at all?

In the label 'near zero', the word *near* seems to be comprehended effortlessly by the human brain, but what of computing systems? What does *near* mean in the context of process control? The range –0.1 to +0.1 or the range –1 to +1, or …? Is there a way we can make number crunching systems understand this? If it is ascertained in a machine that any *error* less than or equal to | 1 | means *near zero*, and anything outside this range is *negative*/*positive*, then does it mean that 1.001 is *not* '*near zero*' while 1 is '*near zero*'? This is an exaggeration in the real world.

The rule-base representing the expert knowledge can be significantly improved if we consider more categories for process variables. For example, linguistic label 'positive' may be subdivided into *positive small*, *positive medium*, and *positive large*. The increased granularity of the categories results in finer formulated rules. There is, however, a trade-off between accuracy and complexity.

Fuzzy logic deals with how we can capture the essence of human comprehension and embed it on the system, allowing for a gradual transition from one category to another. This comprehension as per Lofti Zadeh, the founder of the fuzzy logic concept, confers a higher machine intelligence level to computer systems.

In the previous chapter, our focus was on machine learning problem setting based on the knowledge (information) available in the form of numerical data. Our focus in this chapter is on another machine learning problem setting where language serves as a way of expressing imprecise knowledge, and the tolerance for imprecision about the vague environment we live in. Most human knowledge is imprecise, uncertain and usually expressed in linguistic terms. In addition, human ways of reasoning are approximate, nonquantitative and linguistic in nature. Fuzzy logic is a tool for transforming such linguistically expressed knowledge into workable algorithm called a *fuzzy logic model*. In its newest incarnation, fuzzy logic is called 'computing with words'.

The point of departure in fuzzy logic is the existence of human solution. If there is no human solution, there will be no knowledge to model and, consequently, no sense in applying fuzzy logic. However, the existence of human solution is not sufficient. One must be able to articulate to structure the human solution in the language of fuzzy IF-THEN rules. Almost all structured human knowledge can be expressed in the form of IF-THEN rules. The fuzzy logic modeling is thus a 'white box' design situation in which the solution to the problem is known, that a, structured human knowledge (experience, expertise, heuristics) about the process exists (Fig. 12.1b). Interpretability of the fuzzy logic model for decision making is an important characteristic of this setting of machine learning problems.

Neural networks and fuzzy logic models are modeling tools. They perform in the same way after the learning stage of neural networks or the embedding of human knowledge about some specific task in fuzzy logic structure, is finished. Whether the more appropriate tool for solving a given problem is a neural network or a fuzzy logic model, depends upon the availability of previous expert knowledge (in linguistic form) about the system to be modeled and the amount of measured data. The less previous expert knowledge exists, the more likely it is that a neural network approach will be used to attempt a solution. The more knowledge available, the more suitable the problem will be for fuzzy logic modeling.

Through integration of the techniques of fuzzy logic models and neural networks, we can reap the benefits of both the fuzzy logic models and the neural networks. One such integrated system, a *neuro-fuzzy system*, transforms the burden of designing fuzzy logic systems to the training and learning of neural networks. That is, the neural networks provide learning abilities to the fuzzy logic systems. Neuro-fuzzy systems are functionally fuzzy logic models; they only utilize learning ability of neural networks to realize the key components of the fuzzy logic model. Integrated systems may also be formed by incorporating fuzzy logic into the neural network models. In such an integration, called a *fuzzy-neural network*, the numerical parameters (such as input-output data, weights, etc.) of a neural network are fuzzified. Fuzzy-neural networks are fuzzified neural networks, and thus are functionally neural networks.

Instances involving *some knowledge and some data* correspond to 'grey box' design situation (Fig. 12.1c) covered by the paradigm of neuro-fuzzy and fuzzy-neural models.

Embedding existing structured human knowledge into fuzzy logic models and neuro-fuzzy models, will be the subject of discussion in this chapter.

12.2 FUZZY RULES-BASED LEARNING

In a man–machine system, there arises the problem of processing information with the 'vagueness' that is characteristic of man. We consider here a real-life situation in process control.

The basic structure of a feedback control system is shown in Fig. 12.2a. G represents the system to be controlled (*plant* or *process*). The purpose of the *controller* D is to guarantee a desired response of the *output* y. The process of keeping the output y close to the *set-point* (*reference input*) y_r, despite the presence of disturbances, fluctuations of the system parameters, and noisy measurements, is called *regulation*. The law governing corrective action of the controller is called the *control algorithm*. The output of the controller is the *control action* u.

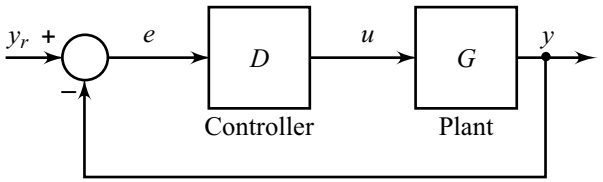
The general form of the control law (implemented using a digital computer) is

$$u(k) = f(e(k), e(k-1), ..., e(k-m), u(k-1), ..., u(k-m)) \tag{12.1}$$

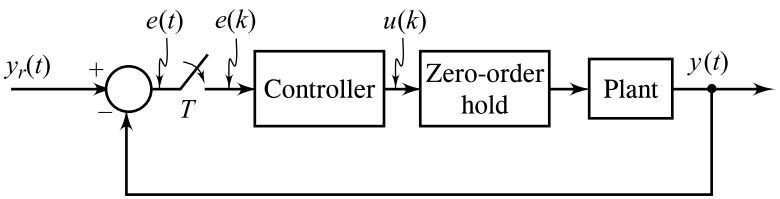
providing a control action that describes the relationship between the input and the output of the controller. In Eqn. (12.1), $e = y_r - y$ represents the error between the desired set-point y_r and the output of the controlled system; parameter m defines the order of the controller; and $f(\cdot)$ is, in general, a nonlinear function. k is an index representing sampling instant; T is the sampling interval used for digital implementation (Fig. 12.2b). To distinguish control law (12.1) from the control schemes based on fuzzy logic/neural networks, we shall call this, *conventional control law*.

A common feature of conventional control is that the control algorithm is analytically described by equations—algebraic, difference, differential, and so on. In general, the synthesis of such control algorithms requires a formalized analytical description of the controlled system by a mathematical model. The concept of analyticity is one of the main paradigms of conventional control theory. We will also refer to conventional control as *model-based* control.

When the underlying assumptions are satisfied, many of the model-based control techniques provide good stability, robustness to model uncertainties and disturbances, and speed of response. However, there are



(a) Basic structure of a feedback control system



(b) Basic structure of a digital control system

Fig. 12.2

many practical deficiencies of these control algorithms. It is, generally, difficult to accurately represent a complex process by a mathematical model. If the process model has parameters whose values are partially known, ambiguous or vague, the control algorithms that are based on such *incomplete information* will not, usually, give satisfactory results. The environment with which the process interacts may not be completely predictable, and it is normally not possible for a model-based algorithm to accurately respond to a condition that it did not anticipate. Skilled human operators are, however, controlling complex plants quite successfully on the basis of their experience, without having quantitative models.

Regulatory control objectives, typical of many industrial applications, are

(1) to remove any significant errors in process output $y(t)$ by appropriate adjustment of the controller output $u(k)$;
(2) to prevent process output from exceeding some user-specified constraint y_c, i.e., for all t, $y(t)$ should be less than or equal to y_c; and
(3) to produce smooth control action near the set-point, i.e., minor fluctuations in the process output are not passed further to the controller.

A conventional PI controller uses an analytical expression of the following form to compute the control action:

$$u(t) = K_c' \left[e(t) + \frac{1}{T_I} \int e(\tau)\, d\tau \right] \tag{12.2}$$

where K_c' is the controller gain, and T_I is integral or reset time.

When this expression is differentiated, we obtain

$$\dot{u}(t) = K_c' \dot{e}\,(t) + \frac{K_c'}{T_I} e(t)$$

The discrete-time version of this equation may be written as

$$\frac{u(k)-u(k-1)}{T} = K_c'\left[\frac{e(k)-e(k-1)}{T}\right]+\frac{K_c' e(k)}{T_I}$$

or

$$\Delta u(k) = K_c v(k) + K_I e(k) \tag{12.3}$$

where

$\Delta u(k)$ = incremental change in control variable = $u(k) - u(k-1)$;
$e(k)$ = error variable = $y_r - y(k)$; and
$v(k)$ = time rate of change of error[1] = $\dfrac{e(k)-e(k-1)}{T}$.

The control objectives, listed earlier, would require variable gains when the process output is in different regions around the set-point. Figure 12.3 illustrates the type of control action desired; Δu should be 'near zero' in the set-point region, very large in the constraint region, and normal in between.

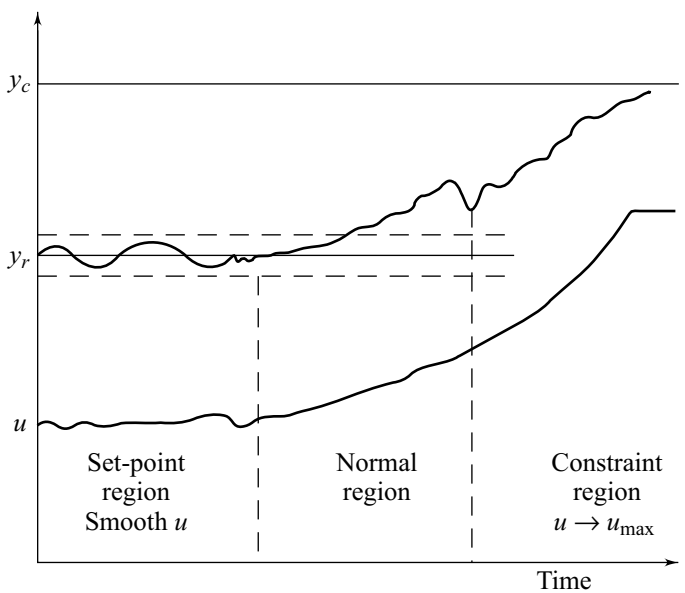


Fig. 12.3 Type of control action desired in different regions around the set-point

A simple PI controller is inherently incapable of achieving all of the above control objectives, and has to be implemented with additional (nonlinear) control laws for set-point and constraint regions, making the control scheme a complex adaptive control scheme which would allow the desired gain modification when required.

On the other hand, an experienced process operator can easily meet all the three control objectives. An expert operator employs, consciously or subconsciously, a set of IF-THEN rules to control a process

[1] A PD controller in position form is

$$u(k) = K_c e(k) + K_D v(k)$$

We see that PD controller in position form is structurally related to PI controller in incremental form.

(Fig. 12.4). He estimates the *error* $e(k)$ and *time rate of change of error* $v(k)$ at a specific time instant, and based on this information he changes the control by $\Delta u(k)$.

A typical *production rule* of the rule-base in Fig. 12.4 is of the form:

IF (process state) THEN (control action) (12.4)

instead of an analytical expression defining the control variable as a function of process state. The 'process state' part of the rule is called the rule *premise* (or *antecedent*), and contains a description of the process state at the kth sampling instant. This description is done in terms of particular values of error $e(k)$, velocity (time rate of change of error) $v(k)$, and the constraint. The 'control action', part of the rule is called the *conclusion* (or *consequent*), and contains a description of the control variable which should be produced given the particular process state in the rule antecedent. This description is in terms of the value of the change-in-control, $\Delta u(k)$.

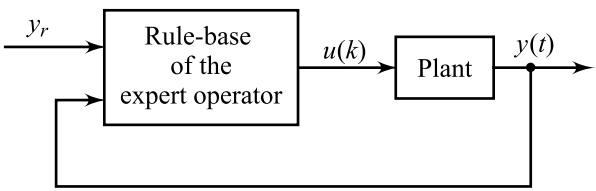


Fig. 12.4 A man-machine control system

Negative values of $e(k)$ mean that the current process output $y(k)$ has a value above the set-point y_r, since $e(k) = y_r - y(k) < 0$. The magnitude of a negative value describes the magnitude of the difference $y_r - y$. On the other hand, positive values of $e(k)$ express the knowledge that the current value of the process output $y(k)$ is below the set-point y_r. The magnitude of such a positive value is the magnitude of the difference $y_r - y$.

Negative values of $v(k)$ mean that the current process output $y(k)$ has increased compared with its previous value $y(k-1)$, since $v(k) = -(y(k) - y(k-1))/T < 0$. The magnitude of such a negative value describes the magnitude of this increase. Positive values of $v(k)$ express the knowledge that $y(k)$ has decreased its value when compared to $y(k-1)$. The magnitude of such a value is the magnitude of the decrease.

Positive values of $\Delta u(k)$ mean that the value of the control $u(k-1)$, has to be increased to obtain the value of the control for the current sampling time k. A value with a negative sign means a decrease in the value of $u(k-1)$. The magnitude of such a value is the magnitude of increase/decrease in the value $u(k-1)$.

The possible combinations of positive/negative values of $e(k)$ and $v(k)$ are as follows:

(1) positive e, negative v
(2) negative e, positive v
(3) negative e, negative v
(4) positive e, positive v

The combination (positive $e(k)$, negative $v(k)$) implies that $y < y_r$, since $e(k) = y_r - y(k) > 0$; and $\dot{y} > 0$, since $v(k) = -(y(k) - y(k-1))/T < 0$. This means that the current process output $y(k)$ is below the set-point and the controller is driving the system upward, as shown by point D in Fig. 12.5. Thus, the current process output is approaching the set-point from below. The combination (negative $e(k)$, positive $v(k)$) implies that $y > y_r$, and $\dot{y} < 0$. This means that the current process output is above the set-point and the controller is driving the system downward, as shown by point B in Fig. 12.5. Thus the current process output is approaching the set-point from above. The combination (negative $e(k)$, negative $v(k)$) implies that $y > y_r$ and $\dot{y} > 0$. This means that the current process output $y(k)$ is above the set-point and the controller is driving the system upward, as shown by point C in Fig. 12.5. Thus the process output

is moving further away from the set-point and approaching overshoot. The combination (positive $e(k)$, positive $v(k)$) implies that $y < y_r$ and $\dot{y} < 0$. This means that the current process output is below the set-point and the controller is driving the system downward, as shown by point A in Fig. 12.5. Thus the process output is moving further away from the set-point and approaching undershoot.

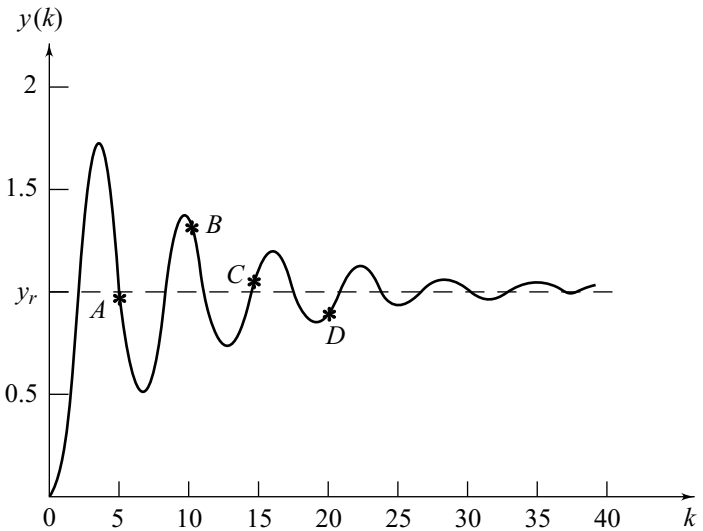


Fig. 12.5 Process output deviations from the set-point

In a man–machine control system of the type shown in Fig. 12.4, experience-based knowledge of the process operator and/or control engineer is instrumental in changing the control by $\Delta u(k)$, for a given estimate of error $e(k)$ and time rate of change of error $v(k)$.

(i) If both $e(k)$ and $v(k)$ (positive or negative) are small (or zero), it means that the current value of the process output variable $y(k)$ has deviated from the set-point but is still close to it. The amount of change $\Delta u(k)$ in the previous control $u(k-1)$ should also be small (or zero) in magnitude, which is intended to correct small deviations from the set-point.

(ii) Consider the situation when $e(k)$ has large negative value (which implies that $y(k)$ is significantly above the set-point). If $v(k)$ is positive at the same time, this means that y is moving towards the set-point. The amount of change Δu to be introduced is intended to either speed up or slow down the approach to the set-point. For example, if $y(k)$ is much above the set-point ($e(k)$ has a large negative value) and it is moving towards the set-point with a small step ($v(k)$ has small positive value), then the magnitude of this step has to be significantly increased ($\Delta u(k) \rightarrow$ large negative value).

(iii) $e(k)$ has either a small value (positive, negative, zero) or a large positive value, which implies that $y(k)$ is either close to the set-point or significantly below it. If $v(k)$ is positive at the same time, this means that y is moving away from the set-point. Then, a positive change $\Delta u(k)$ in the previous control $u(k-1)$ is required to reverse this trend and make y start moving towards it, instead of moving away from the set-point.

(iv) Consider a situation when $e(k)$ has large positive value (which implies that $y(k)$ is significantly below the set-point) and $v(k)$ is negative (which implies that y is moving towards the set-point).

The amount of change Δu to be introduced is intended to either speed up, or slow down, the approach to the set-point. For example, if $y(k)$ is much below the set-point ($e(k)$ has large positive value), and it is moving towards the set-point with somewhat large step ($v(k)$ has large negative value), then the magnitude of this step need not be changed ($\Delta u(k) \rightarrow 0$), or only slightly enlarged ($\Delta u(k) \rightarrow$ small positive value).

(v) $e(k)$ has either a small value (positive, negative, zero) or a large negative value, and this implies that $y(k)$ is either close to the set-point or significantly above it. If $v(k)$ is negative at the same time, y is moving away from the set-point. Thus, a negative change $\Delta u(k)$ in the previous control $u(k-1)$ is required to reverse this trend and make y start moving towards it instead of moving away from the set-point.

The variables e, v and Δu are described as consisting of a finite number of verbally expressed values which these variables can take. Values are expressed as tuples of the form {value sign, value magnitude}. The 'value sign' component of such a tuple takes on either one of the two values: positive or negative. The 'value magnitude' component can take on any number of magnitudes, e.g., {zero, small, medium, big}, or {zero, small, big}, or {zero, very small, small, medium, big, very big}, etc.

The tuples of values may, therefore, look like: Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (ZO), Positive Small (PS), Positive Medium (PM), Positive Big (PB) or an enhanced set/subset of these values.

We consider here a simple rule-based controller which employs only three values of the variables e, v, and Δu: Negative (N), Near Zero (NZ), Positive (P), for e and v; and Negative (N), Zero (Z), Positive (P) for Δu. A typical production rule of the rule-base in Fig. 12.4 is

IF $e(k)$ is Positive **and** $v(k)$ is Positive THEN $\Delta u(k)$ is Positive (12.5)

Let us see now what such a rule actually means. A positive $e(k)$ implies that $y(k)$ is below the set-point. If $v(k)$ is positive at the same time, it means that $y(k)$ is moving away from the set-point. Thus, a positive change $\Delta u(k)$ in the previous control $u(k-1)$ is required to reverse this trend.

Consider another rule:

IF $e(k)$ is Positive **and** $v(k)$ is Negative THEN $\Delta u(k)$ is Zero (12.6)

This rule says that if $y(k)$ is below the set-point, and is moving towards the set-point, then, no change in control is required.

We will present the rule-base in table format, shown in Fig. 12.6. The cell defined by the intersection of the third row and third column represents the rule given in (12.5), and the cell defined by the inter-section of the third row and first column represents the rule given in (12.6).

e \ v	N	NZ	P
N	N	N	Z
NZ	N	Z	P
P	Z	P	P

Fig. 12.6 The rule-base for system of Fig. 12.4

The rule-base shown in Fig. 12.6 is designed to remove any significant errors in process output by appropriate adjustment of the controller output. Note that the rule

IF $e(k)$ is Near Zero **and** $v(k)$ is Near Zero THEN $\Delta u(k)$ is Zero (12.7)

ensures smooth action near the set-point, i.e., minor fluctuations in the process output are not passed further to the controller.

The rule-base of Fig. 12.6 is thus effective for control action in the set-point region and the normal region in Fig. 12.3. However, we require additional rules for the constraint region. The following three rules prescribe a control action when the error is in the constraint region, approaching it or leaving it.

(i) IF $e(k)$ is in constraint region THEN value of $\Delta u(k)$ is drastic change.
This rule specifies the magnitude of additional $\Delta U(k)$ to be added to the one already determined by the rules of Fig. 12.6 when $e(k)$ is in the constraint region.

(ii) IF $e(k)$ enters constraint region, THEN start summing up the values of $\Delta u(k)$ determined by constraint Rule 1.

(iii) IF $e(k)$ leaves constraint region, THEN subtract the total value of $\Delta u(k)$ determined by constraint Rule 2.

The man–machine control system of Fig. 12.4 has the capability of representing and manipulating data that is not precise, but rather *fuzzy*. The error variable is 'near zero', change in control is 'drastic', etc.,— are the type of linguistic information which the expert controller is required to handle. But what is a 'drastic change' in control? The property 'drastic' is inherently *vague*, meaning that the set of signals it is applied to, has no sharp boundaries between 'drastic' and 'not drastic'. The fuzziness of a property lies in the lack of well-defined boundaries of the set of objects to which the property applies.

Problems featuring *uncertainty* and *ambiguity* have been successfully addressed subconsciously by humans. Humans can adapt to unfamiliar situations and they are able to gather information in an efficient manner, and discard irrelevant details. The information gathered need not be complete and precise and could be *general*, *qualitative* and *vague* because humans can *reason*, *infer* and *deduce* new information and knowledge. They can *learn*, *perceive* and improve their skills through experience.

How can humans reason about complex systems, when the complete description of such a system often requires more detailed data than a human could ever hope to recognize simultaneously, and assimilate with understanding? The answer is that humans have the capacity to reason approximately. In reasoning about a complex system, humans reason approximately about its behavior, thereby maintaining only a generic understanding about the problem.

The seminal work by Dr. Lotfi Zadeh (1965) on system analysis based on the theory of fuzzy sets, has provided a mathematical strength to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning. The conventional approaches to knowledge representation, lack the means for representing the meaning of fuzzy concepts. As a consequence, the approaches based on classical logic and probability theory, do not provide an appropriate conceptual framework for dealing with the representation of commonsense knowledge, since such knowledge is by its nature, both, *lexically imprecise* and *non-categorical*. The development of fuzzy logic was motivated, in large measure, by the need for a conceptual framework which can address the issue of uncertainty and lexical imprecision. Fuzzy logic provides an inference morphology, that enables approximate human reasoning capabilities to be applied to knowledge-based systems.

Since the publication of Zadeh's seminal work *Fuzzy Sets* in 1965, the subject has been the focus of many independent research investigations by mathematicians, scientists and engineers from around the world. Fuzzy logic has rapidly become one of the most successful of technologies today, for developing sophisticated control systems. With its aid, complex requirements may be implemented in amazingly simple, easily maintained and inexpensive controllers. Of course, fuzzy logic is not the best approach

for every control problem. As designers look at its power and expressiveness, they must decide where to apply it.

The criteria, in order of relevance, as to when and where to apply fuzzy logic are as follows:

- Human (structured) knowledge is available.
- A mathematical model is unknown or impossible.
- The process is substantially nonlinear.
- There is lack of precise sensor information.
- It is applied at the higher levels of hierarchical control systems.
- It is applied in generic decision-making problems

Possible difficulties in applying fuzzy logic, arise from the following:

- Knowledge is subjective.
- For high-dimensional inputs, the increase in the required number of rules is exponential.
- Knowledge must be structured, but experts bounce between a few extreme poles: they have trouble structuring the knowledge; they are too aware of their 'expertise'; they tend to hide knowledge; and there may be some other subjective factors working against the whole process of human knowledge transfer.

Note that the basic premise of fuzzy logic is that a human solution is good. When applied, for example, in control systems, this premise means that a human being is a good controller. Today, after several thousands successful applications, there is more or less convergence on trustworthiness of this premise.

The word 'fuzzy' may sound to mean intrinsically imprecise, but there is nothing 'fuzzy' about fuzzy logic. It is firmly based on multivalued logic theory and does not violate any well-proven laws of logic. Also fuzzy logic systems can produce answers to any required degree of accuracy. This means that these models can be very precise if needed (There is a trade-off between precision and cost). However, they are aimed at handling imprecise and approximate concepts that cannot be processed by any other known modeling tool. In this sense, fuzzy logic models are invaluable supplements to classical hard computing techniques. For example in a hierarchical control system, classical control at the lowest level, supplemented by fuzzy logic control at higher levels provides good hybrid solution in many situations.

Our focus in this chapter is on the essential ideas and tools necessary for the construction of the fuzzy knowledge-based models, that have been successful in the development of intelligent controllers. Fuzzy control and modeling use only a small portion of the fuzzy mathematics that is available; this portion is also mathematically quite simple and conceptually, easy to understand. This chapter begins with an introduction to some essential concepts, terminology, notations and arithmetic of fuzzy sets and fuzzy logic. We include only a minimum, though adequate, amount of fuzzy mathematics necessary for understanding fuzzy control and modeling. To facilitate easy reading, this background material is presented in a rather informal manner, with simple and clear notation as well as explanation. Whenever possible, excessively rigorous mathematics is avoided. This material is intended to serve as an introductory foundation for the reader to understand not only the fuzzy controllers presented later in this chapter but also others in the literature. We recommend references [137, 138, 142–145] for further reading on fuzzy set theory.

12.3 FUZZY QUANTIFICATION OF KNOWLEDGE

Up to this point we have only quantified, in an abstract way, the knowledge that the human expert has about how to control the plant. Next, we will show how to use fuzzy logic to fully quantify the meaning of linguistic descriptions so that we may automate in the fuzzy controller, the control rules specified by the expert.

12.3.1 What is Fuzzy Logic?

Knowledge is structured information and knowledge acquisition is done through learning and experience, which are forms of high-level processing of information. Knowledge representation and processing are the keys to any intelligent system. In *logic*, knowledge is represented by propositions and is processed through reasoning, by the application of various laws of logic, including an appropriate *rule of inference*.

Fuzzy logic focuses on linguistic variables in natural language, and aims to provide foundations for *approximate reasoning* with imprecise propositions.

In *classical logic*, a proposition is either TRUE, denoted by 1, or FALSE, denoted by 0. Consider the following proposition p:

'Team member is female'

Let X be a collection of 10 people: $x_1, x_2, ..., x_{10}$, who form a project team. The entire object of discussion is

$$X = \{x_1, x_2, ..., x_{10}\}$$

In general, the entire object of discussion is called a 'universe of discourse', and each constituent member x is called an 'element' of the universe (the fact that x is an element of X, is written as $x \in X$).

If x_1, x_2, x_3 and x_4 are female members in the project team, then the proposition p on the universe of discourse X is equally well represented by the *crisp* (nonfuzzy) set A defined below.

$$A = \{x_1, x_2, x_3, x_4\}$$

The fact that A is a subset of X is denoted as $A \subset X$.

The proposition can also be expressed by a mapping μ_A from X into the binary space $\{0, 1\}$.

$$\mu_A : X \rightarrow \{0, 1\}$$

such that

$$\mu_A = \begin{cases} 0\,;\, x = x_5, x_6, x_7, x_8, x_9, x_{10} \\ 1\,;\, x = x_1, x_2, x_3, x_4 \end{cases}$$

That is to say, the value $\mu_A(x) = 1$ when the element x satisfies the attributes of set A; 0 when it does not. μ_A is called the *characteristic function* of A.

Next, supposing that, within X, only x_1 and x_2 are below age 20; we may call them 'minors'. Then

$$B = \{x_1, x_2\}$$

consists of minor team members. In this case

$$\mu_B(x) = \begin{cases} 1\,;\, x = x_1, x_2 \\ 0\,;\, \text{otherwise} \end{cases}$$

B is obviously a subset of A; we write $B \subset A$.

We have considered the 'set of females A', and the 'set of minors B' in X. Is it also possible to consider a 'set of young females C'? If, for convenience, we consider the attribute 'young' to be same as 'minor', then $C = B$; but, in this case, we have created a sharp boundary, under which x_2 who is 19 is still young ($\mu_C(x_2) = 1$), but x_3 who just turned 20 today is no longer young ($\mu_C(x_3) = 0$). In just one day, the value changed from yes (1) to no (0), and x_3 is now an old maid.

However, is it not possible that a young woman becomes an old maid over a period of 10 to 15 years, so that we ought to be patient with her? Prof. Zadeh admitted values such as 0.8 and 0.9 that are intermediate between 0 and 1, thus creating the concept of a 'fuzzy set'. Whereas a crisp set is defined by the characteristic function that can assume only the two values {0, 1}, a *fuzzy set* is defined by a 'membership function' that can assume an infinite number of values; any real number in the closed interval [0, 1].

With this definition, the concept of 'young women' in X can be expressed flexibly in terms of membership function (Fuzzy sets are denoted in this book by a set symbol with a tilde understrike).

$$\mu_{\underset{\sim}{C}} : X \to [0, 1]$$

such that

$$\mu_{\underset{\sim}{C}} = \begin{cases} 1; & x = x_1, x_2 \\ 0.9; & x = x_3 \\ 0.2; & x = x_4 \\ 0; & \text{otherwise} \end{cases}$$

The significance of such terms as 'patient' and 'flexibly' in the above description may be explained as follows. For example, we have taken $\mu_{\underset{\sim}{C}}(x_3) = 0.9$, but suppose that x_3 objects that 'you are being unfair; I really ought to be a 1 but if you insist we can compromise on 0.95'. There is a good amount of subjectivity in the choice of membership values. A great deal of research is being done on the question of assignment of membership values. However, even with this restriction, it has become possible to deal with many problems that could not be handled with only crisp sets.

Since [0, 1] incorporates {0, 1}, the concept of fuzzy set can be considered as an extended concept, which incorporates the concept of crisp set. For example, the crisp set B of 'minors' can be regarded as a fuzzy set $\underset{\sim}{B}$ with the membership function:

$$\mu_{\underset{\sim}{B}}(x) = \begin{cases} 1; & x = x_1, x_2 \\ 0; & \text{otherwise} \end{cases}$$

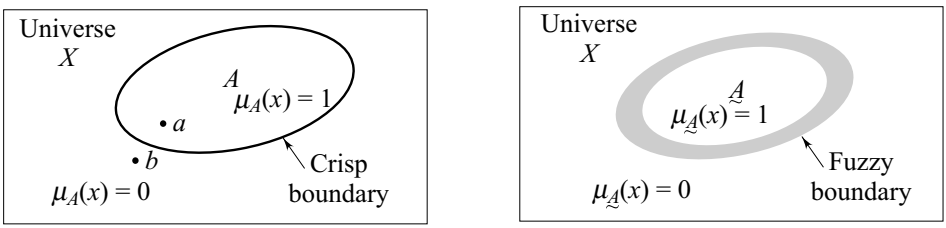
Example 12.1

One of the most commonly used examples of a fuzzy set is the set of tall people. In this case, the universe of discourse is potential heights (the real line), say, from 3 feet to 9 feet. If the set of tall people is given the

well-defined boundary of a crisp set, we might say all people taller than 6 feet are officially considered tall. The characteristic function of the set A = {tall men} then, is

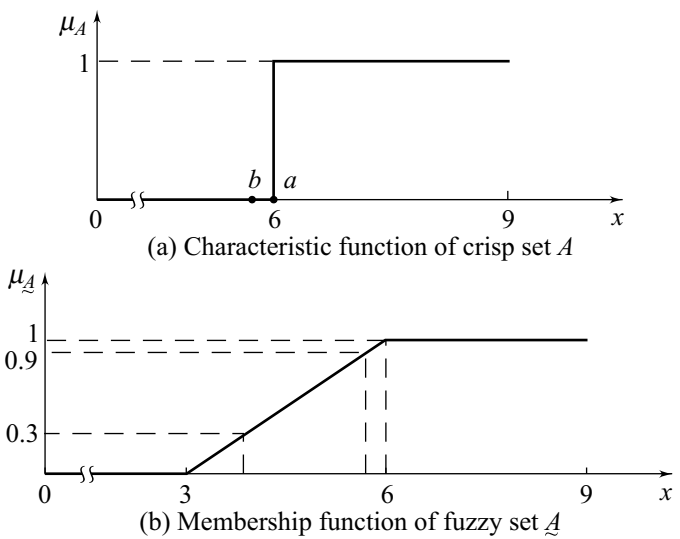
$$\mu_A(x) = \begin{cases} 1 \text{ for } 6 \le x \\ 0 \text{ for } 3 \le x < 6 \end{cases}$$

Such a condition is expressed by a Venn diagram shown in Fig. 12.7a, and a characteristic function shown in Fig. 12.8a.



(a) The crisp set A and the universe of discourse (b) The fuzzy set $\underset{\sim}{A}$ and the universe of discourse

Fig. 12.7



(a) Characteristic function of crisp set A

(b) Membership function of fuzzy set $\underset{\sim}{A}$

Fig. 12.8

For our example of universe X of heights of people, the crisp set A of all people with $x \ge 6$ has a sharp boundary: individual, 'a' corresponding to $x = 6$ is a member of the crisp set A, and individual 'b' corresponding to $x = 5.9$ is unambiguously *not* a member of set A. Is it not an absurd statement for the situation under consideration? A 0.1" reduction in the height of a person has changed μ_A from 1 to 0, and the person is no more tall.

It may make sense to consider the crisp set of all real numbers greater than 6 because the numbers belong to an abstract plane, but when we want to talk about real people, it is unreasonable to call one person

short and another one tall, when they differ in height by the width of a hair. But if this kind of distinction is unworkable, then what is the right way to define the set of all people? Much as with our example of 'set of young females', the word 'tall' would correspond to a curve that defines the degree to which any person is tall. Figure 12.8b shows a possible membership function of this fuzzy set $\underset{\sim}{A}$; the curve defines the transition from not tall to tall. Two people with membership values 0.9 and 0.3 are tall to some degree, but one significantly less than the other.

Note that there is inherent subjectivity in fuzzy set description. Figure 12.9 shows a smoothly varying curve (*S*-shaped) for transition from not tall to tall. Compared to Fig. 12.8b, the membership values are lower for heights close to 3′ and are higher for heights close to 6′. This looks more reasonable; however, the price paid is in terms of a more complex function, which is more difficult to handle.

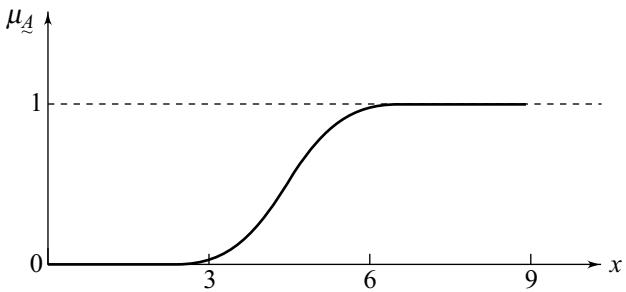


Fig. 12.9 A smoothly varying membership function for fuzzy set $\underset{\sim}{A}$

Figure 12.7b shows the representation of a fuzzy set by a Venn diagram. In the central (unshaded) region of the fuzzy set, $\mu_{\underset{\sim}{A}}(x) = 1$. Outside the boundary region of fuzzy set, $\mu_{\underset{\sim}{A}}(x) = 0$. On the boundary region, $\mu_{\underset{\sim}{A}}(x)$ assumes an intermediate value in the interval (0, 1). Presumably, the membership value of an x in fuzzy set, $\underset{\sim}{A}$, approaches a value of 1 as it moves closer to the central (unshaded) region; it approaches a value of 0 as it moves closer to leaving the boundary region of $\underset{\sim}{A}$.

Thus, so far we have discussed the *representation* of knowledge in logic. We have seen that the concept of fuzzy sets makes it possible to describe vague information (knowledge). But description alone will not lead to the development of any useful products. Indeed, a good deal of time passed after fuzzy sets were first proposed, until they were applied at the industrial level. However, eventually it became possible to apply them in the form of 'fuzzy inference', and fuzzy logic theory has now become legitimized as one component of applied high technology.

In fuzzy logic theory, nothing is done at random or haphazardly. Information containing a certain amount of vagueness is expressed as faithfully as possible, without the distortion produced by forcing it into a 'crisp' mould, and it is then processed by applying an appropriate rule of inference.

'Approximate reasoning' is the best known form of fuzzy logic processing and covers a variety of inference rules.

Fuzziness is often confused with probability. The fundamental difference between them is that fuzziness deals with deterministic plausibility, while probability concerns the likelihood of nondeterministic (stochastic) events. Fuzziness is one aspect of uncertainty. It is the ambiguity (vagueness) found in the definition of a concept, or the meaning of a term. However, the uncertainty of probability generally

relates to the occurrence of phenomena, not the vagueness of phenomena. For example, 'There is a 50–50 chance that he will be there' has the uncertainty of randomness. 'He is a young man', has the uncertainty in definition of 'young man'. Thus, fuzziness describes the ambiguity of an event, whereas randomness describes the uncertainty in the occurrence of an event.

We can now give a formal definition to fuzzy sets.

12.3.2 Fuzzy Sets

A *universe of discourse*, X, is a collection of objects all having the same characteristics. The individual elements in the universe X will be denoted as x.

A universe of discourse and a membership function that spans the universe, completely define a *fuzzy set*. Consider a universe of discourse X with x representing its generic element. A fuzzy set $\underset{\sim}{A}$ in X has the membership function $\mu_{\underset{\sim}{A}}(x)$ which maps the elements of the universe onto numerical values in the interval [0, 1]:

$$\mu_{\underset{\sim}{A}}(x) : X \rightarrow [0, 1] \tag{12.8a}$$

Every element x in X has a membership function $\mu_{\underset{\sim}{A}}(x) \in [0, 1]$. $\underset{\sim}{A}$ is then defined by the set of ordered pairs:

$$\underset{\sim}{A} = \{(x, \mu_{\underset{\sim}{A}}(x)) \mid x \in X, \mu_{\underset{\sim}{A}}(x) \in [0, 1]\} \tag{12.8b}$$

A membership value of zero implies that the corresponding element is definitely *not* an element of the fuzzy set $\underset{\sim}{A}$. A membership function of unity means that the corresponding element is definitely an element of fuzzy set $\underset{\sim}{A}$. A grade of membership greater than zero, and less than unity, corresponds to a noncrisp (or fuzzy) membership of the fuzzy set $\underset{\sim}{A}$. Classical sets can be considered as special case of fuzzy sets with all membership grades equal to unity.

A fuzzy set $\underset{\sim}{A}$ is formally given by its membership function $\mu_{\underset{\sim}{A}}(x)$. We will identify any fuzzy set with its membership function, and use these two terms interchangeably.

Membership functions characterize the fuzziness in a fuzzy set. However, the shape of the membership functions, used to describe the fuzziness, has very few restrictions indeed. It might be claimed that the rules used to describe fuzziness are also fuzzy. Just as there are an infinite number of ways to characterize fuzziness, there are an infinite number of ways to graphically depict the membership functions that describe fuzziness. Although the selection of membership functions is *subjective*, *it cannot be arbitrary*; it should be *plausible*.

To avoid unjustified complications, $\mu_{\underset{\sim}{A}}(x)$ is usually constructed without a high degree of precision. It is advantageous to deal with membership functions involving a small number of parameters. Indeed, one of the key issues in the theory and practice of fuzzy sets is how to define the proper membership functions. Primary approaches include (1) asking the control expert to define them; (2) using data from the system to be controlled, to generate them; and (3) making them in a trial-and-error manner. In more than 25 years of practice, it has been found that the third approach, though ad hoc, works effectively and efficiently in many real-world applications.

Numerous applications in control have shown that only four types of membership functions are needed in most circumstances: trapezoidal, triangular (a special case of trapezoidal), Gaussian, and bell-shaped.

Figure 12.10 shows an example of each type. Among the four, the first two are more widely used. All these fuzzy sets are continuous, normal and convex.

A fuzzy set is said to be *continuous* if its membership function is continuous.

A fuzzy set is said to be *normal* if its *height* is one (The largest membership value of a fuzzy set is called the height of the fuzzy set).

The *convexity* property of fuzzy sets is viewed as a generalization of the classical concept of crisp sets. Consider the universe X to be a set of real numbers $\Re$. A subset A of $\Re$ is said to be *convex* if, and only if, for all $x_1, x_2 \in A$ and for every real number λ satisfying $0 \le \lambda \le 1$, we have

$$\lambda x_1 + (1 - \lambda)x_2 \in A \tag{12.9}$$

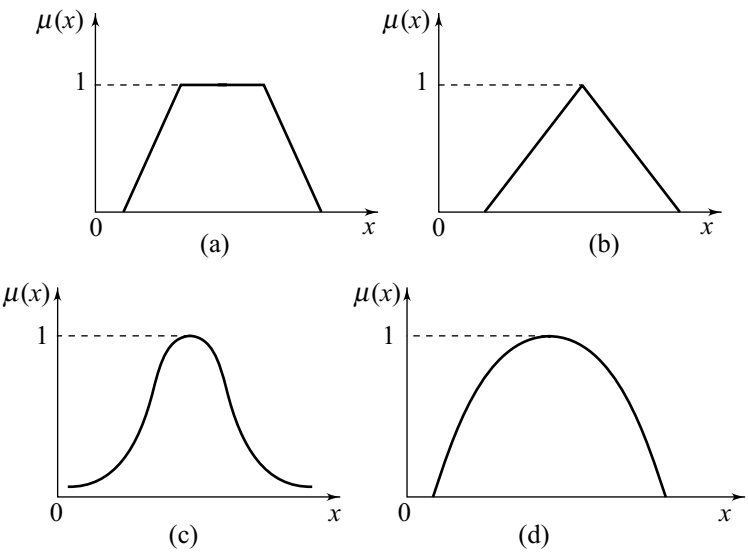
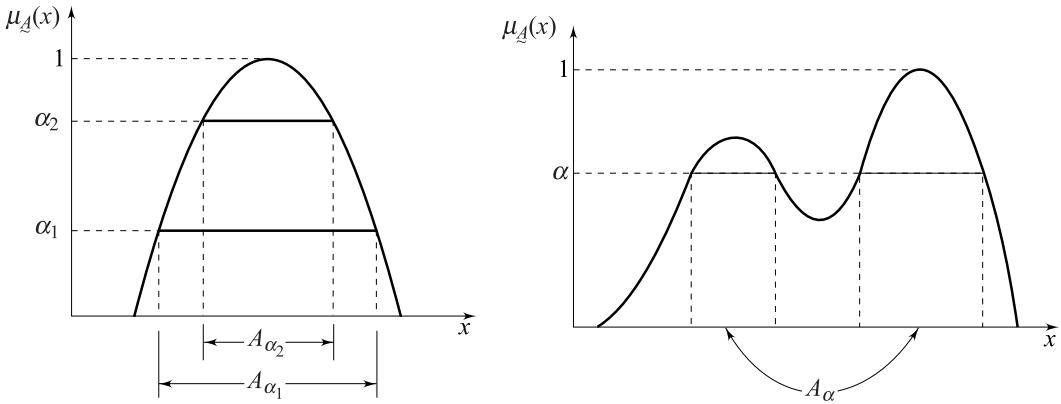


Fig. 12.10 Examples of fuzzy sets: (a) Trapezoidal (b) Triangular (c) Gaussian (d) Bell-shaped

It can easily be established that any set defined by a single interval of real numbers is convex; any set defined by more than one interval, that does not contain some points between the intervals, is not convex.

An *alpha-cut* of a fuzzy set $\underset{\sim}{A}$ is a crisp set A_α that contains all the elements of the universal set X that have a membership grade in $\underset{\sim}{A}$ greater than or equal to α (refer to Fig. 12.11). The convexity property of fuzzy sets is viewed as a generalization of the classical concept of convexity of crisp sets. In order to make the generalized convexity consistent with the classical definition of convexity, it is required that α-cuts of a *convex fuzzy set* be convex for all $\alpha \in (0, 1]$ in the classical sense (0-cut is excluded here since it is always equal to $\Re$ in this sense and thus includes $-\infty$ to $+\infty$). Figure 12.11a shows a fuzzy set that is convex. Two of the α-cuts shown in this figure are clearly convex in the classical sense, and it is easy to see that any other α-cuts for $\alpha > 0$ are convex as well. Figure 12.11b illustrates a fuzzy set that is not convex. The lack of convexity of this fuzzy set can be demonstrated by identifying some of its α-cuts ($\alpha > 0$) that are not convex.



(a) Convex fuzzy set (b) Nonconvex fuzzy set

Fig. 12.11

The *support* of a fuzzy set $\underset{\sim}{A}$ is the crisp set of all $x \in X$ such that $\mu_{\underset{\sim}{A}}(x) > 0$. That is

$$\text{supp}\,(\underset{\sim}{A}) = \{x \in X \mid \mu_{\underset{\sim}{A}}(x) > 0\} \tag{12.10}$$

The element $x \in X$ at which $\mu_{\underset{\sim}{A}}(x) = 0.5$, is called the *crosspoint*.

A fuzzy set $\underset{\sim}{A}$ whose support is a single point in X with $\mu_{\underset{\sim}{A}}(x) = 1$, is referred to as a *fuzzy singleton*.

Example 12.2

Consider the fuzzy set described by membership function depicted in Fig. 12.12, where the universe of discourse is

$$X = [32^\circ\text{F}, 104^\circ\text{F}]$$

This fuzzy set $\underset{\sim}{A}$ is linguistic 'warm', with membership function

$$\mu_{\underset{\sim}{A}}(x) = \begin{cases} 0 \;;\; x < 64^\circ \\ (x - 64^\circ)/6 \;;\; 64^\circ \le x < 70^\circ \\ 1 \;;\; 70^\circ < x \le 74^\circ \\ (78^\circ - x)/4 \;;\; 74 < x \le 78^\circ \\ 0 \;;\; x > 78^\circ \end{cases}$$

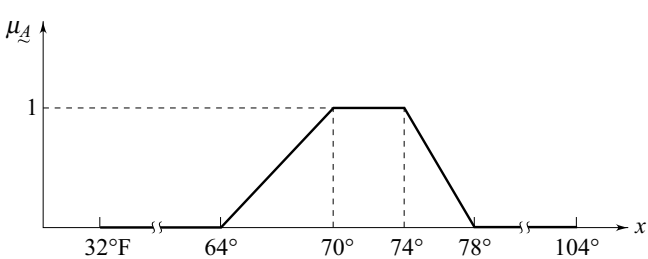


Fig. 12.12 A fuzzy set: linguistic 'warm'

The support of $\underset{\sim}{A}$ is the crisp set

$$\{x \mid 64^\circ < x < 78^\circ\}$$

Example 12.3

Consider a natural language form expression:

'Speed sensor output is very large'

The formal, symbolic translation of this natural language expression, in terms of linguistic variables, proceeds as follows:

(i) An abbreviation 'Speed' may be chosen to denote the physical variable 'Speed sensor output'.
(ii) An abbreviation 'XFast' (i.e., extra fast) may be chosen to denote the particular value 'very large' of speed.
(iii) The above natural language expression is rewritten as 'Speed is XFast'.

Such an expression is an *atomic fuzzy proposition*. The 'meaning' of the atomic proposition is then defined by a fuzzy set $\underset{\sim}{XFast}$, or a membership function $\mu_{\underset{\sim}{XFast}}(x)$, defined on the physical domain X = [0 mph, 100 mph] of the physical variable 'Speed'.

Many atomic propositions may be associated with a linguistic variable, e.g.,

'Speed is Fast'
'Speed is Moderate'
'Speed is Slow'
'Speed is XSlow'

Thus, the set of *linguistic values* that the linguistic variable 'Speed' may take is

$$\{XFast, Fast, Moderate, Slow, XSlow\}$$

These linguistic values are called *terms* of the linguistic variable. Each term is defined by an appropriate membership function.

It is usual in approximate reasoning to have the following frame associated with the notion of a linguistic variable:

$$\left\langle \underset{\sim}{A}, \mathcal{L}\underset{\sim}{A}, X, \mu_{\mathcal{L}\underset{\sim}{A}} \right\rangle \tag{12.11}$$

where $\underset{\sim}{A}$ denotes the symbolic name of a linguistic variable, e.g., speed, temperature, level, error, change-of-error, etc. $\mathcal{L}\underset{\sim}{A}$ is the set of linguistic values that $\underset{\sim}{A}$ can take, i.e., $\mathcal{L}\underset{\sim}{A}$ is the term set of $\underset{\sim}{A}$. X is the actual physical domain over which linguistic variable $\underset{\sim}{A}$ takes its quantitative (crisp) values, and $\mu_{\mathcal{L}\underset{\sim}{A}}$ is a membership function which gives a meaning to the linguistic value in terms of the quantitative elements of X.

Example 12.4

Consider speed, interpreted as a linguistic variable with X = [0mph, 100mph]; i.e., x = 'speed'. Its term set could be

$$\{Slow, Moderate, Fast\}$$

$\underset{\sim}{Slow}$ = the fuzzy set for 'a speed below about 40 miles per hour (mph)', with membership function $\mu_{\underset{\sim}{Slow}}$

$\underset{\sim}{Moderate}$ = the fuzzy set for 'a speed close to 55 mph', with membership function $\mu_{\underset{\sim}{Moderate}}$.

$\underset{\sim}{Fast}$ = the fuzzy set for 'a speed above about 70 mph', with membership function $\mu_{\underset{\sim}{Fast}}$.

The frame of speed is

$$\left\langle Speed,\ \mathcal{L}Speed,\ X,\ \mu_{\mathcal{L}Speed} \right\rangle$$

where

$$\mathcal{L}Speed = \{Slow,\ Moderate,\ Fast\}$$

$$X = [0,\ 100] \text{ mph}$$

$\mu_{Slow}, \mu_{Moderate}, \mu_{Fast}$ are given in Fig. 12.13.

The frame of speed helps us to decide the degree to which an atomic proposition associated with 'speed' is satisfied, given a specific physical value of speed. For example, for crisp input

$$Speed = 50 \text{ mph}$$

$$\mu_{Slow}(50) = 1/3$$

$$\mu_{Moderate}(50) = 2/3$$

$$\mu_{Fast}(50) = 0$$

Therefore, the proposition 'Speed is Slow' is satisfied to a degree of 1/3, the proposition 'Speed is Moderate' is satisfied to a degree of 2/3, and the proposition 'Speed is Fast' is not satisfied.

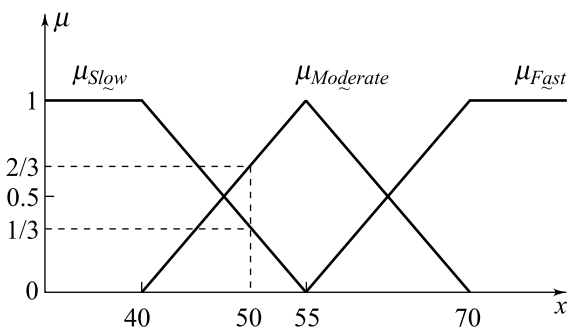


Fig. 12.13 Terms of linguistic variable 'speed'

An extension of ordinary fuzzy sets is to allow the membership values to be a fuzzy set—instead of a crisply defined degree. A fuzzy set whose membership function is itself a fuzzy set, is called a *Type*-2 *fuzzy set* [143]. A *Type*-1 *fuzzy set* is an ordinary fuzzy set. We will limit our discussion to Type-1 fuzzy sets. The reference to a fuzzy set in this chapter, implies a Type-1 fuzzy set.

12.3.3 Fuzzy Operations

There are a variety of fuzzy set theories which differ from one another by the set operations (complement, intersection, union) they employ. The fuzzy complement, intersection and union are not unique operations, contrary to their crisp counterparts; different functions may be appropriate to represent these operations in different contexts. That is, not only membership functions of fuzzy sets, but also operations on fuzzy sets, are context-dependent. The capability to determine appropriate membership functions, and meaningful fuzzy operations in the context of each particular application, is crucial for making fuzzy set theory practically useful.

The intersection and union operations on fuzzy sets are often referred to as *triangular norms* (*t*-norms), and *triangular conorms* (*t*-conorms; also called *s*-norms), respectively. The reader is advised to refer to [143] for the axioms which *t*-norms, *t*-conorms, and the complements of fuzzy sets are required to satisfy.

In the following, we define *standard fuzzy operations*, which are generalizations of the corresponding crisp set operations.

Consider the fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ in the universe X.

$$\underset{\sim}{A} = \{(x, \mu_{\underset{\sim}{A}}(x)) \mid x \in X; \mu_{\underset{\sim}{A}}(x) \in [0, 1]\} \tag{12.12}$$

$$\underset{\sim}{B} = \{(x, \mu_{\underset{\sim}{B}}(x)) \mid x \in X; \mu_{\underset{\sim}{B}}(x) \in [0, 1]\} \tag{12.13}$$

The operations with $\underset{\sim}{A}$ and $\underset{\sim}{B}$ are introduced *via* operations on their membership functions $\mu_{\underset{\sim}{A}}(x)$ and $\mu_{\underset{\sim}{B}}(x)$ correspondingly.

Complement

The standard complement, $\overline{\underset{\sim}{A}}$, of fuzzy set $\underset{\sim}{A}$ with respect to the universal set X, is defined for all $x \in X$ by the equation

$$\mu_{\overline{\underset{\sim}{A}}}(x) \triangleq 1 - \mu_{\underset{\sim}{A}}(x) \; \forall x \in X \tag{12.14}$$

Intersection

The standard operation, $\underset{\sim}{A} \cap \underset{\sim}{B}$ is defined for all $x \in X$ by the equation

$$\mu_{\underset{\sim}{A} \cap \underset{\sim}{B}}(x) \triangleq \min [\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(x)] \equiv \mu_{\underset{\sim}{A}}(x) \wedge \mu_{\underset{\sim}{B}}(x) \; \forall x \in X \tag{12.15}$$

where $\wedge$ indicates the **min** operation.

Union

The standard union, $\underset{\sim}{A} \cup \underset{\sim}{B}$, is defined for all $x \in X$ by the equation

$$\mu_{\underset{\sim}{A} \cup \underset{\sim}{B}}(x) \triangleq \max [\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(x)] \equiv \mu_{\underset{\sim}{A}}(x) \vee \mu_{\underset{\sim}{B}}(x) \; \forall \, x \in X \tag{12.16}$$

where $\vee$ indicates the **max** operation.

12.3.4 Fuzzy Relations

Consider two universes (crisp sets) X and Y. The *Cartesian product* (or *cross product*) of two sets X and Y (in this order) is the set of all ordered pairs, such that, the first element in each pair is a member of X, and the second element is a member of Y. Formally,

$$X \times Y = \{(x, y); x \in X, y \in Y\} \tag{12.17}$$

where $X \times Y$ denotes the Cartesian product.

A *fuzzy relation* on $X \times Y$, denoted by $\underset{\sim}{R}$, or $\underset{\sim}{R}(X, Y)$ is defined as the set

$$\underset{\sim}{R} = \{((x, y), \mu_{\underset{\sim}{R}}(x, y)) \mid (x, y) \in X \times Y, \mu_{\underset{\sim}{R}}(x, y) \in [0, 1]\} \tag{12.18}$$

where $\mu_{\underset{\sim}{R}}(x, y)$ is a function in two variables, called membership function of the fuzzy relation. It gives the degree of membership of the ordered pair (x, y) in $\underset{\sim}{R}$, associating with each pair (x, y) in $X \times Y$, a real number in the interval [0, 1]. The degree of membership indicates the degree to which x is in relation with y. It is clear that a fuzzy relation is basically a fuzzy set.

Example 12.5

Consider an example of fuzzy sets: the set of people with normal weight. In this case, the universe of discourse appears to be all potential weights (the real line). However, the knowledge representation in terms of this universe, is not useful. The normal weight of a person is a function of his/her height.

$$\text{Body Mass Index (BMI)} = \frac{\text{Weight kg}}{(\text{Height, m})^2}$$

Normal BMI for males is 20–25, and for females is 19–24. Values between 25 to 27 in men and 24 to 27 in women indicate overweight; and those over 27 indicate obesity. Of course, values below 20 for men and below 19 for women indicate underweight.

The universe of discourse for this fuzzy set is more appropriately the Cartesian product of two universal sets: X, the set of all potential heights, and Y, the set of all potential weights. The Cartesian product space $X \times Y$ is a universal set which is a set of ordered pairs (x, y), for each $x \in X$ and each $y \in Y$.

A subset of the Cartesian product $X \times Y$, satisfying the knowledge attribute 'normal weight' is a set of (height, weight) pairs. This is called a relation $\underset{\sim}{R}$. The membership value for each element of $\underset{\sim}{R}$ depends on BMI. For men, a BMI of 27 and more could be given a membership value of 0, and a BMI of less than 18 could also be given a membership value of 0; and membership value between 0 and 1 for BMI between 18 and 27.

Example 12.6

Because fuzzy relations, in general, are fuzzy sets, we can define the Cartesian product to be a relation between two or more fuzzy sets. Let $\underset{\sim}{A}$ be a fuzzy set on universe X, and $\underset{\sim}{B}$ be a fuzzy set on universe Y; then the Cartesian product between fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ will result in a fuzzy relation $\underset{\sim}{R}$, which is contained within the full Cartesian product space, or

$$\underset{\sim}{A} \times \underset{\sim}{B} = \underset{\sim}{R} \subset X \times Y \tag{12.19a}$$

where the fuzzy relation $\underset{\sim}{R}$ has membership function

$$\mu_{\underset{\sim}{R}}(x, y) = \mu_{\underset{\sim}{A} \times \underset{\sim}{B}}(x, y) = \min[\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)] \; \forall x \in X, \forall y \in Y \tag{12.19b}$$

Note that the **min** combination applies here because each element (x, y), in the Cartesian product, is formed by taking both elements x, y together, not just the one or the other.

As an example of the Cartesian product of fuzzy sets, we consider premise quantification. Atomic fuzzy propositions do not, usually, make a knowledge base in real-life situations. Many propositions connected by logical connectives may be needed. A set of such *compound propositions*, connected by IF-THEN rules, makes a knowledge base.

Consider two propositions defined by

$$p \triangleq x \text{ is } \underset{\sim}{A}$$

$$q \triangleq y \text{ is } \underset{\sim}{B}$$

where $\underset{\sim}{A}$ and $\underset{\sim}{B}$ are the fuzzy sets:

$$\underset{\sim}{A} = \left\{\left(x, \mu_{\underset{\sim}{A}}(x)\right) \middle| x \in X\right\}$$

$$\underset{\sim}{B} = \left\{\left(y, \mu_{\underset{\sim}{B}}(y)\right) \middle| y \in Y\right\}$$

The meaning of the linguistic terms 'x is $\underset{\sim}{A}$', and 'y is $\underset{\sim}{B}$' is quantified *via* the membership functions $\mu_{\underset{\sim}{A}}(x)$ and $\mu_{\underset{\sim}{B}}(y)$, respectively. Now, we seek to quantify the linguistic premise 'x is $\underset{\sim}{A}$ and y is $\underset{\sim}{B}$' of the rule:

IF x is $\underset{\sim}{A}$ **and** y is $\underset{\sim}{B}$ THEN z is $\underset{\sim}{C}$ (12.20a)

The main item to focus on is, how to quantify the logical **and** operation that combines the meaning of two linguistic terms. As said earlier, there are actually several ways (*t*-norms) to define this quantification. In the following, we use **min** operation:

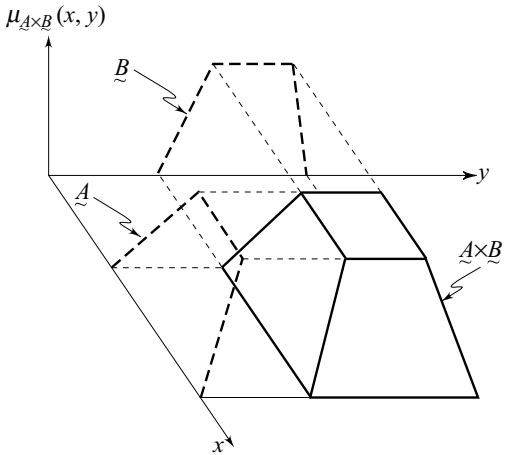


Fig. 12.14 The conjunction operator implemented as Cartesian product

$$\mu_{premise}(x, y) = \min\left[\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)\right] \forall x \in X, \forall y \in Y \tag{12.20b}$$

Does this quantification make sense? Notice that this way of quantifying the **and** operation in the premise, indicates that you can be no more certain about the conjunction of two statements, than you are about the individual terms that make them up.

The *conjunction operator* (logical connective **and**), implemented as Cartesian product, is described in Fig. 12.14.

$$\mu_{premise}(x, y) = \mu_{\underset{\sim}{A}\times\underset{\sim}{B}}(x, y) = \min\left[\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)\right] \forall x \in X, \forall y \in Y \tag{12.20c}$$

Example 12.7

We consider here quantification of 'implication' operator via fuzzy logic. Consider the implication statement

IF pressure is high THEN volume is small

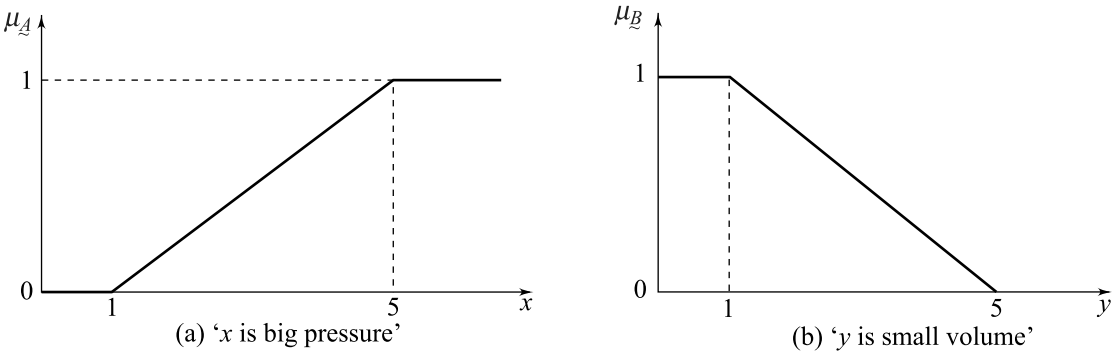
The membership function of the fuzzy set $\underset{\sim}{A}$ = 'big pressure',

$$\mu_{\underset{\sim}{A}}(x) = \begin{cases} 1 & ; x \geq 5 \\ 1-(5-x)/4 & ; 1 \leq x \leq 5 \\ 0 & ; \text{otherwise} \end{cases}$$

is shown in Fig. 12.15a. The membership function of the fuzzy set $\underset{\sim}{B}$ = 'small volume',

$$\mu_{\underset{\sim}{B}}(y) = \begin{cases} 1 & ; y \leq 1 \\ 1-(y-1)/4 & ; 1 \leq y \leq 5 \\ 0 & ; \text{otherwise} \end{cases}$$

is shown in Fig. 12.15b.



(a) 'x is big pressure' (b) 'y is small volume'

Fig. 12.15

If p is a proposition of the form 'x is $\underset{\sim}{A}$' where $\underset{\sim}{A}$ is a fuzzy set on the universe X, e.g., 'big pressure', and q is a proposition of the form 'y is $\underset{\sim}{B}$' where $\underset{\sim}{B}$ is a fuzzy set on the universe Y, e.g., 'small volume', then one encounters the following problem:

How does one define the membership function of the fuzzy implication $\underset{\sim}{A} \rightarrow \underset{\sim}{B}$?

There are different important classes of fuzzy implication operators based on t-norm and t-conorm. In many practical applications, one uses *Mamdani's implication operator* to model causal relationship between fuzzy variables:

$$\mu_{\underset{\sim}{A}\rightarrow\underset{\sim}{B}}(x, y) = \min\left[\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)\right] \ \forall x \in X, \forall y \in Y \tag{12.21}$$

The fuzzy implication $\underset{\sim}{A} \rightarrow \underset{\sim}{B}$ is a fuzzy relation in the Cartesian product space $X \times Y$.

Note that Mamdani's implication operator gives a relation which is symmetric with respect to $\underset{\sim}{A}$ and $\underset{\sim}{B}$. This is not intuitively satisfying, because 'implication' is not a commutative operation. In practice, however, the method provides good, robust results. The justification for the use of the **min** operator to represent the implication, is that *we can be no more certain about our consequent than our premise*.

12.4 FUZZY INFERENCE

Problems featuring uncertainty and ambiguity have been successfully addressed subconsciously by humans. Humans can adapt to unfamiliar situations and they are able to gather information in an efficient manner and discard irrelevant details. The information gathered need not be complete and precise, and could be general, qualitative and vague, because humans can reason, infer and deduce new information and knowledge. They can learn, perceive and improve their skills through experience.

How can humans reason about complex systems, when the complete description of such a system often requires more detailed data than a human could ever hope to recognize simultaneously, and assimilate with understanding? The answer is that humans have the capacity to *reason approximately*. In reasoning about a complex system, humans reason approximately about its behavior, thereby maintaining only a generic understanding about the problem.

The fuzzy set theory has provided a mathematical strength to capture the uncertainties associated with human congnitive processes, such as thinking and reasoning. Fuzzy logic provides an *inference* morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems.

Fuzzy conditional, or fuzzy IF-THEN production rules are symbolically expressed as

$$\{\text{IF (premise } i\text{) THEN (consequent } i\text{)}\}_{i=1}^{N}$$

Here N is the number of rules.

Two major types of fuzzy rules exist: Mamdani fuzzy rules, and Sugeno fuzzy rules.

12.4.1 Mamdani Fuzzy Rules

In Mamdani fuzzy rules, both the premises and the consequents are fuzzy propositions (atomic/compound). Consider first the case of a rule with atomic propositions. For example:

$$\text{'IF } x \text{ is } \underset{\sim}{A} \text{ THEN } y \text{ is } \underset{\sim}{B}\text{'} \tag{12.22a}$$

If we let X be the premise universe of discourse, and Y the consequent universe of discourse, then the relation between the premise $\underset{\sim}{A}$ and consequent $\underset{\sim}{B}$ can be described using fuzzy sets on the Cartesian product space $X \times Y$. Using Mamdani's implication rule,

$$\begin{aligned} \underset{\sim}{R} &= \underset{\sim}{A} \rightarrow \underset{\sim}{B} \\ \mu_{\underset{\sim}{R}}(x, y) &= \mu_{\underset{\sim}{A} \rightarrow \underset{\sim}{B}}(x, y) \\ &= \min[\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)] \quad \forall x \in X, \forall y \in Y \end{aligned} \tag{12.22b}$$

When the rule premise or rule consequent are compound fuzzy propositions, then the membership function, corresponding to each such compound proposition, is first determined. The above operation is applied to represent IF-THEN relation. Quite often, in control applications, we come across logical connective **and** (conjunction operation on atomic propositions), which, as we have seen in Example 12.6, may be implemented by Cartesian product.

The rules of inference in fuzzy logic govern the deduction of final conclusion from IF-THEN rules for known inputs (Fig. 12.16). Consider the statements:

$$\begin{array}{lll} \text{rule} & : & \text{IF } x \text{ is } \underset{\sim}{A} \text{ THEN } y \text{ is } \underset{\sim}{B} \\ \text{input} & : & x \text{ is } \underset{\sim}{A}' \\ \hline \text{inference} & : & y \text{ is } \underset{\sim}{B}' \end{array} \tag{12.23}$$

Here the propositions 'x is $\underset{\sim}{A}$', 'x is $\underset{\sim}{A}'$', 'y is $\underset{\sim}{B}$' and 'y is $\underset{\sim}{B}'$' are characterized by fuzzy sets $\underset{\sim}{A}$, $\underset{\sim}{A}'$, $\underset{\sim}{B}$, and $\underset{\sim}{B}'$, respectively.

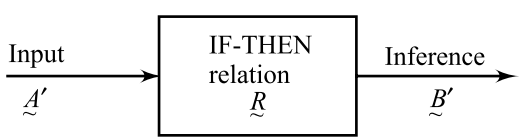


Fig. 12.16 Inference mechanism

$$\begin{aligned} \underset{\sim}{A} &= \{(x, \mu_{\underset{\sim}{A}}(x)) \mid x \in X;\ \mu_{\underset{\sim}{A}} \in [0, 1]\} \\ \underset{\sim}{A}' &= \{(x, \mu_{\underset{\sim}{A}'}(x)) \mid x \in X;\ \mu_{\underset{\sim}{A}'} \in [0, 1]\} \\ \underset{\sim}{B} &= \{(y, \mu_{\underset{\sim}{B}}(y)) \mid y \in Y;\ \mu_{\underset{\sim}{B}} \in [0, 1]\} \\ \underset{\sim}{B}' &= \{(y, \mu_{\underset{\sim}{B}'}(y)) \mid y \in Y;\ \mu_{\underset{\sim}{B}'} \in [0, 1]\} \end{aligned} \tag{12.24}$$

Fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{A}'$ are close but not equal, and same is valid for the sets $\underset{\sim}{B}$ and $\underset{\sim}{B}'$.

Inference mechanism is based on matching of two fuzzy sets $\underset{\sim}{A}'$ and $\underset{\sim}{R}$, and determining membership function of $\underset{\sim}{B}'$ according to the result. Note that X denotes the space in which the input $\underset{\sim}{A}'$ is defined, and it is subspace of the space $X \times Y$ in which the rule-base relation $\underset{\sim}{R}$ is defined. It is, therefore, not possible to take the intersection of $\underset{\sim}{A}'$ and $\underset{\sim}{R}$; an operation required for matching the two sets, to incorporate the knowledge of the membership functions of both the input and the rule base. But when $\underset{\sim}{A}'$ is extended to $X \times Y$, this is possible.

Cylindrical extension of $\underset{\sim}{A}'$ (a fuzzy set defined on X) on $X \times Y$ is the set of all tuples $(x, y) \in X \times Y$, with membership degree equal to $\mu_{\underset{\sim}{A}'}(x)$, i.e.,

$$\mu_{ce(\underset{\sim}{A}')}(x, y) = \mu_{\underset{\sim}{A}'}(x) \text{ for every } y \in Y \tag{12.25}$$

Now, the intersection operation, to incorporate the knowledge of membership functions of input and rule base, is possible. It is given by

$$ce(\underset{\sim}{A}') \cap \underset{\sim}{R}$$

In terms of membership functions, this operation may be expressed as

$$\mu_{ce(\underset{\sim}{A}')}(x, y) \wedge \mu_{\underset{\sim}{R}}(x, y) = \min[\mu_{ce(\underset{\sim}{A}')}(x, y), \mu_{\underset{\sim}{R}}(x, y)] \quad \forall x \in X, \forall y \in Y$$

$$\mu_{\underset{\sim}{R}}(x, y) = \mu_{\underset{\sim}{A} \to \underset{\sim}{B}}(x, y) = \min[\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)]$$

$$\mu_{ce(\underset{\sim}{A}')}(x, y) = \mu_{\underset{\sim}{A}'}(x)$$

Therefore,

$$\mu_{\underset{\sim}{S}}(x, y) = \mu_{ce(\underset{\sim}{A}')}(x, y) \wedge \mu_{\underset{\sim}{R}}(x, y) = \min(\mu_{\underset{\sim}{A}'}(x), \min(\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y))) \tag{12.26}$$

By projecting this matched fuzzy set (defined on $X \times Y$) over the inference subspace Y, we can determine the membership function $\mu_{\underset{\sim}{B}'}(y)$ of the fuzzy set $\underset{\sim}{B}'$ (defined on Y).

Projection of $\mu_{\underset{\sim}{S}}(x, y)$ (a fuzzy set defined on $X \times Y$) on Y, is a set of all $y \in Y$ with membership grades equal to $\max_x\{\mu_{\underset{\sim}{S}}(x, y)\}$; $\max_x$ means maximum with respect to x while y is considered fixed, i.e.,

$$\mu_{proj(\underset{\sim}{S})}(y) = \max_x\{\mu_{\underset{\sim}{S}}(x, y)\} \tag{12.27}$$

Projection on Y means that y_i is assigned the highest membership degree from the tuples (x_1, y_i), (x_2, y_i), (x_3, y_i), ..., where $x_1, x_2, x_2, ... \in X$ and $y_i \in Y$. The rationale for using the **max** operation on the membership functions of $\underset{\sim}{S}$ should be clear in view of the fact that we have a many-to-one mapping.

The combination of fuzzy sets with the aid of cylindrical extension and projection, is called *composition*. It is denoted by $\circ$.

If $\underset{\sim}{A}'$ is a fuzzy set defined on X and $\underset{\sim}{R}$ is a fuzzy relation defined on $X \times Y$, then the composition of $\underset{\sim}{A}'$ and $\underset{\sim}{R}$ resulting in a fuzzy set $\underset{\sim}{B}'$ defined on Y is given by

$$\underset{\sim}{B}' = \underset{\sim}{A}' \circ \underset{\sim}{R} = proj\ (ce(\underset{\sim}{A}') \cap \underset{\sim}{R}) \text{ on } Y \tag{12.28}$$

Note that, in general, intersection is given by a t-norm, and projection by a t-conorm, resulting in many definitions of composition operator. In our applications, we will mostly use **min** operator for t-norm and **max** operator for t-conorm. Therefore, we have the following *compositional rule of inference*:

$$\mu_{\underset{\sim}{B}'}(y) = \max_x\{\min(\mu_{\underset{\sim}{A}'}(x), \min(\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)))\} \tag{12.29}$$

This inference rule, based on *max-min composition*, uses Mamdani's rule for implication operator.

In control applications, as we shall see later, the fuzzy set $\underset{\sim}{A}'$ is fuzzy singleton, i.e.,

$$\mu_{\underset{\sim}{A}'}(x) = \begin{cases} 1 & \text{for } x = x_0 \in X \\ 0 & \text{for all other } x \in X \end{cases} \tag{12.30}$$

This results in a simple inference procedure, as is seen below.

$$\mu_{\underset{\sim}{B}'}(y) = \begin{cases} \min(\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)) & \text{for } x = x_0, \forall y \in Y \\ 0 & \text{for all other } x, \forall y \in Y \end{cases} \tag{12.31}$$

Graphical representation of the procedure is shown in Fig. 12.17.

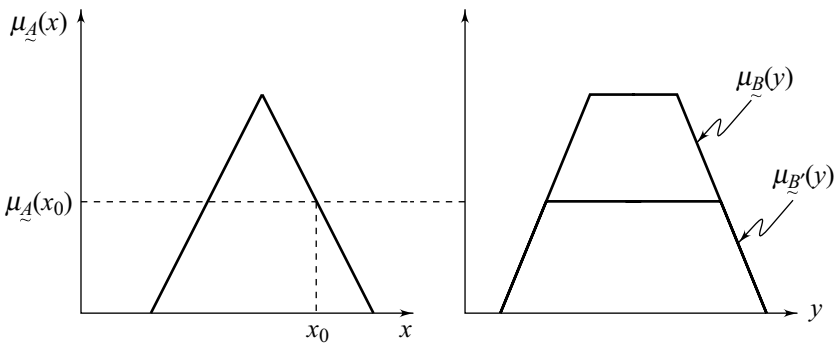


Fig. 12.17 Inference procedure for singleton fuzzy system

When the rule (12.20a) has a compound proposition in premise part, connected by logical connectives, then $\mu_{\underset{\sim}{A}}$ in (12.31) is replaced by $\mu_{premise}$. For rules with AND'ed premise, one might use **min** or **product** *t*-norm for calculating $\mu_{premise}$ (we have used **min** in Eqn. (12.20b)); and for rules with OR'ed premise, we may use **max** *t*-conorm for the calculation of $\mu_{premise}$. Of course, other *t*-norms and *t*-conorms are also premissible.

Singleton fuzzy system is most widely used because of its simplicity and lower computational requirements. However, this kind of fuzzy system may not always be adequate, especially in cases where noise is present in the data. *Nonsingleton fuzzy system* becomes necessary to account for uncertainty in the data.

12.4.2 Sugeno Fuzzy Rules

Unlike Mamdani fuzzy rules, Sugeno rules are functions of input variables on the rule consequent. A typical rule, with two input variables and one output variable, is of the form:

$$\text{IF } x_1 \text{ is } \underset{\sim}{A} \textbf{ and } x_2 \text{ is } \underset{\sim}{B} \text{ THEN } y = f(x_1, x_2) \tag{12.32a}$$

where $f(\cdot)$ is a real function.

In theory, $f(\cdot)$ can be any real function, linear or nonlinear. It seems to be appealing to use nonlinear functions; rules are more general and can potentially be more powerful. Unfortunately, the idea is impractical; properly choosing or determining the mathematical formalism of nonlinear functions for every fuzzy rule in the rule base, is extremely difficult, if not impossible. For this reason, linear functions

have been employed exclusively in theoretical research, and practical development, of Sugeno fuzzy models. For a system with two input variables and one output variable, *i*th rule in the rule base is of the form:

$$\text{IF } x_1 \text{ is } \underset{\sim}{A}^{(i)} \textbf{ and } x_2 \text{ is } \underset{\sim}{B}^{(i)} \text{ THEN } y^{(i)} = \alpha_{i,0} + \alpha_{i,1}\, x_1 + \alpha_{i,2}\, x_2 \tag{12.32b}$$

where the $\alpha_{i,j}$ are real numbers.

We can view the Sugeno fuzzy system as a nonlinear interpolator between the linear mappings that are defined by the functions in the consequents of the rules. It is important to note that a Sugeno fuzzy system may have any linear mapping as its output function which contributes to its generality. One mapping that has proven to be particularly useful, is to have a linear dynamic system as the output function so that the *i*th rule (12.32b) takes the form:

$$\text{IF } x_1 \text{ is } \underset{\sim}{A}^{(i)} \textbf{ and } x_2 \text{ is } \underset{\sim}{B}^{(i)} \text{ THEN } \dot{\mathbf{x}}(t) = \mathbf{A}_i\, \mathbf{x}(t) + \mathbf{b}_i u(t);\ i = 1, 2, ..., R \tag{12.32c}$$

where $\underset{\sim}{A}^{(i)}$ and $\underset{\sim}{B}^{(i)}$ are the fuzzy sets of the *i*th rule, and $\mathbf{A}_i$ and $\mathbf{b}_i$ are state and input matrices (of appropriate dimensions) of the local description of the linear dynamic system. Such a fuzzy system can be thought of as a nonlinear interpolator between *R* linear systems. The premise membership functions for each rule quantify whether the linear system in the consequent is valid for a specific region on the state space. As the state evolves, different rules turn on, indicating that other combinations of linear models should be used. Overall, we find that the Sugeno fuzzy system provides a very intuitive representation of a nonlinear system as a nonlinear interpolation between *R* linear models [145].

We will limit our discussion to the more widely used controllers—the Mamdani type singleton fuzzy logic systems. Sugeno architecture will be employed for data-based fuzzy modeling.

12.5 DESIGNING A FUZZY LOGIC CONTROLLER (MAMDANI ARCHITECTURE)

Figure 12.18 shows the basic configuration of a fuzzy logic controller (FLC), which comprises four principal components: a rule base, a decision-making logic, an input fuzzification interface, and an output defuzzification interface. The rule base holds a set of IF-THEN rules, that quantify the knowledge that human experts have amassed about solving particular problems. It acts as a resource to the decision-making logic, which makes successive decisions about which rules are most relevant to the current situation, and applies the actions indicated by these rules. The input fuzzifier takes the crisp numeric inputs and, as its name implies, converts them into the fuzzy form needed by the decision-making logic. At the output, the defuzzification interface combines the conclusions reached by the decision-making logic and converts them into crisp numeric values as control actions.

We will illustrate the FLC methodology, step by step, on a water-heating system.

Consider a simple water-heating system shown in Fig. 12.19. The water heater has a knob (*HeatKnob*) to control the steam for circulation through the radiator. The higher the setting of the *HeatKnob*, the hotter the water gets, with the value of '0' indicating the steam supply is turned off, and the value of '10' indicating the maximum possible steam supply. There is a sensor (*TempSense*) in the outlet pipe to tell us the temperature of the outflowing water, which varies from 0ºC to 125ºC. Another sensor (*LevelSense*) tells us the level of the water in the tank, which varies from 0 (= empty) to 10 (= full). We assume that

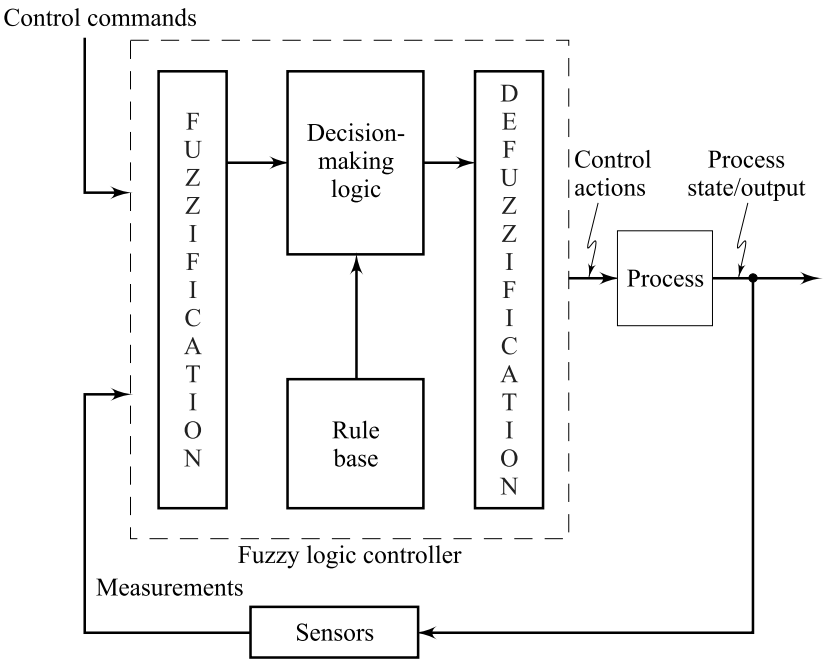


Fig. 12.18 A simple fuzzy logic control system block diagram

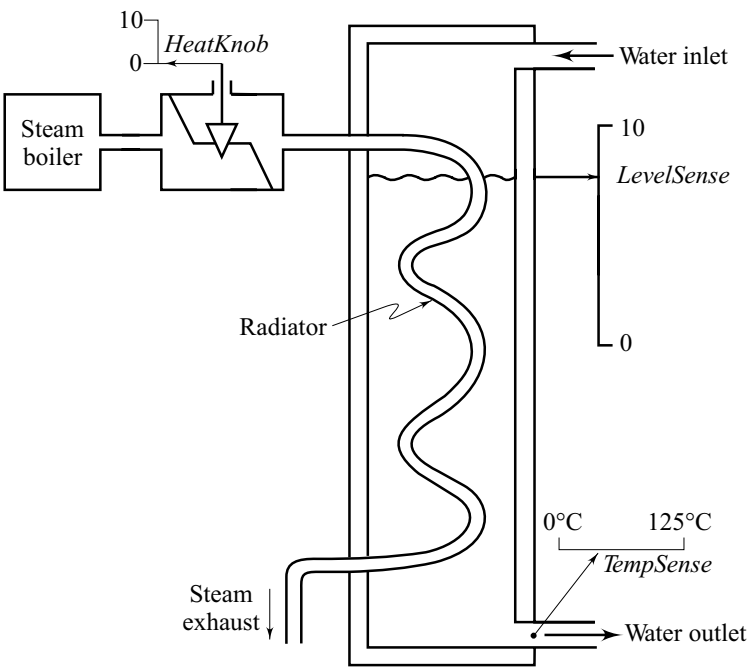


Fig. 12.19 Water-heating system

there is an automatic flow control that determines how much cold water flows into the tank from the main water supply; whenever the level of the tank gets below 4, the flow control turns ON, and turns OFF when the level of the water gets above 9.5.

Figure 12.20 shows a FLC diagram for the water-heating system.

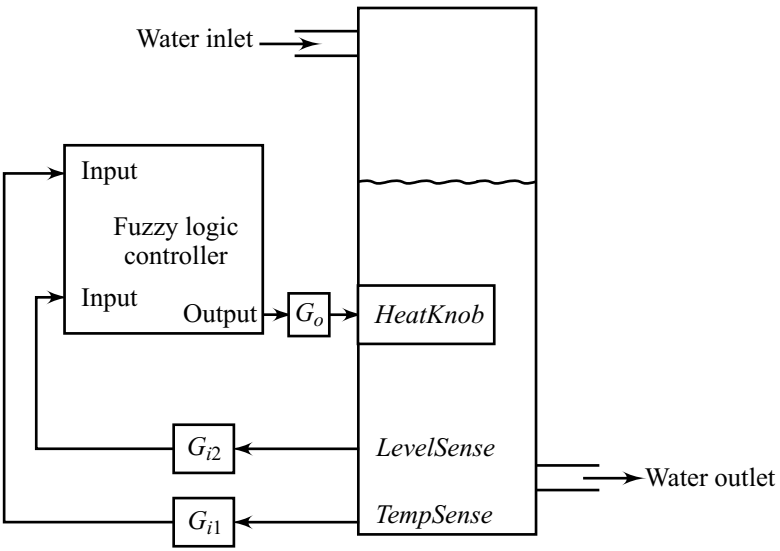


Fig. 12.20 Fuzzy control of a water heater

The design objective can be stated as:

Keep the water temperature as close to 80ºC as possible, in spite of changes in the hot water flowing out of the tank, and the cold water flowing into the tank.

Step 1: Define Inputs and Outputs for the FLC Three fuzzy variables characterize the behavior of the water-heating system.

Input Variables: *TempSense* and *LevelSense*

Output Variable: *HeatKnob*

For x = outlet water temperature (linguistic variable *TempSense*), the universe of discourse is

$$X = [0ºC, 125ºC]$$

For y = level of water in the tank (linguistic variable *LevelSense*), the universe of discourse is

$$Y = [0, 10]$$

For z = HeatKnob setting (linguistic variable *HeatKnob*), the universe of discourse is

$$Z = [0, 10]$$

Step 2: Define Frames for Fuzzy Variables The frame of *TempSense is*

$$\left\langle \underset{\sim}{TempSense}, \mathcal{L}\underset{\sim}{TempSense}, X, \mu_{\mathcal{L}\underset{\sim}{TempSense}} \right\rangle$$

where $\mathcal{L}TempSense$ is the set of linguistic values that *TempSense* can take. We may use the following fuzzy subsets to describe the linguistic values:

XSmall (*XS*); *Small* (*S*); *Medium* (*M*); *Large* (*L*); *XLarge* (*XL*)

i.e.,

$$\mathcal{L}TempSense = \{XSmall, Small, Medium, Large, XLarge\}$$

The frame of *LevelSense* is

$$\left\langle LevelSense, \mathcal{L}LevelSense, Y, \mu_{\mathcal{L}LevelSense} \right\rangle$$

$$\mathcal{L}\,LevelSense = \{XSmall, Small, Medium, Large, XLarge\}$$

In our system, we have just one output which is the *HeatKnob*. We take the following frame for this linguistic variable:

$$\left\langle HeatKnob, \mathcal{L}HeatKnob, Z, \mu_{\mathcal{L}HeatKnob} \right\rangle$$

$$\mathcal{L}HeatKnob = \{VeryLittle, ALittle, AGoodAmount, ALot, AWholeLot\}$$

Step 3: Assign Membership Values to Fuzzy Variables Since the membership function essentially embodies all fuzziness for a particular fuzzy set, its description is the essence of a fuzzy property or operation. Because of the importance of the 'shape' of the membership function, a great deal of attention has been focussed on development of these functions. Many ways to develop membership functions, i.e., to assign membership values to fuzzy variables, have been reported in the literature—methods based on Inference, Neural Networks, Genetic Algorithms, Inductive Reasoning, etc. The assignment process can be intuitive, or it can be based on some algorithmic or logical operations. We shall rely on intuition in our application examples.

Table 12.1 Fuzzy variable ranges for *TempSense*

Crisp Input Range	*Fuzzy Variable*
0–20	*XSmall*
10–35	*Small*
30–75	*Medium*
60–95	*Large*
85–125	*XLarge*

The input variables *TempSense* and *LevelSense*, as well as the output variable *HeatKnob*, are restricted to positive values. In Table 12.1 and Fig. 12.21, we show a possible assignment for ranges and triangular membership functions for *TempSense*. Similarly, we assign ranges and fuzzy membership functions for *LevelSense* in Table 12.2 and Fig. 12.22; and *HeatKnob* in Table 12.3 and Fig. 12.23. The optimization of these assignments is often done through trial and error for achieving optimum performance of FLC.

The following guidelines were kept in mind while determining range of fuzzy variables as related to the crisp inputs.

(i) Symmetrically distribute the fuzzified values across the universe of discourse.

Table 12.2 Fuzzy variable ranges for *LevelSense*

Crisp Input Range	*Fuzzy Variable*
0–2	*XSmall*
1.5–4	*Small*
3–7	*Medium*
6–8.5	*Large*
7.5–10	*XLarge*

Table 12.3 Fuzzy variable ranges for *HeatKnob*

Crisp Input Range	*Fuzzy Variable*
0–2	*VeryLittle*
1.5–4	*ALittle*
3–7	*AGoodAmount*
6–8.5	*ALot*
7.5–10	*AWholeLot*

(ii) Use an odd number of fuzzy sets for each variable so that some set is assured to be in the middle. The use of 5 to 7 sets is fairly typical.

(iii) Overlap adjacent sets (by 15% to 25%, typically).

Step 4: Create a Rule Base Now that we have the inputs and the outputs in terms of fuzzy variables, we need only specify what actions to take, under what conditions; i.e., we need to construct a set of rules that describe the operation of the FLC. These rules usually take the form of IF-THEN rules, and can be obtained from a human expert (heuristics).

The rule-base matrix for our example is given in Table 12.4. Our heuristic guidelines, in determining this matrix, are the following:

(i) When the temperature is low, the *HeatKnob* should be set higher than when the temperature is high.

(ii) When the volume of water is low, the *HeatKnob* does not need to be as high as when the volume of water is high.

Table 12.4 Decision table

TempSense → *LevelSense* ↓	*XS*	*S*	*M*	*L*	*XL*
XS	*AGoodAmount*	*ALittle*	*VeryLittle*		
S	*ALot*	*AGoodAmount*	*VeryLittle*	*VeryLittle*	
M	*AWholeLot*	*ALot*	*AGoodAmount*	*VeryLittle*	
L	*AWholeLot*	*ALot*	*ALot*	*ALittle*	
XL	*AWholeLot*	*ALot*	*ALot*	*AGoodAmount*	

In FLCs we do not need to specify all the cells in the matrix. No entry signifies that no action is taken. For example, in the column for *TempSense* = *XLarge*, no action is required since the temperature is already at or above the target temperature.

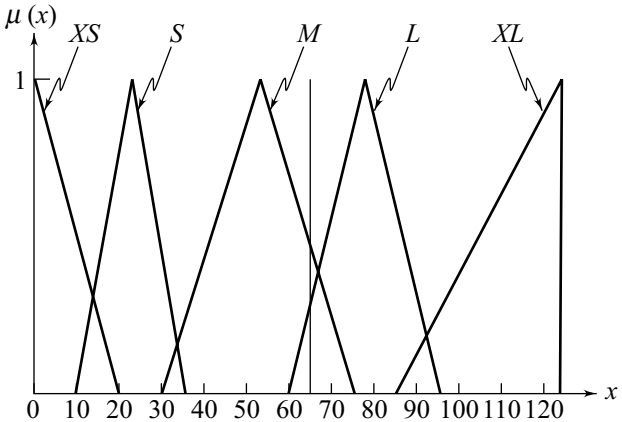


Fig. 12.21 Fuzzy membership functions for *TempSense*

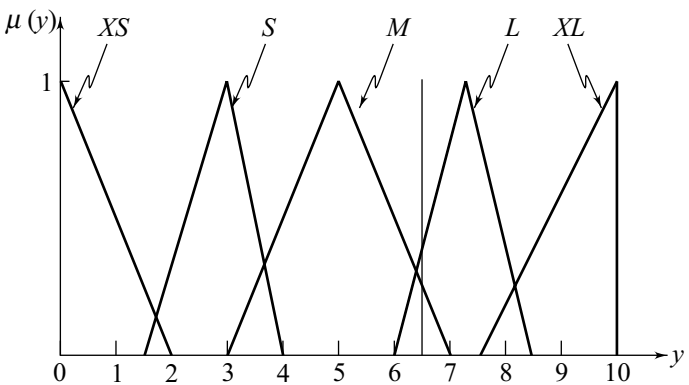


Fig. 12.22 Fuzzy membership functions for *LevelSense*

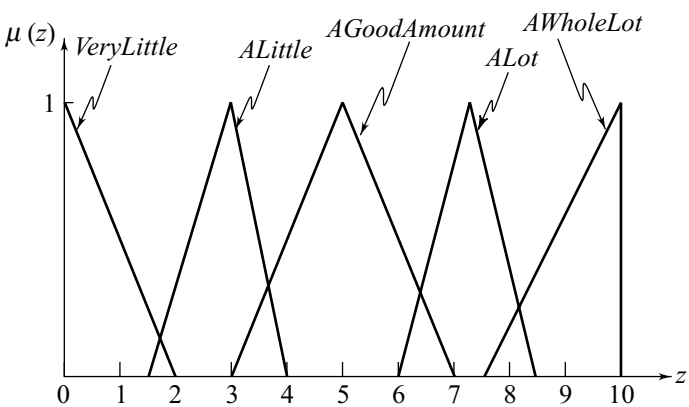


Fig. 12.23 Fuzzy membership functions for the output *HeatKnob*

Let us examine a couple of typical entries in the table: For *LevelSense* = *Small*, and *TempSense* = *XSmall*, the output is *HeatKnob* = *ALot*. Now for the same temperature, as the water level rises, the setting on HeatKnob should also rise—to compensate for the added volume of water. We can see that for *LevelSense* = *Large* and *TempSense* = *XSmall*, the output *HeatKnob* = *AWholeLot*.

We can translate the table entries into IF-THEN rules. We give here a couple of rules.

- IF *TempSense* is *Small* **and** *LevelSense* is *Small* THEN set *HeatKnob* to *ALot*.
- IF *TempSense* is *XSmall* **and** *LevelSense* is *Large* THEN set *HeatKnob* to *AWholeLot*.

Step 5: Choose Scaling Gains for the Variables Using standard ideas from control engineering, we have introduced gains G_{i1} and G_{i2} with the input variables, as shown in Fig. 12.20, and at the same time we also put a gain G_o between FLC and the plant. Change in the *scaling gains*, at the input and output of FLC, can have a significant impact on the performance of the resulting control system, and hence they are often convenient parameters for *tuning*.

First, let us consider the effect of input scaling gain G_{i1}. Note that we can actually achieve the same effect as scaling *via* G_{i1}, by simply changing the labeling of the temperature axis for the membership function of the input variable *TempSense*. The case when $G_{i1} = 1$ corresponds to our original choice of the membership functions in Fig. 12.21. The choice of $G_{i1} = 0.5$ as the scaling gain for the FLC with these membership functions, is equivalent to having the membership functions shown in Fig. 12.24 with $G_{i1} = 1$. Thus, the choice of a scaling gain G_{i1} results in scaling the horizontal axis of the membership functions by $1/G_{i1}$ (multiplication of each number on the horizontal axis of Fig. 12.21 by 1/0.5 produces Fig. 12.24; membership functions are uniformly 'spread out' by a factor of 1/0.5). Similar statements can be made about G_{i2} (Fig. 12.25).

Figure 12.23 shows our choice of output membership functions with $G_o = 1$. There is a proportional effect between the scaling gain G_o and the output membership functions as shown in Fig. 12.26 for $G_o = 2$.

If, for the process under consideration, the effective universes of discourse for all inputs and output are common, say, [0, 1], then we may say that the FLC is *normalized*. Clearly, scaling gains can be used to normalize the given FLC. Denormalization of the output of such a FLC will yield the required control action.

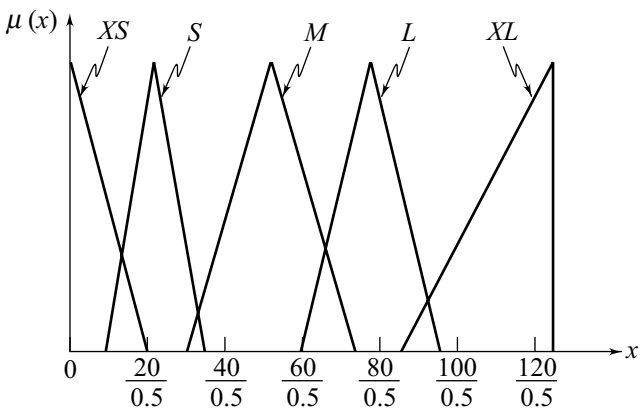


Fig. 12.24 Scaled membership functions for *TempSense*

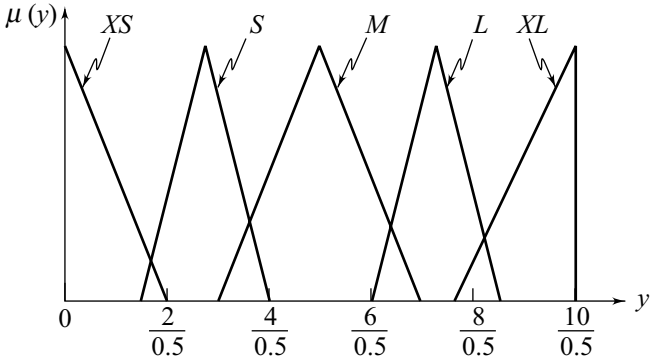


Fig. 12.25 Scaled membership functions for *LevelSense*

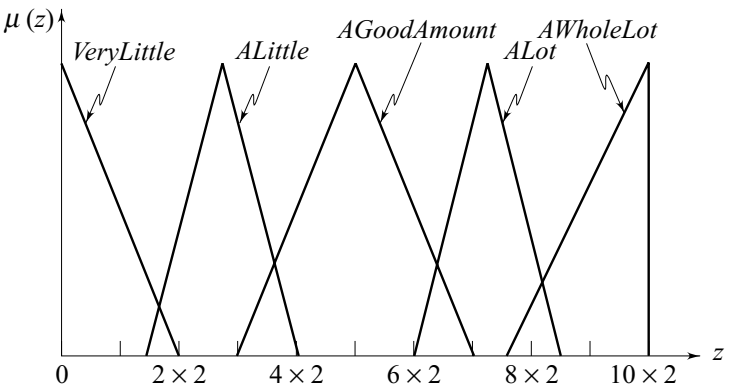


Fig. 12.26 The effect of scaling gain G_o on the spacing of the output membership functions

It is important to realize that the scaling gains are not the only parameters that can be tuned to improve the performance of the fuzzy control system. Membership function shapes, positioning, and number and type of rules are often the other parameters to tune.

We set $G_{i1} = G_{i2} = G_o = 1$ for our design problem.

Step 6: Fuzzify Inputs to the FLC Fuzzy sets are used to quantify information in the rule base, and the inference mechanism operates on fuzzy sets and produces fuzzy sets. Inputs to the FLC are the measured output variables of the controlled process, which are crisp variables. And input to the controlled process (control action) is required to be crisp. Therefore, we must specify how the fuzzy system will convert the numeric (crisp) inputs to the FLC into fuzzy sets (a process called 'fuzzification'). Also we must specify how the fuzzy system will convert the fuzzy sets produced by inference mechanism into numeric (crisp) FLC output (a process called 'defuzzification'), which is the input to the controlled process.

Fuzzification can be defined as a mapping from an observed input space to fuzzy sets in a specified universe of discourse. A natural and simple fuzzification approach is to convert a crisp measurement into a fuzzy singleton within the specified universe of discourse. This approach is based on the assumption that the observed data is crisp, and not corrupted by random noise.

To understand fuzzification, we consider an example. Assume that at a particular point in time, *LevelSense* = 6.5 and *TempSense* = 65ºC. These are the crisp inputs directly from the sensors. Figures 12.21 and 12.22 show the membership functions for the input variables and indicate with vertical lines the measured values of *LevelSense* and *TempSense*. These vertical lines are, in fact, graphical representation of the two fuzzy singletons obtained by the fuzzification process.

Step 7: Determine which Rules Fire We see that with singleton fuzzification, combining the fuzzy sets that were created by the fuzzification process to represent the inputs with the premise membership functions for the rules, is particularly simple. It simply reduces to computing the membership values of the input fuzzy sets for the given inputs.

From Fig. 12.21 we find that for input *TempSense* = 65ºC, $\mu_{\underset{\sim}{M}}(65) = 0.45$, $\mu_{\underset{\sim}{L}}(65) = 0.28$, and all other membership functions are off (i.e., their values are zero). Therefore, the proposition '*TempSense* is *Medium*' is satisfied to a degree of 0.45, the proposition '*TempSense* is *Large*' is satisfied to a degree of 0.28; all other atomic propositions associated with *TempSense* are not satisfied.

From Fig. 12.22 we find that for input *LevelSense* = 6.5, $\mu_{\underset{\sim}{M}}(6.5) = 0.25$, $\mu_{\underset{\sim}{L}}(6.5) = 0.38$; all other membership functions are off.

We next form membership values of premises of all the rules. From the induced decision table (Table 12.5), we observe that the rules that have the premise terms:

(i) *TempSense* is *Medium* **and** *LevelSense* is *Medium*
(ii) *TempSense* is *Large* **and** *LevelSense* is *Medium*
(iii) *TempSense* is *Medium* **and** *LevelSense* is *Large*
(iv) *TempSense* is *Large* **and** *LevelSense* is *Large*

have $\mu_{premise} > 0$. For all other rules, $\mu_{premise} = 0$.

Determining applicability of each rule is called 'firing'. We say that, a rule fires at time t if its premise membership value at time t is greater than zero. The inference mechanism seeks to determine which rules fire, to find out which rules are relevant to the current situation. The inference mechanism combines the recommendations of all the rules, to come up with a single conclusion.

Table 12.5 Induced Decision table

TempSense→ *LevelSense* ↓	$\mu_{\underset{\sim}{XS}} = 0$	$\mu_{\underset{\sim}{S}} = 0$	$\mu_{\underset{\sim}{M}} = 0.45$	$\mu_{\underset{\sim}{L}} = 0.28$	$\mu_{\underset{\sim}{XL}} = 0$
$\mu_{\underset{\sim}{XS}} = 0$	0	0	0	0	0
$\mu_{\underset{\sim}{S}} = 0$	0	0	0	0	0
$\mu_{\underset{\sim}{M}} = 0.25$	0	0	*AGoodAmount*	*VeryLittle*	0
$\mu_{\underset{\sim}{L}} = 0.38$	0	0	*ALot*	*ALittle*	0
$\mu_{\underset{\sim}{XL}} = 0$	0	0	0	0	0

For crisp input *TempSense* = 65ºC, and *LevelSense* = 6.5, four rules fire. $\mu_{premise}$ for the four rules (refer to Table 12.5), which amounts to *firing strength* in each case, can be calculated as follows:

(i) $\mu_{TempSense \times LevelSense} = \min(0.45,0.25) = 0.25$
(ii) $\mu_{TempSense \times LevelSense} = \min(0.28,0.25) = 0.25$
(iii) $\mu_{TempSense \times LevelSense} = \min(0.45,0.38) = 0.38$
(iv) $\mu_{TempSense \times LevelSense} = \min(0.28,0.38) = 0.28$

Step 8: Infer the Output Recommended by Each Rule From the induced decision table (Table 12.5), we observe that only four cells contain nonzero terms. Let us call these cells *active*. The active cells correspond to the following rules:

(i) *TempSense* is *Medium* **and** *LevelSense* is *Medium* : p_1
Set *HeatKnob* to *AGoodAmount* : q_1
IF p_1 THEN q_1
$\mu_{premise(1)} = 0.25$
$\mu_{inference(1)}$ is obtained by 'chopping off' the top of $\mu_{AGoodAmount}$ function of the output variable *HeatKnob*, as shown in Fig. 12.27a.

(ii) *TempSense* is *Large* **and** *LevelSense* is *Medium* : p_2
Set *HeatKnob* to *VeryLittle* : q_2
IF p_2 THEN q_2
$\mu_{premise(2)} = 0.25$
$\mu_{inference(2)}$ is shown in Fig. 12.27b.

(iii) *TempSense* is *Medium* **and** *LevelSense* is *Large* : p_3
Set *HeatKnob* to *ALot* : q_3
IF p_3 THEN q_3
$\mu_{premise(3)} = 0.38$
$\mu_{inference(3)}$ is shown in Fig. 12.27c.

(iv) *TempSense* is *Large* **and** *LevelSense* is *Large* : p_4
Set *HeatKnob* to *ALittle* : q_4
IF p_4 THEN q_4
$\mu_{premise(4)} = 0.28$
$\mu_{inference(4)}$ is shown in Fig. 12.27d.

The reader should note that for different crisp measurements *TempSense* and *LevelSense*, there will be different values of $\mu_{premise}$ and, hence, different $\mu_{inference}$ functions will be obtained.

Step 9: Aggregate the Fuzzy Outputs Recommended by Each Rule In the previous step, we noticed that the input to the inference process is the set of rules that fire; its output is the set of fuzzy sets that represent the inference reached by all the rules that fire. We now combine all the recommendations of all the rules to determine the control action. This is done by aggregating (union operation) the inferred fuzzy sets. Aggregated fuzzy set, obtained by drawing all the inferred fuzzy sets on one axis, is shown in Fig. 12.28. This fuzzy set represents the desired control action.

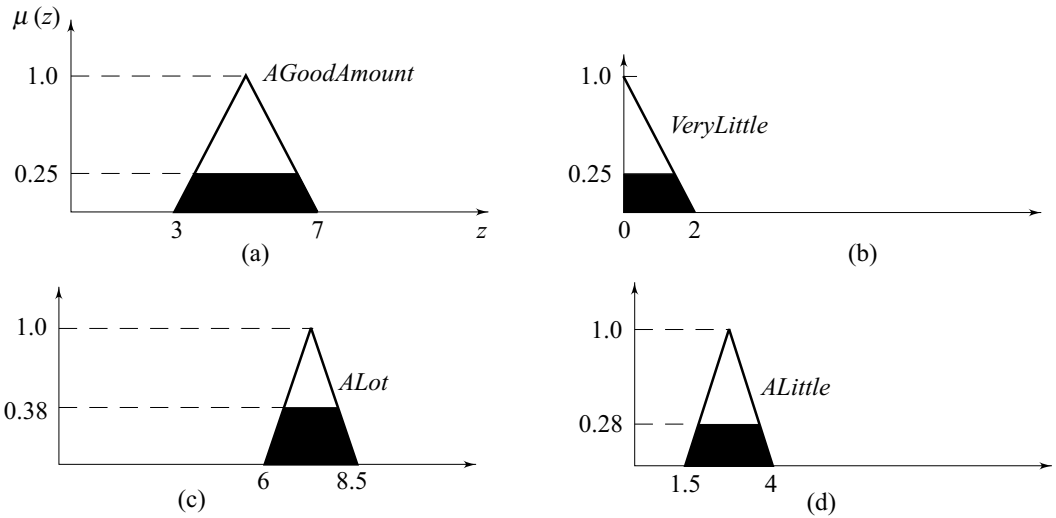


Fig. 12.27 Inference for each rule

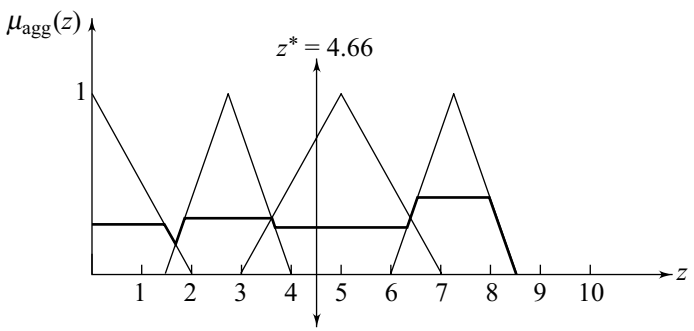


Fig. 12.28 Aggregated fuzzy set

Step 10: Defuzzify the Aggregated Fuzzy Set to form Crisp Output from the FLC Defuzzification is a mapping from a space of fuzzy control actions defined by fuzzy sets on an output universe of discourse, into nonfuzzy (crisp) control actions. This process is necessary because crisp control action is required to actuate the control.

There are many approaches to defuzzification. We will consider here the 'Center of Area' (COA) method, which is known to yield superior results.

We may discretize the universe Z into q equal (or almost equal) subintervals by the points $z_1, z_2, ..., z_{q-1}$. The crisp value z^*, according to this method is

$$z^* = \frac{\sum_{k=1}^{q-1} z_k \mu_{\text{agg}}(z_k)}{\sum_{k=1}^{q-1} \mu_{\text{agg}}(z_k)} \tag{12.33}$$

From Fig. 12.28, we obtain

$$\Sigma z_k \mu_{\text{agg}} = 1 \times 0.25 + 1.5 \times 0.25 + 2 \times 0.28 + 3 \times 0.28 + 4 \times 0.25 + 5 \times 0.25 + 6 \times 0.25 + 7 \times 0.38 + 8 \times 0.38 = 11.475$$

$$\Sigma \mu_{\text{agg}} = 0.25 + 0.25 + 0.28 + 0.28 + 0.25 + 0.25 + 0.25 + 0.38 + 0.38 = 2.57$$

Therefore,

$$z^* = \frac{11.475}{2.57} = 4.46$$

The physical interpretation of z^* is that, if the area is cut of a thin piece of metal or wood, the center of the area will be the center of gravity.

In fact, there is hardly any need of discretization of the universe for situations like the one shown in Fig. 12.28; we can split up geometry into pieces and place a straight edge (centroid) through the figure to have it perfectly balanced with equal area of the figure on either side. Analytical expression for z^* is

$$z^* = \frac{\int_z \mu_{\text{agg}}(z) z dz}{\int_z \mu_{\text{agg}}(z) dz} \tag{12.34}$$

This completes the design for the simple example we chose.

12.6 DATA BASED FUZZY MODELING (SUGENO ARCHITECTURE)

The Mamdani architecture is widely used for capturing expert knowledge. It allows us to describe the expertise in more intuitive, more human-like manner. On the other hand, the Sugeno architecture is by far the most popular candidate for data-based fuzzy modeling.

Basically, a fuzzy model is a 'Fuzzy Inference System (FIS)' composed of four principal components: a fuzzification interface, a knowledge base, a decision-making logic, and a defuzzification interface. Figure 12.29 shows the basic configuration of FIS for data-based modeling.

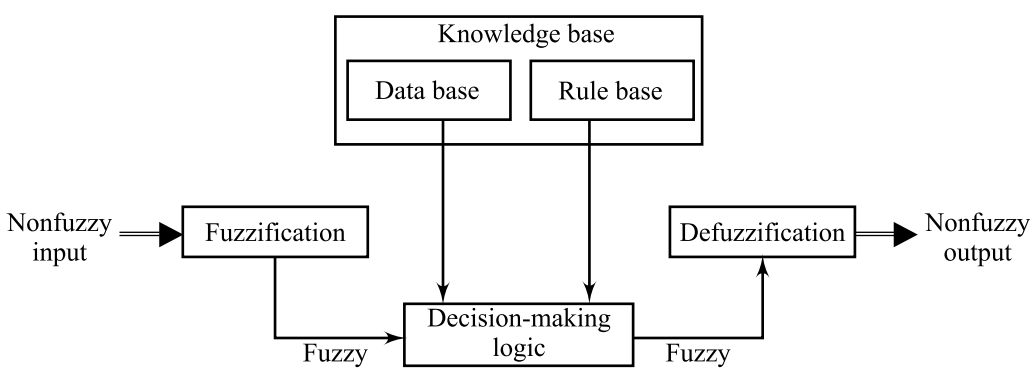


Fig. 12.29 Basic configuration of a fuzzy inference system

We consider here, a single-output FIS in the n-dimensional input space. Let us assume that the following P input-output pairs are given as training data for constructing FIS model:

$$\left\{\mathbf{x}^{(p)}, y^{(p)} \,|\, p = 1, 2, \ldots, P\right\} \tag{12.35}$$

where $\mathbf{x}^{(p)} = \left[x_1^{(p)} x_2^{(p)} \cdots x_n^{(p)}\right]^T$ is the input vector of pth input-output pair and $y^{(p)}$ is the corresponding output.

The *fuzzification* interface performs a mapping that converts crisp values of input variables into fuzzy singletons. A singleton is a fuzzy set with a membership function that is unity at a single particular point on the universe of discourse (the numerical-data value), and zero everywhere else. Basically, a fuzzy singleton is a precise value and hence no fuzziness is introduced by fuzzification in this case. This strategy, however, has been widely used in fuzzy modeling applications because it is easily implemented.

There are two factors that determine a *database* (i) a fuzzy partition of an input space, and (ii) membership functions of antecedent fuzzy sets. Assume that the domain interval of the ith input variable x_i, is equally divided into K_i fuzzy sets labeled $\underset{\sim}{A}_{i1}$, $\underset{\sim}{A}_{i2}$, ..., $\underset{\sim}{A}_{iK_i}$, for $i = 1, 2, ..., n$. Then the n-dimensional input space is divided into $K_1 \times K_2 \times \cdots \times K_n$ fuzzy partition spaces:

$$(\underset{\sim}{A}_{1j_1}, \underset{\sim}{A}_{2j_2}, \ldots, \underset{\sim}{A}_{nj_n}); j_1 = 1, 2, ..., K_1; ...; j_n = 1, ..., K_n \tag{12.36}$$

Though any type of membership functions (e.g., triangle-shaped, trapezoid-shaped, bell-shaped, etc.) can be used for fuzzy sets, we employ the symmetric triangle-shaped fuzzy sets, $\underset{\sim}{A}_{ij_i}$, with the following membership functions:

$$\mu_{\underset{\sim}{A}_{ij_i}}(x_i) \triangleq \mu_{ij_i}(x_i) = 1 - \frac{2\left|x_i - c_{(i,j_i)}\right|}{w_{(i,j_i)}}; j_i = 1, 2, \ldots, K_i \tag{12.37}$$

$c_{(i,j_i)}$ is the center of the membership function, where the membership grade is equal to 1, and $w_{(i,j_i)}$ denotes the width of the membership function (Fig. 12.30).

By means of the input-output data, the range $\left[x_i^{\min}, x_i^{\max}\right]$ of the ith input variable is determined, where

$$x_i^{\min} = \min_{p=\{1,\ldots,P\}} x_i^{(p)},\ x_i^{\max} = \max_{p=\{1,\ldots,P\}} x_i^{(p)} \tag{12.38a}$$

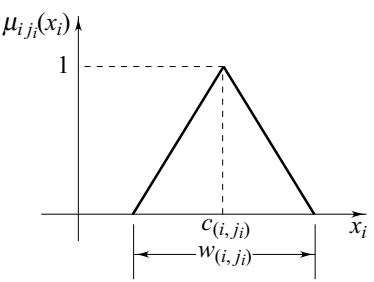


Fig. 12.30 Parameters of a membership function

The center position of each membership function with respect to the ith variable is determined by

$$c_{(i,j_i)} = x_i^{\min} + (j_i - 1)\left[(x_i^{\max} - x_i^{\min})/(K_i - 1)\right]; c_{(i,1)} = x_i^{\min}; c_{(i,K_i)} = x_i^{\max} \tag{12.38b}$$

To achieve sufficient overlap from one linguistic label to another, we take

$$w_{(i,j_i)} = 2(c_{(i,j_i+1)} - c_{(i,j_i)}) \tag{12.38c}$$

Figure 12.31 shows an example where the domain interval of x_1 is divided into $K_1 = 5$ fuzzy sets.

The rule base consists of a set of fuzzy IF-THEN rules in the form 'IF a set of conditions are satisfied THEN a set of consequences can be inferred'. Different types of consequent parts have been used in

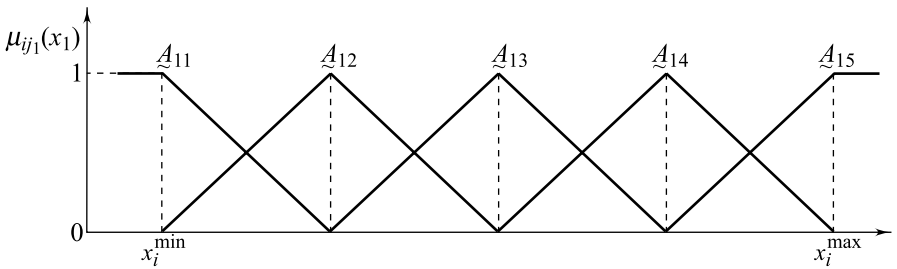


Fig. 12.31 Fuzzy partition of an input space and membership functions of fuzzy sets

fuzzy rules; the two commonly used fuzzy models are based on Mamdani's approach and Sugeno's approach. We restrict our discussion to Sugeno architecture: the domain interval of y is represented by R linear functions, giving rise to R fuzzy rules. All the rules corresponding to the possible combinations of the inputs are implemented. The total number of rules R for an n-input system is : $K_1 \times K_2 \times \cdots \times K_n$.

The format of fuzzy rules is,

Rule r: IF x_1 is $\underset{\sim}{A}_{1j_1}$ and $\cdots$ and x_n is $\underset{\sim}{A}_{nj_n}$ THEN

$$\hat{y}^{(r)} = a_0^{(r)} + a_1^{(r)} x_1 + \cdots + a_n^{(r)} x_n; r = 1, 2, \ldots, R \tag{12.39}$$

The consequent part is a linear function of the input variables x_i; a_0, a_1, ..., a_n are the $(n + 1)$ parameters that determine the real consequent value. The aim of the linear function is to describe the local linear behavior of the system. Each rule r gives rise to a local linear model. The selected R rules are required to approximate the function that theoretically underlines the system behavior most consistently, with the given sample of input-output data (12.35) (When $\hat{y}$ is a constant in (12.39), we get a Sugeno model in which the consequent of a rule is specified by a singleton).

The decision-making logic employs fuzzy IF-THEN rules from the rule base to infer the output by a fuzzy reasoning method. The contribution of each local linear model (i.e., each rule) in the estimated output of the FIS is dictated by the *firing strength* of the rule. We use *product* strategy to assign firing strength $\mu^{(r)}$ to each rule $r = 1, 2, \ldots, R$.

Given an input vector, $\mathbf{x}^{(p)} = \left[x_1^{(p)}, x_2^{(p)}, ..., x_n^p\right]^T$, the degree of compatibility of $\mathbf{x}^{(p)}$ to the rth fuzzy IF-THEN rule is the firing strength $\mu^{(r)}$ of the rule, and is given by (note that we have used **product** t-norm operator on the premise part of the rule)

$$\begin{aligned} \mu^{(r)}(\mathbf{x}^{(p)}) &= \mu_{1j_1}(x_1^{(p)}) \times \mu_{2j_2}(x_2^{(p)}) \times \cdots \times \mu_{nj_n}(x_n^{(p)}) \\ &= \prod_{(i,j_i) \in I_r} \mu_{ij_i}(x_i^{(p)}) \end{aligned} \tag{12.40}$$

where I_r is the set of all $\underset{\sim}{A}_{ij_i}$ associated with the premise part of rule r.

The main idea of the Sugeno architecture is that in each input fuzzy region $\underset{\sim}{A}_{1j_1} \times \underset{\sim}{A}_{2j_2} \times \cdots \times \underset{\sim}{A}_{nj_n}$ of the input domain, a local linear system is formed. The membership function $\mu^{(r)}(\mathbf{x}^{(p)})$ of each region is a map indicating the degree of the output of the associated linear system to the region. A simple *defuzzification* procedure is to take the output of the system as the 'fuzzy' combination of the outputs of local systems in all regions:

$$\hat{y} = \frac{\sum_{r=1}^{R}(a_0^{(r)} + a_1^{(r)}x_1 + \cdots + a_n^{(r)}x_n)\mu^{(r)}}{\sum_{r=1}^{R}\mu^{(r)}} \tag{12.41a}$$

$$= \sum_{r=1}^{R}(a_0^{(r)} + a_1^{(r)}x_1 + \cdots + a_n^{(r)}x_n)\bar{\mu}^{(r)} \tag{12.41b}$$

where

$$\bar{\mu}^{(r)} = \frac{\mu^{(r)}}{\sum_{r=1}^{R}\mu^{(r)}} \tag{12.41c}$$

is the *normalized firing strength* of rule *r*; a ratio of firing strength of rule *r* to the sum of the firing strengths of all the rules.

Note that the output of the fuzzy model can be determined only if the parameters in rule consequents are known. However, it is often difficult or even impossible to specify a rule consequent in a polynomial form. Fortunately, it is not necessary to have any prior knowledge of rule consequent parameters for the Sugeno fuzzy modeling approach to deal with a problem. These parameters can be determined using *least squares estimation method* as follows.

Given the values of the membership parameters and a training set of *P* input-output patterns $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1,2,...,P\}$, we can form *P* linear equations in terms of the consequent parameters.

$$y^{(p)} = \bar{\mu}^{(1)}(\mathbf{x}^{(p)})\left[a_0^{(1)} + a_1^{(1)}x_1^{(p)} + \cdots + a_n^{(1)}x_n^{(p)}\right] + \bar{\mu}^{(2)}(\mathbf{x}^{(p)})\left[a_0^{(2)} + a_1^{(2)}x_1^{(p)} + \cdots + a_n^{(2)}x_n^{(p)}\right] + \cdots$$
$$+ \bar{\mu}^{(R)}(\mathbf{x}^{(p)})\left[a_0^{(R)} + a_1^{(R)}x_1^{(p)} + \cdots + a_n^{(R)}x_n^{(p)}\right];\ p = 1,2,\cdots P \tag{12.42}$$

where $\bar{\mu}^{(r)}(\mathbf{x}^{(p)})$ is the normalized firing strength of rule *r*, fired by the input pattern $\mathbf{x}^{(p)}$.

In terms of vectors

$$\bar{\mathbf{x}}^{(p)} = \begin{bmatrix}1 & x_1^{(p)} & x_2^{(p)}\cdots x_n^{(p)}\end{bmatrix}^T$$
$$\boldsymbol{\theta}^{(r)} = \begin{bmatrix}a_0^{(r)} & a_1^{(r)} & \cdots a_n^{(r)}\end{bmatrix} \tag{12.43}$$
$$\boldsymbol{\Theta} = \begin{bmatrix}a_0^{(1)} & a_1^{(1)}\cdots a_n^{(1)} & a_0^{(2)}\cdots a_n^{(2)}\cdots a_0^{(R)}\cdots a_n^{(R)}\end{bmatrix}$$

we can write the *P* linear equations as follows:

$$\begin{aligned}
y^{(1)} &= \bar{\mu}^{(1)}(\mathbf{x}^{(1)})\left[\boldsymbol{\theta}^{(1)}\,\bar{\mathbf{x}}^{(1)}\right] + \bar{\mu}^{(2)}(\mathbf{x}^{(1)})\left[\boldsymbol{\theta}^{(2)}\,\bar{\mathbf{x}}^{(1)}\right] + \cdots + \bar{\mu}^{(R)}(\mathbf{x}^{(1)})\left[\boldsymbol{\theta}^{(R)}\,\bar{\mathbf{x}}^{(1)}\right] \\
y^{(2)} &= \bar{\mu}^{(1)}(\mathbf{x}^{(2)})\left[\boldsymbol{\theta}^{(1)}\,\bar{\mathbf{x}}^{(2)}\right] + \bar{\mu}^{(2)}(\mathbf{x}^{(2)})\left[\boldsymbol{\theta}^{(2)}\,\bar{\mathbf{x}}^{(2)}\right] + \cdots + \bar{\mu}^{(R)}(\mathbf{x}^{(2)})\left[\boldsymbol{\theta}^{(R)}\,\bar{\mathbf{x}}^{(2)}\right] \\
&\vdots \\
y^{(P)} &= \bar{\mu}^{(1)}(\mathbf{x}^{(P)})\left[\boldsymbol{\theta}^{(1)}\,\bar{\mathbf{x}}^{(P)}\right] + \bar{\mu}^{(2)}(\mathbf{x}^{(P)})\left[\boldsymbol{\theta}^{(2)}\,\bar{\mathbf{x}}^{(P)}\right] + \cdots + \bar{\mu}^{(R)}(\mathbf{x}^{(P)})\left[\boldsymbol{\theta}^{(R)}\,\bar{\mathbf{x}}^{(P)}\right]
\end{aligned} \tag{12.44}$$

These P equations can be rearranged into a single vector-matrix equation:

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(P)} \end{bmatrix} = \begin{bmatrix} \left[\bar{\mathbf{x}}^{(1)}\right]^T \bar{\mu}^{(1)}(\mathbf{x}^{(1)}) & \left[\bar{\mathbf{x}}^{(1)}\right]^T \bar{\mu}^{(2)}(\mathbf{x}^{(1)}) \cdots & \left[\bar{\mathbf{x}}^{(1)}\right]^T \bar{\mu}^{(R)}(\mathbf{x}^{(1)}) \\ \left[\bar{\mathbf{x}}^{(2)}\right]^T \bar{\mu}^{(1)}(\mathbf{x}^{(2)}) & \left[\bar{\mathbf{x}}^{(2)}\right]^T \bar{\mu}^{(2)}(\mathbf{x}^{(2)}) \cdots & \left[\bar{\mathbf{x}}^{(2)}\right]^T \bar{\mu}^{(R)}(\mathbf{x}^{(2)}) \\ \vdots & & \\ \left[\bar{\mathbf{x}}^{(P)}\right]^T \bar{\mu}^{(1)}(\mathbf{x}^{(P)}) & \left[\bar{\mathbf{x}}^{(P)}\right]^T \bar{\mu}^{(2)}(\mathbf{x}^{(P)}) \cdots & \left[\bar{\mathbf{x}}^{(P)}\right]^T \bar{\mu}^{(R)}(\mathbf{x}^{(P)}) \end{bmatrix} \begin{bmatrix} [\boldsymbol{\theta}^{(1)}]^T \\ [\boldsymbol{\theta}^{(2)}]^T \\ \vdots \\ [\boldsymbol{\theta}^{(R)}]^T \end{bmatrix} \tag{12.45a}$$

or

$$\mathbf{y} = \mathbf{X}^T \boldsymbol{\Theta}^T \tag{12.45b}$$

In the Sugeno fuzzy model given above, we have used most intuitive approach of implementing all possible combinations of the given fuzzy sets as rules. In fact, if data is not uniformly distributed, some rules may never be fired. This and other drawbacks are handled by many variants of the basic ANFIS model, described in the next section.

12.7 SYSTEM IDENTIFICATION AND CONTROL WITH NEURO-FUZZY SYSTEMS

Fuzzy logic and neural networks are natural complementary tools in building intelligent systems. While neural networks are computational structures that perform well when dealing with raw data, fuzzy logic deals with reasoning, using linguistic information acquired from domain experts. However, fuzzy systems lack the ability to learn and cannot adjust themselves to a new environment. On the other hand, although neural networks can learn, they are opaque to the user. The merger of a neural network with a fuzzy system into one integrated system, therefore, offers a promising approach to building intelligent systems. Integrated systems can combine the parallel computation and learning abilities of neural networks, with the human-like knowledge representation and explanation abilities of fuzzy systems. As a result, neural networks become more transparent, while fuzzy systems become capable of learning.

The structure of a *neuro-fuzzy system* is similar to a multilayer neural network. In general, a neuro-fuzzy system has input terminals, output layer, and hidden layers that represent membership functions and fuzzy rules.

Roger Jang [142] proposed an integrated system that is functionally equivalent to a Sugeno fuzzy inference model. He called it an Adaptive Neuro-Fuzzy Inference System or ANFIS. Similar network structures have also been proposed for Mamdani fuzzy inference model [137]. However, the Sugeno model is by for the most popular candidate for data-based fuzzy modeling. Our brief presentation of the subject is, therefore, focused on ANFIS based on Sugeno fuzzy model.

12.7.1 ANFIS Architecture

Figure 12.32 shows the ANFIS architecture. For simplicity, we assume that the ANFIS has two inputs, x_1 and x_2, and one output y. Each input is represented by two fuzzy sets, and the output by a first-order polynomial. The ANFIS implements the following four rules:

$$
\begin{aligned}
&\textit{Rule 1}: \quad \text{IF } x_1 \text{ is } \underset{\sim}{A}_{11} \text{ and } x_2 \text{ is } \underset{\sim}{A}_{21} \text{ THEN } y^{(1)} = a_0^{(1)} + a_1^{(1)} x_1 + a_2^{(1)} x_2 \\
&\textit{Rule 2}: \quad \text{IF } x_1 \text{ is } \underset{\sim}{A}_{12} \text{ and } x_2 \text{ is } \underset{\sim}{A}_{22} \text{ THEN } y^{(2)} = a_0^{(2)} + a_1^{(2)} x_1 + a_2^{(2)} x_2 \\
&\textit{Rule 3}: \quad \text{IF } x_1 \text{ is } \underset{\sim}{A}_{12} \text{ and } x_2 \text{ is } \underset{\sim}{A}_{21} \text{ THEN } y^{(3)} = a_0^{(3)} + a_1^{(3)} x_1 + a_2^{(3)} x_2 \\
&\textit{Rule 4}: \quad \text{IF } x_1 \text{ is } \underset{\sim}{A}_{11} \text{ and } x_2 \text{ is } \underset{\sim}{A}_{22} \text{ THEN } y^{(4)} = a_0^{(4)} + a_1^{(4)} x_1 + a_2^{(4)} x_2
\end{aligned}
\tag{12.46}
$$

where $\underset{\sim}{A}_{11}$ and $\underset{\sim}{A}_{12}$ are fuzzy sets on the universe of discourse, of input variable x_1, $\underset{\sim}{A}_{21}$ and $\underset{\sim}{A}_{22}$ are fuzzy sets on the universe of discourse of input variable x_2; $a_0^{(r)}$, $a_1^{(r)}$ and $a_2^{(r)}$ is a set of parameters specified for rule r.

Let us now discuss the purpose of each layer in ANFIS of Fig. 12.32.

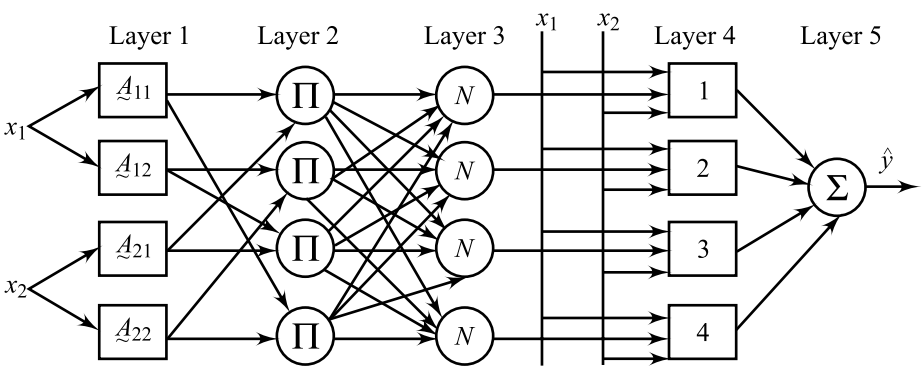


Fig. 12.32 An Adaptive Neuro-Fuzzy Inference System (ANFIS)

Layer 1 The inputs to the nodes in the first layer are the input fuzzy sets of the ANFIS. Since these fuzzy sets are fuzzy singletons, numerical inputs are directly transmitted to the first-layer nodes.

Nodes in this layer represent the membership functions associated with each linguistic term of input variables. Every node here is an *adaptive* node. Links in this layer are fully connected between input terminals and their corresponding membership function nodes. Membership functions can be any appropriate parameterized function; we use Gaussian function.

$$
\mu_{\underset{\sim}{A}_{i j_i}}(x_i) \triangleq \mu_{ij_i}(x_i) = \exp\left[-\left(\frac{x_i - c_{(i,j_i)}}{w_{(i,j_i)}} \right)^2 \right] \tag{12.47}
$$

The nodes are labeled $\underset{\sim}{A}_{i j_i}$; $i = 1, 2$; $j_i = 1, 2$. Total number of nodes in this layer is, therefore, four. $c_{(i,j_i)}$ is the center (mean) and $w_{(i,j_i)}$ is the width (variance), respectively, of the membership function corresponding to the node $\underset{\sim}{A}_{i j_i}$; x_i is the input and $\mu_{i j_i}$ is the output of the node. The adjusted weights in Layer 1 are $c_{(i,j_i)}$'s and $w_{(i,j_i)}$'s. As the values of these parameters change, the Gaussian function varies accordingly; thus exhibiting various forms of membership functions of fuzzy set $\underset{\sim}{A}_{i j_i}$. Parameters in this layer are referred to as *premise parameters*.

Layer 2 Every node in this layer is a fixed node labeled Π, whose output is the product of all the incoming signals. Each node output represents firing strength of a rule. In fact, other *t*-norm operators could also be used as node functions.

Each node, representing a single Sugeno fuzzy rule, has the output

$$\mu^{(r)}(\mathbf{x}) \triangleq \prod_{(i,j_i)\in I_r} \mu_{ij_i}(x_i) \tag{12.48}$$

where I_r is the set of all $\mathcal{A}_{ij_i}$ associated with the premise part of rule r.

Layer 3 Every node in this layer is a *fixed* node labeled N. The rth node calculates the ratio of the rth rule's firing strength, to the sum of all rules' firing strengths:

$$\bar{\mu}^{(r)} = \frac{\mu^{(r)}}{\sum_{r=1}^{R} \mu^{(r)}} = \text{Normalized firing strength of rule } r \tag{12.49}$$

Layer 4 Every node is this layer is an *adaptive* node, is connected to the respective normalization node in the previous layer, and also receives inputs x_1 and x_2. It calculates the weighted consequent value of a given rule as

$$\hat{y}^{(r)} = \bar{\mu}^{(r)}\left[a_0^{(r)} + a_1^{(r)}x_1 + a_2^{(r)}x_2\right] \tag{12.50}$$

where $\bar{\mu}^{(r)}$ is the normalized firing strength from layer 3, and $a_0^{(r)}$, $a_1^{(r)}$ and $a_2^{(r)}$ are the parameters of this node. Parameters in this layer are referred to as *consequent parameters*.

Each node in Layer 4 is a local linear model of the Sugeno fuzzy system; integration of outputs of all local linear models yields global output.

Layer 5 The single node in this layer is a *fixed* mode labeled Σ, which computes the overall output as the summation of all incoming signals:

$$\hat{y} = \sum_{r=1}^{R} (a_0^{(r)} + a_1^{(r)}x_1 + a_2^{(r)}x_2)\bar{\mu}^{(r)}; R = 4 \tag{12.51}$$

12.7.2 How Does an ANFIS Learn?

An ANFIS uses a hybrid learning algorithm that combines the least squares estimator and the gradient descent method. First, initial activation functions are assigned to each membership neuron. The function centers of the neurons connected to input x_i, are set so that the domain of x_i is divided equally, and the widths are set to allow sufficient overlapping of the respective functions.

In an ANFIS training algorithm, each epoch is composed of a forward pass and a backward pass. In the forward pass, a training set of input patterns (input vector $\mathbf{x}$) is presented to the ANFIS, neurons outputs are calculated on the layer-by-layer basis, and the rules consequent parameters are identified by the least squares estimator. In the Sugeno fuzzy inference, an output $\hat{y}$ is a linear function. Thus, given the values of the membership parameters and a training set of P input-output patterns, we can form P linear equations in terms of the consequent parameters (refer to Eqns (12.45)). Least-squares solution of these equations yields the consequent parameters.

As soon as the rule consequent parameters are established, we can compute actual network output, $\hat{y}$, and determine the error

$$e = y - \hat{y} \tag{12.52}$$

In the backward pass, the backpropagation algorithm in applied. The error signals are propagated back, and the premise parameters are updated according to the chain rule.

The goal is to minimize the error function

$$E = \tfrac{1}{2}(y - \hat{y})^2 \tag{12.53}$$

The error at Layer 5:

$$\frac{\partial E}{\partial \hat{y}} = (\hat{y} - y) \tag{12.54}$$

Back propagating to Layer 3 via Layer 4 (refer to Eqn. (12.51)),

$$\frac{\partial E}{\partial \bar{\mu}^{(r)}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \bar{\mu}^{(r)}} = \frac{\partial E}{\partial \hat{y}} \left[(a_0^{(r)} + a_1^{(r)} x_1 + a_2^{(r)} x_2) \right] \tag{12.55}$$

Back propagating to Layer 2 (refer to Eqn. (12.49)),

$$\frac{\partial E}{\partial \mu^{(r)}} = \frac{\partial E}{\partial \bar{\mu}^{(r)}} \frac{\partial \bar{\mu}^{(r)}}{\partial \mu^{(r)}} = \frac{\partial E}{\partial \bar{\mu}^{(r)}} \left[\frac{\bar{\mu}^{(r)} (1 - \bar{\mu}^{(r)})}{\mu^{(r)}} \right] \tag{12.56}$$

The error at Layer 1:

I_r is the set of all $A_{i j_i}$ associated with the premise part of rule r. Reverse pass: $I_{(i,j_i)}$ is the set of all rule nodes in Layer 2 connected to $(i, j_i)^{\text{th}}$ node (corresponding to $A_{i j_i}$) of Layer 1.

Back propagating error to Layer 1 (refer to Eqn. (12.48)),

$$\begin{aligned} \frac{\partial E}{\partial \mu_{ij_i}} &= \sum_{r \in I_{(i,j_i)}} \frac{\partial E}{\partial \mu^{(r)}} \frac{\partial \mu^{(r)}}{\partial \mu_{ij_i}} \\ &= \sum_{r \in I_{(i,j_i)}} \frac{\partial E}{\partial \mu^{(r)}} \left[\frac{\mu^{(r)}}{\mu_{ij_i}} \right] \end{aligned} \tag{12.57a}$$

From Eqn. (12.47), we obtain

$$\frac{\partial \mu_{ij_i}}{\partial c_{(i,j_i)}} = 2\mu_{i j_i} (x_i - c_{(i,j_i)}) / w_{(i,j_i)}^2 \tag{12.57b}$$

$$\frac{\partial \mu_{i j_i}}{\partial w_{(i,j_i)}} = 2\mu_{i j_i} (x_i - c_{(i,j_i)})^2 / w_{(i,j_i)}^3 \tag{12.57c}$$

Denoting the iteration index by k (refer to Eqn. (11.27)),

$$c_{(i,j_i)}(k+1) = c_{(i,j_i)}(k) - \eta \frac{\partial E(k)}{\partial c_{(i,j_i)}(k)} \tag{12.58a}$$

$$w_{(i,j_i)}(k+1) = w_{(i,j_i)}(k) - \eta \frac{\partial E(k)}{\partial w_{(i,j_i)}(k)} \tag{12.58b}$$

where η is the learning rate.

For given input-output pairs $(\mathbf{x}^{(p)}, y^{(p)}; p = 1, 2, ..., P)$, the batch-updating algorithm back propagates the cumulative error resulting from the difference between $y^{(p)}; p = 1,..., P$, and $\hat{y}^{(p)}; p = 1,..., P$, from output layer to the previous layers to update weights of the network.

In this section, we have described a method that can be used to construct identifiers of dynamical systems that, in turn, could be employed to construct *neuro-fuzzy control* systems. The idea behind the method is to apply the backpropagation algorithm to a fuzzy logic system.

Neuro-fuzzy control refers to the design methods for fuzzy logic controllers that employ neural network techniques. The design methods for neuro-fuzzy control are derived directly from methods for neural control. Thus, if we replace the NN blocks in Figs 11.23–11.25 with ANFIS blocks, then we end up with neuro-fuzzy control systems.

REVIEW EXAMPLES

Review Example 12.1

We consider here the simplest fuzzy PI control scheme for a servo motor with the control model (refer to Fig. 12.33a)

$$\frac{Y(s)}{U(s)} = G(s) = \frac{1}{s(s+3.6)} \tag{12.59}$$

The objective of the fuzzy controller is to control angular position $y(t)$ of the servo motor to achieve a given set-point y_r, within desired accuracy.

The discretized model for the plant (refer to Chapter 3) is

$$G_{h0}G(z) = \frac{Y(z)}{U(z)} = \frac{0.0237z^{-1} + 0.0175z^{-2}}{1 - 1.407z^{-1} + 0.407z^{-2}}$$

$$y(k) = 1.407y(k-1) - 0.407y(k-2) + 0.0237u(k-1) + 0.0175u(k-2) \tag{12.60}$$

The proposed fuzzy controller (refer to Fig. 12.33b) has the following two input variables:

$e(k)$ = error between the set-point and actual position of the shaft;

$v(k)$ = rate of change of error;

and one output variable:

$\Delta u(k)$ = incremental voltage signal to the driver circuit of the motor.

Universe of discourse for $e(k)$ = $\{-L_e, L_e\}$

Universe of discourse for $v(k)$ = $\{-L_v, L_v\}$

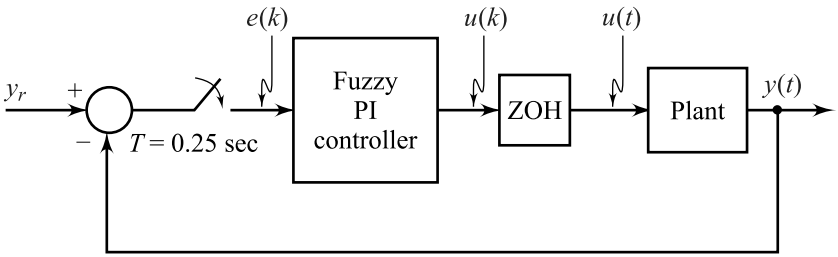
Universe of discourse for $\Delta u(k)$ = $\{-H_{\Delta u}, H_{\Delta u}\}$

Clockwise and counterclockwise rotations are defined as positive and negative, respectively.

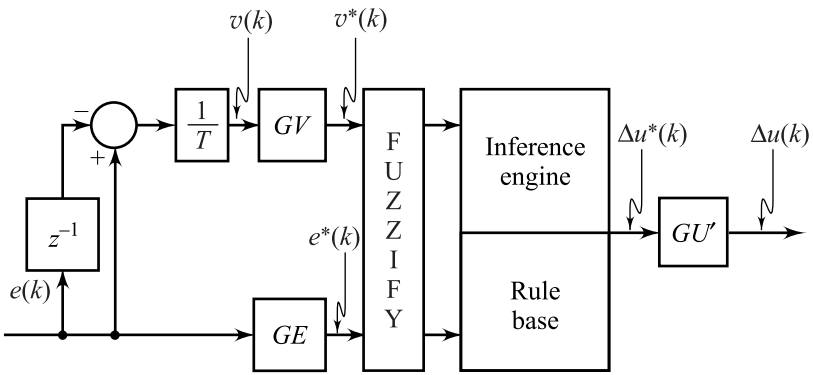
The two input variables are quantized to two fuzzy subsets: Positive ($\underset{\sim}{P}$), Negative ($\underset{\sim}{N}$); and the output variable is quantized to three fuzzy subsets: Positive ($\underset{\sim}{P}$), Zero ($\underset{\sim}{Z}$), Negative ($\underset{\sim}{N}$). Triangular membership functions are used.

The scaling factors *GE* (gain for error variable), and *GV* (gain for velocity variable) describe input normalization:

$$e^*(k) = GE \times e(k);\ GE = L/L_e$$



(a)



(b)

Fig. 12.33 Structure for a fuzzy controller

$$v^*(k) = GV \times v(k);\ GV = L/L_v$$

where $e^*, v^* \in \{-L, L\}$

The output Δu^* of the fuzzy controller is denormalized to Δu, by the relation

$$\Delta u(k) = GU' \times \Delta u^*(k);\ GU' = H_{\Delta u}/H$$

where $\Delta u^* \in \{-H, H\}$

Without loss of generality, we take $L = H = 1$ (refer to Fig. 12.34).

The fuzzy PI controller uses the following four fuzzy rules:

IF $e^*(k)$ is $\underset{\sim}{P}$ **and** $v^*(k)$ is $\underset{\sim}{P}$ THEN $\Delta u^*(k)$ is $\underset{\sim}{P}$

IF $e^*(k)$ is $\underset{\sim}{P}$ **and** $v^*(k)$ is $\underset{\sim}{N}$ THEN $\Delta u^*(k)$ is $\underset{\sim}{Z}$

IF $e^*(k)$ is $\underset{\sim}{N}$ **and** $v^*(k)$ is $\underset{\sim}{P}$ THEN $\Delta u^*(k)$ is $\underset{\sim}{Z}$

IF $e^*(k)$ is $\underset{\sim}{N}$ **and** $v^*(k)$ is $\underset{\sim}{N}$ THEN $\Delta u^*(k)$ is $\underset{\sim}{N}$

The initial value of the system output and the initial velocity are set to zero, as is the initial output of the fuzzy PI controller.

The scaling factors *GE*, *GV* and *GU′* of the fuzzy controller may be tuned by trial and error. Refer to Appendix B for realization of the controller.

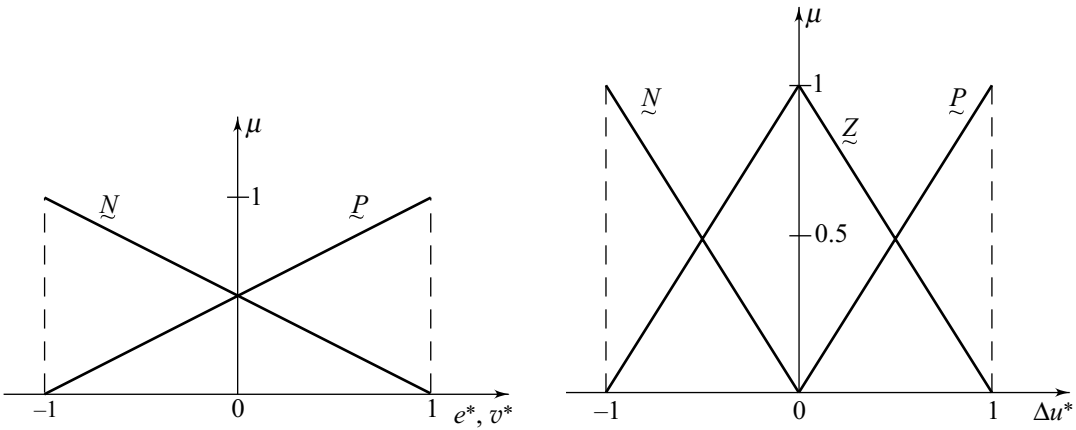


Fig. 12.34 Membership functions for input and output variables of a fuzzy controller

Review Example 12.2

Figure 12.32 shows the schematic diagram of an ANFIS, used to model a process with two inputs, x_1 and x_2, and one output y. Two fuzzy sets $\underset{\sim}{A}_{11}$ and $\underset{\sim}{A}_{12}$ have been utilized to represent x_1; and x_2 has been expressed using two other fuzzy sets $\underset{\sim}{A}_{21}$ and $\underset{\sim}{A}_{22}$. The membership function distributions of x_1 and x_2 are shown in Fig. 12.35.

There is a maximum of 2 × 2 possible rules (refer to (12.46)); the values of the coefficients of the consequent part of the rules are as follows:

$$a_0^{(1)} = 0.10,\ a_0^{(2)} = 0.11,\ a_0^{(3)} = 0.13,\ a_0^{(4)} = 0.14,\ a_1^{(1)} = 0.2,\ a_1^{(2)} = 0.2,\ a_1^{(3)} = 0.3,$$
$$a_1^{(4)} = 0.3,\ a_2^{(1)} = 0.3,\ a_2^{(2)} = 0.4,\ a_2^{(3)} = 0.3,\ a_2^{(4)} = 0.4.$$

The objective is to determine the predicted output $\hat{y}$ of ANFIS when $x_1 = 1.1$ and $x_2 = 6.0$.

For given values of x_1 and x_2, we find, using the principle of similar triangles, from Fig. 12.35 (Layer 1 in Fig. 12.32):

$$\mu_{\underset{\sim}{A}_{11}}(x_1) = \left(\frac{2.01 - 1.1}{2.01 - 1}\right) \times 1 = 0.900990$$

$$\mu_{\underset{\sim}{A}_{12}}(x_1) = \left(\frac{1.1 - 1}{2.01 - 1}\right) \times 1 = 0.099010$$

$$\mu_{\underset{\sim}{A}_{21}}(x_2) = \left(\frac{10 - 6}{10 - 5}\right) \times 1 = 0.8$$

$$\mu_{\underset{\sim}{A}_{22}}(x_2) = \left(\frac{6 - 5}{10 - 5}\right) \times 1 = 0.2$$

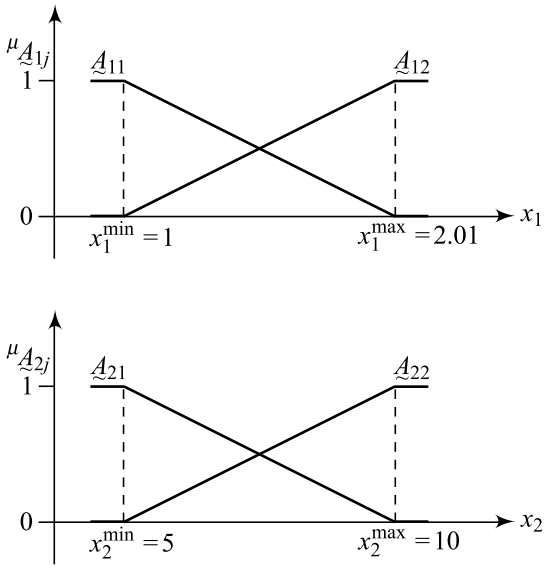


Fig. 12.35 Membership function distributions

All the possible four rules, given in (12.46), will be fired. Firing strengths of the rules are (Layer 2 in Fig. 12.32; Eqn. (12.48)):

$$\mu^{(1)}(\mathbf{x}) = 0.900990 \times 0.8 = 0.720792$$
$$\mu^{(2)}(\mathbf{x}) = 0.099010 \times 0.2 = 0.019802$$
$$\mu^{(3)}(\mathbf{x}) = 0.099009 \times 0.8 = 0.079208$$
$$\mu^{(4)}(\mathbf{x}) = 0.900990 \times 0.2 = 0.180198$$

The normalized firing strengths of the rules are (Layer 3 in Fig. 12.32; Eqn. (12.49)):

$$\bar{\mu}^{(1)} = \mu^{(1)} \Big/ \sum_{r=1}^{4} \mu^{(r)} = \mu^{(1)} = 0.720792$$
$$\bar{\mu}^{(2)} = \mu^{(2)},\ \bar{\mu}^{(3)} = \mu^{(3)},\ \bar{\mu}^{(4)} = \mu^{(4)}$$

Weighted consequent values of the rules are (Layer 4 in Fig. 12.32; Eqn. (12.50)):

$$\hat{y}^{(1)} = 0.720792(0.10 + 0.2 \times 1.1 + 0.3 \times 6.0) = 1.528079;$$
$$\hat{y}^{(2)} = 0.054059;\ \hat{y}^{(3)} = 0.179010;\ \hat{y}^{(4)} = 0.517168$$

Predicted output of the ANFIS, is (Layer 5 in Fig. 12.32; Eqn. (12.51)):

$$\hat{y} = 2.278316$$

PROBLEMS

12.1 (a) In the following, we suggest a membership function for fuzzy description of the set 'real numbers *close* to 2':

$$\underset{\sim}{A} = \{x, \mu_{\underset{\sim}{A}}(x)\}$$

where

$$\mu_{\underset{\sim}{A}}(x) = \begin{cases} 0 & ; x < 1 \\ -x^2 + 4x - 3 & ; 1 \le x \le 3 \\ 0 & ; x > 3 \end{cases}$$

Sketch the membership function (arc of a parabola) and determine its supporting interval, and α-cut interval for $\alpha = 0.5$.

(b) Sketch the piecewise quadratic membership function

$$\mu_{\underset{\sim}{B}}(x) = \begin{cases} 2(x-1)^2 & ; 1 \le x < 3/2 \\ 1 - 2(x-2)^2 & ; 3/2 \le x < 5/2 \\ 2(x-3)^2 & ; 5/2 \le x \le 3 \\ 0 & ; \text{otherwise} \end{cases}$$

and show that it also represents 'real number *close* to 2'. Determine its support, and α-cut for $\alpha = 0.5$.

12.2 (a) The well known *Gaussian distribution* in probability is defined by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} ; -\infty < x < \infty$$

where μ is the mean and σ is the standard deviation of the distribution. Construct a normal, convex membership function from this distribution (select parameters μ and σ) that represents 'real numbers *close* to 2'. Find its support, and α-cut for $\alpha = 0.5$. Show that the membership function

$$\mu_{\underset{\sim}{A}}(x) = \frac{1}{1 + (x-2)^2}$$

also represents 'real numbers *close* to 2'. Find its support, and α-cut for $\alpha = 0.5$.

12.3 Consider the piecewise quadratic function

$$f(x) = \begin{cases} 0 & ; x < a \\ 2\left(\dfrac{x-a}{b-a}\right)^2 & ; a \le x < \dfrac{a+b}{2} \\ 1 - 2\left(\dfrac{x-b}{b-a}\right)^2 & ; \dfrac{a+b}{2} \le x < b \\ 1 & ; b \le x < c \end{cases}$$

Construct a normal, convex membership function from $f(x)$ (select parameters a, b and c) that represents the set 'tall men' on the universe $\{3, 9\}$. Determine the crosspoints and support of the membership function.

12.4 (a) Write an analytical expression for the membership function $\mu_{\underset{\sim}{A}}(x)$ with supporting interval $[-1, 9]$ and α-cut interval for $\alpha = 1$ given as $[4, 5]$.

(b) Define what we mean by a normal membership function and a convex membership function. Is the function described in (a) above (i) normal, (ii) convex?

12.5 (a) Let the fuzzy set $\underset{\sim}{A}$ be the linguistic 'warm' with membership function

$$\mu_{\underset{\sim}{A}}(x) = \begin{cases} 0 & ; \quad x < a_1 \\ \dfrac{x - a_1}{b_1 - a_1} & ; \quad a_1 \le x \le b_1 \\ 1 & ; \quad b_1 \le x \le b_2 \\ \dfrac{x - a_2}{b_2 - a_2} & ; \quad b_2 \le x \le a_2 \\ 0 & ; \quad x \ge a_2 \end{cases}$$

$a_1 = 64°\text{F}$, $b_1 = 70°\text{F}$, $b_2 = 74°\text{F}$, $a_2 = 78°\text{F}$

(i) Is $\underset{\sim}{A}$ a normal fuzzy set?

(ii) Is $\underset{\sim}{A}$ a convex fuzzy set?

(iii) Is $\underset{\sim}{A}$ a singleton fuzzy set?

If answer to one or more to these is 'no', then give an example of such a set.

(b) For fuzzy set $\underset{\sim}{A}$ described in part (a), assume that $b_1 = b_2 = 72°\text{F}$.

Sketch the resulting membership function and determine its support, crosspoints and α-cuts for $\alpha = 0.2$ and 0.4.

12.6 Consider two fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$; membership functions $\mu_{\underset{\sim}{A}}(x)$ and $\mu_{\underset{\sim}{B}}(x)$ are shown in Fig. P12.6. The fuzzy variable x is temperature.

Sketch the graph of $\mu_{\bar{\underset{\sim}{A}}}(x)$, $\mu_{\underset{\sim}{A} \cap \underset{\sim}{B}}(x)$ and $\mu_{\underset{\sim}{A} \cup \underset{\sim}{B}}(x)$.

Which t-norm and t-conorm have you used?

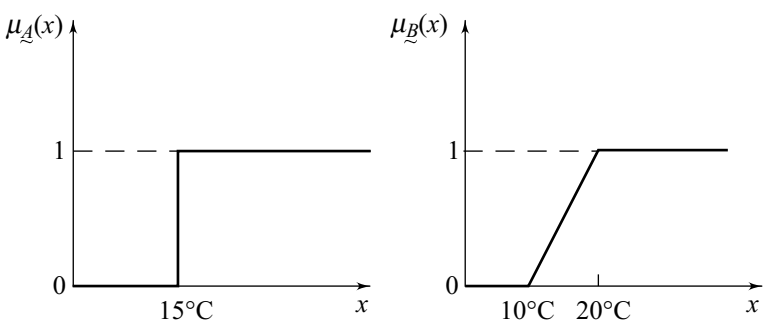


Fig. P12.6

12.7 Consider the fuzzy relation $\underset{\sim}{R}$ on the universe $X \times Y$, given by the membership function

$$\mu_{\underset{\sim}{R}}(x, y) = \frac{1}{[1+100(x-3y)^4]},$$

vaguely representing the crisp relation $x = 3y$. All elements satisfying $x = 3y$ have unity grade of membership; elements satisfying, for example, $x = 3.1y$ have membership grades less than 1. The farther away the elements are from the straight line, the lower are the membership grades.

Give a graphical representation of the fuzzy relation $\underset{\sim}{R}$.

12.8 Assume the membership function of the fuzzy set $\underset{\sim}{A}$, *big pressure*, is

$$\mu_{\underset{\sim}{A}}(x) = \begin{cases} 1 & ; \quad x \geq 5 \\ 1 - \dfrac{5-x}{4} & ; \quad 1 \leq x \leq 5 \\ 0 & ; \quad \text{otherwise} \end{cases}$$

Assume the membership function of the fuzzy set $\underset{\sim}{B}$, *small volume*, is

$$\mu_{\underset{\sim}{B}}(y) = \begin{cases} 1 & ; \quad y \leq 1 \\ 1 - \dfrac{y-1}{4} & ; \quad 1 \leq y \leq 5 \\ 0 & ; \quad \text{otherwise} \end{cases}$$

Find the truth values of the following propositions:

(i) 4 is big pressure.

(ii) 3 is small volume.

(iii) 4 is big pressure **and** 3 is small volume.

(iv) 4 is big pressure $\rightarrow$ 3 is small volume.

Explain the conjunction and implication operations you have used for this purpose.

12.9 Consider the following statements:

Input	:	$\underset{\sim}{A}'$ is very small
Rule	:	IF $\underset{\sim}{A}'$ is small THEN $\underset{\sim}{B}'$ is large
Inference	:	$\underset{\sim}{B}'$ is very large

If $\underset{\sim}{R}$ is a fuzzy relation from X to Y representing the implication rule, and $\underset{\sim}{A}'$ is a fuzzy subset of X, then the fuzzy subset $\underset{\sim}{B}'$ of Y, which is induced by $\underset{\sim}{A}'$, is given by

$$\underset{\sim}{B}' = \underset{\sim}{A}' \circ \underset{\sim}{R}$$

where $\circ$ operation (composition) is carried out by taking cylindrical extension of $\underset{\sim}{A}'$, taking the intersection with $\underset{\sim}{R}$, and projecting the result onto Y.

Define cylindrical extension, intersection and projection operations that lead to max-min compositional rule of inference.

12.10

Input	:	x is $\underset{\sim}{A}'$ **and** y is $\underset{\sim}{B}'$
Rule 1	:	IF x is $\underset{\sim}{A}_1$ **and** y is $\underset{\sim}{B}_1$ THEN z is $\underset{\sim}{C}_1$
Rule 2	:	IF x is $\underset{\sim}{A}_2$ **and** y is $\underset{\sim}{B}_2$ THEN z is $\underset{\sim}{C}_2$
Inference	:	z is $\underset{\sim}{C}'$

Taking arbitrary membership functions for $\underset{\sim}{A}_1$, $\underset{\sim}{B}_1$, $\underset{\sim}{C}_1$, $\underset{\sim}{A}_2$, $\underset{\sim}{B}_2$ and $\underset{\sim}{C}_2$, outline the procedure of determining $\underset{\sim}{C}'$ corresponding to the crisp inputs $x = x_0$ and $y = y_0$. Use t-norm 'min' for conjunction operation, Mamdani's implication operation and max-min compositional rule of inference.

12.11 Fig. P12.11 shows the fuzzy output of a certain control problem. Defuzzify by using the center of area method, to obtain the value of crisp control action.

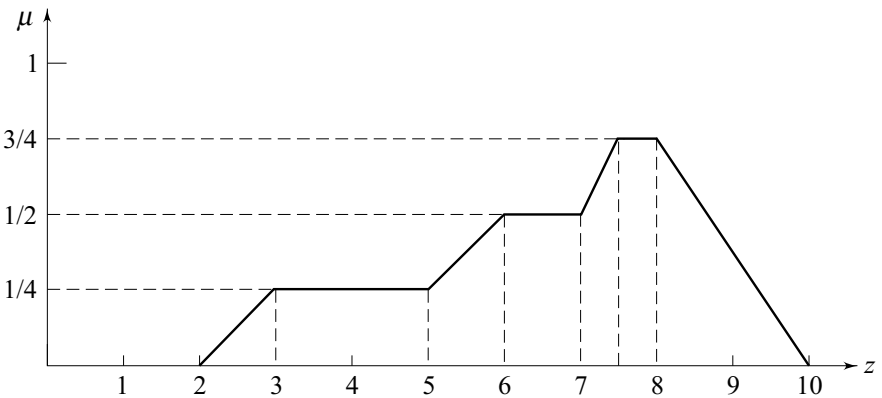


Fig. P12.11

12.12 Consider the fuzzy system concerning the terminal voltage and speed of an electric motor, described by the membership functions

x	100	150	200	250	300
$\mu_{\underset{\sim}{A}}(x)$	1	0.8	0.5	0.2	0.1
y	1600	1800	2000	2200	2400
$\mu_{\underset{\sim}{B}}(y)$	1	0.9	0.7	0.3	0

Input	:	Voltage is rather small (x is $\underset{\sim}{A}'$)
Rule	:	IF voltage is small (x is $\underset{\sim}{A}$) THEN speed is small (y is $\underset{\sim}{B}$)
Inference	:	Speed is rather small (y is $\underset{\sim}{B}'$)

Assume that the input fuzzy set $\underset{\sim}{A}'$ is a singleton at $x_0 = 125$. Determine the inference fuzzy set $\underset{\sim}{B}'$ of the fuzzy system. Defuzzify this set to obtain crisp value for speed.

Use piecewise continuous approximations of graphs of $\mu_{\underset{\sim}{A}}(x)$ and $\mu_{\underset{\sim}{B}}(y)$ to describe your solution.

12.13 Consider the two-input, one-output fuzzy system:

Input	:	x is $\underset{\sim}{A}'$ **and** y is $\underset{\sim}{B}'$
Rule 1	:	IF x is $\underset{\sim}{A}_1$ **and** y is $\underset{\sim}{B}_1$ THEN z is $\underset{\sim}{C}_1$
Rule 2	:	IF x is $\underset{\sim}{A}_2$ **and** y is $\underset{\sim}{B}_2$ THEN z is $\underset{\sim}{C}_2$
Inference	:	z is $\underset{\sim}{C}'$

The fuzzy sets $\underset{\sim}{A}_i$, $\underset{\sim}{B}_i$ and $\underset{\sim}{C}_i$; $i = 1, 2$, have the membership functions

$$\mu_{\underset{\sim}{A}_1}(x) = \begin{cases} \dfrac{x-2}{3} & ; \quad 2 \le x \le 5 \\ \dfrac{8-x}{3} & ; \quad 5 \le x \le 8 \end{cases} \qquad \mu_{\underset{\sim}{A}_2}(x) = \begin{cases} \dfrac{x-3}{3} & ; \quad 3 \le x \le 6 \\ \dfrac{9-x}{3} & ; \quad 6 \le x \le 9 \end{cases}$$

$$\mu_{\underset{\sim}{B}_1}(y) = \begin{cases} \dfrac{y-5}{3} & ; \quad 5 \le y \le 8 \\ \dfrac{11-y}{3} & ; \quad 8 \le y \le 11 \end{cases} \qquad \mu_{\underset{\sim}{B}_2}(y) = \begin{cases} \dfrac{y-4}{3} & ; \quad 4 \le y \le 7 \\ \dfrac{10-y}{3} & ; \quad 7 \le y \le 10 \end{cases}$$

$$\mu_{\underset{\sim}{C}_1}(z) = \begin{cases} \dfrac{z-1}{3} & ; \quad 1 \le z \le 4 \\ \dfrac{7-z}{3} & ; \quad 4 \le z \le 7 \end{cases} \qquad \mu_{\underset{\sim}{C}_2}(z) = \begin{cases} \dfrac{z-3}{3} & ; \quad 3 \le z \le 6 \\ \dfrac{9-z}{3} & ; \quad 6 \le z \le 9 \end{cases}$$

Assume fuzzy sets $\underset{\sim}{A}'$ and $\underset{\sim}{B}'$ are singletons at $x_0 = 4$ and $y_0 = 8$. Determine the inference fuzzy set $\underset{\sim}{C}'$ of the fuzzy system. Defuzzify $\underset{\sim}{C}'$.

12.14 The control objective is to design an automatic braking system for motor cars. We need two analog signals: vehicle speed (V), and a measure of distance (D) from the vehicle in the front. A fuzzy logic control system will process these, giving a single output, braking force (B), which controls the brakes.

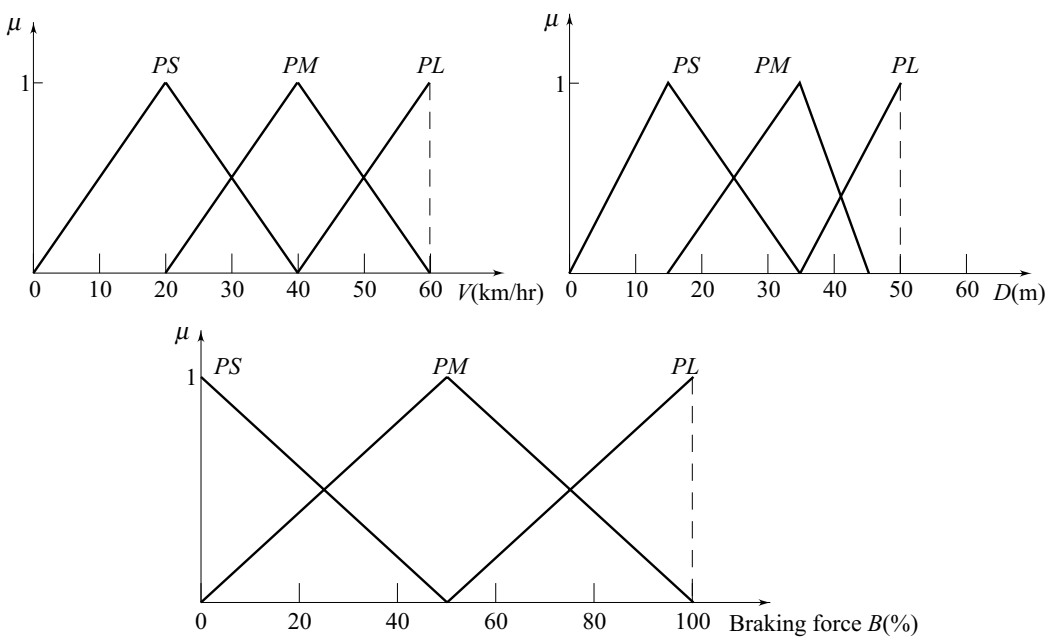


Fig. P12.14

Term set for each of the variables (V, D, and B) is of the form:

{*PS* (positive small), *PM* (positive medium), *PL* (positive large)}

Membership functions for each term-set are given in Fig. P12.14.

Suppose that for the control problem, two rules have to be fired:

Rule 1: IF $D = PS$ **and** $V = PM$ THEN $B = PL$

Rule 2: IF $D = PM$ **and** $V = PL$ THEN $B = PM$

For the sensor readings of $V = 55$ km/hr, and $D = 27$ m from the car in front, find graphically

(i) the firing strengths of the two rules;

(ii) the aggregated output; and

(iii) defuzzified control action.

12.15 The control objective is to automate the *wash time* when using a washing machine. Experts select for inputs *dirt* and *grease* of the clothes to be washed, and for output parameter the *wash time*, as follows:

Dirt $\triangleq$ {*SD* (small dirt), *MD* (medium dirt), *LD* (large dirt)}

Grease $\triangleq$ {*NG* (no grease), *MG* (medium grease), *LG* (large grease)}

Washtime $\triangleq$ {*VS* (very short), *S* (short), *M* (medium), *L* (long), *VL* (very long)}

The degrees of the dirt and grease are measured on a scale from 0 to 100; washtime is measured in minutes from 0 to 60.

$$\mu_{\underset{\sim}{SD}}(x) = \frac{50 - x}{50};\ 0 \le x \le 50 \qquad \mu_{\underset{\sim}{VS}}(z) = \frac{10 - z}{10};\ 0 \le z \le 10$$

$$\mu_{\underset{\sim}{MD}}(x) = \begin{cases} \dfrac{x}{50} & ;\ 0 \le x \le 50 \\ \dfrac{100 - x}{50} & ;\ 50 \le x \le 100 \end{cases} \qquad \mu_{\underset{\sim}{S}}(z) = \begin{cases} \dfrac{z}{10} & ;\ 0 \le z \le 10 \\ \dfrac{25 - z}{15} & ;\ 10 \le z \le 25 \end{cases}$$

$$\mu_{\underset{\sim}{LD}}(x) = \frac{x - 50}{50};\ 50 \le x \le 100 \qquad \mu_{\underset{\sim}{M}}(z) = \begin{cases} \dfrac{z - 10}{15} & ;\ 10 \le z \le 25 \\ \dfrac{40 - z}{15} & ;\ 25 \le z \le 40 \end{cases}$$

$$\mu_{\underset{\sim}{NG}}(y) = \frac{50 - y}{50};\ 0 \le y \le 50 \qquad \mu_{\underset{\sim}{L}}(z) = \begin{cases} \dfrac{z - 25}{15} & ;\ 25 \le z \le 40 \\ \dfrac{60 - z}{20} & ;\ 40 \le z \le 60 \end{cases}$$

$$\mu_{\underset{\sim}{MG}}(y) = \begin{cases} \dfrac{y}{50} & ;\ 0 \le y \le 50 \\ \dfrac{100 - y}{50} & ;\ 50 \le y \le 100 \end{cases} \qquad \mu_{\underset{\sim}{VL}}(z) = \frac{z - 40}{20};\ 40 \le z \le 60$$

$$\mu_{\underset{\sim}{LG}}(y) = \frac{y - 50}{50};\ 50 \le y \le 100$$

The selected rules are as follows:

Grease → *Dirt* ↓	*NG*	*MG*	*LG*
SD	*VS*	*M*	*L*
MD	*S*	*M*	*L*
LD	*M*	*L*	*VL*

Find a crisp control output for the following sensor readings:
Dirt = 60; *Grease* = 70

12.16 A fuzzy controller is acting according to the following rule base (*N* = *negative*, *M* = *medium*, *P* = *positive*):

R_1 : If x_1 is *N* AND x_2 is *N*, THEN *u* is *N*

R_2 : If x_1 is *N* OR x_2 is *P*, THEN *u* is *M*

R_3 : If x_1 is *P* OR x_2 is *N*, THEN *u* is *M*

R_4 : If x_1 is *P* AND x_2 is *P*, THEN *u* is *P*

The membership functions of the input and output variables are given in Fig. P12.16. Actual inputs are $x_1 = 2.5$ and $x_2 = 4$. Which rules are active and what will be the controller action *u*? Find *u* by applying standard fuzzy operations: **min** for AND, and **max** for OR.

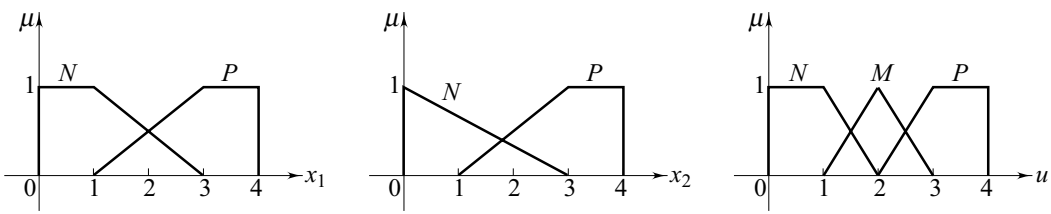


Fig. P12.16

12.17 Consider the following fuzzy model of a system with inputs *x* and *y* and outpur *z*:

Rule 1 : If *x* is A_3 OR *y* is B_1, THEN *z* is C_1

Rule 2 : If *x* is A_2 AND *y* is B_2, THEN *z* is C_2

Rule 3 : If *x* is A_1, THEN *z* is C_3

The membership functions of the input and output variables are given in Fig. P12.17. Actual inputs are x_1 and y_1. Find the output *z* by applying standard fuzzy operation: **min** for AND, and **max** for OR.

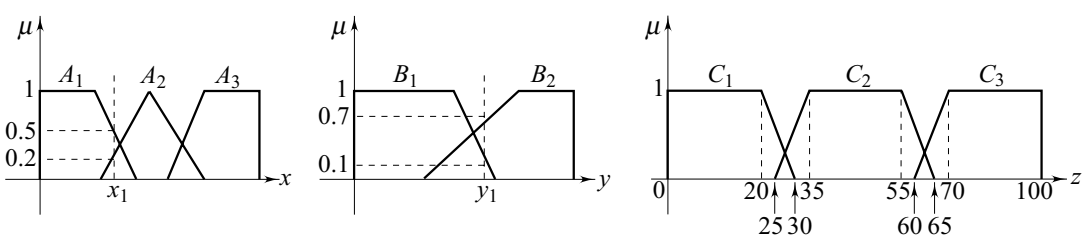


Fig. P12.17

12.18 A fuzzy controller is acting according to the following rule base (N = *negative*, P = positive):

R_1 : If x_1 is N AND x_2 is N, THEN u is k_1

R_2 : If x_1 is N OR x_2 is P, THEN u is k_2

R_3 : If x_1 is P OR x_2 is N, THEN u is k_2

R_4 : If x_1 is P AND x_2 is P, THEN u is k_3

The membership functions of the input variables are given in Fig. P12.16 and the membership functions of the output variable (which is a controller action) u are singletons placed at $k_1 = 1$, $k_2 = 2$, $k_3 = 3$. Actual inputs are $x_1 = 2.5$ and $x_2 = 4$. Find u by applying standared fuzzy operations: **min** for AND, and **max** for OR.

12.19 Consider a two-dimensional *sinc* equation defined by

$$y = \text{sinc}(x_1, x_2) = \frac{\sin(x_1)\sin(x_2)}{x_1 x_2}$$

Training data are sampled uniformly from the input range [–10, 10] × [–10, 10]. With two symmetric triangular membership functions assigned to each input variable, construct a Sugeno fuzzy model architecture for the *sinc* function. Give defining equations for determination of the premise and consequent parameters of the model.

12.20 To identify the nonlinear system

$$y = \left(1 + (x_1)^{0.5} + (x_2)^{-1} + (x_3)^{-1.5}\right)^2$$

we assign two membership functions to each input variable. Training and testing data are sampled uniformly from the input ranges [1,6] × [1,6] × [1,6], and [1.5,5.5] × [1.5,5.5] × [1.5,5.5], respectively. Extract Sugeno fuzzy rules from the numerical input-output training data that could be employed in an ANFIS model.

12.21 Assume that a fuzzy inference system has two inputs x_1 and x_2, and one output y. The rule base contains two Sugeno fuzzy rules as follows:

Rule 1: IF x_1 is $\underset{\sim}{A}_{11}$ and x_2 is $\underset{\sim}{A}_{21}$ THEN $y^{(1)} = a_0^{(1)} + a_1^{(1)} x_1 + a_2^{(1)} x_2$

Rule 2: IF x_1 is $\underset{\sim}{A}_{12}$ and x_2 is $\underset{\sim}{A}_{22}$ THEN $y^{(2)} = a_0^{(2)} + a_1^{(2)} x_1 + a_2^{(2)} x_2$

$\underset{\sim}{A}_{ij}$ are Gaussian functions.

For given input values x_1 and x_2, the inferred output is calculated by

$$\hat{y} = \frac{\mu^{(1)} y^{(1)} + \mu^{(2)} y^{(2)}}{\mu^{(1)} + \mu^{(2)}}$$

where $\mu^{(r)}$, $r = 1, 2$ are firing strengths of the two rules. Product inference is used to calculate the firing strengths of the rules.

Develop ANFIS architecture for this modeling problem, and derive learning algorithms based on least squares estimation and the gradient-descent methods.

12.22 Consider a fuzzy model (Mamdani architecture) for a manufacturing process. The process is characterized by two input variables, x_1 and x_2, and one output variable y. The membership function distribution (isosceles triangles of base widths θ_1, θ_2, θ_3) of x_1, x_2 and y are shown in Fig. P12.22, and a rule base is given in Table P12.22. Determine the output of the model for $x_1 = 10$, $x_2 = 28$.

Table P12.22

$x_2 \rightarrow$ x_1 $\downarrow$	$\underset{\sim}{A}_{21}$	$\underset{\sim}{A}_{22}$	$\underset{\sim}{A}_{23}$	$\underset{\sim}{A}_{24}$
$\underset{\sim}{A}_{11}$	$\underset{\sim}{S}$	$\underset{\sim}{S}$	$\underset{\sim}{M}$	$\underset{\sim}{L}$
$\underset{\sim}{A}_{12}$	$\underset{\sim}{S}$	$\underset{\sim}{M}$	$\underset{\sim}{L}$	$\underset{\sim}{L}$
$\underset{\sim}{A}_{13}$	$\underset{\sim}{M}$	$\underset{\sim}{M}$	$\underset{\sim}{L}$	$\underset{\sim}{XL}$
$\underset{\sim}{A}_{14}$	$\underset{\sim}{M}$	$\underset{\sim}{L}$	$\underset{\sim}{XL}$	$\underset{\sim}{XL}$

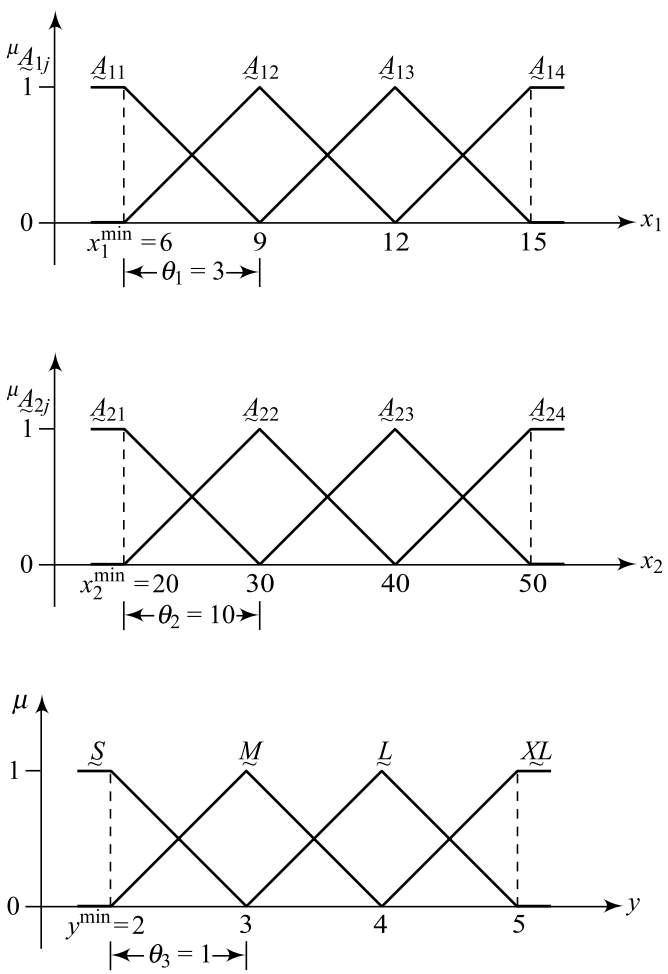


Fig. P12.22

12.23 Consider a fuzzy model (Sugeno architecture) for a manufacturing process. The process is characterized by two input variables, x_1 and x_2, and one output variable, y. The membership function distributions of x_1 and x_2 are shown in Fig. P12.23. Domain intervals of x_i are divided into $K_i = 3$ fuzzy sets. Therefore, there is a maximum of $K_1 \times K_2 = 9$ feasible rules. The output of the rth rule is expressed as

$$\hat{y}^{(r)} = a_j^{(r)} x_1 + b_k^{(r)} x_2$$

where $j, k = 1, 2, 3$; $a_1^{(r)} = 1$, $a_2^{(r)} = 2$ and $a_3^{(r)} = 3$ if x_1 is found to be $\underset{\sim}{A}_{11}$, $\underset{\sim}{A}_{12}$ and $\underset{\sim}{A}_{13}$, respectively; $b_1^{(r)} = 1$, $b_2^{(r)} = 2$, $b_3^{(r)} = 3$ if x_2 is found to be $\underset{\sim}{A}_{21}$, $\underset{\sim}{A}_{22}$ and $\underset{\sim}{A}_{23}$, respectively. Determine the output of the model if $x_1 = 6.0$ and $x_2 = 2.2$.

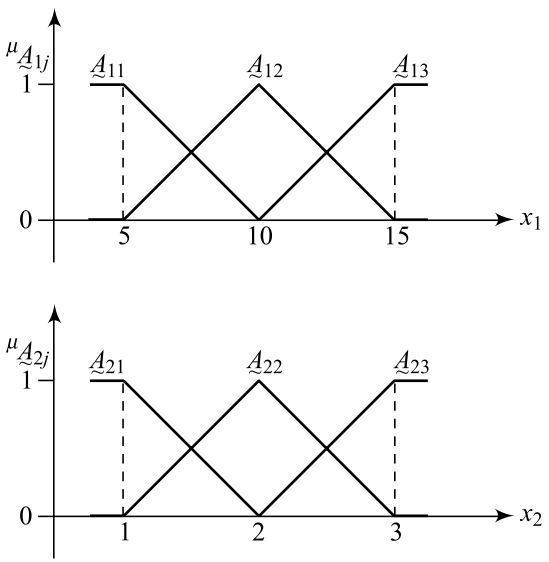


Fig. P12.23

Chapter 13

Optimization with Genetic Algorithms

13.1 EVOLUTIONARY ALGORITHMS

The adaptation of creatures to their environments results from the interaction of two processes: *evolution* and *learning*. Evolution is a slow stochastic process at the population level that determines the basic structures of a species. Evolution operates on biological entities, rather than on the individuals themselves. At the other end, learning is a process of gradually improving an individual's adaptation capability to its environment by tuning the structure of the individual.

Evolution is based on the Darwinian model, also called the *principal of natural selection* or *survival of the fittest*. All the living organisms have a specific genetic material containing information about them and allowing them to transfer their features to new generations. During reproduction, a new organism is created, which takes certain features after its parents, but also has certain features specific to itself. This organism starts to live in a given environment. If it turns out that it is well fit to the environment, it will transfer its genetic material to its offspring. The individual that is poorly fit to the environment will find it difficult to live in this environment and hence, transfer its genetic material to subsequent generations.

The presented idea has been applied to solve optimization problems. It turns out that an analogous approach to numerical calculations can be proposed using so-called *evolutionary algorithms*. The environment is defined on the basis of the problem to be solved. A population of individuals constituting potential solutions of a given problem lives in this environment. With the use of appropriately defined fitness function, we check to what extent they are adapted to the environment. Individuals exchange solutions (genetic material) and generate new solutions. Among the potential solutions, only the best-fit ones 'survive'. A family of evolutionary algorithms constitutes classical *genetic algorithms*, *evolution strategies, evolutionary programming*, and *genetic programming*.

Recently, more and more computational techniques inspired by biological adaptive systems (such as the collective behavior of animals and insects, as well as the immune systems of mammals) are emerging. The three well-known population-based optimization methods in this category are *particle swarm optimization,* the *immune algorithm*, and the *ant-colony optimization*. All these algorithms belong to a branch of *swarm intelligence*, an emergent collective intelligence of groups of simple agents. They are general optimization methods and can be used for discrete and continuous function optimization.

Our focus in this chapter is on genetic algorithm—an evolutionary algorithm which is both the simplest and the most general, for optimization. The application of genetic algorithm to the learning of neural networks as well as to the structural and parametric adaptations of fuzzy systems, will also be described.

13.2 GENETIC ALGORITHMS

13.2.1 Introduction

Biological evolution is an appealing source of inspiration for addressing optimization problems. Evolution is, in effect, a method of searching among an enormous number of possibilities for 'solutions'. In biology, the enormous set of possibilities is the set of possible genetic sequences, and the desired 'solutions' are highly fit organisms—organisms well able to survive and reproduce in their environments. Of course, the fitness of a biological organism depends on many factors—for example, how well it can weather the physical characteristics of the environment, and how well it can compete with, or cooperate with, the other organisms around it. The fitness criteria continually changes as creatures evolve; so evolution is searching a constantly changing set of possibilities. Searching for solutions in the face of changing conditions is precisely what is required for adaptive computer programs. Furthermore, evolution is a massively parallel search method: rather than work on one species at a time, evolution tests and changes millions of species in parallel. Finally, the 'rules' of evolution are remarkably simple: species evolve by means of random variation (via mutation, recombination and other parameters), followed by natural selection in which the fittest tend to survive and reproduce; thus, propagating their genetic material to future generations. Yet, these simple rules are thought to be responsible, in large part, for the extraordinary variety and complexity we see in the biosphere.

Knowledge of biological terminology, though not necessary, may help better appreciation of genetic algorithms. All living organisms consist of cells, and each cell contains the same set of one or more *chromosomes*—strings of DNA (deoxyribonucleic acid). A chromosome can be conceptually divided into *genes*—functional blocks of DNA, each of which encodes a particular protein. Very roughly, one can think of a gene as encoding a *trait*, such as eye color. The different possible 'settings' for a trait (e.g., blue, brown, hazel) are called *alleles*. Each gene is located at a particular *locus* (position) on the chromosome.

Many organisms have multiple chromosomes in each cell. The complete collection of genetic material (all chromosomes taken together) is called the organism's *genome*. The term *genotype* refers to the particular set of genes contained in a genome. The genotype gives rise, under foetus and later development, to the organism's *phenotype*—its physical and mental characteristics, such as eye color, height, brain size and intelligence.

Organisms whose chromosomes are arrayed in pairs are called *diploid*; organisms whose chromosomes are unpaired are called *haploid*. In nature, most sexually reproducing species are diploid, including human beings. In diploid sexual reproduction, *recombination* (or *crossover*) occurs: in each parent, genes are exchanged between each pair of chromosomes to form a *gamete* (a single chromosome), and then gametes from the two parents pair up to create a full set of diploid chromosomes. In haploid sexual reproduction, genes are exchanged between the two parents' single-strand chromosomes. Offsprings are subject to *mutation*, in which single nucleotides (elementary bits of DNA) are changed from the parents

to offsprings; mutation may cause the chromosomes of children to be different from those of their biological parents. The *fitness* of an organism is typically defined as the probability that the organism will live to reproduce (*viability*), or as a function of the number of offspring the organism has (*fertility*).

The basic idea of a genetic algorithm is very simple. The term *chromosome* typically refers to a candidate solution to a problem, typically stored as strings of binary digits (1s and 0s) in the computer's memory. The 'genes' are short blocks of adjacent bits that encode a particular element of the candidate solution (e.g., in the context of multiparameter function optimization, the bits encoding a particular parameter might be considered to be a gene). An 'allele' in a bit string, is either 0 or 1. Crossover typically consists of exchanging genetic material between two single-chromosome haploid parents. Mutation consists of flipping the bit at a randomly-chosen locus.

Most applications of genetic algorithms employ haploid individuals, particularly, single-chromosome individuals. The genotype of an individual, in a genetic algorithm using bit strings, is simply the configuration of bits in that individual's chromosome.

13.2.2 How are Genetic Algorithms Different from Traditional Methods?

The current literature identifies three main types of search methods: calculus-based, enumerative and random. Calculus-based methods have been studied extensively. These subdivide into two main classes: indirect and direct. Indirect methods seek local extrema by solving the usually nonlinear set of equations, resulting from setting the gradient of the objective function equal to zero. Given a smooth, unconstrained function, finding a possible peak starts by restricting search to those points with slopes of zero in all directions. On the other hand, direct (search) methods seek local optima by hopping on the function and moving in a direction related to the local gradient. This is simply the notion of *hill climbing*: to find the local best, climb the function in the steepest permissible direction.

Both the calculus-based methods are local in scope: the optima they seek are the best in a neighborhood of the current point. Clearly, starting the search procedures in the neighborhood of the lower peak will cause us to miss the main event (the higher peak). Furthermore, once the lower peak is reached, further improvement must be sought through random restart or other trickery. Another problem with calculus-based methods is that, they depend upon the existence of derivatives (well-defined slope values). Even if we allow numerical approximation of derivatives, this is a severe shortcoming. The real world of search is fraught with discontinuities and vast multimodal (i.e., consisting of many 'hills') noisy search spaces; methods depending upon restrictive requirements of continuity and derivative existence, are unsuitable for all, but a very limited, problem domain.

Enumerative schemes have been considered in many shapes and sizes. The idea is fairly straightforward: within a finite search space, the search algorithm starts looking at objective function values at every point in the space, one at a time. Although the simplicity of the type of algorithm is attractive, and enumeration is a very human kind of search, such schemes have applications wherein the number of possibilities is small. Even the highly touted enumerative scheme, *dynamic programming*, breaks down on problems of moderate size and complexity.

Random walks and random schemes that search and save the best, in the long run, can be expected to do no better than enumerative schemes. We must be careful to separate the strictly random search

methods from randomized techniques. The genetic algorithm is an example of a search procedure that uses random choice as a tool, to guide a highly exploitative search through a coding of parameter space. Using random choice as a tool in a directed search process seems strange at first, but nature contains many examples.

The traditional schemes have been used successfully in many applications; however, as more complex problems are attacked, other methods will be necessary. We shall soon see how genetic algorithms help attack complex problems [146].

The GA literature describes a large number of successful applications, but there are also many cases in which GAs perform poorly. Given a potential application, how do we know if a GA is a good method to use? There is no rigorous answer, though many researchers share the intuitions that if the space to be searched is large, is known not to be perfectly smooth and unimodal, or is not well understood; or if the fitness function is noisy; and if the task does not require a global optimum to be found—i.e., if quickly finding a sufficiently good solution is enough—a GA will have a good chance of being competitive or surpassing other methods. If the space is not large, it can be searched exhaustively by enumerative search methods, and one can be sure that the best possible solution has been found, whereas a GA might give only a 'good' solution. If the space is smooth and unimodal, a gradient ascent algorithm will be much more efficient than a GA. If the space is well understood, search methods using domain-specific heuristics can often be designed to outperform any general-purpose method such as a GA. If the fitness function is noisy, a one-candidate-solution-at-a-time search method such as simple hill climbing might be irrecoverably led astray by the noise; but GAs, since they work by accumulating fitness statistics over many generations, are thought to outperform robustly in the presence of small amounts of noise.

These intuitions, of course, do not rigorously predict when a GA will be an effective search procedure, competitive with other procedures. It would be useful to have a mathematical characterization of how the genetic algorithm works, that is, predictive. Research on this aspect of genetic algorithms has not yet produced definite answers.

13.2.3 Basics of Genetic Algorithms

Encoding

Simple genetic algorithms require the natural parameter set of the problem to be coded as a finite-length string of binary bits 0 and 1. For example, given a set of two-dimensional data ((x, y) data points), we want to fit a linear curve (straight line) through the data. To get a linear fit, we encode the parameter set for a line $y = \theta_1 x + \theta_2$, by creating independent bit strings for the two unknown constants θ_1 and θ_2 (parameter set describing the line) and then joining them (concatenating the strings). A bit string is a combination of 0s and 1s, which represents the value of a number in binary form. An n-bit string can accommodate all integers up to the value $2^n - 1$.

For problems that are solved by the genetic algorithm, it is usually known that the parameters, that are manipulated by the algorithm, will lie in a certain fixed range, say $\{\theta_{\min}, \theta_{\max}\}$. A bit string may then be mapped to the value of a parameter, say θ_i, by the mapping

$$\theta_i = \theta_{\min i} + \frac{b}{2^L - 1}(\theta_{\max i} - \theta_{\min i}) \tag{13.1}$$

where 'b' is the number in decimal form that is being represented in binary form (e.g., 152 may be represented in binary form as 10011000), L is the length of the bit string (i.e., the number of bits in each string), and $\theta_{\max}$ and $\theta_{\min}$ are user-specified constants, which depend on the problem in hand.

The length of the bit strings is based on the handling capacity of the computer being used, i.e., how long a string (strings of each parameter are concatenated to make one long string representing the whole parameter set) the computer can manipulate at an optimum speed.

Let us consider the data set in Table 13.1. For performing a line ($y = \theta_1 x + \theta_2$) fit, as mentioned earlier, we encode the parameter set (θ_1, θ_2) in the form of binary strings. We take the string length to be 12 bits. The first six bits encode the parameter θ_1, and the next six bits encode the parameter θ_2.

Table 13.1 Data set through which a line fit is required

Data number	*x*	*y*
1	1.0	1.0
2	2.0	2.0
3	4.0	4.0
4	6.0	6.0

The strings (000000, 000000) and (111111, 111111), represent the points ($\theta_{\min 1}$, $\theta_{\min 2}$) and ($\theta_{\max 1}$, $\theta_{\max 2}$), respectively, in the parameter space for the parameter set (θ_1, θ_2). Decoding of (000000) and (111111) to decimal form gives 0 and 63, respectively. However, problem specification may impose different values of minimum and maximum for θ_i. We assume that the minimum value to which we would expect θ_1 or θ_2 to go would be –2, and the maximum would be 5.

Therefore,

$$\theta_{\min i} = -2, \text{ and } \theta_{\max i} = 5$$

Consider a string (a concatenation of two substrings)

$$000111\ 010100 \tag{13.2}$$

representing a point in the parameter space for the set (θ_1, θ_2). The decimal value of the substring (000111) is 7 and that of (010100) is 20. This, however, does not give the value of the parameter set (θ_1, θ_2) corresponding to the string in (13.2). The mapping (13.1) gives the value:

$$\theta_1 = \theta_{\min 1} + \frac{b}{2^L - 1}(\theta_{\max 1} - \theta_{\min 1}) = -2 + \frac{7}{2^6 - 1}(5 - (-2)) = -1.22$$

$$\theta_2 = \theta_{\min 2} + \frac{b}{2^L - 1}(\theta_{\max 2} - \theta_{\min 2}) = -2 + \frac{20}{2^6 - 1}(5 - (-2)) = 0.22$$

Fitness Function

A fitness function takes a chromosome (binary string) as an input, and returns a number that is a measure of the chromosome's performance on the problem to be solved. Fitness function plays the same role in GAs as the environment plays in natural evolution. The interaction of an individual with its environment, provides a measure of fitness to reproduce. Similarly, the interaction of a chromosome with a fitness function, provides a measure of fitness that the GA uses when carrying out reproduction. Genetic algorithm is a maximization routine; the fitness function must be a non-negative figure of merit.

It is often necessary to map the underlying natural objective function to a fitness function form through one or more mappings. If the optimization problem is to minimize cost function $\bar{J}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the parameter set, then the following cost-to-fitness transformation may be used:

$$J(\boldsymbol{\theta}) = \frac{1}{\bar{J}(\boldsymbol{\theta}) + \varepsilon} \tag{13.3}$$

where ε is a small positive number. Maximization of J can be achieved by minimization of $\bar{J}$; so the desired effect is achieved.

Another way to define the fitness function is to let

$$J(\boldsymbol{\theta}(k)) = -\bar{J}(\boldsymbol{\theta}(k)) + \max_{\boldsymbol{\theta}(k)} \left\{ \bar{J}(\boldsymbol{\theta}(k)) \right\} \tag{13.4}$$

The minus sign in front of the $\bar{J}(\boldsymbol{\theta}(k))$ term turns the minimization problem into a maximization problem and $\max_{\boldsymbol{\theta}(k)} \left\{ \bar{J}(\boldsymbol{\theta}(k)) \right\}$ term is needed to shift the function up, so that $\bar{J}(\boldsymbol{\theta}(k))$ is always positive; k is the iteration index.

A fitness function can be any nonlinear, nondifferentiable, discontinuous, positive function because the algorithm only needs a fitness value assigned to each string.

For the problem in hand (fit a line through a given data set), let us choose a fitness function. Using decoded values of θ_1 and θ_2 of a chromosome, and the four data values of x given in Table 13.2, calculate

$$\hat{y}^{(p)} = \theta_1 x^{(p)} + \theta_2;\ p = 1, 2, 3, 4$$

These computed values of $\hat{y}^{(p)}$ are compared with the correct values $y^{(p)}$, given in Table 13.2, and square of errors in estimating the y's is calculated for each string. The summation of the square of errors is subtracted from a large number (400 in this problem) to convert the problem into a maximization problem:

$$J(\boldsymbol{\theta}) = 400 - \sum_{p} \left(\hat{y}^{(p)} - y^{(p)} \right)^2;\ \boldsymbol{\theta} = [\theta_1 \quad \theta_2] \tag{13.5}$$

The fitness value of the string (13.2) is calculated as follows:

$\theta_1 = -1.22,\ \theta_2 = 0.22$

For $x = 1.0,\ \hat{y}^{(1)} = \theta_1 x + \theta_2 = -1.00$

For $x = 2.0,\ \hat{y}^{(2)} = -2.22$

For $x = 4.0,\ \hat{y}^{(3)} = -4.66$

For $x = 6.0,\ \hat{y}^{(4)} = -7.10$

$$J(\boldsymbol{\theta}) = 400 - \sum_{p=1}^{4} \left(\hat{y}^{(p)} - y^{(p)}\right)^2 = 131.586$$

Initialization of Population

The basic element processed by a GA is the string formed by concatenating substrings, each of which is a binary coding of a parameter of the search space. If there are N decision variables in an optimization problem, and each decision variable is encoded as an n-digit binary number, then a chromosome is a string of $n \times N$ binary digits. We start with a randomly selected initial population of such chromosomes; each chromosome in the population represents a point in the search space, and hence, a possible solution to the problem. Each string is then decoded to obtain its fitness value, which determines the probability of the chromosome being acted on by genetic operators. The population then evolves, and a new generation is created through the application of genetic operators (The total number of strings included in a population, is kept unchanged throughout generations, for computational economy and efficiency). The new generation is expected to perform better than the previous generation (better fitness values). The new set of strings is again decoded and evaluated, and another generation is created using the basic genetic operators. This process is continued until convergence is achieved within a population.

Let $\theta^j(k)$ be a single parameter in chromosome j of generation k. Chromosome j is composed of N of these parameters:

$$\boldsymbol{\theta}^j(k) = [\theta_1{}^j(k),\ \theta_2{}^j(k),\ \ldots,\ \theta_N^j(k)] \tag{13.6}$$

The population of chromosomes, in generation k:

$$P(k) = \{\boldsymbol{\theta}^j(k) |\ j = 1, 2, ..., S\} \tag{13.7}$$

where S represents the number of chromosomes in the population. We want to pick S to be big enough, so that the population elements can cover the search space. However, we do not want S to be too big, since this increases the number of computations we have to perform.

For the problem in hand, Table 13.2 gives an initial population of 4 strings, the corresponding decoded values of θ_1 and θ_2, and the fitness value for each string.

Table 13.2 Initial population

String number	*String*	θ_1	θ_2	J
1	000111010100	–1.22	0.22	131.586
2	010010001100	0.00	–0.67	323.784
3	010101101010	0.33	2.67	392.41
4	100100001001	2.00	–1.00	365.00
			ΣJ	1212.8
			Av.J	303.2
			Max.J	392.41

Evolution occurs as we go from generation k to the next generation $k + 1$. Genetic operations of *selection*, *crossover* and *mutation* are used to produce a new generation.

Selection

Basically, according to Darwin, the most qualified (fittest) creatures survive to mate. Fitness is determined by a creature's ability to survive predators, pestilence, and other obstacles to adulthood and subsequent reproduction. In our unabashedly artificial setting, we quantify 'most qualified' via a chromosome's fitness $J(\boldsymbol{\theta}^j(k))$. The fitness function is the final arbiter of the string-creature's life or death. Selecting strings according to their fitness values means that the strings with a higher value have a higher probability of contributing one or more offspring in the next generation.

Selection is a process in which good-fit strings in the population are selected to form a mating pool, which we denote by

$$M(k) = \{\mathbf{m}^j(k) \mid j = 1, 2, ..., S\} \tag{13.8}$$

The mating pool is the set of chromosomes that are selected for mating. A chromosome is selected for mating pool according to the probability proportional to its fitness value. The probability for selecting the ith string is

$$p_i = \frac{J\left(\boldsymbol{\theta}^i(k)\right)}{\sum_{j=1}^{S} J\left(\boldsymbol{\theta}^j(k)\right)} \tag{13.9}$$

For the initial population of four strings in Table 13.2, the probability for selecting each string is calculated as follows:

$$p_1 = \frac{131.586}{131.586 + 323.784 + 392.41 + 365} = \frac{131.586}{1212.8} = 0.108$$

$$p_2 = \frac{323.784}{1212.8} = 0.267; \; p_3 = \frac{392.41}{1212.8} = 0.324; \; p_4 = \frac{365.00}{1212.8} = 0.301$$

To clarify the meaning of the formula and, hence, the selection strategy, Goldberg [146] uses the analogy of spinning a unit-circumference roulette wheel; the wheel is cut like a pie into S regions where the ith region is associated with the ith element of $P(k)$. Each pie-shaped region has a portion of the circumference that is given by p_i in Eqn. (13.9).

The roulette wheel for the problem in hand is shown in Fig. 13.1. String 1 has solution probability of 0.108. As a result, String 1 is given 10.8% slice of the roulette wheel. Similarly, String 2 is given 26.7% slice, String 3 is given 32.4% slice and String 4 is given 30.1% of the roulette wheel.

You spin the wheel, and if the pointer points at region i when the wheel stops, then you place $\boldsymbol{\theta}^i$ into the mating pool $M(k)$. You

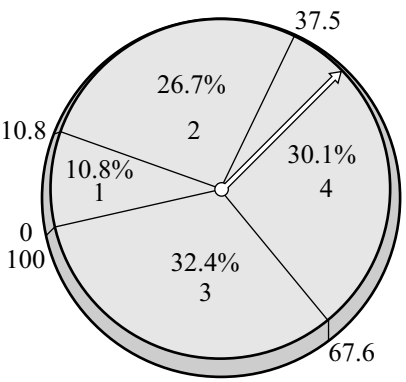


Fig. 13.1 Roulette Wheel

spin the wheel S times, so that S strings end up in the mating pool. Clearly, the strings which are more fit will end up with more copies in the mating pool; hence, chromosomes with larger-than-average fitness, will embody a greater portion of the next generation. At the same time, due to the probabilistic nature of the selection process, it is possible that some relatively unfit strings may end up in the mating pool.

For the problem in hand, the four spins might choose strings 3, 3, 4 and 2 as parents (String 1 also may be selected in the process of roulette wheel spin; it is just the luck of the draw. If the roulette wheel were spun many times, the average results would be closer to the expected values).

Reproduction Phase; Crossover

We think of crossover as mating in biological terms, which, at the fundamental biological level, involves the process of combining chromosomes. The crossover operation operates on the mating pool $M(k)$. First, specify the 'crossover probability' p_c (usually chosen to be near one, since, when mating occurs in biological systems, genetic material is swapped between the parents).

The procedure for crossover consists of the following steps:

(i) Randomly pair off the strings in the mating pool $M(k)$. If there are an odd number of strings in $M(k)$, then, for instance, simply take the last string and pair it off with another string which has already been paired off.

(ii) Consider chromosome pair $(\boldsymbol{\theta}^j, \boldsymbol{\theta}^i)$ that was formed in Step 1. Generate a random number $r \in [0, 1]$.

  (a) If $r < p_c$, then crossover $\boldsymbol{\theta}^j$ and $\boldsymbol{\theta}^i$. To crossover these chromosomes, select at random a 'cross site' and exchange all bits to the right of the cross site of one string, with those of the other. This process is pictured in Fig. 13.2. In this example, the cross site is position four on the string, and hence we swap the last eight bits between the two strings. Clearly, the cross site is a random number between one and the number of bits in the string, minus one.

  (b) If $r > p_c$, then the crossover will not take place; hence, we do not modify the strings.

(iii) Repeat Step 2 for each pair of strings that is in $M(k)$.

Fig. 13.2 Crossover operation example

For the problem in hand, Table 13.3 shows the power of crossover. The first column shows the four strings selected for mating pool. We randomly pair off the strings. Suppose that random choice of mates has selected the first string in the mating pool, to be mated with the fourth. With a cross site 4, the two strings cross and yield two new strings as shown in Table 13.3. The remaining two strings in the mating pool are crossed at site 9; the resulting strings are given in the table.

Table 13.3 Selection and crossover processes

String number	*Mating pool*	*Couples*	*After crossover*	θ_1	θ_2	J
3	010101 101010	0101 | 01 101010	010110 001100	0.44	– 0.67	370.574
3	010101 101010	0100 | 10 001100	010001 101010	–0.11	2.67	378.311
4	100100 001001	010101 101 | 010	010101 101001	0.33	2.56	392.794
2	010010 001100	100100 001 | 001	100100 001010	2.00	–0.89	362.972
					ΣJ	1504.7
					Av.J	376.163
					Max.J	392.794

In nature, an offspring inherits genes from both the parents. The crossover process creates children strings from the parent strings. The children strings thus produced, may or may not, have combination of good substrings from parents strings, but we don't worry about this too much, because if good strings are not created by crossover, they will not survive too long because of the selection operator. If good strings are created by crossover, there will be more copies of it in the next mating pool generated by the selection operator.

Besides the fact that crossover helps to model the mating part of the evolution process, why should the genetic algorithm perform crossover? Basically, the crossover operation perturbs the parameters near good positions to try to find better solutions to the optimization problem. It tends to help perform a localized search around the more fit strings (since, on average, the strings in the generation k mating pool are more fit than the ones in the generation k population).

Reproduction Phase; Mutation

Selection according to fitness, combined with crossover, gives genetic algorithms the bulk of their processing power. Mutation plays a secondary role in the operation of GAs. Mutation is needed because, occasionally, chromosomes may lose some potentially useful genetic material. In artificial genetic systems, mutation is realized by inverting a randomly chosen bit in a string. This is illustrated in Fig. 13.3.

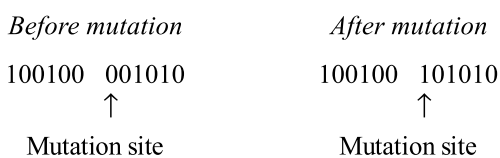


Fig. 13.3 Mutation operation example

Besides the fact that this helps to model mutation in a biological system, why should the genetic algorithm perform mutation? Basically, it provides random excursions into new parts of the search space. It is possible that we will get lucky and mutate to a good solution. It is the mechanism that tries to make sure that we do not get stuck at a local maxima, and that we seek to explore other areas of the search space to help find a global maximum for $J(\boldsymbol{\theta})$. Usually, the mutation probability p_m is chosen to be quite small (e.g., less than 0.01) since this will help guarantee that all the strings in the mating pool are not mutated so that any search progress that was made is lost (i.e., we keep it relatively low to avoid degradation to exhaustive search via a random walk in the search space).

After mutation, we get a modified mating pool $M(k)$. To form the next generation for the population, we let

$$P(k+1) = M(k) \tag{13.10}$$

where this $M(k)$ is the one that was formed by selection and modified by crossover and mutation. Then the above steps repeat, successive generations are produced, and we thereby model evolution (of course, it is a very crude model).

Terminal Conditions

While the biological evolutionary process continues, perhaps indefinitely, we would like to terminate our artificial one and find the following:

(1) The population string—say, $\boldsymbol{\theta}^*(k)$—that best maximizes the fitness function. Notice that, to determine this, we also need to know the generation number k where the most fit string existed (it is not necessarily in the last generation). A computer code, implementing the genetic algorithm, keeps track of the highest J value, and the generation number and string that achieved this value of J.
(2) The value of the fitness function $J(\boldsymbol{\theta}^*(k))$.

There is then the question of how to terminate the genetic algorithm. There are many ways to terminate a genetic algorithm, many of them similar to termination conditions used for conventional optimization algorithms. To introduce a few of these, let $\varepsilon > 0$ be a small number and $n_1 > 0$ and $n_2 > 0$ be integers. Consider the following options for terminating the GA:

(1) Stop the algorithm after generating the generation $P(n_1)$—that is, after n_1 generations.
(2) Stop the algorithm after at least n_2 generations have occurred and, at least n_1 steps have occurred when the maximum (or average) value of J for all population members has increased by no more than ε.
(3) Stop the algorithm once J takes on a value above some fixed value.

The above possibilities are easy to implement on a computer but, sometimes, you may want to watch the parameters evolve and decide yourself when to stop the algorithm.

Working Parameters

A set of parameters is predefined to guide the genetic algorithm, such as follows:

(1) the length of each decision variable encoded as a binary string;
(2) the number of chromosomes to be generated and operated in each generation, i.e., population size;
(3) the crossover probability p_c;
(4) the mutation probability p_m; and
(5) and the stopping criterion.

Example 13.1

Consider the problem of maximizing the function

$$J(\theta) = \theta^2 \tag{13.11}$$

where θ is permitted to vary between 0 and 31.

To use a GA, we must first code the decision variables of our problem as some finite length string. For this problem, we will code the variable θ simply as a binary unsigned integer of length 5. With a five-bit unsigned integer, we can obtain numbers between 0 (00000) and 31 (11111). The fitness function is simply defined as the function $J(\theta)$.

To start off, we select an initial population at random. We select a population of size 4. Table 13.4 gives the selected initial population, decoded θ values, and the fitness function values $J(\theta)$. As an illustration of the calculations done, let's take a look at the third string of the initial population, string 01000. Decoding this string gives $\theta = 8$, and the fitness $J(\theta) = 64$. Other θ and $J(\theta)$ values are obtained similarly.

The mating pool of the next generation may be selected by spinning a roulette wheel. Alternatively, the roulette-wheel technique may be implemented using a computer algorithm:

(i) Sum the fitness of all the population members, and call this result the total fitness ΣJ.
(ii) Generate r, a random number between 0 and total fitness.
(iii) Return the first population member whose fitness, added to the fitness of the preceding population members (running total), is greater than or equal to r.

We generate numbers randomly from the interval [0, 1170] (refer to Table 13.4). For each number, we choose the first chromosome for which the running total of fitness is greater than, or equal to, the random number. Four randomly generated numbers are 233, 9, 508, 967; String 1 and String 4 give one copy to the mating pool, String 2 gives two copies, and String 3 gives no copies.

With the above active pool of strings looking for mates, simple crossover proceeds in two steps: (1) strings are mated randomly, and (2) mated-strings couples crossover. We take the crossover probability $p_c = 1$. Looking at Table 13.5, we find that, random choice of mates has selected the second string in the mating pool to be mated with the first. With a crossing site of 4, the two strings 01101, and 11000 cross and yield two new strings, 01100 and 11001. The remaining two strings in the mating pool are crossed at site 2; the resulting strings are given in Table 13.5.

Table 13.4 Selection process

String number	*Initial population*	θ	$J(\theta)$	*Running total*
1	01101	13	169	169
2	11000	24	576	745
3	01000	8	64	809
4	10011	19	361	1170
		ΣJ	1170	
		Av.J	293	
		Max.J	576	

Table 13.5 Crossover process

Mating pool	*New population*	θ	$J(\theta)$
0110 ⋮ 1	01100	12	144
1100 ⋮ 0	11001	25	625
11 ⋮ 000	11011	27	729
10 ⋮ 011	10000	16	256
		ΣJ	1754
		Av. J	439
		Max. J	729

The last operator, mutation, is performed on a bit-by-bit basis. We assume that the probability of mutation in this test is 0.001. With 20 transferred bit positions, we should expect $20 \times 0.001 = 0.02$ bits to undergo mutation during a given generation. Simulation of this process indicates that no bits undergo mutation for this probability value. As a result, no bit positions are changed from 0 to 1, or vice versa, during this generation.

Following selection, crossover and mutation, the new population is ready to be tested. To do this, we simply decode the new strings created by the simple GA, and calculate the fitness function values from the θ values thus decoded. The results are shown in Table 13.5. While drawing concrete conclusions from a single trial of a stochastic process is, at best, a risky business, we start to see how GAs combine high-performance notions to achieve better performance. Both the maximal and average performance have improved in the new population. The population average fitness has improved from 293 to 439 in one generation. The maximum fitness has increased from 576 to 729 during that same period.

13.3 GENETIC-FUZZY SYSTEMS

Fuzzy inference systems (discussed in Chapter 12) are highly nonlinear systems with many input and output variables. The knowledge base for the design of these systems (refer to Fig.12.29) consists of data base (membership functions for input and output variables) and rule base. Crucial issues in the design are the tasks of selecting appropriate membership functions, and the generation of fuzzy rules. These tasks require experience and expertise. Genetic algorithms may be employed for

- tuning of membership functions, while the rule base remains unchanged;
- generating a rule base when a set of membership functions for input/output variables remains unchanged; or
- for both of these tasks simultaneously.

We will limit our presentation to the first task, i.e., tuning of membership functions while the rule base remains unchanged.

13.3.1 Tuning of Membership Functions

The effectiveness of the fuzzy system operation could be increased by appropriate tuning of the fuzzy sets. The GA modifies membership functions by changing the location of characteristic points of their shapes. The information on characteristic points of the membership functions is coded in chromosomes. After the appropriate representation of fuzzy sets in the chromosome has been selected, the GA operates on the population of individuals, i.e., on the population of chromosomes containing coded shapes of fuzzy membership functions, according to the genetic cycle comprising the following steps:

1. Decoding each of the individuals (chromosomes) of the population, recreating the set of membership functions, and constructing an appropriate fuzzy system. The rule base is predefined.
2. Evaluating the performance of the fuzzy system on the basis of the difference (error) between the system's responses and the desired values. This error defines the individual's (chromosome's) fitness.
3. Selection and application of genetic operators, such as crossover and mutation, and obtaining a new generation.

Example 13.2

Let us consider the application of GA to the fuzzy model of a manufacturing process, described in Problem 12.22. The process is characterized by two input variables, x_1 and x_2, and one output variable, y. The membership function distributions of the inputs and the output are shown in Fig. P12.22, and the predefined rule base is given in Table P12.22.

The membership functions have the shape of isosceles triangles, which may be described by means of characteristic points in the following manner: the vertices of the triangles are fixed, and the base-widths θ_1, θ_2, and θ_3 (refer to Fig. P12.22) are tunable. The ranges of the tunable parameters are assumed to be

$$2 \le \theta_1 \le 4;\ 5 \le \theta_2 \le 15;\ 0.5 \le \theta_3 \le 1.5 \tag{13.12}$$

Let us code these fuzzy sets in chromosomes by placing characteristic parameters one by one, next to each other (Fig.13.4). Starting from the leftmost position, L bits are assigned for parameter θ_1. Each of the parameters θ_1, θ_2, θ_3, may be assigned different number of bits depending on their ranges. However, for simplicity of presentation, we assign $L = 5$ in each of the three cases. Thus, the GA-string is 15 bits long.

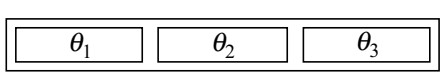


Fig. 13.4 Chromosome with encoded parameters

An initial population for the GA is created at random. We assume that the first chromosome of this randomly selected population is

$$10110 \quad 01101 \quad 11011 \tag{13.13}$$

The mapping rule (13.1) is used to determine the real values of the parameters θ_i; $i = 1, 2, 3$, represented by this string. The decoded value b of the binary substring 10110 is equal to 22. Therefore, the real value of θ_1 is given by (refer to Fig.P12.22, and parameter values (13.12))

$$\theta_1 = \theta_1^{\min} + \frac{b}{2^L - 1}\left(\theta_1^{\max} - \theta_1^{\min}\right) = 2 + \frac{22}{2^5 - 1}(4 - 2) = 3.419355$$

The real values of θ_2 and θ_3, corresponding to their respective substrings in (13.13), are 9.193548 and 1.370968, respectively. Figure 13.5 shows the modified membership distributions of input and output variables.

The GA optimizes the database (tunes the membership functions) with the help of a set of training examples. Assume that we are given P training examples $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1,2,\ldots,P\}$. Further, we take first training example ($p = 1$) as $\{x_1 = 10, x_2 = 28, y = 3.5\}$.

For the inputs $x_1 = 10, x_2 = 28$, we calculate the predicted value of the output, $\hat{y}$, of the fuzzy model when the model parameters are given by the first chromosome in the initial population. This is done using the procedure given in Section 12.4. This will give us the absolute value of error in prediction: $e_1 = |3.5 - \hat{y}|$.

From this procedure, repeated on all the training examples, we can obtain the average value of absolute errors in prediction,

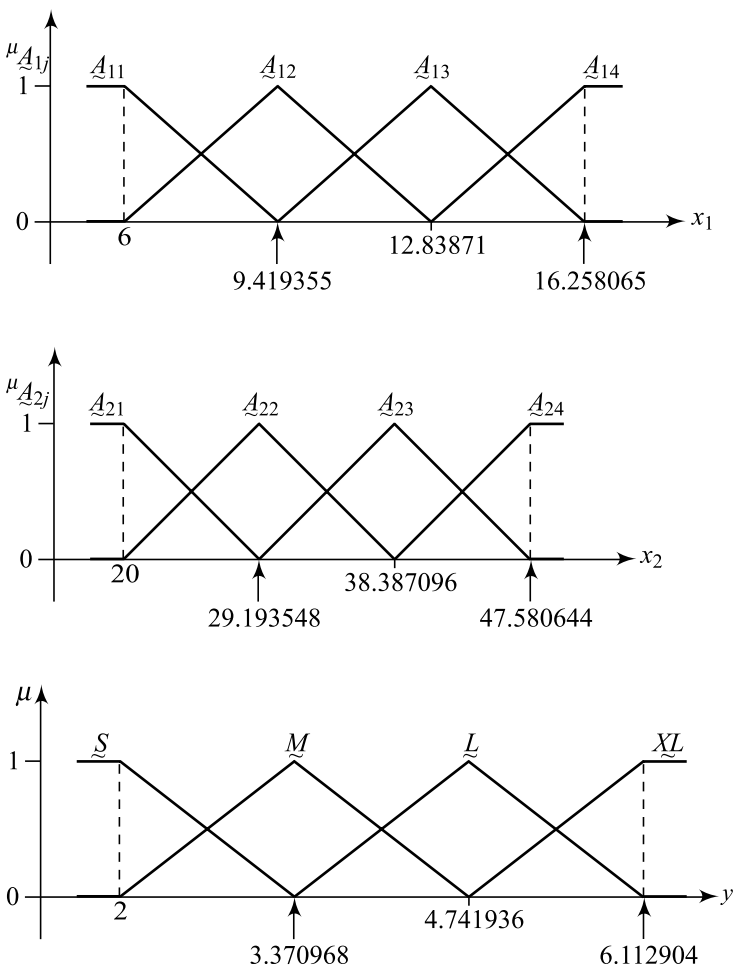


Fig. 13.5 Modified membership distributions of inputs and output

$$\bar{e} = \frac{1}{P}\sum_{p=1}^{P} e_p$$

Since GA is a maximization algorithm, we may choose the fitness function

$$J = \frac{1}{\bar{e} + \varepsilon}$$

where ε is a small positive number,

The population of GA-strings is then modified using different operators, such as selection, crossover and mutation, and after a few generations, the GA will be able to evolve an optimal fuzzy system.

13.4 GENETIC-NEURAL SYSTEMS

Neural-network learning is a search process for the minimization of a performance criterion (error function). In order to make use of existing learning algorithms, one needs to select a lot of parameters such as the number of layers, the number of units in each layer, the manner of their connection, the activation functions, as well as learning parameters. Learning process is usually carried out with the use of error backpropagation for connection weights, and trial-and-error approach for the other parameters. These design steps sometimes need quite a lot of time and experience, but genetic algorithms can be helpful here.

Genetic algorithms can be introduced into neural networks at many different levels:

- learning of connection weights including biases;
- determination of optimal architecture; or
- the simultaneous determination of architecture and weights.

We will limit our presentation to the first task, i.e., the use of genetic algorithms to the problems of optimization of neural network weights.

13.4.1 Optimization of Neural Network Weights

The gradient-based algorithms for learning weights of neural networks usually run multiple times to avoid local minima, and also gradient information must be available. Two of the most important arguments for the use of genetic algorithms to the problems of optimization of neural network weights, are

- a global search of space of weights, avoiding local minima; and
- useful for problems where obtaining gradient information is difficult or expensive.

It is important to mention that when gradient information is readily available, the gradient-based methods could be more effective in terms of computation speed, than the GA for weight optimization of neural networks. In fact, there is no clear winner in terms of the best training algorithm, since the best method is always problem dependent. The hybrid of genetic algorithm and gradient algorithm is an effective alternative.

With a fixed topology, the weights of a neural network are coded in a chromosome. Each individual of the population is determined by a total set of neural network weights. The order of placing the weights in the chromosome is arbitrary, but cannot be changed after the process of learning begins.

The fitness of individuals will be evaluated on the basis of the fitness function, defined as the sum of squares of errors, being the differences between the network desired signal and network output signal for different input data.

The genetic algorithm operates on the population of individuals (chromosomes representing neural networks with the same architecture but with different weights values) according to the typical genetic cycle comprising the following steps:

1. Decoding each individual of the current population to the set of weights and constructing the corresponding neural network with this set of weights; while the network architecture and the learning rule are predefined.
2. Calculating the total mean squared error of the difference between the desired signals and output signals for all the input data. This error determines the fitness of the individual (constructed network).
3. Selection and application of genetic operators, such as crossover and mutation, and obtaining a new generation.

REVIEW EXAMPLES

Review Example 13.1

Although applied in many complex industrial processes, fuzzy logic-based expert systems experience a deficiency in knowledge acquisition, and rely, to a great extent, on empirical and heuristic knowledge which, in many cases, cannot be elicited objectively. Fuzziness describes event ambiguity. It measures the degree to which an event occurs, not whether it occurs. Fuzzy controller design involves the determination of the linguistic state space, definition of membership grades of each linguistic term, and the derivation of the control rules. The information on the above aspects can be gathered by interviewing process operators, process knowledge experts, and other sources of domain knowledge and theory.

The choice of a FLC depends more on the intuition of the designer, and its effectiveness depends on the following parameters:

(i) Selection of rule set.
(ii) Number, shape and size of the membership functions of the input and output variables.
(iii) Value of the normalizing factors for the input variables to the FLC.
(iv) Value of the denormalizing factors for the output variables of the FLC.

Genetic Algorithm (GA) has a capability to guide in poorly understood, irregular spaces. In the following, we illustrate the use of GA in designing a FLC for the thermal system described in Review Example 11.2. We design FLC by tuning only the normalizing and denormalizing factors.

The proposed fuzzy controller has two input variables (refer to Figs 12.33):

$e(k)$ – error between set-point and actual temperature of the tank,

$v(k)$ – rate of change of error;

and one output variable:

$\Delta u(k)$ – incremental heat input to the tank.

The universe of discourse for all the three variables may be taken as [–1, 1]. Proposed membership functions are shown in Fig. 13.6.

The selected rules are as follows:

Rate of change of error→ Error ↓	*N*	*NZ*	*P*
N	*N*	*N*	*Z*
NZ	N	Z	P
P	*Z*	*P*	*P*

The initial value of the system output is Y_0. The initial velocity, and the initial output of the Fuzzy PI controller are set to zero.

The scaling factors *GE*, *GV* and *GU′* of the fuzzy controller may be tuned using genetic algorithm. Refer to Appendix B for realization of the controller.

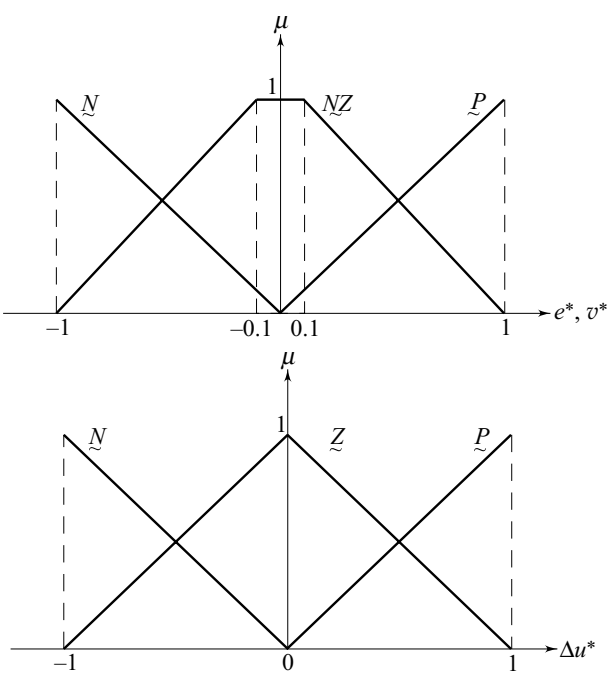


Fig. 13.6 Membership functions for the input and output variables of the fuzzy controller

Review Example 13.2

Problem P11.9 is concerned with optimization of the connection weights of the neural network shown in Fig. P11.9, using gradient algorithm. In the following, we describe how a binary-coded GA could be used (instead of gradient algorithm) to update the connection weights of this network.

One of the GA-strings is given below, in which five bits ($L = 5$) are used to represent each connection weight (all the weights are assumed to vary in the range 0.0 to 1.0):

$\{w_{11}\ w_{12}\ w_{21}\ w_{22}\ w_{31}\ w_{32}\ v_1\ v_2\ v_3\} =$

$$10110\ \ 01011\ \ 01101\ \ 11011\ \ 10001\ \ 00011\ \ 11001\ \ 11110\ \ 11101 \tag{13.14}$$

The parameter w_{11} is represented by the binary substring 10110. Its decoded value is $b = 22$. It varies in the range of $\{w_{11}^{\min}, w_{11}^{\max}\} = \{0.0, 1.0\}$. Using the mapping rule (13.1), its real value can be determined as follows:

$$w_{11} = w_{11}^{\min} + \frac{b}{2^L - 1}\left(w_{11}^{\max} - w_{11}^{\min}\right) = 0.0 + \frac{22}{2^5 - 1}(1.0 - 0.0) = 0.709677$$

Similarly, the real values of all the parameters represented by the GA-string (13.14) can be calculated. The real values are:

$\{w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}, v_1, v_2, v_3\} =$

$$\{0.709677, 0.354839, 0.419355, 0.870968, 0.548387, 0.096774, 0.806452,$$
$$0.967742, 0.935484\} \tag{13.15}$$

The first training pattern of the data $\{\mathbf{x}^{(p)}, y^{(p)}, p = 1,2,\ldots, P\}$ is assumed to be $\{x_1 = 0.6, x_2 = 0.7, y = 0.9\}$. The outputs of the hidden units for an input $\{x_1 = 0.6, x_2 = 0.7\}$ and the connection weights given by (13.15), are found as follows:

$$a_1 = 0.674194;\ a_2 = 0.861291;\ a_3 = 0.396774;\ z_1 = 0.662442;\ z_2 = 0.702930;\ z_3 = 0.597912$$

The activation value a of the neuron in the output layer is obtained as follows:

$$a = v_1 z_1 + v_2 z_2 + v_3 z_3 = 1.773820$$

and the predicted output of the network is

$$\hat{y} = \frac{e^a - e^{-a}}{e^a + e^{-a}} = 0.9440$$

Since the target output for this training pattern is equal to 0.9, the error in prediction is found to be equal to 0.0440.

A population of GA-strings represents a number of candidate neural networks. When the batch mode of training is adopted, the whole training data is passed through the neural network represented by a GA string. This gives Mean Square Error (MSE):

$$MSE = \frac{1}{P}\sum_{p=1}^{P}(y^{(p)} - \hat{y}^{(p)})^2 \tag{13.16}$$

Since GA is a maximization algorithm, we may choose the fitness function

$$J = \frac{1}{MSE + \varepsilon} \tag{13.17}$$

where ε is a small positive number.

The population of GA-strings is then modified using the selection, crossover, and mutation operators. The GA, through its search, is expected to evolve an optimal neural network.

PROBLEMS

13.1 The objective is to use GA to find the value of x that maximizes the function

$$f(x) = \sin(\pi x/256)$$

over the range $0 \le x \le 255$, where values of x are restricted to integers. The true solution to the problem is $x = 128$, having function value equal to one.

Explain the steps involved in GA. Use a random population of size 8, represent each individual in the population with an 8-bit binary string (8 strings is the population), choose fitness function

$$F(x) = f(x)/\Sigma f(x)$$

with summation over 8 strings, take crossover probability $p_c = 0.75$ and zero mutation probability. Show only one iteration by hand calculation.

13.2 The objective is to minimize the function:

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_2^2 + x_1 - 7)^2$$

in the interval $0 \le x_1, x_2 \le 6$. The true solution to the problem is $[3, 2]^T$ having a function value equal to zero.

Take up this problem to explain the steps involved in GA: maximizing the function

$$F(x_1, x_2) = \frac{1.0}{1.0 + f(x_1, x_2)}; \; 0 \le x_1, x_2 \le 6.$$

Step 1: Take 10 bits to code each variable. With 10 bits, what is the solution accuracy in the interval (0, 6)?

Step 2: Take population size equal to total string length, i.e., 20. Create a random population of strings.

Step 3: Consider the first string of the initial random population. Decode the two substrings and determine the corresponding parameter values. What is the fitness function value corresponding to each string? Similarly for other strings, calculate the fitness values.

Step 4: Select good strings in the population to form the mating pool.

Step 5: Perform crossover on random pairs of strings (the crossover probability is 0.8).

Step 6: Perform bitwise mutation with probability 0.05 for every bit.

The resulting population is the new population. This completes one iteration of GA and the generation count is incremented by 1.

13.3 A fuzzy logic-based expert system is to be developed that will work based on Sugano's architecture to predict the output of a process. The Data Base of the fuzzy system is shown in Fig.P13.3; x_1 and x_2 are two inputs with specified minimum values $x_1^{\min}$ and $x_2^{\min}$ respectively. The base-widths θ_1 and θ_2 are assumed to vary in the ranges:

$$0.8 \le \theta_1 \le 1.5;\ 4.0 \le \theta_2 \le 6.0$$

There is a maximum of $R = 4$ feasible rules; the output of rth rule ($r = 1, 2, \ldots, R$) is expressed as follows:

$$\hat{y}^{(r)} = a_0^{(r)} + a_1^{(r)} x_1 + a_2^{(r)} x_2$$

The parameters $a_0^{(r)}$, $a_1^{(r)}$, $a_2^{(r)}$ are assumed to vary in the range:

$$0.001 \le a_0^{(r)}, a_1^{(r)}, a_2^{(r)} \le 1.0$$

To optimize the performance of the fuzzy system using GA, a set of training examples $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1, \ldots, P\}$ is used. A typical GA-string in the population of solutions is of the form:

$$\{\theta_1\ \theta_2\ a_0^{(1)}\ a_1^{(1)} a_2^{(1)}\ a_0^{(2)}\ a_1^{(2)}\ a_2^{(2)}\ a_0^{(3)}\ a_1^{(3)}\ a_2^{(3)} a_0^{(4)}\ a_1^{(4)}\ a_2^{(4)}\}$$

with 4 binary bits assigned to represent each of the parameters.

Randomly select an initial population of solutions, and determine the deviation in prediction for the training example $\{\mathbf{x}^{(1)}, y^{(1)}\} = \{x_1^{(1)} = 1.1, x_2^{(1)} = 6.0, y^{(1)} = 5.0\}$ using the first GA-string.

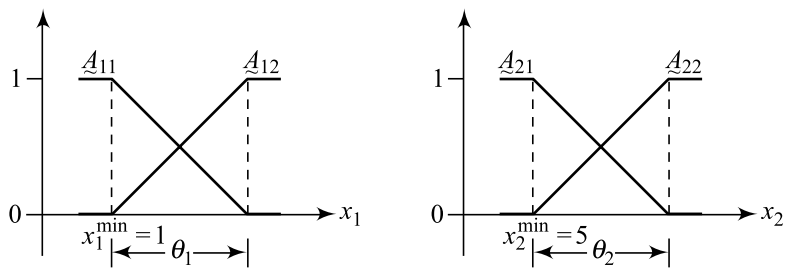


Fig. P13.3

13.4 A fuzzy logic-based expert system is to be developed that will work based on Mamdani's architecture to predict the output of a process. The Data Base of the fuzzy system is shown in Figs P13.3 and P13.4; x_1 and x_2 are two inputs with specified minimum values $x_1^{\min}$ and $x_2^{\min}$, respectively, and y is the output with specified minimum value $y^{\min}$. The basewidths θ_1, θ_2 and θ_3 of these isosceles triangles are tunable. The ranges of the tunable parameters are assumed to be

$$0.8 \le \theta_1 \le 1.5;\ 4.0 \le \theta_2 \le 6.0,;\ 0.5 \le \theta_3 \le 3$$

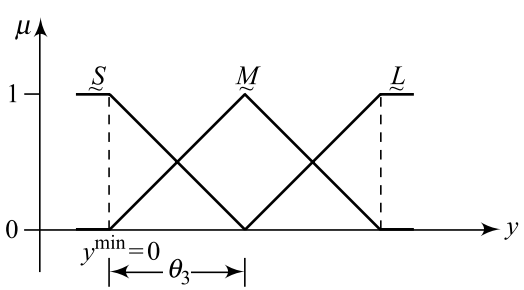


Fig. P13.4

The Rule Base of the fuzzy system is given in Table P13.4.

Table P13.4

$x_2 \rightarrow$ x_1 $\downarrow$	$\underset{\sim}{A}_{21}$	$\underset{\sim}{A}_{22}$
$\underset{\sim}{A}_{11}$	$\underset{\sim}{S}$	$\underset{\sim}{M}$
$\underset{\sim}{A}_{12}$	$\underset{\sim}{M}$	$\underset{\sim}{L}$

To optimize the performance of the fuzzy system using GA, a set of training examples $\{\mathbf{x}^{(p)}, y^{(p)}; p = 1,\ldots, P\}$ is used. A typical GA-string in the population of solutions is of the form

$$\{\theta_1\ \theta_2\ \theta_3\}$$

with 4 binary bits assigned to represent each of the parameters.

Randomly select an initial population of solutions, and determine the deviation in prediction for the training example $\{\mathbf{x}^{(1)}, y^{(1)}\} = \{x_1^{(1)} = 1.1, x_2^{(1)} = 6.0, y^{(1)} = 5.0\}$ using the first GA-string.

13.5 Reconsider the neural network shown in Fig. P11.9, modified to include the bias weights: w_{10}, w_{20} and w_{30}, for the hidden units and bias weight, v_0, for the output unit. All the bias weights vary in the range 0.0 to 1.0.

A binary-coded GA is used to update connection weights including biases. Extend the procedure given in Review Example 13.2 to this modified network.

Chapter 14

Intelligent Control with Reinforcement Learning

14.1 INTRODUCTION

Reinforcement learning is a machine intelligence approach that emphasizes on learning by the individual from direct interaction with its environment. This contrasts with classical approaches (discussed earlier in Chapters 11 and 12) to machine learning which have focused on learning from exemplary supervision or from expert knowledge of the environment. In this chapter, the coverage of reinforcement learning is to be regarded as an introduction to the subject; a springboard to advanced studies. The inclusion of the topic has been motivated by the observation that reinforcent learning control has the potential of solving many nonlinear control problems.

Reinforcement learning is based on the common sense idea that if an action is followed by a satisfactory state of affairs, or by an improvement in the state of affairs (as determined in some clearly defined way), then the tendency to produce that action is strengthened, i.e., *reinforced*. Extending this idea to allow action selections to depend on state information, introduces aspects of feedback. A reinforcement learning system is, thus, any system that through interaction with its environment *improves its performance* by receiving *feedback* in the form of a scalar reward (or penalty)—a *reinforcement signal*, that is commensurate with the appropriateness of the response. The learning system is not told which action to take, as in forms of machine learning discussed earlier in Chapters 11 and 12, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the *immediate reward* but also the next situation, and through that all subsequent rewards. These two characteristics—trial-and-error search and *cumulative reward*—are the two important distinguishing features of reinforcement learning. Although the system's initial performance may be poor, with enough interaction with the environment, it will eventually learn an effective strategy for maximizing cumulative reward.

Reinforcement learning is emerging as an important alternative to classical problem-solving approaches to intelligent control (Chapters 11 and 12), because it possesses many of the properties for intelligent control that classical approaches lack. Much of the classical intelligent control is an empirical science—the asymptotic effectiveness of the learning systems has been validated only empirically. Recent advances

relating reinforcement learning to dynamic programming are providing solid mathematical foundation; mathematical results that guarantee optimality in the limit for an important class of reinforcement learning systems, are now available [147].

Reinforcement learning systems do not depend upon models of the environment, because they learn through trial-and-error experience with the environment. However, when available, they can exploit this knowledge to determine a good initial control policy; this results in faster convergence to optimal policy.

The use of neural networks (or other associative memory structures such as fuzzy systems) makes reinforcement learning tractable on the realistic control problems with large state spaces. A neural network has the key feature of *generalization*; experience with a limited subset of state space is usefully generalized to produce a good approximation over a much larger subset. Intelligent control architectures incorporating aspects of both the reinforcement learning and the supervised learning, generalize from previously experienced states to ones that have never been experimented with. Empirical results based on such architectures, have shown robust, efficient learning on a variety of nonlinear control problems.

14.2 ELEMENTS OF REINFORCEMENT LEARNING CONTROL

Consider building a controller for stabilization of dynamic system; the controller has a set of sensors to observe the state of the dynamic system. In classical adaptive control strategies, the controller adjusts its behavior on-line, in real-time, to the changing properties of the controlled process, measured as a deviation of the actual process response from the process-model response. Reinforcement learning control is, in fact, a new adaptive control strategy wherein the controller's performance depends on a sequence of decisions made by experimenting with the controlled process (model not known *a priori*) and observing the consequences.

In a general formulation of reinforcement learning framework for solving *sequential decision problems*, we see the reinforcement learning problem as a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and the decision-maker is called an *agent*. The thing it interacts with, comprising everything outside the agent, is called the *environment*. These interact continually; the agent selecting *actions* and the environment responding to these actions and presenting new situations (*states* of the environment) to the agent. Figure 14.1 diagrams a generic agent *perceiving* its environment through *sensors* and *acting* upon that environment through *effectors*. Reinforcement learning is learning what to do—how to map states to actions—so as to maximize a numerical *reward*. The agent is not told which actions to take; it must instead discover which actions yield the most reward by trying them. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before. The agent has to *exploit* what it already knows by being *greedy* to maximize reward, but it has also to *explore* in order to make better action selections in the future. The dilemma is that neither exploitation nor exploration can be pursued exclusively without failing at the task. The agent must try a variety of actions and progressively favor those that appear to be most effective. Although the agent's initial performance may be poor, with enough interaction with the environment, it will eventually learn an effective *policy* for maximizing reward.

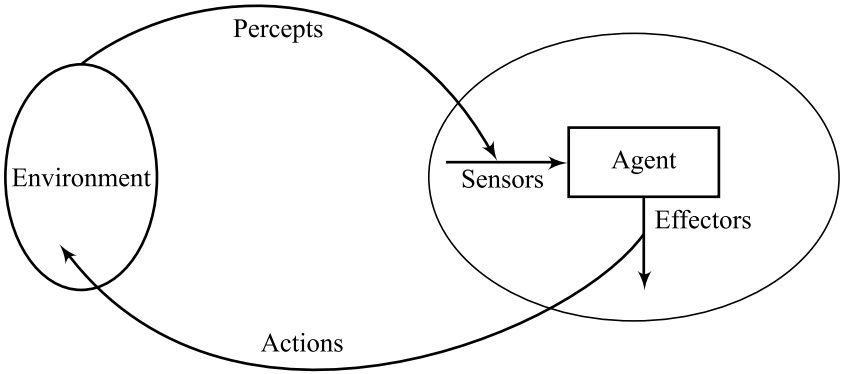


Fig. 14.1 A generic agent

Beyond the agent and the environment, one can identify four main sub-elements of a reinforcement learning system—a *policy*, a *reward function,* a *value function* and *horizon* of decisions. A *policy* defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. A *reward function* defines *immediate reward* for an action responsible for the current state of the environment. Roughly speaking, it maps states of the environment to a scalar, a *reward*, indicating the intrinsic desirability of the state. Whereas a reward function indicates what is good in the immediate sense, a *value function* specifies what is good in the long run. Roughly speaking, the *value* of a state is the *cumulative reward* an agent can expect to accumulate over the future as a result of sequence of its actions, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states. An agent's sole objective is to maximize the cumulative reward (value) it receives in the long run.

The value function depends on whether there is a *finite horizon* or an *infinite horizon* for decision making. A finite horizon means that there is a *fixed* time after which nothing matters—the game is over, so to speak. With a finite horizon, the optimal action for a given state could change over time. We say that the optimal policy for a finite horizon is *nonstationary*.

With no fixed time limit, on the other hand, there is no reason to behave differently in the same state at different times. Hence, the optimal action depends only on the current state, and the optimal policy is *stationary*. Polices for the infinite-horizon case are, therefore, simpler than those for finite-horizon case.

Note that 'infinite horizon' does not necessarily mean that all state sequences are infinite; it just means that there is no fixed deadline. If the environment contains *terminal states* and if the agent is guaranteed to get to one eventually, then we will never come across infinite sequences.

Our focus in this chapter is on reinforcement learning solutions to control problems. The controller (agent) has a set of sensors to observe the state of the controlled process (environment); the learning task is to learn a control strategy (policy) for choosing control signals (actions) that achieve minimization of a performance measure (maximization of cumulative reward).

In control problems, we minimize a performance measure; frequently referred to as *cost function*. The reinforcement learning control solution seeks to minimize the long-term accumulated cost the controller

incurs over the task time. The general reinforcement learning solution seeks to maximize the long-term accumulated reward the agent receives over the task time. Since in control problems, reference of optimality is a cost function, we assign *cost* to the reward structure of the reinforcement learning process; *the reinforcement learning solution then seeks to minimize the long-term accumulated cost the agent incurs over the task time*. The value function of the reinforcement learning process is accordingly defined with respect to cost structure.

The stabilizing control problems we have been discussing in this book, are all infinite-horizon problems. Here also, we will limit our discussion to this class of control problems.

Some reinforcement learning systems have one more element—a *model* of the environment. This is something that mimics the behavior of the environment. For example, given a state and action, the model might predict the resultant next state and next cost.

Early reinforcement learning systems were explicitly model-free, trial-and-error learners. Nevertheless, it gradually became clear that reinforcement learning methods are closely related to dynamic programming methods, which do use models. *Adaptive dynamic programming* has emerged as a solution method for reinforcement learning problems wherein the agent learns the models through trial-and-error interaction with the environment, and then uses these models in dynamic programming methods.

We have used the vector $\mathbf{x}$ to represent the *state* of a physical system: $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^T$, where x_i: $i = 1, ..., n$, are *state variables* of the system. State $\mathbf{x}$, a vector of real numbers, is a point in the state space. In reinforcement learning (RL) framework, we will represent the state by 's'; thus s is a point in the n-dimensional state space. Similarly, the vector $\mathbf{u}$ has been used for control. We will represent this by the action 'a' in our RL framework.

If the environment is *deterministic*, then an agent's action a will transit the state of the environment from s to s' deterministically; there is no probability involved. In fact, the transfer function models or state variable models, used in the book so far, for plants/controlled processes, are approximate models based on the assumption of deterministic behavior.

If the environment is *stochastic*, then transition of s to s' under action a will be different each time action a is applied in state s. This is captured by a probabilistic model. If the environment is deterministic, but uncertain, then also transition of s to s' under action a will not be unique each time action a is applied in state s. Since uncertainty in environments is the major issue leading to complexity of the control problem, we will be concerned with probabilistic models.

(1) A specification of the outcome probabilities for each admissible action in each possible state is called the *transition model*.

$P(s, a, s')$: probability of reaching state s' if action a is applied in state s.

(2) In control problems, the transitions are *Markovian*—the probability of reaching state s' from s depends only on s and not on the history of earlier states.

(3) In each state s, the agent receives a *reinforcement* $r(s)$, which measures the immediate cost of the action.

(4) The specification of a sequential decision problem for a fully observable environment, with a Markovian transition model and cost for each state, is called a *Markov Decision Process (MDP)*.

(5) The basis of our reinforcement learning framework is Markov decision processes.

14.3 METHODS FOR SOLVING THE REINFORCEMENT LEARNING PROBLEM

Dynamic programming is a well-known, general-purpose method to deal with complex systems to find optimal control strategies for nonlinear and stochastic dynamic systems. It addresses the problem of designing closed-loop policies off-line under the assumption that an accurate model of the stochastic dynamic system is available. The off-line design procedure typically yields a computationally efficient method for determining each action as a function of the observed system state.

There are two practical issues related to the use of dynamic programming:

(1) For many real-world problems, the number of possible states and admissible actions in each state are so large that the computational requirements of dynamic programming are overwhelming ('curse of dimensionality').
(2) Dynamic programming algorithms require accurate model of the dynamic system; this prior knowledge is not always available ('curse of modeling').

Over the past three decades, a focus of researchers has been to develop methods capable of finding high-quality approximate solutions to problems where exact solutions via classic dynamic programming are not attainable in practice due to high computational complexity and lack of accurate knowledge of system dynamics. In fact, *reinforcement learning* is a field that represents this stream of activities. All of the reinforcement learning can be viewed as attempts to achieve the same effect as dynamic programming, only with less computation and without assuming a perfect model of the dynamic system. By focusing computational effort along behavioral patterns of interactions with the environment, and by using function approximation (neural network) for generalization of experience to states not reached through interactions, reinforcement learning can be used on-line for problems with large state spaces and with lack of accurate knowledge of system dynamics.

There is a close relationship between reinforcement learning and using dynamic programming to solve sequential decision problems. In both, the environment is characterized by a set of states, a set of admissible actions, and a cost function. In both, the objective is to find a decision policy that minimizes the cumulative cost over time. There is an important difference though. When solving a sequential decision problem using dynamic programming, the agent (presumably the designer of the eventual control system) has a complete (albeit stochastic) model of the environment's behavior. Given this information, the agent can compute the optimal control policy with respect to the model, as will be outlined in the next section. In reinforcement learning, the set of states, and the set of admissible actions are known *a priori*, but the effects of action on the environment and on the cost is not known. Thus, the agent cannot compute an optimal policy *a priori* (off-line). Instead, the agent must learn an optimal policy by experimenting in the environment. Reinforcement learning system is, thus, an on-line system.

In an on-line learning system, the learner moves about the real environment and observes the results. In this case, our primary concern is usually the number of real-world actions that the agent must perform to converge to an acceptable policy (rather than the number of computational cycles, as in off-line learning). The reason is that in many practical domains, the costs in time and in dollars of performing actions in the external world dominate the computational costs.

On-line learning can be performed in two elementary ways:

(1) Temporal difference learning
(2) Adaptive dynamic programming

14.3.1 Temporal Difference Learning

If one had to identify an idea as central and novel to reinforcement learning, it would undoubtedly be *Temporal Difference* (TD) learning. Temporal difference learning can be thought of as a version of dynamic programming, with the difference that TD methods can learn on-line in real-time, from raw experience without a model of the environment's dynamics. TD methods do not assume complete knowledge of the environment; they require only *experience*—sample sequences of states, actions and costs from actual interaction with the environment. Learning from *actual* experience is striking because it requires no prior knowledge of the environment's dynamics, yet can obtain optimal behavior.

The principle advantage of dynamic programming is that, if a problem can be specified in terms of Markov decision process, then it can be analyzed and an optimal policy obtained *a priori*. The two principle disadvantages of dynamic programming are as follows: (1) for many tasks, it is difficult to specify the dynamic model; and (2) because dynamic programming determines a fixed control policy *a priori*, it does not provide a mechanism for adapting the policy to compensate for disturbances and/or modeling errors (nonstationary dynamics).

Reinforcement learning has complimentary advantages as follows: (1) it does not require a prior dynamical model of any kind, but learns on experience gained directly from the environment; and (2) to some degree, it can track the dynamics of nonstationary systems. The principle disadvantage of reinforcement learning is that, in general, many trials (repeated experiences) are required to learn an optimal control strategy, especially if the system starts with a poor initial policy.

This suggests that the respective weaknesses of these two approaches may be overcome by integrating them. That is, if a complete, possibly inaccurate, model of the task is available *a priori*, model-based methods (including dynamic programming) can be used to develop initial policy for a reinforcement learning system. A reasonable initial policy can substantially improve the system's initial performance and reduce the time required to reach an acceptable level of performance. Conversely, adding an adaptive reinforcement learning component to an otherwise model-based fixed controller, can compensate for an inaccurate model.

In this chapter, we limit our discussion to *naive* reinforcement learning systems.

14.3.2 Adaptive Dynamic Programming

An adaptive dynamic programming agent works by learning the transition model of the environment through interaction with the environment. It then plugs the transition model and the observed costs in the dynamic programming algorithm. Adaptive dynamic programming is, thus, an on-line learning system.

The process of learning the model itself is easy when the environment is fully observable. In the simplest case, we can represent the transition model as a table of probabilities. We keep track of how often each action outcome occurs, and estimate the transition probability $P(s, a, s')$ from the frequency with which state s' is reached when executing action a in state s.

Our focus in this chapter is on temporal difference learning. We begin with an introduction to dynamic programming, and then using this platform, develop temporal difference methods of learning.

14.4 BASICS OF DYNAMIC PROGRAMMING

We first define a general formulation of the problem of learning sequential control strategies. To do so, we consider building a learning controller for stabilization of an inverted pendulum (Fig. 5.16). The controller, or *agent*, has a set of sensors to observe the *state* of its *environment* (the dynamic system: inverted pendulum mounted on a cart). For example, a controller may have sensors to measure angular position θ and velocity $\dot{\theta}$ of the pendulum, and horizontal position z and velocity $\dot{z}$ of the cart; and actions implemented by applying a force of u newtons to the cart. Its task is to learn control strategy, or *policy*, for choosing actions that achieve its goals.

A common way of obtaining approximate solutions for continuous state and action tasks is to quantize the state and action spaces, and apply finite-state dynamic programming (DP) methods. The methods we explore later in this chapter make learning tractable on the realistic control problems with continuous state spaces (infinitely large set of quantized states).

Suppose that our stabilization problem demands that the pendulum must be kept within ± 12° from vertical, and the cart must be kept within ± 2.4m from the center of the track.

We define the following finite sets of possible states S and available actions A.

State →	1	2	3	4	5	6
Pend. angle(deg); θ	< -6	-6 to -1	-1 to 0	0 to 1	1 to 6	> 6
Pend. velocity; $\dot{\theta}$	< -50	-50 to 50	> 50			
Cart position(m); z	< -0.8	-0.8 to 0.8	> 0.8			
Cart velocity; $\dot{z}$	< -0.5	-0.5 to 0.5	> 0.5			

Actions →	1	2	3	4	5	6	7
Apply force of u newtons	–10	–6	–2	0	2	6	10

Define: $x_1, = \theta, x_2 = \dot{\theta}, x_3 = z, x_4 = \dot{z}$. Vector $\mathbf{x} = [x_1\ x_2\ x_3\ x_4]^T$ defines a point in the state space; the distinct point corresponding to $\mathbf{x}$ is the distinct state s of the environment (pendulum on a cart). Therefore, there are $6 \times 3 \times 3 \times 3 = 162$ distinct states: $s^{(1)}, s^{(2)},..., s^{(162)}$, of our environment. The finite set of states, in our learning problem, is thus given as

$$S : \{s^{(1)}, s^{(2)},..., s^{(162)}\}$$

The action set size is seven: $a^{(1)}, a^{(2)},..., a^{(7)}$. The finite set of available actions in our learning problem, is thus given as

$$A : \{a^{(1)}, a^{(2)},..., a^{(7)}\}$$

We assume the knowledge of *state transition model*:

$P(s, a, s')$: probability of reaching state s' if action a is applied in state s;

$$\text{for all } s \in S, \text{ and for all } a \in A$$

Note that our model is stochastic; it captures the uncertainties involved in the environment.

In each state s, the agent receives a *reinforcement* $r(s)$, which measures the immediate *cost* of action. For the particular inverted pendulum example, a cost of '–1' may be assigned to *failure states* ($\theta > 12°$; $\theta < -12°$), and a cost of '0' may be assigned to every other state. Note that cost structure for a learning problem is an important design parameter. It controls the convergence speed of a learning algorithm. The functions $P(\cdot)$ and $r(\cdot)$ are part of the environment and not necessarily known to the agent.

The specification of a sequential design problem for a fully observable (the agent knows where it is) environment with a Markovian decision model and cost for each state, is a *Markov Decision Process* (MDP). An MDP is defined by the tuple (S,A,P,r) where S is the set of possible states the environment can occupy; A is the set of admissible actions the agent may execute to change the state of the environment, P is the state transition probability, and r is the cost function. Usually S and A are distinct and finite; we assume that

$$S : \{s^{(1)}, s^{(2)}, ..., s^{(N)}\}; A : \{a^{(1)}, a^{(2)}, ..., a^{(M)}\}$$

where N represents the total number of distinct states of the environment, and M represents the total number of admissible actions in each state.

Let us now consider the structure of solution to the problem. Any fixed action sequence (open-loop structure) will not solve the problem because due to uncertainties in the behavior of the environment, the agent might end up in a failure state; i.e., the scheme lacks the robustness properties. Therefore, a solution must specify what the agent should do far *any* state that the environment might reach. The resulting feedback loop is a source of a measure of internal/external disturbances. A solution of this kind is called a *policy*. We usually denote a policy by π.

A stationary policy π for an MDP is a mapping $\pi : S \to \Omega(A)$, where $\Omega(A)$ is the set of all probability distributions over A. $\pi(a,s)$ stands for the probability that policy π chooses action a in state s. Since each action $a^{(1)}, a^{(2)}, ..., a^{(M)}$ is a candidate for state s, policy $\pi(a,s)$ for s is a set of action-selection probabilities associated with $a^{(1)}, ..., a^{(M)}$; their sum equals one.

A stationary deterministic policy π is a policy that commits to a single action choice per state, that is, a mapping $\pi : S \to A$ from states to actions. In this case, *$\pi(s)$ indicates the action that the agent takes in state s*. For every MDP, there exists an *optimal deterministic policy*, which minimizes the *expected, total discounted cost* (to be defined shortly) from any initial state. It is, therefore, sufficient to restrict the search for the optimal policy only within the space of deterministic policies.

The next question we must decide is how to calculate the *value of a state*. Recall that the value of a state is the *cumulative cost* an agent can expect to incur over the future as a result of sequence of its actions, starting from that state. A sequence of actions for a given task will force the environment through a sequence of states. Let us call it *environment trajectory* of a given task. In an infinite-horizon problem, the number of actions for a task is not fixed; therefore, number of distinct states in an environment

trajectory is not fixed. A typical state sequence in a trajectory may be expressed as $\{s_0, s_1, s_2,...\}$ where, each s_t; $t = 0,1,2,3,...$, could be any of the possible environment states $s^{(1)},..., s^{(N)}$.

Given the initial state s_t and the agent's policy π. The agent selects an action $\pi(s_t)$, and the result of this action is next state s_{t+1}. The state transition model, $P(s, a, s')$, gives a probability that the next state s_{t+1} will be $s' \in S$, given that the current state $s_t = s$ and the action $a_t = a$. Since each state $s^{(1)}, s^{(2)},..., s^{(N)}$ is a candidate to be the next state s', the environment simulator gives a set of probabilities: $P(s_t, a_t, s^{(1)}), ..., P(s_t, a_t, s^N)$; their sum equals one. Thus, a given policy π generates not one state sequence (environment trajectory), but a whole range of possible state sequences, each with a specific probability determined by the transition model of the environment.

The quality of a policy is, therefore, measured by the *expected value* (cumulative cost) of a state, where the expectation is taken over all possible state sequences that could occur. For MDPs, we can define the 'value of a state under policy π' formally as

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{t=0}^{\infty}\gamma^t r(s_t)\right\} \tag{14.1}$$

where $E_{\pi}\{\cdot\}$ denotes the expected value given that the agent follows policy π. This is a *discounted cost* value function; the *discount factor* γ is a number between 0 and 1($0 \le \gamma < 1$)

Note that

$$\sum_{t=0}^{\infty}\gamma^t r(s_t) \le \sum_{t=0}^{\infty}\gamma^t r_{\max} = r_{\max}/(1-\gamma)$$

Thus, the infinite sequence converges to a finite limit when costs are bounded and $\gamma < 1$.

The discount factor γ determines the relative value of delayed versus immediate costs. In particular, costs incurred t steps into the future are discounted exponentially by a factor of γ^t. Note that if we set $\gamma = 0$, only the immediate cost is considered. If we set γ closer to 1, future costs are given greater emphasis relative to the immediate cost. The meaning of γ substantially less than 1 is that future costs matter to us less than the costs paid at this present time. The discount factor is an important design parameter in reinforcement learning scheme.

The final step is to show how to choose between policies. An *optimal policy* is a policy that yields the highest expected value. We use π^* to denote an optimal policy.

$$\pi^* = \arg\min_{\pi} E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t)\right] \tag{14.2}$$

The 'arg min' notation denotes the values of π at which $E_{\pi}[\cdot]$ is minimized. $\pi^*(s)$ is, thus, a solution (obtained off-line) to the sequential decision problem. Given π^*, the agent decides what to do in real time by observing the current state s and executing the action $\pi^*(s)$. This is the simplest kind of agent, selecting fixed actions on the basis of the current state. A reinforcement learning agent, as we shall see shortly, is *adaptive*; it improves its policy on the basis of on-line, real-time interactions with the environment.

In the following we describe algorithms for finding optimal policies of the dynamic programming agent.

14.4.1 Finding Optimal Policies

The dynamic programming technique rests on a very simple idea known as the *principle of optimality* [105].

An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the previous decisions.

Consider a state sequence (environment trajectory) resulting from the execution of optimal policy π^*: $\{s_0, s_1, s_2,...\}$ where each $s_t: t = 0,1,2...$, could be any of the possible environment states $s^{(1)}, s^{(2)},..., s^{(N)}$. The index t represents *stages of decisions* in the sequential decision problem.

The dynamic programming algorithm expresses a generalization of the principle of optimality. It states that *the optimal value of a state is the immediate cost for that state plus the expected discounted optimal value of the next state, assuming that the agent chooses the optimal action*. That is, the optimal value of a state is given by

$$V^*(s) = r(s) + \gamma \min_a \sum_{s'} P(s,a,s')V^*(s') \tag{14.3}$$

This is one form of the *Bellman optimality equation* for V^*. For finite MDPs, this equation has a unique solution.

The Bellman optimality equation is actually a system of N simultaneous *nonlinear* equations in N unknowns, where N is the number of possible environment states. If the dynamics of the environment ($P(s,a,s')$) and the immediate costs underlying the decision process ($r(s)$) are known, then, in principle, one can solve this system of equations for V^* using any one of the variety of methods for solving systems of nonlinear equations. Once one has V^*, it is relatively easy to determine an optimal policy:

$$\pi^*(s) = \arg\min_a \sum_{s'} P(s,a,s')V^*(s') \tag{14.4}$$

Note that $V^*(s) = V^{\pi^*}(s)$:

$$V^*(s) = \min_\pi V^\pi(s) \quad \text{for all } s \in S \tag{14.5}$$

The solution of Bellman optimality equation (14.3) directly gives the values V^* of states with respect to optimal policy π^*. From this solution, one can obtain optimal policy using Eqn. (14.4).

Equation (14.5) suggests an alternative route to finding optimal policy π^*. It uses *Bellman equation for* V^π, given below.

$$V^\pi(s) = r(s) + \gamma \sum_{s'} P(s,\pi(s),s')V^\pi(s') \tag{14.6}$$

Note that this equation is a system of N simultaneous *linear* equations in N unknowns, where N is the number of possible environment states (Eqns (14.6) are same as Eqns (14.3) with 'min' operator removed). We can solve these equations for $V^\pi(s)$ by standard linear algebra methods.

Given an initial policy π_0, one can solve (14.6) for $V^{\pi_0}(s)$. Once we have V^{π_0}, we can obtain improved policy π_1, using the strategy given by Eqn. (14.4):

$$\pi_1(s) = \arg\min_a \sum_{s'} P(s,a,s')V^{\pi_0}(s') \tag{14.7}$$

The process is continued:

$$\pi_0 \to V^{\pi_0} \to \pi_1 \to V^{\pi_1} \to \pi_2 \to \cdots \to \pi^* \to V^*$$

Each policy is guaranteed to be a strict improvement over the previous one (unless it is already optimal). Because a finite MDP has only a finite number of policies, this process must converge to an optimal policy π^* and optimal value function V^* in a finite number of iterations.

Thus, given a complete and accurate model of MDP in the form of knowledge of the state transition probabilities $P(s, a,s')$ and immediate costs $r(s)$ for all states $s \in S$ and all actions $a \in A$, it is possible—at least in principle—to solve the decision problem off-line. There is one problem: the Bellman equations (14.3) are nonlinear because of the 'min' operator; solution of nonlinear equations is problematic. The Bellman equations (14.6) are linear and therefore, can be solved relatively quickly. For large state spaces, time might be prohibitive even in this relatively simpler case.

In the following, we describe basic forms of two dynamic programming algorithms: *value iteration* and *policy iteration*—a step towards answering the computational complexity problems of solving Bellman equations.

14.4.2 Value Iteration

As used for solving Markov decision problems, *value iteration* is a successive approximation procedure for solving the Bellman optimality equation (14.3), whose basic operation is 'backing up' estimates of optimal state values. We can solve Eqn. (14.3) using a simple iterative algorithm:

$$V_{(l+1)}(s) \leftarrow r(s) + \gamma \min_a \sum_{s'} P(s,a,s')V_l(s') \tag{14.8}$$

The algorithm begins with arbitrary guess $V_0(s)$ for each $s \in S$. The sequence of $V_1(s),V_2(s),\ldots$, is then obtained. The algorithm converges to the optimal values $V^*(s)$ as the number of iterations l approaches infinity (We use the index l for the stages of iteration algorithm, whereas we have used earlier the index t to denote the stages of decisions in the sequential decision problem). In practice, we stop once the value function changes by a small amount. Then a *greedy policy* (choosing the action with the lowest estimated cost) with respect to the optimal set of values is obtained as an optimal policy.

The computation (14.8) is done off-line, i.e., before the real system starts operating. An optimal policy, that is, an optimal choice of $a \in A$ for each $s \in S$, is computed either simultaneously with V^*, or in real time, using Eqn.(14.4).

A sequential implementation of iteration algorithm (14.8) requires temporary storage locations so that all the iteration-$(l + 1)$ values are computed based on the iteration-l values. The optimal values V^* are then stored in a *lookup table*. In addition to a problem of the memory needed for large tables, there is another problem of time needed to accurately fill them. If there are N states, and M is the largest number of admissible actions for any state, then each iteration which consists of backing up the value of each state exactly once requires about $M \times N^2$ operations. For the large state sets, typical in many control problems, it is difficult to try to complete even one iteration, let alone repeat the process until it converges to V^* (curse of dimensionality).

The iteration of *synchronous* DP algorithm defined in (14.8) *backs up* the value of every state once to produce the new approximate value function. We call this kind of operation as *full backup*; it is based on all possible next states rather than on a sample next state. We think of the backups as being done in a *sweep* through the state space.

Asynchronous DP algorithms are not organized in terms of systematic sweep of the entire set of states in each iteration. These algorithms backup the values of the states in any order whatsoever, using whatever values of other states happen to be available. The values of some states may be backed up several times before the values of others are backed up once. To converge correctly, however, an asynchronous algorithm must continue to back up the values of all the states.

Of course, avoiding sweeps does not necessarily mean that we can get away with less computation. It just means that our algorithm does not need to get locked into any hopelessly long sweep before it can make progress. We can try to take advantage of this flexibility by selecting the states to which we apply backups so as to improve the algorithm's rate of progress. We can try to order the backups to let value information propagate from state to state in an efficient way. Some states may not need their values backed up as often as other states. Some state orderings produce faster convergence than others, depending on the problem.

14.4.3 Policy Iteration

A *policy iteration algorithm* operates by alternating between two steps (the algorithm begins with arbitrary initial policy π_0).

(i) *Policy evaluation step*

Given the current policy π_k, we perform policy evaluation step that computes V^{π_k}(s) for all $s \in S$, as the solution of the linear system of equations (Bellman equation)

$$V^{\pi_k}(s) = r(s) + \gamma \sum_{s'} P(s, \pi_k(s), s') V^{\pi_k}(s') \tag{14.9}$$

in the N unknowns $V^{\pi_k}(s)$.

To solve these equations, an iteration procedure similar to the one used in value iteration algorithm (given by (14.8)) may be used.

$$V_{l+1}^{\pi_k}(s) \leftarrow r(s) + \gamma \sum_{s'} P(s, \pi_k(s), s') V_l^{\pi_k}(s') \tag{14.10}$$

(ii) *Policy improvement step*

Once we have V^{π_k}, we can obtain improved policy π_{k+1} (refer to Eqn.(14.7)) as follows:

$$\pi_{k+1}(s) = \arg\min_a \sum_{s'} P(s, a, s') V^{\pi_k}(s') \tag{14.11}$$

The two-step procedure is repeated with policy π_{k+1} used in place of π_k, unless we have $V^{\pi_{k+1}}(s) \approx V^{\pi_k}(s)$ for all s; in which case, the algorithm is terminated with optimal policy $\pi^* = \pi_k$.

Policy iteration algorithm can be viewed as an *actor–critic system*. In this interpretation, the policy evaluation step is viewed as the work of a *critic*, who evaluates the performance of the current policy

π_k, i.e., generates an estimate of the value function V^{π_k} from states and reinforcement supplied by the environment as inputs. The policy improvement step is viewed as the work of an *actor*, who takes into account the latest evaluation of the critic, i.e., the estimate of the value function, and acts out the improved policy π_{k+1}.

The algorithm we have described so far requires updating the values/policy for all states at once. It turns out that this is not strictly necessary. In fact, on each iteration, we can pick any subset of states and apply updating to that subset. This algorithm is called *asynchronous policy iteration*. Given certain conditions on the initial policy and value function, asynchronous policy iteration is guaranteed to converge to an optimal policy. The freedom to choose any states to work on means that we can design much more efficient heuristic algorithms—for example, algorithms that concentrate on updating the values of states that are likely to be reached by a good policy.

14.5 TEMPORAL DIFFERENCE LEARNING

The novel aspect of learning that we address now is that it assumes the agent does *not* have knowledge of $r(s)$ and $P(s,a,s')$, and therefore it cannot learn solely by simulating actions with environment model (off-line learning not possible). It has no choice but to interact with the environment and learn by observing consequences.

Figure 14.2 gives a general setting of the agent-environment interaction process. Time advances by discrete unit length quanta; $t = 0,1,2,\ldots$ At each time step t, the agent senses the current state $s_t \in S$ of the environment, chooses an action $a_t \in A$, and performs it. The environment responds by giving the agent a cost $r_t = r(s_t)$, and by producing the succeeding state $s_{t+1} \in S$.

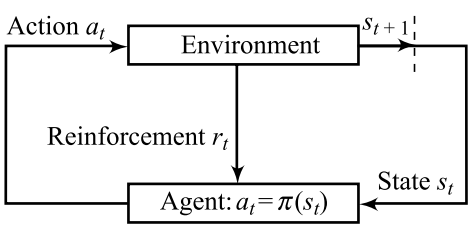


Fig. 14.2 The agent-environment interaction

The environment is stochastic in nature—each time the action a_t is applied in the state s_t, the succeeding state s_{t+1} could be any of the possible states in $S : s^{(1)}, s^{(2)},\ldots, s^{(N)}$. For the stochastic environment, the agent, however, explores in the space of deterministic policies (a deterministic optimal policy is known to exist for Markov decision process). Therefore, for each observed environment state s_t, the agent's policy suggests a deterministic action $a_t = \pi(s_t)$.

The task of the agent is to learn a policy $\pi : S \to A$ that produces the lowest possible cumulative cost over time (*greedy policy*). To state this requirement more precisely, the agent's task is to learn a policy π that minimizes the value V^{π} given by (14.1).

Reinforcement learning methods specify how the agent updates its policy as a result of its experience. The agent could use alternative methods for gaining experience and using it for improvement of its

policy. In the so called *Monte Carlo* method, the agent executes a set of *trials* in the environment using its current policy π. In each trial, the agent starts in state $s^{(i)}$ (any point $s^{(1)},\ldots,s^{(N)}$ of state space) and experiences a sequence of state transitions until at reaches a terminal state. In infinite-horizon discounted cost problems under consideration, terminal state corresponds to the *equilibrium state*. A learning episode (trial) is infinitely long, because the learning is continual. For the purpose of viewing the infinite-horizon problem in terms of episodic learning, we may define a stability region around the equilibrium point and say that the environment has terminated at a *success state* if the state continues to be in stability region for a prespecified time period (In a real-time control, any uncertainty (internal or external) will pull the system out of stability region and a new learning episode begins). Failure states (situations corresponding to 'the game is over and it is lost') if any, are also terminal states of the learning process.

In a learning episode, agent's percepts supply both the current state and the cost incurred in that state. Typical state sequences (environment trajectories) resulting from trials might look like this:

$$(1)\ \left(s^{(1)}\right)_{r(1)} \to \left(s^{(5)}\right)_{r(5)} \to \left(s^{(9)}\right)_{r(9)} \to \left(s^{(5)}\right)_{r(5)} \to \left(s^{(9)}\right)_{r(9)} \to \left(s^{(10)}\right)_{r(10)} \to$$
$$\to \left(s^{(11)}\right)_{r(11)} \to \left(s^{(SUCCESS)}\right)_{r(SUCCESS)}$$

$$(2)\ \left(s^{(1)}\right)_{r(1)} \to \left(s^{(5)}\right)_{r(5)} \to \left(s^{(9)}\right)_{r(9)} \to \left(s^{(10)}\right)_{r(10)} \to \left(s^{(11)}\right)_{r(11)} \to$$
$$\to \left(s^{(7)}\right)_{r(7)} \to \left(s^{(11)}\right)_{r(11)} \to \left(s^{(SUCCESS)}\right)_{r(SUCCESS)}$$

$$(3)\ \left(s^{(1)}\right)_{r(1)} \to \left(s^{(2)}\right)_{r(2)} \to \left(s^{(3)}\right)_{r(3)} \to \left(s^{(7)}\right)_{r(7)} \to \left(s^{(FAILURE)}\right)_{r(FAILURE)}$$

Note that each state percept is subscripted with the cost incurred. The objective is to use the information about costs to learn the expected value $V^{\pi}(s)$ associated with each state. The value is defined to be the expected sum of (discounted) costs incurred if policy π is followed (refer to Eqn.(14.2)).

When a nonterminal state is visited, its value is estimated based on what happens after that visit. Thus, the value of a state is the expected total cost from that state onward, and each trial (episode) provides *samples* of the value for each state visited. For example, the first trial in the set of three given above, provides one sample of value for state $s^{(1)}$:

(i) $r(1) + \gamma r(5) + \gamma^2 r(9) + \gamma^3 r(5) + \gamma^4 r(9) + \gamma^5 r(10) + \gamma^6 r(11) + \gamma^7 r(SUCCESS)$;

two samples of values for state $s^{(5)}$:

(i) $r(5) + \gamma r(9) + \gamma^2 r(5) + \gamma^3 r(9) + \gamma^4 r(10) + \gamma^5 r(11) + \gamma^6 r(SUCCESS)$;

(ii) $r(5) + \gamma r(9) + \gamma^2 r(10) + \gamma^3 r(11) + \gamma^4 r(SUCCESS)$;

two samples of values for state $s^{(9)}$:

(i) $r(9) + \gamma r(5) + \gamma^2 r(9) + \gamma^3 r(10) + \gamma^4 r(11) + \gamma^5 r(SUCCESS)$;

(ii) $r(9) + \gamma r(10) + \gamma^2 r(11) + \gamma^3 r(SUCCESS)$;

and so on.

Thus, at the end of each episode, the algorithm calculates the observed total cost for each state visited, and updates the estimated value for that state accordingly just by keeping a running average for each state in a table. In the limit of infinitely many trials, the sample average will converge to the true expectation of Eqn. (14.2).

The Monte Carlo method differs from dynamic programming in the following two ways:

(i) First, it operates on *sample experience*, and thus can be used for direct learning without a model.

(ii) Second, it does not build its value estimates for a state on the basis of estimates of the possible successor states (refer to Eqn. (14.6)); it must wait until the end of the trial to determine the update in value estimates of states. In dynamic programming methods, the value of each state equals its own cost plus the discounted expected value of its successor states.

The *Temporal Difference* (TD) learning methods combine the sampling of Monte Carlo, with the value estimation scheme of dynamic programming. TD methods update value estimates based on cost of one-step real-time transition and learned estimate of successor state, without waiting for the final outcome. Typically, when a transition occurs from state s to state s', we apply the following update to $V^{\pi}(s)$:

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \eta\,(r(s) + \gamma V^{\pi}(s') - V^{\pi}(s)) \tag{14.12}$$

where η is the learning parameter.

Because the update uses the difference in values between successive states, it is called the temporal-difference or TD equation, TD methods have an advantage over dynamic programming methods in that they do not require a model of the environment. Advantage of TD methods over Monte Carlo is that they are naturally implemented in an on-line fully incremental fashion. With Monte Carlo methods, one must wait until the end of a sequence, because only then is the value known, whereas with TD methods, one need only wait one time step.

Note that the update (14.12) is based on *one* state transition that just happens with a certain probability, whereas in (14.6), the value function is updated for all states simultaneously using all possible next states, weighted by their probabilities. This difference disappears when the effects of TD adjustments are averaged over a large number of transitions. The interaction with the environment can be repeated several times by restarting the experiment after success/failure state is reached. For one particular state, the next state and received reinforcement can be different each time the state is visited. Because the frequency of each successor in the set of transitions is approximately proportional to its probability, TD can be viewed as a crude but efficient approximation to dynamic programming.

The TD equation (14.12) is, in fact, approximation of policy-evaluation step of policy iteration algorithm of dynamic programming (refer to previous section for a recall), where the agent's policy is fixed and the task is to learn the values of states. This, as we have seen, can be done without a model of the system. However, improving the policy using (14.11) still requires the model.

One of the most important breakthroughs in reinforcement learning was the development of model-free TD control algorithm, known as Q-learning.

14.6 Q-LEARNING

In addition to recognizing the intrinsic relationship between reinforcement learning and dynamic programming, Watkins [148,150] has made an important contribution to reinforcement learning by suggesting a new algorithm called *Q-learning*. The significance of Q-learning is that when applied to a Markov decision process, can be shown to converge to the optimal policy, under appropriate conditions. Q-learning is the first reinforcement learning algorithm to be shown convergent to the optimal policy for decision problems involving cumulative cost.

The Q-learning method learns an *action–value* representation instead of learning value function. We will use the notation $Q(s,a)$ *to denote the value of doing action a in state s*. Q-function is directly related to value function as follows:

$$V(s) = \min_a Q(s,a) \tag{14.13}$$

Q-functions may seem like just another way of storing value information, but they have a very important property: *a TD agent that learns a Q-function does not need a model for either learning or action selection.* For this reason, Q-learning is called a *model-free* method.

The connections between Q-learning and dynamic programming are strong: Q-learning is motivated directly by value-iteration, and its convergence proof is based on a generalization of the convergence proof for value-iteration.

We can use the value-iteration algorithm (14.8) directly as an update equation for an iteration process that calculates exact Q-values, given an estimated model:

$$Q_{l+1}(s,a) \leftarrow r(s) + \gamma \sum_{s'} P(s,a,s') \left[\min_{a'} Q_l(s',a') \right] \tag{14.14}$$

It converges to the optimal Q-values, $Q^*(s,a)$.

Once one has $Q^*(s,a)$ for all $s \in S$ and all $a \in A$, it is relatively easy to determine an optimal policy:

$$\pi^*(s) = \arg\min_a Q^*(s,a) \tag{14.15}$$

This does, however, require that a model is given (or is learned (adaptive dynamic programming)), because Eqn. (14.14) uses $P(s, a, s')$.

The temporal-difference approach, on the other hand, requires no model. The update equation for TD Q-learning is (refer to Eqn. (14.12))

$$Q(s,a) \leftarrow Q(s,a) + \eta \left(r(s) + \gamma \left[\min_{a'} Q(s',a') \right] - Q(s,a) \right) \tag{14.16}$$

which is calculated whenever action a is executed in state s leading to s'.

The Q-learning algorithm (14.16) backs up the Q-value for only a single state-action pair at each time step of control, where the state-action pair consists of the observed current state and the action actually executed. Specifically, assume that at time step t in real-time control, the agent observes state s_t and has available the estimated Q-values produced by all the preceding stages of real-time Q-learning (the estimates stored in a *lookup table* with one entry for each state-action pair). We denote these estimates by $Q_t(s,a)$ for all admissible state-action pairs. The agent selects an action $a_t \in A$ using this information available in lookup table:

$$a_t = \arg\min_a Q_t(s_t, a)$$

After executing a_t, the agent receives the immediate cost $r_t = r(s_t)$ while the environment state changes to s_{t+1}. The Q-values in the lookup table are then updated as follows.

For the state-action pair (s_t, a_t):

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \eta_t \left[r_t + \gamma \left(\min_a Q_t(s_{t+1}, a) \right) - Q_t(s_t, a_t) \right] \tag{14.17a}$$

For other admissible state-action pairs, the Q-values remain unchanged:

$$Q_{t+1}(s, a) = Q_t(s, a) \; \forall \; (s, a) \neq (s_t, a_t) \tag{14.17b}$$

Watkins [148, 150] has shown that the Q-learning system that

(1) decreases its learning parameter at an appropriate rate (e.g., $\eta_t = 1/t^{\beta}$, where $0.5 < \beta < 1$); and
(2) visits each state-action pair infinitely often, is guaranteed to converge to an optimal policy.

Convergence, thus, requires that *the agent selects actions in such a fashion that it visits every possible state-action pair infinitely enough*. By this we mean that if action a is an admissible action from state s, then over time the agent must execute action a from state s repeatedly with nonzero frequency as the length of its action sequence approaches infinity.

In the Q-learning algorithm given by Eqns (14.17), the strategy for the agent in state s_t at time step t is to select the action a that minimizes $Q_t(s_t,a)$, thereby exploiting the current approximation of Q^* by following a *greedy policy*. However, with this strategy, the agent runs the risk that it will over commit to actions that are found during early stages to have low Q-values, while failing to *explore* other actions that can have even lower values. In fact, the convergence condition requires that each state-action transition occurs infinitely often. This will clearly not happen if the agent always follows the greedy policy.

The Q-learning agent must, therefore, follow the policy of *exploration* and *exploitation*: exploration ensures that all admissible state-action pairs are visited enough to satisfy the Q-learning convergence condition, and exploitation seeks to minimize the cost by following a greedy policy.

Many exploration schemes have been used in the RL literature. The simplest one is to behave greedily most of the time, but every once a while, say with small probability ε, instead select an action at random, uniformly, independently of the action-value estimates. We call methods using this near-greedy action selection rule *ε-greedy methods*.

14.6.1 Generalization

We have so far assumed that the Q-values learned by the agent are represented in a tabular form with one entry for each state-action pair. This is a particularly clear and instructive case, but of course, it is limited to tasks with small numbers of states and actions. The problem is not just the memory needed for large tables, but the computational time needed to experience all the state-action pairs for generation of data to accurately fill the tables.

Very few decision and control problems in the real world fit into *lookup table representation* strategy for solution; the number of possible states and actions in the real world is often much too large to accommodate the computational and storage requirements. The problem is more severe when state/action spaces include continuous variables—to use a table, they should be discretized to finite size, which may cause errors. The only way to learn anything at all on these tasks is to *generalize* from previously experienced states to ones that have never been seen. In other words, experience with a limited subset of state space be usefully generalized to produce a good approximation over a much larger subset.

Fortunately, generalization from examples has already been extensively studied, and we do not need to invent totally new methods for use in Q-learning. To a large extent, we need only combine Q-learning with off-the-shelf architectures for inductive generalization—often called *function approximation* because it takes examples from desired Q-function and attempts to generalize from them to construct

an approximation of the entire function. Function approximation is an instance of *supervised learning* (discussed in Chapters 11 and 12). In principle, any of the methods studied in this field can be used in Q-learning.

In parametric methods, the tabular (exact) representation of the real-valued functions $Q(s,a)$ is replaced by a generic parametric function approximator $\hat{Q}(s, a; \mathbf{w})$ where $\mathbf{w}$ are the adjustable parameters of the approximator. Learning $Q(s,a)$ for all $s \in S$ and $a \in A$ amounts to learning parameters $\mathbf{w}$ of $\hat{Q}(s, a; \mathbf{w})$ The new version of Q-learning equation (14.16) is

$$w_i \leftarrow w_i + \eta\left(r(s) + \gamma\left[\min_{a'} Q(s', a'; \mathbf{w})\right] - Q(s, a; \mathbf{w})\right)\frac{\partial Q(s, a, \mathbf{w})}{\partial w_i} \tag{14.18}$$

This update rule can be shown to converge to the closest possible approximation to the true function when the function approximator is *linear* in the parameters.

Unfortunately, all bets are off when *nonlinear* functions—such as neural networks–are used. For many tasks, Q-learning fails to converge once a nonlinear function approximator is introduced. Fortunately, however, the algorithm does converge for large number of applications. The theory of Q-learning with nonlinear function approximator still contains many open questions; at present it remains an empirical science.

For Q-learning, it makes more sense to use an incremental learning algorithm that updates the parameters of function approximator after each trial. Alternatively, examples may be collected to form a training set and learned in batch mode, but it slows down learning as no learning happens while a sufficiently large sample is being collected.

We give an example of *neural Q-learning*. Let $\hat{Q}_t(s, a; \mathbf{w})$denote the approximation to $Q_t(s,a)$ for all admissible state-action pairs, computed by means of a neural network at time step t. The state s is input to the neural network with parameter vector $\mathbf{w}$ producing the output $\hat{Q}_t(s, a; \mathbf{w}) \forall\ a \in A$. We assume that the agent uses the training rule of (14.17) after initialization of $\hat{Q}(s, a; \mathbf{w})$ with arbitrary finite values of $\mathbf{w}$.

Treating the expression inside the square bracket in (14.17a) as the error signal involved in updating the current value of parameter vector $\mathbf{w}$, we may identify the target (desired) value of $\hat{Q}_t$ at time step t as

$$\hat{Q}_t^{\text{target}}(s_t, a_t; \mathbf{w}) = r_t + \gamma\left(\min_{a} Q_t(s_{t+1}, a; \mathbf{w})\right) \tag{14.19}$$

At each iteration of the algorithm, the weight vector $\mathbf{w}$ of the neural network is changed slightly in a way that brings the output $\hat{Q}_t(s_t, a_t; \mathbf{w})$ closer to the target $\hat{Q}_t^{\text{target}}(s_t, a_t; \mathbf{w})$for the current (s_t,a_t) pair. For other state-action pairs, Q-values remain unchanged (Eqn. (14.17b)).

14.7 SARSA-LEARNING

The Q-learning algorithm, described in the previous section, is an *off-policy* TD method: the learned action-value function Q directly approximates Q^*, the optimal action-value function, independent of the policy being followed; optimal action for state s is then obtained from Q^*. The Q-learning is motivated by value iteration algorithm in dynamic programming.

The alternative approach, motivated by policy iteration algorithm in dynamic programming, is an *on-policy* TD method. The distinguishing feature of this method is that it attempts to evaluate and improve the same policy that it uses to make decisions.

In Section 14.5 on TD learning, we considered transitions from state to state and learned the value of states (Eqn. (14.12)) when following a policy π. The relationship between states and state-action pairs is symmetrical. Now we consider transitions from state-action pair to state-action pair and learn the value of state-action pairs, following a policy π. In particular, for on-policy TD method, we must estimate $Q^{\pi}(s, a)$ for the current policy π and for all states $s \in S$ and actions $a \in A$. We can learn Q^{π} using essentially the same TD method used in Eqn. (14.12) for learning V^{π}:

$$Q^{\pi}(s,a) \leftarrow Q^{\pi}(s,a) + \eta\Big(r(s) + \gamma Q^{\pi}(s',a') - Q^{\pi}(s,a)\Big) \tag{14.20}$$

where a' is the action executed in state s'.

This rule uses every element of the quintuple of events, (s, a, r, s', a'), that make up a transition from one state-action pair to the next. This quintuple (State-Action-Reinforcement-State-Action) gives rise to the name SARSA for this algorithm. Unlike Q-learning, here the agent's policy does matter. Once we have $Q^{\pi}(s,a)$, improved policy can be obtained as follows:

$$\pi_{k+1}(s) = \arg\min_{a} Q^{\pi}(s,a) \tag{14.21}$$

Since tabular (exact) representation is impractical for large state and action spaces, function approximation methods are used. Approximations in the policy-iteration framework can be introduced at the following two places:

(i) *The representation of the Q-function*: The tabular representation of the real-valued function $Q^{\pi}(s,a)$ is replaced by a generic parametric function approximation $\hat{Q}^{\pi}(s, a; \mathbf{w})$ when $\mathbf{w}$ are the adjustable parameters of the approximator.

(ii) *The representation of the policy*: The tabular representation of the policy $\pi(s)$ is replaced by a parametric representation $\hat{\pi}(s; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ are the adjustable parameters of the representation

The difficulty involved in use of these approximate methods within policy iteration is that the off-the-shelf architectures and parameter adjustment methods cannot be applied blindly; they have to be fully integrated into the policy-iteration framework.

References

APPLICATIONS

1. Siouris, G.M.; *Missile Guidance and Control Systems*; New York: Springer-Verlag, 2004.
2. Lawrence, A.; *Modern Inertial Technology: Navigation, Guidance, and Control*; 2nd Edition, New York: Springer-Verlag, 1998.
3. Leonhard, W.; *Control of Electric Drives*; 3rd Edition, New York: Springer-Verlag, 2001.
4. Bose, B.K.; *Modern Power Electronics and AC Drives*; Englewood Cliffs, NJ : Prentice-Hall, 2001.
5. Sen, P.C.; *Thyristor DC Drives*; 2nd Edition, New York: Wiley-Interscience, 1991.
6. Seborg, D.E., T.F. Edgar, and D.A. Mellichamp; *Process Dynamics and Control*; 2nd Edition, New York: John Wiley, 2004.
7 Lee, J., W. Cho, and T.F.Edgar, *An Improved Technique for PID Controller Tuning from Closed-Loop Tests*, *AICHE J*., Vol. 36, No. 36, No. 12,pp.1891–1895, 1990.
8 Ziegler, J.G., and N.B. Nichols, *Optimum Settings for Automatic Controllers*, *Trans*. *ASME*, Vol. 64, pp. 759–768, 1942.
9 Perry, R.H., and D. Green (ed.), *Perry's Chemical Engineering Handbook*, 6th Edition, New York: McGraw-Hill, 1997.
10. Shinskey, F.G.; *Process Control Systems*; 4th Edition, New York: McGraw-Hill, 1996.
11. Astrom, K.J., and T. Hagglund; *PID Controllers: Theory, Design, and Tuning*; 2nd Edition, Seattle, WA: International Society for Measurement and Control, 1995.
12. Corripio, A.B.; *Tuning of Industrial Control Systems*; Research Tringle Park, North Carolina: Instrument Society of America, 1990.
13. Stephanopoulos, G.; *Chemical Process Control—An Introduction to Theory and Practice*; Englewood Cliffs, NJ: Prentice-Hall, 1984.
14. Steven, B., and F. Lewis; *Aircraft Control and Simulation*; New York: Wiley-Interscience, 2003.
15. Nelson, R.C.; *Flight Stability and Automatic Control*; 2nd Edition, New York: McGraw-Hill, 1997.
16. Etkin, B., and L.D. Reid; *Dynamics of Flight: Stability and Control*; 3rd Edition, New York: John Wiley, 1996.
17. Craig, J.J.; *Introduction to Robotics*: *Mechanics and Control*; 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall, 2003.
18. Koivo, A.J.; *Fundamental for Control of Robotics Manipulators*; New York: John Wiley, 1989.
19. Asada, H., and K. Youcef-Toumi; *Direct Drive Robots: Theory and Practice*; Cambridge, Massachusetts: The MIT Press, 1987.

20. Valentino, J.V., and J. Goldenberg; *Introduction to Computer Numerical Control* (CNC) 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall, 2002.
21. Groover, M.K; *Automation, Production Systems*, *and Computer-Integrated Manufacturing*; 2nd Edition, Englewood Cliffs, NJ: Prentice-Hall, 2000.
22. Olsson, G., and G. Piani; *Computer Systems for Automation and Control*; London: Prentice-Hall International, 1992.
23. Hughes, T.A.; *Programmable Controllers*; 4th Edition, North Caroline: The Instrumentation, Systems, and Automation Society, 2004.
24. Petruzella, F.D.; *Programmable Logic Controllers*; 3rd Edition, New York: McGraw-Hill, 2004.
25. Bolton, W.; *Programmable Logic Controllers*; 3rd Edition, Buslingfon, MA: Newnes Publication, 2003.
26. Beards, C.F.; *Vibration Analysis and Control System Dynamics*; 2nd Edition, Englewood Cliffs, NJ: Prentice-Hall, 1995.

MATHEMATICAL BACKGROUND

27. Noble, B, and J.W. Daniel; *Applied Linear Algebra*; 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall 1988.
28. Lancaster, P., and M. Tismenetsky; *The Theory of Matrices*; 2nd Edition, Orlando, Florida: Academic Press, 1985.
29. Lathi, B.P.; *Linear Systems and Signals*; 2nd Edition, New York; Oxford University Press, 2005.
30. Oppenheim, A.V., R.W. Shafer, and J.R. Buck; *Discrete-Time Signal Processing*; 2nd Edition, Englewood Cliffs, NJ: Prentice-Hall, 1999.
31. Oppenheim, A.V., A.S. Willsky and S. Hamid Nawab; *Signals and Systems*; 2nd Edition, Upper Saddle River, NJ : Prentice-Hall, 1997.
32. Brown, J.W., and R.V. Churchill; *Complex Variables and Applications*; 7th Edition, New York: McGraw-Hill, 2003.
33. Lefschetz, S.; *Differential Equations: Geometric Theory*; 2nd Edition, New York : Dover Publications, 1977.

DYNAMICAL SYSTEMS AND MODELING

34. Palm, W.J., III; *System Dynamics*; New York: McGraw-Hill, 2004.
35. Zak, S.H.; *Systems and Control*; New York: Oxford University Press, 2003.
36. Ogata, K.; *System Dynamics*; 4th Edition, Englewood Cliffs, NJ: Prentice-Hall, 2003.
37. Clark, R.N.; *Control System Dynamics*; New York: Cambridge University Press, 1996.
38. Belanger, P.R.; *Control Engineering*: *A Modern Approach*; Orlando, Florida: Saunders College Publishing, 1995.
39. Moschytz, G.S.; *Linear Integrated Networks: Fundamentals*; New York: Van Nostrand Reinhold, 1974.
40. Schwarz, R.J., and B. Friedland; *Linear Systems*; New York: McGraw-Hill, 1965.
41. Mason, S.J., *Feedback Theory: Further Properties of Signal Flow Graphs*, Proc., IRE, Vol.44, No. 7, pp. 920-926, July 1956.
42. Mason, S.J., *Feedback Theory: Some Properties of Signal Flow Graphs*, Proc. IRE, Vol. 41, No. 9, pp.1144–1156, Sept.1953.

INDUSTRIAL CONTROL DEVICES

43. Necsulescu, D.; *Mechatronics*; Singapore: Pearson Education, 2002.
44. Gupta, S.; *Elements of Control Systems*; Upper Saddle River, N.J: Pearson Education, 2002.
45. Kilian, C.T.; *Modern Control Technology*: *Components and Systems*; 2nd Edition, Singapore: Delmar Publishers, 2002.
46. Morris, N.M.; *Control Engineering*; 4th Edition, London: McGraw-Hill, 1991.
47. De Silva, C.W.; *Control Sensors and Actuators*; Englewood Cliffs, NJ: Prentice-Hall, 1989.
48. Anderson, W.R.; *Controlling Electrohydraulic Systems*; New York: Marcel Dekker, 1988.
49. Parr, E.A.; *Industrial Control Handbook*, Vol.1–Vol.3; Oxford: BSP Professional Books, 1987.
50. Schuler, C.A., and W.L. McNamee; *Industrial Electronics and Robotics*; New York: McGraw-Hill, 1986.
51. Kenjo, T.; and S. Nagamori; *Permanent-magnet and Brushless DC Motors*, Oxford: Clarendon Press, 1985.
52. Kenjo, T.; *Stepping Motors and their Microprocessor Controls*; Oxford Clarendon Press, 1984.
53. Ahrendt, W.R., and C.J. Savant, Jr.; *Servomechanism Practice*; 2nd Edition, New York: McGraw-Hill, 1960.
54. Kuo, S., and W.S.Gan; *Digital Signal Processors: Architectures, Implementations, and Applications*; Upper Saddle River, NJ: Pearson Education, 2005.
55. Pack, D., and S. Barrett; *68HCI2 Microcontroller: Theory and Applications*; Upper Saddle River, NJ: Prentice-Hall, 2002.
56. Wolf, W.; *Computers as Components: Principles of Embedded Computing System Design*; San Franscisco: Morgan Kaufmann, 2001.
57. Mackenzie, I.; *The 8051 Microcontroller*; Upper Saddle River, NJ: Prentice-Hall, 1998.

FEEDBACK CONTROL THEORY

58. Franklin, G.F., J.D. Powell, and A. Emami-Naeini; *Feedback Control of Dynamical Systems*; 5th Edition, Upper Saddle River, NJ: Pearson Education, 2005.
59. Dorf, R.C., and R.H. Bishop; *Modern Control Systems*, 10th Edition, Upper Saddle River, NJ: Pearson Education, 2004.
60. Nise, N.S.; *Control Systems Engineering*; 4th Edition, Danvers, MA: John Wiley, 2003.
61. Kuo, B.C., and F. Golnaraghi; *Automatic Control Systems*; 8th Edition, Danvers, MA: John Wiley, 2003.
62. D'Azzo, J.J., C.H. Houpis, and S.N.Sheldon; *Linear Control System Analysis and Design with MATLAB*; 5th Edition, New York: Marcel Dakker, 2003.
63. Wilkie, J.,M. Johnson, and R. Ketabi; *Control Engineering: An Introductory Course*; New York: Palgrave, 2003.
64. Dorsey, J.; *Continuous and Discrete Control Systems*; New York: McGraw-Hill, 2002.
65. Stefani, R.T., B. Shahian, C.J. Savant, and G.H. Hostetter; *Design of Feedback Control Systems*; 4th Edition, New York: Oxford University Press, 2002.
66. Ogata, K.; *Modern Control Engineering*; 4th Edition, Englewood Cliffs, NJ: Prentice-Hall, 2001.
67. Goodwin, G.C., S.F. Graebe, and M.E. Salgado; *Control System Design*; Upper Saddle River, NJ: Pearson Education, 2001.

68. Phillips, C.L., and R.D. Harbor; *Feedback Control Systems*; 4th Edition, Englewood Cliffs, NJ: Prentice-Hall, 1999.
69. Shinners, S.M.; *Modern Control System Theory and Design*; 2nd Edition, New York: John Wiley, 1998.
70. Raven, F.H.; *Automatic Control Engineering*; 5th Edition, New York: McGraw-Hill, 1995.
71. Wolovich, W.A.; *Automatic Control Systems*: *Basic Analysis and Design*; Orlando, Florida: Saunders College Publishing, 1994.
72. Doebelin, E.O.; *Control System Principles and Design*; New York; John Wiley, 1986.
73. Truxal, J.G.; *Automatic Feedback Control System Synthesis*; New York: McGraw-Hill, 1995.

ROBUST CONTROL

74. Dullerud, G.E., and F. Paganini; *A Course in Robust Control Theory*; New York: Springer-Verlag, 2000.
75. Zhou, K., and J.C. Doyle; *Essentials of Robust Control*; Englewood Cliffs, NJ: Prentice-Hall, 1997.
76. Saberi, A., B.M. Chen, and P. Sannuti; *Loop Transfer Recovery–Analysis and Design*; London: Springer-Verlag, 1993.
77. Doyle, J.C., B.A. Francis, and A.R. Tannenbaum; *Feedback Control Theory*; New York: MacMilian; *Publishing Company*, 1992.
78. Francis, B.A.; *A course in the Control Theory, Lecture Notes in Control and Information Seciences*; No. 88, Berlin: Springer-Verlag, 1987.
79. Rosenwasser, E., and R. Yusupov; *Sensitivity of Automatic Control Systems*; Boca Ratno, FL: CRC Press, 2004.

DIGITAL CONTROL

80. Franklin, G.F., J.G. Powell, and M.L. Workman; *Digital Control of Dynamic Systems*; 3rd Edition, San Diejo, CA: Addision-Wesley, 1997.
81. Astrom, K.J., and B. Wittenmark; *Computer-Controlled Systems*; 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall, 1996.
82. Santina, M.S., and A.R. Stubberud; *Sample-Rate Selection*; Boca Raton, FL: CRC Press, 1996.
83. Santina, M.S., and A.R. Stubberud; *Quantization Effects*; Boca Raton, FL: CRC Press, 1996.
84. Ogata, K.; *Discrete-Time Control Systems*; 2nd Edition, Upper Saddle River, NJ: Pearson Education, 1995.
85. Santina, M.S., A.R. Stubberud, and G.H. Hostetter; *Digital Control System Design*; 2nd Edition, Stamford, CT: International Thomson Publishing, 1994.
86. Phillips, C.L., and H.T. Nagla, Jr.; *Digital Control System Analysis and Design*; 3rd Edition, Upper Saddle River, NJ: Pearson Education, 1994.
87. Kuo, B.C.; *Digital Control Systems*; 2nd Edition, Orlando, Florida: Saunders College Publishing, 1992.
88. Houpis, C.H., and G.B. Lemont; *Digital Control Systems*: *Theory, Hardware, and Software*; 2nd Edition, New York: McGraw-Hill, 1992.

89. Hristu-Varsakelis, D., and W.S. Levine(eds); *Handbook of Networked and Embedded Control Systems*; Boston: Birkhauser Publishers, 2005.
90. Ibrahim, D.; *Microcontroller Based Temperature Monitoring and Control*; Woburn, MA: Newnes, 2002.
91. Chidambaram, M.; *Computer Control of Processes*; New Delhi: Narosa Publishers, 2002.
92. Ozkul, T.; *Data Acquisition and Process Control using Personal Computers*; New York: Marcel Dekkar, 1996.
93. Rigby, W.H., and T. Dalby; *Computer Interfacing: A Practical Approach to Data Acquisition and Control*; Englewood Cliffs, NJ: Prentice-Hall, 1995.
94. Tooly, M.; *PC-Based Instrumentation and Control*; 2nd Edition, Oxford: Newnes: Publications, 1995.
95. Gupta, S., and J.P. Gupta; *PC Interfacing for Data Acquisition and Process Control*; 2nd Edition, Research Triangle Park, North Carolina : Instrument Society of America, 1994.
96. Gopal, M.; *Digital Control Engineering*; New Delhi: New Age International, 1988.
97. Rao, M.V.C, and A.K. Subramanium; "Elimination of singular cases in Jury's test" *IEEE Trans. Automatic Control*, Vol.AC-21, pp.1q14–115, 1976.
98. Jury, E.I., and J. Blanchard; "*A Stability Test for Linear Discrete-time Systems in Table Form*", Proc. IRE., Vol.49, pp.1947–1948, 1961.

STATE SPACE AND LINEAR SYSTEMS

99. Tewari, A.; *Modern Control Design with MATLAB and Simulink*; Singapore: John Wiley, 2003.
100. Chen, C-T.; *Linear System Theory and Design*; 3rd Edition, New York: Oxford University Press, 1998.
101. DeRusso, P.M., R.J. Roy, C.M. Close, and A.A. Desrochers; *State Variables for Engineers*; 2nd Edition, New York: John Wiley, 1998.
102. Szidarovszky, F., and A.T. Bahill; *Linear System Theory*; 2nd Edition, Boca Raton, FL: CRC Press, 1998.
103. Antsaklis, P.J., and A.N. Michel; *Linear Systems*; New York: McGraw-Hill, 1997.
104. Rugh, W.I.; *Linear System Theory*; 2nd Edition, Upper Saddle River, NJ: Prentice-Hall, 1995.
105. Gopal, M.; *Modern Control System Theory*; 2nd Edition, New Delhi: New Age International, 1993.
106. Brogan, W.L.; *Modern Control Theory*; 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall, 1991.
107. Friedland, B.; *Control System Design*: *An Introduction to State-Space Methods*; New York: McGraw-Hill, 1986.
108. Zadeh, L.A., and C.A. Desoer; *Linear System Theory: A State Space Approach*; New York: McGraw-Hill, 1963.

MULTIVARIABLE AND OPTIMAL CONTROL

109. Bertsekas, D.P.; *Dynamic Programming and Optical Control*; 3rd Edition, Belnont: Athena Sientific, 2005.
110. Naidu, D.S.; *Optimal Control Systems; Boca Raton*, FL: CRC Press, 2003.
111. Camacho, E.F., and C. Bordons; *Model Predictive Control*; 2nd Edition, London: Springer-Verlag, 2003.

112. Bryson, A.E.; *Applied Linear Optimal Control: Examples and Algorithms*; Cambridge, UK : Cambridge University Press, 2002.
113. Locatelli, A.; *Optimal Control: An Introduction*; Basel, Switzerland: Birkhauser Verlag, 2001.
114. Chen, T., and B. Francis; *Optimal Sampled-Data Control Systems*; 2nd Edition, London: Springer, 1996.
115. Lewis, F.L., and V.L. Syrmos; *Optimal Control*; 2nd Edition, New York: John Wiley, 1995.
116. Zhou, K., J.C. Doyle, and K. Glover; *Robust and Optimal Control*; Upper Saddle River, NJ: Prentice-Hall, 1996.
117. Anderson, B.D.O., and J.B. Moore; *Optimal Control: Linear Quadratic Methods*; Englewood Cliffs, NJ: Prentice-Hall, 1990.
118. Grimble, M.J., and M.A., Johnson; *Optimal Control and Stochastic Estimation: Theory and Applications*; Vol.I, Chichester: John Wiley, 1988.
119. Albertos, P., and A. Sala; *Multivariable Control Systems: An Engineering Approach*; Springer, 2004.
120. Clarke, D.W., C. Mothadi, and P.S. Tuffs "Generalized predictive control–Part I. The basic algorithm", *Automatica*, vol. 23, No. 2, pp. 137–148; 1987.
121. Clarke, D.W., C. Mothadi, and P.S. Tuffs "Generalized predictive control–Part II. Extensions and interpretations", *Automaticam* vol. 23, No. 2, pp. 149–160; 1987.
122. Fortman, T.E., and K.L. Hitz, *An Introduction to Linear Control Systems*, New York: Marcel Dekker, 1977.
123. Kautsky, J., N.K. Nichols, and P. Dooren; "Robust pole assignment by linear state feedback", *Int., J.Control*, Vol.41, No.5, pp.1129–1155, 1985.
124. Doyle, J.C., and G. Stein; "Robustness with Observers", *IEEE Trans. Automatic Control*, Vol.AC-24, No.4, pp.607-611, 1979.

NONLINEAR CONTROL SYSTEMS

125. Khalil, H.K.; *Nonlinear Systems*; 3rd Edition, Upper Saddle River, NJ: Prentice-Hall, 2001.
126. Slotine, J-J.E., and W.Li.; *Applied Nonlinear Control*; Englewood Cliffs, NJ: Prentice-Hall, 1991.
127. Itkis,U.; *Control Systems of Variable Structure*; New York: John Wiley, 1976.
128. Atherton, D.P.; *Nonlinear Control Engineering*; London: Van Nostrand Reinhold, 1975.
129. Minorsky N.; *Theory of Nonlinear Control Systems*; New York: McGraw-Hill, 1969.

SYSTEM IDENTIFICATION AND ADAPTIVE CONTROL

130. Narendra, K.S., and A.M. Annaswamy; *Stable Adaptive Systems*; New York: Dover Publications, 2005.
131. Ljung, L.; *System Identification: Theory for the User*; 2nd Edition, Englewood Cliffs, NJ: Prentice-Hall, 1998.
132. Astrom, K.J., and B. Wittenmark; *Adaptive Control*; 2nd Edition, Reading, MA: Prentice-Hall, 1994.
133. Landau, I.D.; *System Identification and Control Design*; Englewood Cliffs, NJ: Prentice-Hall, 1990.

134. Sastry, S., and M. Bodson; *Adaptive Control: Stability*, *Convergence and Robustness*; Englewood Cliffs, NJ: Prentice-Hall, 1989.
135. Grimble, M.J., and M.A. Johnson; *Optimal Control and Stochastic Examination: Theory and Applications*; Vol.II, Chichester: John Wiley, 1988.
136. Harris, C.J., and S.A. Billings (eds); *Self-Tuning and Adaptive Control: Theory and Applications*; 2nd Edition, London: Peter Peregrinus, 1985.

INTELLIGENT CONTROL

137. Negnevitsky, M.; *Artificial Intelligence: A Guide to Intelligent Systems*; 2nd Edition, Essex, UK: Pearson Education, 2005.
138. Kccman, V.; *Learning and Soft Computing*; Cambridge, MA: The MIT Press, 2001.
139. Norgaard, M.O., Ravn, N.K. Poulsen, and L.K. Hansen; *Neural Networks for Modelling and Control of Dynamic Systems*; London: Springer-Verlag, 2000.
140. Lewis, F.L., S. Jagannathan, and A. Yesildirek; *Neural Network Control of Robot Manipulators and Nonlinear Systems*; London, Taylor & Francis, 1999.
141. Haykin, S.; *Neural Networks: A Comprehensive Foundation*, 2nd Edition, Prentice-Hall, 1998.
142. Jang, J-S.R., C-T. Sun, and E. Mizutani; *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*; Upper Saddle River, NJ: Pearson Education, 1997.
143. Lin C.T., and C.S. George Lee; *Neural Fuzzy Systems*; Upper Saddle River, NJ: Prentice-Hall, 1996.
144. Ying, Hao; *Fuzzy Control and Modelling: Analytical Foundations and Applications*; New York: IEEE Press, 2000.
145. Passino, K.M., and S. Yurkovich; *Fuzzy Control*; California: Addison-Wesley, 1990.
146. Goldberg, D.E.; *Genetic Algorithms in Search Optimization, and Machine Learning*; Reading, Massachusetts, Addison-Wesley, 1989.
147. Werbos, P., *A. Handbook of Learning and Approximate Dynamic Programming*, New York: Wiley-IEEE Press, August 2004.
148. Sutton. R.S., and A.G. Barto, *Reinforcement Learning: An Introduction* Cambridge, Mass.: MIT Press, 1998.
149. Mitchell, T.M., Machine Learning, Singapore: McGraw-Hill, 1997.
150. Bertsekas, D.P., and J.N. Tsitsiklis, *Neruo-Dynamic Programming*, Belmont, Mass: Athena Scientific, 1996.

TECHNICAL COMPUTING

151. National Programme on technology enhancement learnning
Course: Electrical Engineering (control Engineering)
Faculty Coordinator: Prof. M. Gopal
Web Content: Matlab Modules for Control System Principles and Design
www.nptel.iitm.ac.in

152. MATLAB & Simulink Software
www.mathworks.com
The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098, USA
153. MATHEMATICA Software
www.wolfram.com
Wolfram Research, Inc.
100 Trade Center Drive
Champaign, IL 61820-7237, USA
154. MAPLESOFT Software
www.maplesoft.com
Maplesoft
615 Kumpf Drive
Waterloo, Ontario
Canada N2V1K8

COMPANION BOOK

155. Gopal, M.; *Control Systems: Principles and Design*; 4th Edition, New Delhi: Tata McGraw-Hill, 2012.

Answers to Problems

Caution

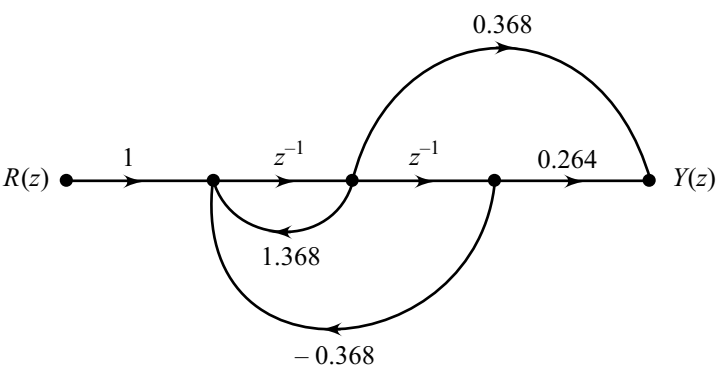
For some problems (especially the design problems) of the book, many alternative solutions are possible. The answers given in the present section, correspond to only one possible solution for such problems.

2.1 (a) x_1, x_2: Outputs of unit delayers, starting at the right and proceeding to the left.

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -0.368 & 1.368 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{c} = [0.264 \quad 0.368]$$

(b)



$$\frac{Y(z)}{R(z)} = \frac{0.368z + 0.264}{z^2 - 1.368z + 0.368}$$

2.2 (a) $y(k+2) + 5y(k+1) + 3y(k) = r(k+1) + 2r(k)$

(b) x_1, x_2: Outputs of unit delayers, starting at the right and proceeding to the left.

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -3 & -5 \end{bmatrix}; \mathbf{g} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}; \mathbf{c} = [1 \quad 0]$$

(c) $$\frac{Y(z)}{R(z)} = \frac{z+2}{z^2 + 5z + 3}$$

2.3 (a) $y(k+1)+\frac{1}{2}y(k)=-r(k+1)+2r(k);\ \frac{Y(z)}{R(z)}=\frac{-z+2}{z+\frac{1}{2}}$

(b) $y(k)=\begin{cases}-1 & ;k=0\\ \frac{5}{2}\left(-\frac{1}{2}\right)^{k-1} & ;k\geq 1\end{cases}$ (c) $y(k)=\frac{2}{3}-\frac{5}{3}\left(-\frac{1}{2}\right)^{k};\ k\geq 0$

2.4 (a) $y(k)=A\beta(\alpha)^{k-1};\ k\geq 1$ (b) $y(k)=\frac{A\beta}{1-\alpha}[1-(\alpha)^{k}];\ k\geq 0$

(c) $y(k)=\frac{A\beta}{(1-\alpha)^{2}}[\alpha^{k}+(1-\alpha)k-1];\ k\geq 0$

(d) $y(k)=\text{Re}\left[\frac{A\beta}{\alpha-e^{j\Omega}}(\alpha^{k}-e^{j\Omega k})\right]$

2.5 (a) $y(k)=(-1)^{k}-(-2)^{k};\ k\geq 0$

(b) $y(k)=1+\frac{1}{2}\left(\frac{1}{\sqrt{2}}\right)^{k}\left[\sin\frac{k\pi}{4}-\cos\frac{k\pi}{4}\right];\ k\geq 0$

2.6 $y(0)=0, y(1)=0.3679, y(2)=0.8463, y(k)=1;\ k=3, 4, 5, \ldots$

2.7 $y(k)=3(2)^{k}-2;\ k\geq 0$

2.8 (a) $y(k)=-40\delta(k)+20\delta(k-1)-10(0.5)^{k}+50(-0.3)^{k};\ k\geq 0$

(b) $y(k)=-16+(0.56)^{k}[7.94\sin(0.468k)+16\cos(0.468k)];\ k\geq 0$

2.9 (a) $y(k)=-0.833(0.5)^{k}-0.41(-0.3)^{k}+0.476(-1)^{k}+0.769;\ k\geq 0$

(b) $y(k)=-10k(0.5)^{k}+2.5(0.5)^{k}-6.94(0.1)^{k}+4.44;\ k\geq 0$

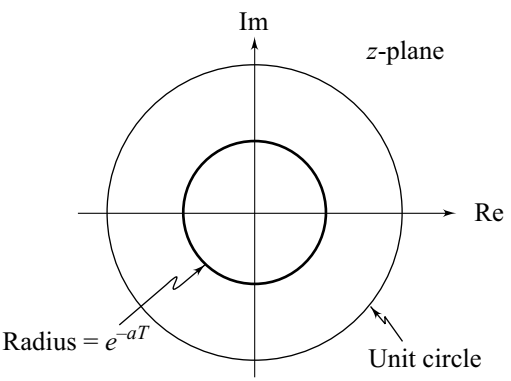
2.10 $y(\infty)=K$

2.13 (a) No (b) Yes

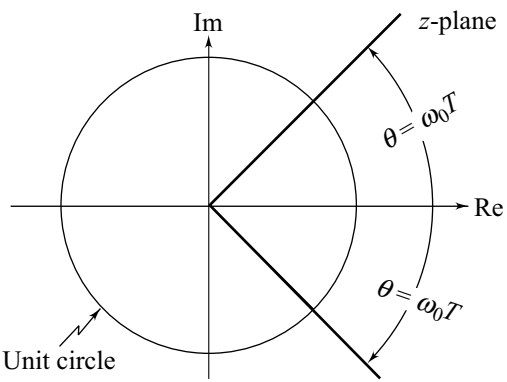
2.14 No

2.16 (b) $T=\pi/\sqrt{2}$

2.19 (i)



(ii)



2.20 $G(z) = \dfrac{1 - e^{-T}}{z - e^{-T}}$; $y(k) = 1 - e^{-kT}$; $k \geq 0$

2.21 $\dfrac{Y(z)}{R(z)} = \dfrac{10}{16}\left[\dfrac{z + 0.76}{(z-1)(z-0.46)}\right]$

2.25 $u(k) - u(k-1) = K_c\left\{\left(1 + \dfrac{T}{T_I} + \dfrac{T_D}{T}\right)e(k) - \left(1 + \dfrac{2T_D}{T}\right)e(k-1) + \dfrac{T_D}{T}e(k-2)\right\}$

$U(z) = K_c\left[1 + \dfrac{T}{T_I}\left(\dfrac{1}{1 - z^{-1}}\right) + \dfrac{T_D}{T}(1 - z^{-1})\right]E(z)$

2.26 (ii) $D(z) = 0.4074\left(\dfrac{z - 0.9391}{z - 0.9752}\right)$

2.27 (a) $U(z) = K_c\left[1 + \dfrac{T}{2T_I}\left(\dfrac{z+1}{z-1}\right) + \dfrac{T_D}{T}\left(\dfrac{z-1}{z}\right)\right]E(z)$

(b) $u(k) = K_c\left\{e(k) + \dfrac{T}{T_I}\sum_{i=1}^{k}\dfrac{e(i-1) + e(i)}{2} + \dfrac{T_D}{T}[e(k) - e(k-1)]\right\}$

2.28 (a) $y(k) = \dfrac{1}{1 + aT}y(k-1) + \dfrac{T}{1 + aT}r(k)$ (b) $y(k) = (1 - aT)y(k-1) + Tr(k-1)$

2.29 $\left(b + \dfrac{a}{T} + \dfrac{1}{T^2}\right)y(k) - \left(\dfrac{a}{T} + \dfrac{2}{T^2}\right)y(k-1) + \dfrac{1}{T^2}y(k-2) = 0; \quad y(0) = \alpha, y(-1) = \alpha - T\beta$

3.1 $\dfrac{Y(z)}{R(z)} = \dfrac{G_{h0}G_1(z)\,G_{h0}G_2(z)}{1 + G_{h0}G_1(z)\,G_{h0}G_2H(z)}$

3.2 $\dfrac{Y(z)}{R(z)} = \dfrac{G_{h0}G(z)}{1 + G_{h0}G(z)\,H(z)}$

3.3 $Y(z) = \dfrac{G_{h0}G_2(z)G_1R(z)}{1 + G_{h0}G_2HG_1(z)}$

3.4 $Y(z) = G_pH_2R(z) + \dfrac{D(z)G_{h0}G_p(z)}{1 + D(z)G_{h0}G_p(z)}\,[H_1R(z) - G_pH_2R(z)]$

3.5 $\dfrac{Y(z)}{R(z)} = \dfrac{D(z)G_{h0}G_1G_2(z)}{1 + D(z)[G_{h0}G_1(z) + G_{h0}G_1G_2(z)]}$; $\dfrac{X(z)}{R(z)} = \dfrac{D(z)G_{h0}G_1(z)}{1 + D(z)[G_{h0}G_1(z) + G_{h0}G_1G_2(z)]}$

3.6 $Y(z) = \dfrac{GW(z)}{1 + D(z)G_{h0}G(z)}$

3.7 $G_{h0}G(z) = 0.0288\left[\dfrac{z + 0.92}{(z-1)\,(z - 0.7788)}\right]$; $\dfrac{\theta_L(z)}{\theta_R(z)} = \dfrac{G_{h0}G(z)}{1 + G_{h0}G(z)}$

3.8 $\dfrac{\omega(z)}{\omega_r(z)} = 159.97\left(\dfrac{K_F + K_P}{z + 159.97K_P - 0.1353}\right)$

3.9 $\dfrac{Y(z)}{R(z)} = 0.0095\left[\dfrac{z(z+0.92)}{z^3 - 2.81z^2 + 2.65z - 0.819}\right]$

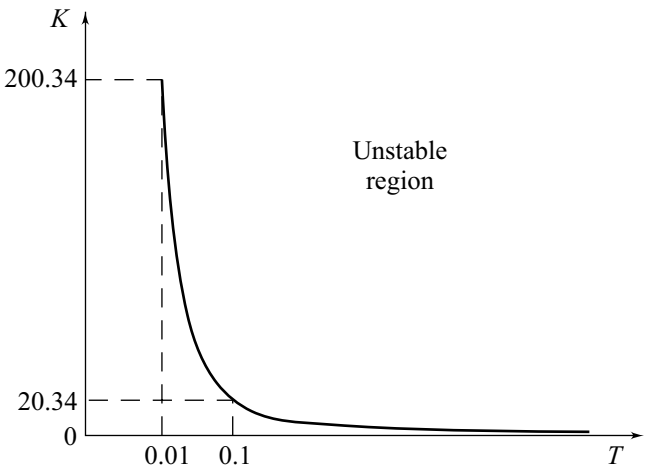
3.10 (i) $\dfrac{Y(z)}{R(z)} = \dfrac{0.45z + 0.181}{z^2 + 0.081z + 0.181}$ (ii) $\dfrac{Y(z)}{R(z)} = \dfrac{0.45z + 0.181}{z^2 - 0.368z^2 + 0.45z + 0.181}$

3.13 $0 < K < 4.293$

3.14 (i) The system is stable for $0 < K < 4.723$.

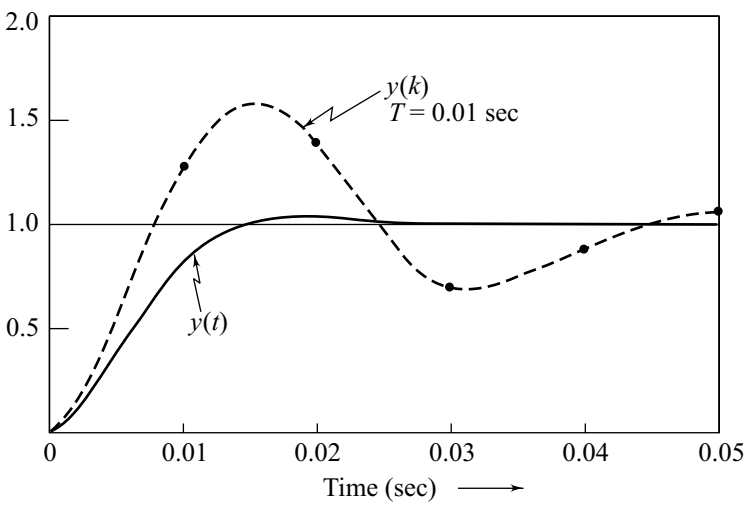
(ii) The system is stable for $0 < K < 3.315$.

3.15



3.16 $0 < K < 0.785$.

3.17 For $T = 0.001$ sec, the response $y(k)$ is very close to $y(t)$.



3.18 $y(k) = 1.02\ (0.795)^k \sin(0.89k);\ k \geq 0$

3.19 $y(0) = 0;\ y(0.5T) = 0.393;\ y(T) = 0.632;\ y(1.5T) = 0.528$
$y(2T) = 0.465; y(2.5T) = 0.493;\ y(3T) = 0.509;\ y(3.5T) = 0.502;\ \cdots$

3.22 (a) $K = 30^\circ\text{C/(kg/min)};\ \tau_D = 5$ min; $\tau = 60$ min

(b) $K_c = 0.545;\ T_I = 13.75$ min; $\tau_D = 2.2$ min

3.23 $K_c = 2.13;\ T_I = 666.66$ sec

4.1 $K_p = \infty;\ K_v = K_1/K_2;\ K_a = 0.$

4.2 $D_1(s) = \dfrac{25s+1}{62.5s+1};\ D_1(z) = 0.4\left(\dfrac{z-0.939}{z-0.975}\right)$

Velocity error constants are equal

4.3 0, 1/3.041, ∞

4.4 Underdamped response with $\zeta = 0.199$ and $\omega_n = 8.93$.

4.5 (a) $Y(z) = \dfrac{[D_2(z) + D_1(z)D_3(z)]G_{h0}G(z)}{1 + D_1(z)G_{h0}G(z)}\ R(z) + \dfrac{GW(z)}{1 + D_1(z)G_{h0}G(z)}$

(b) $Y(z) = D_3(z)R(z) + \dfrac{GW(z)}{1 + D_1(z)G_{h0}G(z)}$

(c) $D_1(z)$ can be made large to reject the disturbances

4.7 $S(z) = \dfrac{z-0.607}{z-0.214};\ \omega_b = 2$ rad/sec

4.8 $GM = 8$ dB; $\Phi M = 28^\circ$; $\nu_b = 1.6$ rad/sec; $\omega_b = 1.35$ rad/sec

4.9 (a) Increase plant gain by a factor of 10; $\Phi M = 30^\circ$

(b) $D(z) = 4.2423\left(\dfrac{z-0.8187}{z-0.2308}\right)$

(c) $D(z) = 0.141\left(\dfrac{z-0.98}{z-0.998}\right)$

(d) $\nu_{b1} = 4.8;\ \nu_{b2} = 9.8;\ \nu_{b3} = 1.04$

(e) Yes

4.10 $D(z) = 37.333\left(\dfrac{z-0.9048}{z-0.1111}\right);\ K_v = \infty$

In the low-frequency range, $\angle G_{h0}G(j\nu)$ is about -180°; therefore, a lag compensator cannot fulfil the requirement of 50° phase margin.

4.11 (a) $K = 50;\ \Phi M = 20^\circ;\ \omega_b = 9.27$ rad/sec.

(c) With lag compensator $D_1(z) = 0.342\,\dfrac{z-0.923}{z-0.973}$, $\Phi M = 54^\circ$, $\omega_b = 4.23$ rad/sec

With lag-lead compensator $D_1(z)D_2(z);\ D_2(z) = 2.49\dfrac{z-0.793}{z-0.484};\ \Phi M = 60^\circ,\ \omega_b = 7.61$ rad/sec

4.12 (a) $K = 61.25$ (b) Unstable

(c) $D(z) = 0.122K\left(\dfrac{z-0.6}{z-0.951}\right)$

4.13 (b) $0 < K < 2.1$; $K = 0.38$

4.14 (a) $K = 2.3925$

(b) (i) $K = 1.4557$; (ii) $K = 0.9653$

4.15 (a) $K = 0.88$; 1.33 rad/sec (b) $K = 0.072$; $\tau = 2.3$ sec

(c) $K = 0.18$; $\omega_n = 0.644$ rad/sec

4.16 $0 < A < 3.33$

4.17 (a) $\tau = 0.4854$ (b) $K = 5.1223$

4.18 (a) $D_1(z) = 13.934\left(\dfrac{z-0.8187}{z-0.1595}\right)$ (b) $K_v = 3$

(c) $D_2(z) = \dfrac{z-0.94}{z-0.98}$

4.19 $D(z) = 1.91\left(\dfrac{z-0.84}{z-0.98}\right)$

4.20 (a) $z_{1,2} = 0.78 \pm j0.18$

(b) Pole of $D(z)$ at $z = 0$; $(K/2) = 0.18$

(c) $K_a = 0.072$

(d) $z_3 = 0.2$; the third pole causes the response to slow down.

4.21 $D(z) = 150\left(\dfrac{z-0.72}{z+0.4}\right)$

4.22 (a) $D(z) = 135.22\left(\dfrac{(z-0.9048)\,(z-0.6135)}{(z+0.9833)\,(z-0.7491)}\right)$

(b) $D(z) = 104.17\left(\dfrac{(z-0.9048)\,(z+1)}{(z+0.9833)\,(z+0.5)}\right)$; 0.15

4.23 $D(z) = \dfrac{4.8-3.9z^{-1}}{1-z^{-1}}$

5.1 $x_1 = \theta_M$, $x_2 = \dot{\theta}_M$, x_3 = motor armature current i_a, x_4 = generator field current i_f; $y = \theta_L$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.025 & 3 & 0 \\ 0 & -12 & -190 & 1000 \\ 0 & 0 & 0 & -4.2 \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.2 \end{bmatrix}; \quad \mathbf{c} = [0.5 \quad 0 \quad 0 \quad 0]$$

5.2 $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 20 \\ 0 & 0 & -5 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 2.5 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0 \quad 0]$

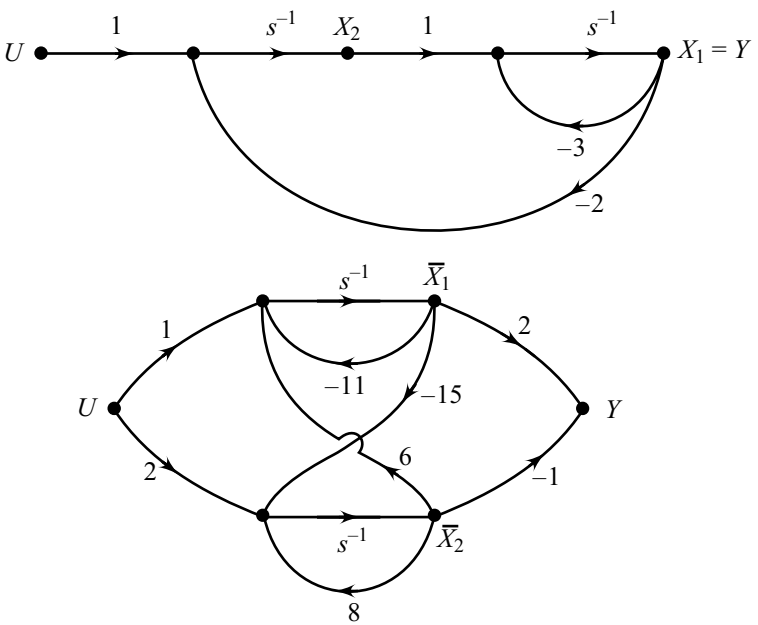
5.3 $x_1 = \theta_M$, $x_2 = \dot{\theta}_M$, $x_3 = i_a$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -0.5 & 19 \\ -\dfrac{k_1}{40} & \dfrac{-(k_2+0.5)}{2} & \dfrac{-21}{2} \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \dfrac{k_1}{2} \end{bmatrix}; \quad \mathbf{c} = \begin{bmatrix} \dfrac{1}{20} & 0 & 0 \end{bmatrix}$$

5.4 $\mathbf{A} = \begin{bmatrix} \dfrac{-B}{J} & \dfrac{K_T}{J} \\ \dfrac{-(k_1 K_t K_c + K_b)}{L_a} & \dfrac{-(R_a + k_2 K_c)}{L_a} \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ \dfrac{k_1 K_c}{L_a} \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

5.5 (a) $\bar{\mathbf{A}} = \begin{bmatrix} -11 & 6 \\ -15 & 8 \end{bmatrix}$; $\bar{\mathbf{b}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$; $\bar{\mathbf{c}} = [2 \quad -1]$

(b)



(c) $\dfrac{Y(s)}{U(s)} = \dfrac{1}{s^2 + 3s + 2}$

5.6 (a) $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ (b) $\bar{\mathbf{A}} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$; $\bar{\mathbf{b}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

(c) $|\lambda\mathbf{I} - \mathbf{A}| = |\lambda\mathbf{I} - \bar{\mathbf{A}}| = \lambda^2$

5.7 $\mathbf{G}(s) = \dfrac{1}{\Delta}\begin{bmatrix} s(s+3) & s+3 & 1 \\ -1 & s(s+3) & s \\ -s & -1 & s^2 \end{bmatrix}$

$\mathbf{H}(s) = \dfrac{1}{\Delta}\begin{bmatrix} 1 \\ s \\ s^2 \end{bmatrix}$; $\Delta = s^3 + 3s^2 + 1$

5.9 x_1, x_2, x_3: outputs of integrators, starting at the right and proceeding to the left.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 1 \\ -2 & 1 & -2 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{c} = [2 \quad -2 \quad 1]$$

5.10 x_1, x_2, x_3, x_4: outputs of integrators

Top row: $x_1, x_2 = y_1$; Bottom row: $x_3, x_4 = y_2$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -4 \\ 1 & -3 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -4 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} 3 & 0 \\ 1 & 2 \\ 0 & 3 \\ 0 & 0 \end{bmatrix}; \mathbf{x}(0) = \begin{bmatrix} 0 \\ y_1(0) \\ 0 \\ y_2(0) \end{bmatrix}; \mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.11 (a) $G(s) = \dfrac{s+3}{(s+1)(s+2)}$ (b) $G(s) = \dfrac{1}{(s+1)(s+2)}$

5.12 $\mathbf{G}(s) = \dfrac{1}{\Delta}\begin{bmatrix} -3s+5 & 4(s-3) \\ -s^2+2s & 2(s^2-3s+1) \end{bmatrix}$; $\Delta = s^3 - 4s^2 + 6s - 5$

5.13 (a) $\mathbf{A} = \begin{bmatrix} -5 & 0.5 & -3.5 \\ 4 & -5 & 0 \\ 0 & 1 & 0 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$; $\mathbf{c} = [0 \quad 1 \quad 0]$

(b) $G(s) = \dfrac{14}{(s+1)(s+2)(s+7)}$

5.14 (a) x_1 = output of lag $1/(s+2)$; x_2 = output of lag $1/(s+1)$

$$\mathbf{A} = \begin{bmatrix} -2 & 1 \\ -1 & -1 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \mathbf{c} = [-1 \quad 1]; d = 1$$

(b) x_1 = output of lag $1/(s+2)$; x_2 = output of lag $1/s$; x_3 = output of lag $1/(s+1)$.

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & 0 & -1 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}; \mathbf{c} = [0 \quad 1 \quad 1]$$

5.15 x_1 = output of lag $1/(s+1)$; x_2 = output of lag $5/(s+5)$; x_3 = output of lag $0.4/(s+0.5)$; x_4 = output of lag $4/(s+2)$.

$$\mathbf{A} = \begin{bmatrix} -1-K_1 & -K_1 & 0 & 0 \\ 0 & -5 & -5K_2 & -5K_2 \\ -0.4K_1 & -0.4K_1 & -0.5 & 0 \\ 0 & 0 & -4K_2 & -2-4K_2 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} K_1 & 0 \\ 0 & 5K_2 \\ 0.4K_1 & 0 \\ 0 & 4K_2 \end{bmatrix}; \mathbf{C} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

5.16 (i) $\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \mathbf{c} = [2 \quad -1]$

(ii) $\mathbf{A} = \begin{bmatrix} 0 & 0 & -2 \\ 1 & 0 & -5 \\ 0 & 1 & -4 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}; \mathbf{c} = [0 \quad 0 \quad 1]$

(iii) $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{c} = [2 \quad 6 \quad 2]; d = 1$

5.17 (i) $\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -2 \\ 0 & 1 & -3 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}; \mathbf{c} = [0 \quad 0 \quad 1]$

(ii) $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -11 & -6 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \quad 0 \quad 0]$

(iii) $\mathbf{\Lambda} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; \mathbf{c} = [-1 \quad 2 \quad 1]; d = 1$

5.18 (a) $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -100 & -52 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \mathbf{c} = [5000 \quad 1000 \quad 0]$

(b) $\mathbf{\Lambda} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -50 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 50 \\ -31.25 \\ -18.75 \end{bmatrix}; \mathbf{c} = [1 \quad 1 \quad 1]$

5.19 (a) $\mathbf{\Lambda} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2 \end{bmatrix}; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}; \mathbf{c} = [1 \quad 1 \quad 1]$

(b) $y(t) = 2.5 - 2e^{-t} - te^{-t} - 0.5e^{-2t}$

5.20 (i) $\lambda_1 = 1, \lambda_2 = 2; \mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

(ii) $\lambda_1 = -1, \lambda_2 = -2; \mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \mathbf{v}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

(iii) $\lambda_1 = -1, \lambda_2 = -2, \lambda_3 = -3; \mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}; \mathbf{v}_2 = \begin{bmatrix} 1 \\ -2 \\ \frac{1}{2} \end{bmatrix}; \mathbf{v}_3 = \begin{bmatrix} 1 \\ -3 \\ 3 \end{bmatrix}$

5.21 (b) $\lambda_1 = -2, \lambda_2 = -3, \lambda_3 = -4; \mathbf{v}_1 = \begin{bmatrix} 1 \\ -2 \\ 4 \end{bmatrix}; \mathbf{v}_2 = \begin{bmatrix} 1 \\ -3 \\ 9 \end{bmatrix}; \mathbf{v}_3 = \begin{bmatrix} 1 \\ -4 \\ 16 \end{bmatrix}$

5.22 (a) $\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 \\ -1+j1 & -1-j1 & -1 \\ -j2 & j2 & 1 \end{bmatrix}$

(b) $\mathbf{Q} = \begin{bmatrix} \frac{1}{2} & -j\frac{1}{2} & 0 \\ \frac{1}{2} & j\frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

5.23 $\lambda_1 = -1, \lambda_2 = 2; \mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \mathbf{v}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}; e^{\mathbf{A}t} = \begin{bmatrix} 2e^{-t} - e^{2t} & -e^{-t} + e^{2t} \\ 2e^{-t} - 2e^{2t} & -e^{-t} + 2e^{2t} \end{bmatrix}$

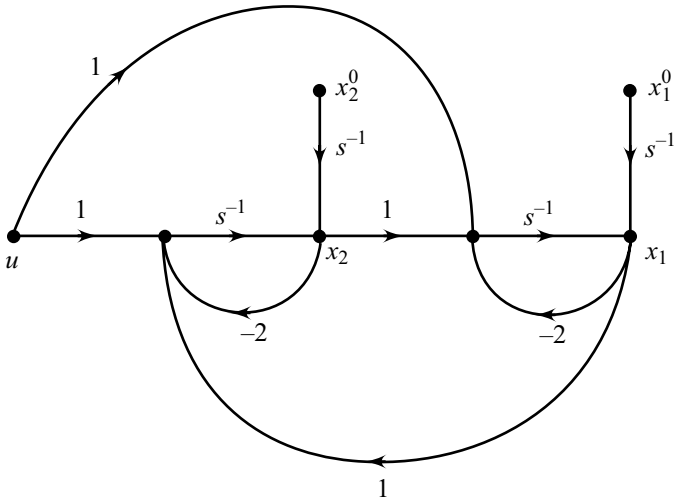
5.24 (a) $e^{\mathbf{A}t} = \begin{bmatrix} \frac{3}{2}e^{-t} - \frac{1}{2}e^{-3t} & -\frac{3}{2}e^{-t} + \frac{3}{2}e^{-3t} \\ \frac{1}{2}e^{-t} - \frac{1}{2}e^{-3t} & -\frac{1}{2}e^{-t} + \frac{3}{2}e^{-3t} \end{bmatrix}$

(b) $e^{\mathbf{A}t} = \begin{bmatrix} \frac{3}{2}e^{-t} - \frac{1}{2}e^{-3t} & \frac{1}{2}e^{-t} - \frac{1}{2}e^{-3t} \\ -\frac{3}{2}e^{-t} + \frac{3}{2}e^{-3t} & -\frac{1}{2}e^{-t} + \frac{3}{2}e^{-3t} \end{bmatrix}$

5.25 (a) $e^{\mathbf{A}t} = \begin{bmatrix} 3e^{-2t} - 2e^{-3t} & e^{-2t} - e^{-3t} \\ -6e^{-2t} + 6e^{-3t} & -2e^{-2t} + 3e^{-3t} \end{bmatrix}$

(b) $e^{\mathbf{A}t} = \begin{bmatrix} (1+2t)e^{-2t} & 2te^{-2t} \\ -2te^{-2t} & (1-2t)e^{-2t} \end{bmatrix}$

5.26 (a)



(b) $\dfrac{X_1(s)}{x_1^0} = G_{11}(s) = \dfrac{1/2}{s+1} + \dfrac{1/2}{s+3}$; $\dfrac{X_1(s)}{x_2^0} = G_{12}(s) = \dfrac{1/2}{s+1} + \dfrac{-1/2}{s+3}$

$\dfrac{X_2(s)}{x_1^0} = G_{21}(s) = \dfrac{1/2}{s+1} + \dfrac{-1/2}{s+3}$; $\dfrac{X_2(s)}{x_2^0} = G_{22}(s) = \dfrac{1/2}{s+1} + \dfrac{1/2}{s+3}$

$\dfrac{X_1(s)}{U(s)} = H_1(s) = \dfrac{1}{s+1}$; $\dfrac{X_2(s)}{U(s)} = H_2(s) = \dfrac{1}{s+1}$

(c) (i) $\mathbf{x}(t) = \dfrac{1}{2}\begin{bmatrix} e^{-t}(x_1^0 + x_2^0) + e^{-3t}(x_1^0 - x_2^0) \\ e^{-t}(x_1^0 + x_2^0) + e^{-3t}(-x_1^0 + x_2^0) \end{bmatrix}$ (ii) $\mathbf{x}(t) = \begin{bmatrix} 1 - e^{-t} \\ 1 - e^{-t} \end{bmatrix}$

5.27 $x_1(t) = \dfrac{1}{3} - e^{-2t} + \dfrac{2}{3}e^{-3t}$; $x_2(t) = 2(e^{-2t} - e^{-3t})$; $x_3(t) = -2(2e^{-2t} - 3e^{-3t})$; $y(t) = x_1(t)$

5.28 (a) Asymptotically stable (b) $y(t) = \dfrac{1}{2} + 2e^{-t} - \dfrac{3}{2}e^{-2t}$

5.29 $y_1(t) = 3 - \dfrac{5}{2}e^{-t} - e^{-2t} + \dfrac{1}{2}e^{-3t}$; $y_2(t) = 1 + e^{-2t} - 2e^{-3t}$

5.30 (a) $\mathbf{A} = \begin{bmatrix} -6 & 0.5 \\ 4 & -5 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 7 \\ 0 \end{bmatrix}$; $\mathbf{c} = [0 \quad 1]$

(b) $y(t) = \dfrac{28}{3}\left[\dfrac{1}{4}(1 - e^{-4t}) - \dfrac{1}{7}(1 - e^{-7t})\right]$

5.31 $\begin{bmatrix} x_1(1) \\ x_2(1) \end{bmatrix} = \begin{bmatrix} 2.7183 - k \\ 2k \end{bmatrix}$ for any $k \neq 0$

5.32 (a) Modes: $e^{\lambda_1 t}$, $e^{\lambda_2 t}$, $e^{\lambda_3 t}$, $\lambda_1 = -1$, $\lambda_2 = -2$, $\lambda_3 = -3$

(b) $\mathbf{x}(0) = \begin{bmatrix} k \\ -3k \\ 3k \end{bmatrix}$; $k \neq 0$

5.33 (b) $\mathbf{x}(0) = \begin{bmatrix} k \\ -k \end{bmatrix}$; $k \neq 0$

5.34 $e^{\mathbf{A}t} = \begin{bmatrix} 2e^{-t} - e^{-2t} & e^{-t} - e^{-2t} \\ 2e^{-2t} - 2e^{-t} & 2e^{-2t} - e^{-t} \end{bmatrix}$

$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$

5.37 Controllable but not observable.

5.38 (i) Controllable but not observable (ii) Controllable but not observable
(iii) Both controllable and observable (iv) Both controllable and observable
(v) Both controllable and observable

5.39 (i) Observable but not controllable (ii) Controllable but not observable
(iii) Neither controllable nor observable

5.40 (i) $G(s) = \dfrac{1}{s+2}$; state model is not controllable

(ii) $G(s) = \dfrac{s+4}{(s+2)(s+3)}$; state model is not observable

5.41 (a) $\lambda_1 = 1$, $\lambda_2 = -2$, $\lambda_3 = -1$; unstable

(b) $G(s) = \dfrac{1}{(s+1)(s+2)}$; stable

5.42 (a) $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [10 \quad 0]$

(b) $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [20 \quad 10 \quad 0]$

(c) $\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -2 \\ 0 & 1 & -3 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 20 \\ 10 \\ 0 \end{bmatrix}$; $\mathbf{c} = [0 \quad 0 \quad 1]$

6.1 $\mathbf{G}(z) = \dfrac{1}{\Delta}\begin{bmatrix} z^2 & z & 1 \\ -(4z+1) & z(z+3) & z+3 \\ -z & -1 & z(z+3)+4 \end{bmatrix} z;\ \Delta = z^3 + 3z^2 + 4z + 1$

$$\mathbf{H}(z) = \frac{1}{\Delta}\begin{bmatrix} -3z^2 - 7z \\ -7z^2 - 9z + 3 \\ 3z + 7 \end{bmatrix}$$

6.3 $G(z) = \dfrac{2z+2}{z^2 - z + \frac{1}{2}}$

6.4 $\mathbf{G}(z) = \begin{bmatrix} \dfrac{2z+2}{\Delta} & 4 + \dfrac{-4z-14}{\Delta} & \dfrac{-30}{\Delta} \\ \dfrac{z+\frac{1}{2}}{\Delta} & \dfrac{-3z-4}{\Delta} & -2 + \dfrac{-3z-9}{\Delta} \end{bmatrix};\ \Delta = z^2 - z + \frac{1}{2}$

6.5 x_1, x_2, x_3: Outputs of unit delayers, starting at the top of the column of unit delayers and proceeding to the bottom.

$$\mathbf{F} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & 2 \\ 0 & -\frac{1}{2} & -1 \\ 0 & 3 & \frac{1}{3} \end{bmatrix};\ \mathbf{g} = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix};\ \mathbf{c} = [5 \quad 6 \quad -7];\ d = 8$$

6.6 x_1, x_2, x_3: Outputs of unit delayers. x_1 and x_2 in first row, starting at the left and proceeding to the right.

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 2 \\ -12 & -7 & -6 \end{bmatrix};\ \mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix};\ \mathbf{C} = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix};\ \mathbf{D} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

6.7 (i) $\mathbf{F} = \begin{bmatrix} 0 & 1 \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix};\ \mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix};\ \mathbf{c} = [-1 \quad -2];\ d = 3$

(ii) $\mathbf{F} = \begin{bmatrix} 0 & 0 & \frac{3}{4} \\ 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix};\ \mathbf{g} = \begin{bmatrix} 0.5 \\ -3 \\ 4 \end{bmatrix};\ \mathbf{c} = [0 \quad 0 \quad 1];\ d = -2$

6.8 (i) $\mathbf{F} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix};\ \mathbf{g} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix};\ \mathbf{c} = [-1 \quad 2 \quad 1];\ d = 1$

(ii) $\mathbf{F} = \begin{bmatrix} \frac{1}{3} & 1 & 0 \\ 0 & \frac{1}{3} & 1 \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$; $\mathbf{C} = [5 \quad -2 \quad 3]$; $d = 0$

6.9 (i) $\mathbf{F} = \begin{bmatrix} 0 & 0 & -3 \\ 1 & 0 & -7 \\ 0 & 1 & -5 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$; $\mathbf{c} = [0 \quad 0 \quad 1]$; $d = 0$

(ii) $\mathbf{F} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; $\mathbf{c} = [-2 \quad 7]$; $d = 0$

(iii) $\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -3 & -7 & -5 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [2 \quad 1 \quad 0]$; $d = 0$

6.10 $\mathbf{F}^k = \begin{bmatrix} 1.5 - 0.5(3)^k & 0.5[(3)^k - 1] \\ -1.5[(3)^k - 1] & -0.5 + 1.5(3)^k \end{bmatrix}$

6.11 $y(k) = -\frac{17}{6}(-0.2)^k + \frac{22}{9}(-0.8)^k + \frac{25}{18}$; $k \geq 0$

6.12 $y_1(k) = 5\left(\frac{1}{2}\right)^k + 10\left(-\frac{1}{2}\right)^k + 2$; $k \geq 0$; $y_2(k) = 3\left(\frac{1}{2}\right)^k + 2\left(-\frac{1}{2}\right)^k + 1$; $k \geq 0$

6.13 (a) $\lambda_1 = -1$, $\lambda_2 = -1$, $\lambda_3 = -2$; Modes: $(-1)^k$, $k(-1)^{k-1}$, $(-2)^k$

(b) $\mathbf{x}(k) = \begin{bmatrix} k(-1)^{k-1} \\ (-1)^k \\ (-2)^k \end{bmatrix}$

6.14 (a) $G(z) = \mathscr{Z}[G_{h0}(s)G_a(s)] = \dfrac{0.2838z + 0.1485}{z^2 - 1.1353z + 0.1353}$

$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -0.1353 & 1.1353 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [0.1485 \quad 0.2838]$

(b) From controllable companion form continuous-time model:

$\mathbf{F} = \begin{bmatrix} 1 & 0.4323 \\ 0 & 0.1353 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0.2838 \\ 0.4323 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

6.15 $\mathbf{F} = \begin{bmatrix} 0.696 & 0.246 \\ 0.123 & 0.572 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} -0.021 \\ 0.747 \end{bmatrix}$; $\mathbf{c} = [2 \quad -4]$; $d = 6$

6.16 $\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{g}_1 u(k) + \mathbf{g}_2 w(k)$; $\mathbf{F} = \begin{bmatrix} 1 & 0.1 \\ 0 & 0.99 \end{bmatrix}$; $\mathbf{g}_1 = \begin{bmatrix} 0.005 \\ 0.1 \end{bmatrix}$; $\mathbf{g}_2 = \begin{bmatrix} 0 \\ 0.01 \end{bmatrix}$

6.17 $\mathbf{F} = \begin{bmatrix} 0.741 & 0 \\ 0.222 & 0.741 \end{bmatrix}$; $\mathbf{G} = \begin{bmatrix} 259.182 & 0 \\ 36.936 & 259.182 \end{bmatrix}$; $\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

6.18 (a) $0.3679\left[\dfrac{z+0.7181}{(z-1)(z-0.3679)}\right]$

(b) $\dfrac{Y(z)}{R(z)} = \dfrac{0.3679z+0.2642}{z^2-z+0.6321}$; $\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -0.6321 & 1 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [0.2642 \quad 0.3679]$

6.19 (a) $G(z) = \mathcal{Z}[G_{h0}(s)G_a(s)] = \dfrac{0.4512z+0.1809}{z^2-0.3679z}$;

$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0.3679 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [0.1809 \quad 0.4512]$

(b) $\dot{y}(t) = -y(t) + u(t-0.4)$; $x_1(k) = y(k)$; $x_2(k) = u(k-1)$;

$\mathbf{F} = \begin{bmatrix} 0.3679 & 0.1809 \\ 0 & 0 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0.4512 \\ 1 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

6.20 $x_1(k) = x(k)$; $x_2(k) = u(k-3)$; $x_3(k) = u(k-2)$; $x_4(k) = u(k-1)$;

$$\mathbf{F} = \begin{bmatrix} 0.3679 & 0.2387 & 0.3935 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}; \quad \mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

6.21 $x_1 = y$; $x_2 = \dot{y}$; $x_3(k) = u(k-1)$;

$$\mathbf{F} = \begin{bmatrix} 1 & T & \tau_D\,(T-\tau_D/2\,) \\ 0 & 1 & \tau_D \\ 0 & 0 & 0 \end{bmatrix}; \quad \mathbf{g} = \begin{bmatrix} (T-\tau_D\,)^2/2 \\ T-\tau_D \\ 1 \end{bmatrix}; \quad \mathbf{c} = [1 \quad 0 \quad 0]$$

6.23 (a) x_1 = output of lag $1/s$; x_2 = output of lag $1/(s+1)$; x_3 = output of lag $1/(s+2)$.

$$\mathbf{x}(k+1) = \begin{bmatrix} \frac{7}{4} - \frac{1}{2}T - e^{-T} + \frac{1}{4}e^{-2T} & 1 - e^{-T} & \frac{1}{2}(1 + e^{-2T} - 2e^{-T}) \\ -\frac{1}{2} + e^{-T} - \frac{1}{2}e^{-2T} & e^{-T} & e^{-T} - e^{-2T} \\ -\frac{1}{2} + \frac{1}{2}e^{-2T} & 0 & e^{-2T} \end{bmatrix} \mathbf{x}(k)$$

$$+ \begin{bmatrix} -\frac{3}{4} + \frac{1}{2}T + e^{-T} - \frac{1}{4}e^{-2T} \\ \frac{1}{2} - e^{-T} + \frac{1}{2}e^{-2T} \\ \frac{1}{2} - \frac{1}{2}e^{-2T} \end{bmatrix} r(k)$$

6.24 $\mathbf{x}(k+1) = \begin{bmatrix} 0.4 & 0.233 \\ -0.698 & -0.0972 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} 0.2 \\ 0.233 \end{bmatrix} r(k);\ y(k) = [1 \quad 0]\mathbf{x}(k)$

6.25 $\mathbf{x}(k+1) = \begin{bmatrix} 0.6 & 0.233 & 0.2 \\ -0.465 & -0.0972 & 0.233 \\ -1 & 0 & -2 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} r(k)$

6.26 $u(k) = 50e(k) - 41e(k-1)$

$x_1 = y;\ x_2 = \dot{y};\ x_3(k) = e(k-1)$

$$\mathbf{x}(k+1) = \begin{bmatrix} 0.75 & 0.1 & -0.205 \\ -5 & 0.99 & -4.1 \\ -1 & 0 & 0 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} 0.25 \\ 5 \\ 1 \end{bmatrix} r(k)$$

6.27 (a) Both controllable and observable

(b) Both controllable and observable

6.28 $T = n\pi;\ n = 1, 2, \ldots$

6.29 $T \neq n;\ n = 1, 2, 3, \ldots$

6.30 (a) $\lambda_1 = \dfrac{1}{4},\ \lambda_2 = \dfrac{1}{2}$

(b) $G(z) = \dfrac{1}{z - \dfrac{1}{4}}$

(c) Controllable but not observable

7.2 (a) $k_1 = 74,\ k_2 = 25,\ k_3 = 3$

(b) $\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\,\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y;\ \mathbf{m}^T = [3 \quad 7 \quad -1]$

(c) With reference to Fig. 7.7:

$$\hat{\mathbf{x}}_e = \begin{bmatrix} \hat{x}_2 \\ \hat{x}_3 \end{bmatrix};\ \begin{bmatrix} a_{11} & \mathbf{a}_{1e} \\ \mathbf{a}_{e1} & \mathbf{A}_{ee} \end{bmatrix} = \left[\begin{array}{c|cc} 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ -6 & -11 & -6 \end{array}\right];\ \begin{bmatrix} b_1 \\ \mathbf{b}_e \end{bmatrix} = \left[\begin{array}{c} 0 \\ \hline 0 \\ 1 \end{array}\right];\ \mathbf{m} = \begin{bmatrix} -2 \\ 17 \end{bmatrix}$$

7.3 (a) $\mathbf{k} = [3 \quad 7 \quad -1];\ \dot{\mathbf{x}} = (\mathbf{A} - \mathbf{bk})\mathbf{x}$

(b) $\mathbf{m}^T = [74 \quad 25 \quad 3]$

(c) $$\begin{bmatrix} \dot{x}_3 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \left[\begin{array}{c|cc} -6 & 0 & 1 \\ \hline -6 & 0 & 0 \\ -11 & 1 & 0 \end{array}\right]\begin{bmatrix} x_3 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} u = \begin{bmatrix} a_{11} & \mathbf{a}_{1e} \\ \mathbf{a}_{e1} & \mathbf{A}_{ee} \end{bmatrix}\begin{bmatrix} x_3 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ \mathbf{b}_e \end{bmatrix} u$$

With reference to Fig. 7.7:

$$\hat{\mathbf{x}}_e = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix};\ \mathbf{m} = \begin{bmatrix} 16 \\ 4 \end{bmatrix}$$

7.4 $\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\ \hat{\mathbf{x}} + \mathbf{Bu} + \mathbf{m}(y - \mathbf{du})$; $\mathbf{m}^T = [-1 \quad 3]$

7.5 With reference to Fig. 7.7:

$$\hat{\mathbf{x}}_e = \hat{z}_2;\ \begin{bmatrix} a_{11} & a_{1e} \\ a_{e1} & A_{ee} \end{bmatrix} = \left[\begin{array}{c:c} 2 & 2 \\ \hdashline -1 & -1 \end{array}\right];\ \begin{bmatrix} b_1 \\ b_e \end{bmatrix} = \left[\begin{array}{c} 1 \\ \hdashline 0 \end{array}\right];\ m = 4.5;\ \hat{\mathbf{x}} = \begin{bmatrix} y + \hat{z}_2 \\ y + 2\hat{z}_2 \end{bmatrix}$$

7.6 (a) $\mathbf{A} = \begin{bmatrix} 0 & 9 \\ 1 & 0 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 9 \\ 0 \end{bmatrix}$; $\mathbf{c} = [0 \quad 1]$

(b) $\mathbf{k} = \begin{bmatrix} \frac{2}{3} & 3 \end{bmatrix}$

(c) $\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\,\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y$; $\mathbf{m}^T = [81 \quad 12]$

(d) $\mathbf{k} = \begin{bmatrix} \frac{1}{9} & \frac{2}{9} \end{bmatrix}$

7.7 (a) $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

(b) $k_1 = 3\omega_0^2$; $k_2 = 4\omega_0$

(c) $\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\ \hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y$; $\mathbf{m}^T = [20\omega_0 \quad 99\omega_0^2]$

(d) With reference to Fig. 7.7:

$$\hat{x}_e = \hat{x}_2;\ \begin{bmatrix} a_{11} & a_{1e} \\ a_{e1} & A_{ee} \end{bmatrix} = \left[\begin{array}{c:c} 0 & 1 \\ \hdashline -\omega_0^2 & 0 \end{array}\right];\ \begin{bmatrix} b_1 \\ b_e \end{bmatrix} = \left[\begin{array}{c} 0 \\ \hdashline 1 \end{array}\right];\ m = 10\omega_0$$

7.8 (a) $\mathbf{k} = [29.6 \quad 3.6]$

(b) $\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y$; $\mathbf{m}^T = [16 \quad 84.6]$

(c) With reference to Fig. 7.9:

$$\frac{U(s)}{-Y(s)} = D(s) = \frac{778.16s + 3690.72}{s^2 + 19.6s + 151.2}$$

(d) $$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 20.6 & 0 & -29.6 & -3.6 \\ 16 & 0 & -16 & 1 \\ 84.6 & 0 & -93.6 & -3.6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}$$

7.9 (a) $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

(b) $\mathbf{k} = [1 \quad \sqrt{2}]$

(c) $\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{mc})\,\hat{\mathbf{x}} + \mathbf{b}u + \mathbf{m}y$; $\mathbf{m}^T = [5 \quad 25]$

(d) With reference to Fig. 7.9:

$$\frac{U(s)}{-Y(s)} = D(s) = \frac{40.4(s+0.619)}{s^2+6.414s+33.07}$$

(e) With reference to Fig. 7.7:

$$\hat{x}_e = \hat{x}_2\,;\ \begin{bmatrix} a_{11} & a_{1e} \\ a_{e1} & A_{ee} \end{bmatrix} = \left[\begin{array}{c:c} 0 & 1 \\ \hdashline 0 & 0 \end{array}\right];\ \begin{bmatrix} b_1 \\ b_e \end{bmatrix} = \begin{bmatrix} 0 \\ \hdashline 1 \end{bmatrix};\ m = 5$$

(f) $\dfrac{U(s)}{-Y(s)} = D(s) = \dfrac{8.07(s+0.62)}{s+6.41}$

7.10 (a) $k_1 = 4$; $k_2 = 3$; $k_3 = 1$; $N = k_1$

(b) $\mathbf{m}^T = [5 \quad 7 \quad 8]$

7.11 $\mathbf{k} = [-1.4 \quad 2.4]$; $k_i = 1.6$

7.12 $k_1 = 4$; $k_2 = 1.2$; $k_3 = 0.1$

7.13 $k_1 = 1.2$; $k_2 = 0.1$; $k_3 = 4$

7.14 $\mathbf{m}^T = [5 \quad 6 \quad 5]$

7.15 $K_A = 3.6$; $k_2 = 0.11$; $k_3 = 0.33$

7.16 $K_A = 40$; $k_2 = 0.325$; $k_3 = 3$

7.17 $k_1 = a_2/\beta$; $k_2 = (a_1 - \alpha)/\beta$; $N = k_1$

7.18 $k_1 = -0.38$; $k_2 = 0.6$; $k_3 = 6$

7.19 (a) $k_1 = 3$; $k_2 = 1.5$

(b) For a unit-step disturbance, the steady-state error in the output is 1/7.

(c) $k_1 = 2$; $k_2 = 1.5$; $k_3 = 3.5$, Steady-state value of the output = 0

7.20 (a) $\mathbf{k} = [3 \quad 1.5]$ (b) $N = 7$

(c) For a unit-step disturbance, the steady-state error in the output is 1/7.

(d) $k_1 = 2$; $k_2 = 1.5$; $k_3 = 3.5$

7.21 (a) $K = 0.095$; $N = 0.1$ (b) For $A + \delta A = -0.6$, $\omega(\infty) = \dfrac{10}{10.1}r$

(c) $K_1 = 0.105$; $K_2 = 0.5$

7.22 $k_1 = -4$; $k_2 = -3/2$; $k_3 = 0$

7.23 $\hat{\mathbf{x}}(k+1) = (\mathbf{F} - \mathbf{mc})\hat{\mathbf{x}}(k) + \mathbf{Gu}(k) + \mathbf{m}[y(k) - \mathbf{du}(k)]$; $\mathbf{m}^T = \begin{bmatrix} \dfrac{3}{2} & -\dfrac{11}{16} & 0 \end{bmatrix}$

7.24 $\mathbf{k} = [-0.5 \quad -0.2 \quad 1.1]$; $\mathbf{x}(k+1) = (\mathbf{F} - \mathbf{gk})\,\mathbf{x}(k)$

7.25 $\overline{\mathbf{x}}(k+1) = \mathbf{F}\hat{\mathbf{x}}(k) + \mathbf{g}u(k)$

$\hat{\mathbf{x}}(k+1) = \overline{\mathbf{x}}\,(k+1) + \mathbf{m}[y(k+1) - \mathbf{c}\overline{\mathbf{x}}(k+1)]$; $\mathbf{m}^T = [6.25 \quad -5.25]$

7.26 $\begin{bmatrix} x_2(k+1) \\ x_1(k+1) \\ x_3(k+1) \end{bmatrix} = \left[\begin{array}{c:cc} 0 & 0 & 1 \\ \hdashline 1 & 0 & 0 \\ -0.2 & -0.5 & 1.1 \end{array}\right] \begin{bmatrix} x_2(k) \\ x_1(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k) = \begin{bmatrix} f_{11} & \mathbf{f}_{1e} \\ \mathbf{f}_{e1} & \mathbf{F}_{ee} \end{bmatrix} \begin{bmatrix} x_2(k) \\ x_1(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} g_1 \\ \mathbf{g}_e \end{bmatrix} u(k)$

$\hat{\mathbf{x}}_e(k) = [\hat{x}_1(k) \quad \hat{x}_3(k)]^T$

$\hat{\mathbf{x}}_e(k+1) = (\mathbf{F}_{ee} - \mathbf{m}\mathbf{f}_{1e})\hat{\mathbf{x}}_e(k) + (\mathbf{g}_e - \mathbf{m}g_1)u(k) + (\mathbf{f}_{e1} - \mathbf{m}\, f_{11})y(k) + \mathbf{m}y(k+1)$; $\mathbf{m}^T = [0 \quad 1.1]$

7.27 (a) $\mathbf{k} = \begin{bmatrix} \dfrac{111}{76} & -\dfrac{18}{19} \end{bmatrix}$

(b) $\hat{\mathbf{x}}(k+1) = (\mathbf{F} - \mathbf{mc})\,\hat{\mathbf{x}}\,(k) + \mathbf{b}u(k) + \mathbf{m}[y(k) - du(k)]$; $\mathbf{m}^T = [8 \quad -5]$

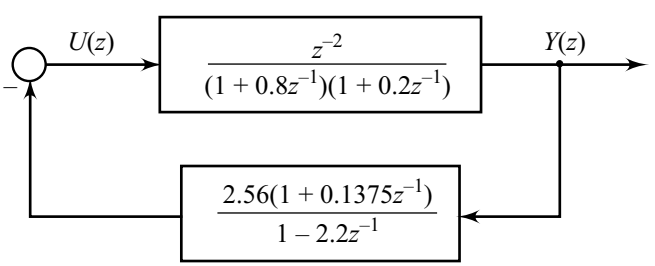
$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \hat{x}_1(k+1) \\ \hat{x}_2(k+1) \end{bmatrix} = \begin{bmatrix} 2 & -1 & -5.84 & 3.79 \\ -1 & 1 & -4.38 & 2.84 \\ 8 & 8 & -11.84 & -5.21 \\ -5 & -5 & -0.38 & 8.84 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \hat{x}_1(k) \\ \hat{x}_2(k) \end{bmatrix}$$

7.28 (a) $\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -0.16 & -1 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

(b) $k_1 = 0.36$; $k_2 = -2.2$

(c) $\hat{x}_2(k+1) = (-1 - m)\,\hat{x}_2(k) + u(k) - 0.16\,y(k) + m\,y(k+1)$; $m = -1$

(d)



7.29 (a) $\mathbf{x}(k) = \mathbf{Q}\mathbf{z}(k)$; $\mathbf{Q} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$; $u = -0.36\,z_1(k) + 2.2z_2(k)$

(b) $\hat{z}_2\,(k+1) = (-1 - m)\hat{z}_2(k) + u(k) - 0.16\,y(k) + my(k+1)$; $m = -1$

(c) $\dfrac{U(z)}{-Y(z)} = D(z) = \dfrac{2.56(1+0.1375z^{-1})}{1-2.2z^{-1}}$

7.30 (a) $\mathbf{F} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0.005 \\ 0.1 \end{bmatrix}$

(b) $k_1 = 13$; $k_2 = 3.95$

(c) $\hat{\mathbf{x}}(k+1) = (\mathbf{F} - \mathbf{mc})\hat{\mathbf{x}}(k) + \mathbf{g}u(k) + \mathbf{m}y(k)$; $\mathbf{m}^T = [2 \quad 10]$

(d) $\dfrac{U(z)}{-Y(z)} = D(z) = \dfrac{65.5(z-0.802)}{z^2 + 0.46z + 0.26}$

7.31 (a) $\mathbf{F} = \begin{bmatrix} 1 & 0.0952 \\ 0 & 0.905 \end{bmatrix}$; $\mathbf{g} = \begin{bmatrix} 0.00484 \\ 0.0952 \end{bmatrix}$; $\mathbf{c} = [1 \quad 0]$

(b) $k_1 = 105.1$; $k_2 = 14.625$

(c) $\hat{\mathbf{x}}(k+1) = (\mathbf{F} - \mathbf{mc})\,\hat{\mathbf{x}}(k) + \mathbf{g}u(k) + \mathbf{m}y(k)$; $\mathbf{m}^T = [1.9 \quad 8.6]$

(d) $\hat{x}_2(k+1) = (0.905 - 0.0952m)\hat{x}_2(k) + (0.0952 - 0.00484m)u(k) - my(k) + my(k+1)$; $m = 9.51$

7.32 (a) $y(k+1) = 0.368y(k) + 0.632u(k) + 0.632w(k)$

(b) $K = 0.3687$; $N = 1.37$

(c) Steady-state error for a unit-step disturbance is 0.73.

(d) $K_1 = 0.553$; $K_2 = 2.013$

8.1 Asymptotically stable in-the-large.

8.2 Unstable.

8.3 Equilibrium state $\mathbf{x}^e = [2 \quad 0]^T$ is asymptotically stable.

8.4 $K > 0$

8.5 $0 < K < 8$

8.6 Asymptotically stable.

8.7 Asymptotically stable.

8.8 $\mathbf{K} = [1 \quad \sqrt{3}]$

8.9 $\mathbf{K} = [1 \quad \sqrt{2}]$

8.10 Sufficient conditions not satisfied.
$u = -[-1 \quad 4]\mathbf{x}$; optimal closed-loop system is asymptotically stable.

8.11 Asymptotically stable optimal solution does not exist.

8.12 $u = -x_1 - 0.23x_2 + r$; r = desired output $y_d = 1$

8.13 $u = -x_1 - \sqrt{6}x_2$

8.14 $u = -x_1 - x_2$; $\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{MC})\hat{\mathbf{x}} + \mathbf{B}u + \mathbf{M}y$; $\mathbf{M}^T = [5 \quad 4]$

8.15 $k_1 = \sqrt{2}$; $k_2 = 0.275$; $J_0 = 93.2375$

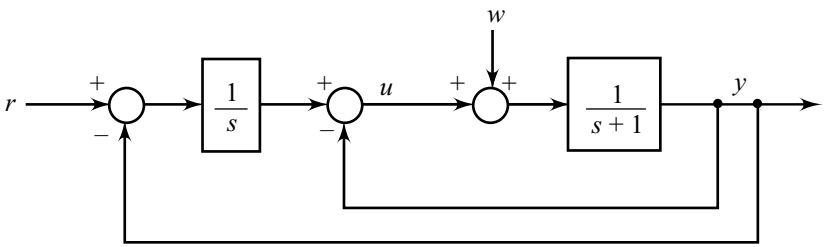
8.16 $\rho = 0.1$; $\mathbf{K} = [3.1623 \quad 0.5968]$; Poles: $-0.6377, -4.9592$
$\rho = 0.01$; $\mathbf{K} = [10 \quad 1.7082]$; Poles: $-2.2361, -4.4721$
$\rho = 0.001$; $\mathbf{K} = [31.6228 \quad 4.3939]$; Poles: $-4.6970 \pm j3.0921$

8.17 (a) $K = \sqrt{2} - 1$ (b) $N = \sqrt{2}$

(c) Steady-state error to unit-step disturbance is $1/\sqrt{2}$

(d) $K = K_1 = 1$

(e)



8.18 (a) $K = 0.095$; $N = 0.1$ (b) For $A + \delta A = -0.6$, $\omega(\infty) = \dfrac{10}{10.1}r$

(c) $K = 0.105$; $K_1 = 0.1$

8.19 $u(k) = -0.178x(k)$

8.20 (a) $K = 0.277$ (b) $y(\infty) = 0.217r$

(c) $N = 1.277$

8.21 (a) $K = 0.2$ (b) $N = 0.45$

(c) For $F + \delta F = 0.3$, $x(\infty) = \dfrac{0.9}{1.1}r$

9.2 $\dfrac{8M}{\pi}$; 1 rad/sec; $y(t) = \dfrac{-8M}{\pi}\sin t$

9.3 0.3; 10 rad/sec

9.4 $\Delta < 0.131$

9.6 4.25; $\sqrt{2}$ rad/sec, stable limit cycle

9.7 3.75; 1 rad/sec

9.8 (a) Stable node; (0, 0) point in $(y, \dot{y})$-plane

(b) Stable node; (1,0) point in $(y, \dot{y})$-plane

(c) Unstable focus; (2,0) point in $(y, \dot{y})$-plane

9.9 For $\theta = 0$, the singular point is center; for $\theta = \pi$, it is saddle.

9.10 (i) Singularity (1,0) in $(y, \dot{y})$-plane is a center

(ii) Singularity (1,0) in $(y, \dot{y})$-plane is a stable focus.

9.11 (a) For $-0.1 < x_1 < 0.1$, $\ddot{x}_1 + \dot{x}_1 + 7x_1 = 0$

For $x_1 > 0.1$, $\ddot{x}_1 + \dot{x}_1 + 0.7 = 0$

For $x_1 < -0.1$, $\ddot{x}_1 + \dot{x}_1 - 0.7 = 0$

(b) Isocline equations:

$x_2 = \dfrac{-7x_1}{m+1}$; $-0.1 < x_1 < 0.1$

$x_2 = \dfrac{-0.7}{m+1}$; $x_1 > 0.1$

$x_2 = \dfrac{0.7}{m+1}$; $x_1 < -0.1$

(c) A singular point at the origin

9.12 (a) For $-0.1 < x_1 < 0.1$, $\ddot{x}_1 + \dot{x}_1 = 0$

For $x_1 < -0.1$, $\ddot{x}_1 + \dot{x}_1 + 7(x_1 + 0.1) = 0$

(b) Isocline equations:

$m = -1$; $-0.1 < x_1 < 0.1$

For $x_1 > 0.1$, $\ddot{x}_1 + \dot{x}_1 + 7\,(x_1 - 0.1) = 0$

$$x_2 = \frac{-7x_1 - 0.7}{m+1};\ x_1 < -0.1$$

$$x_2 = \frac{-7x_1 + 0.7}{m+1};\ x_1 > 0.1$$

(c) Singular point at $(x_1 = \pm\, 0.1,\ x_2 = 0)$

9.13 (a) $\ddot{x}_1 + \dot{x}_1 + 0.7 \operatorname{sgn} x_1 = 0$

(b) Isocline equations:

$$x_2 = \frac{-0.7}{m+1};\ x_1 > 0$$

$$x_2 = \frac{0.7}{m+1};\ x_1 < 0$$

(c) No singular points

9.14 (a) For $-0.1 < x_1 < 0.1$, $\ddot{x}_1 + \dot{x}_1 = 0$

For $x_1 > 0.1$, $\ddot{x}_1 + \dot{x}_1 + 0.7 = 0$

For $x_1 < -0.1$, $\ddot{x}_1 + \dot{x}_1 - 0.7 = 0$

(b) Isocline equations:

$m = -1 \quad ;\ -0.1 < x_1 < 0.1$

$$x_2 = \frac{-0.7}{m+1};\ x_1 > 0.1$$

$$x_2 = \frac{0.7}{m+1};\ x_1 < -0.1$$

(c) No singular points

9.15 (a) $\ddot{x}_1 + 0.1 \operatorname{sgn} \dot{x}_1 + x_1 = 0$

(b) Isocline equation:

$$x_2 = \frac{-x_1 - 0.1 \operatorname{sgn} x_2}{m}$$

(c) Singular point at $(x_1 = \mp\, 0.1,\ x_2 = 0)$

9.16 Steady-state error to unit-step input = – 0.2 rad; Maximum steady-state error = ± 0.3 rad.

9.17 Deadzone helps to reduce system oscillations, and introduces steady-state error.

9.18 Saturation has a slowing down effect on the transient.

9.20 The system has good damping and no oscillations but exhibits chattering behavior. Steady-state error is zero.

9.21 (b) (i) Deadzone provides damping; oscillations get reduced.
(ii) Deadzone introduces steady-state error; maximum error = ± 0.2.
(c) By derivative-control action, (i) settling time is reduced, but (ii) chattering effect appears.

9.23 $x_1 = e$, $x_2 = \dot{e}$; Switching curve: $x_1 = -x_2 + \dfrac{x_2}{|x_2|} \ln\left(1 + \dfrac{x_2^2}{|x_2|}\right)$

9.24 $|x_1| < 1$; origin is the equilibrium state.

9.25 Asymptotically stable in-the-large; origin is the equilibrium state.

9.26 Asymptotically stable; origin is the equilibrium state.

9.27 $1 > 2x_1x_2$; origin is the equilibrium state

9.28 Locally unstable

9.29 Origin is asymptotically stable for $x_1^2 + x_2^2 < 1$

9.30 (i) Globally asymptotically stable (ii) Unstable (iii) Asymptotically stable for $0 < x_2^2 < K_1/K_2$

10.1 $K_P = 100$; $K_D = 14.14$

10.4 $a_1 = 0.8187$; $b_1 = 1.0877$

10.5 $a_1 = 0.9845$; $a_2 = -0.1222$; $b_1 = 0.0579$; $b_2 = 0.1011$

10.6 $a_1 = 1.5$; $a_2 = 2$; $b_1 = 1$; $b_2 = 3$

11.1 (a) $w_1 = 1$, $w_2 = 3$ (b) $w_1 = 0.86442$, $w_2 = 2.8892$

(c) 0.21626

11.4 $w = 0.36$; $w_0 = 0.666$

11.6 (b) $\mathbf{w}(1) = [0.974 \quad -0.948 \quad 0 \quad 0.526]^T$;
$\mathbf{w}(2) = [0.974 \quad -0.956 \quad 0.002 \quad 0.531]^T$
$\mathbf{w}(3) = [0.947 \quad -0.929 \quad 0.016 \quad 0.505]^T$

(c) $\mathbf{w} = [0.9482 \quad -0.9298 \quad 0.0155 \quad 0.505]^T$

11.7 (b) $w_{10} = 1.00043$; $w_{11} = 3.00043$;
$w_{12} = 4$; $w_{20} = -5.9878$; $w_{21} = 6.0123$;
$w_{22} = 5$; $v_0 = -3.9078$; $v_1 = 2.012$;
$v_2 = 4.0061$

(c) With initial weights, $\hat{y} = 0.51$; With updated weights, $\hat{y} = 0.5239$

11.9 (a) $v_l(k+1) = v_l(k) + \eta(y - \hat{y})\left[\dfrac{4}{(e^a + e^{-a})^2}\right] z_\ell$

$$w_{li}(k+1) = w_{li}(k) + \eta(y - \hat{y})\left[\frac{4}{(e^a + e^{-a})^2}\right] v_\ell \left[\frac{e^{-a_\ell}}{(1 + e^{-a_\ell})^2}\right] x_i$$

(b) $v_1 = 0.095474$, $v_2 = 0.195694$, $v_3 = 0.095716$; $w_{11} = 0.199895$,
$w_{12} = 0.100084$, $w_{21} = 0.399785$, $w_{22} = 0.600172$, $w_{31} = 0.299895$, $w_{32} = 0.500084$

12.1 (a) Supporting Interval: [1 3] (b) Support: [1 3]

α-cut interval: $\left[2 - \sqrt{0.5} \quad 2 + \sqrt{0.5}\right]$ α-cut: [1.5 2.5]

12.2 (a) $\sigma = \dfrac{1}{\sqrt{2\pi}}$; $\mu = 2$; Support unbounded ($-\infty \quad \infty$); α-cut: [1.53 2.47]

(b) Support: ($-\infty \quad \infty$); α-cut: [1 3]

12.3 $a = 3$ ft; $b = 6$ ft; $c = 9$ ft; Support: [3 9]; Cross point: 4.5

12.4 (a) $\mu_{\underset{\sim}{A}} = \begin{cases} \dfrac{x+1}{5} & ; -1 \le x \le 4 \\ 1 & ; 4 \le x < 5 \\ \dfrac{x-9}{-4} & ; 5 \le x \le 9 \end{cases}$ (b) It is normal and convex.

12.5 (a) (i) Yes (ii) Yes (iii) No

(b) Support: [64 78]; Cross points: 68, 75; α-cut$\big|_{\alpha=0.2}$: [65.6 76.8]; α-cut$\big|_{\alpha=0.4}$: [67.2 75.6]

12.8 (i) $\mu_{\underset{\sim}{A}}(4) = 0.75$ (ii) $\mu_{\underset{\sim}{B}}(3) = 0.5$

(iii) $\mu_{\underset{\sim}{A}\times\underset{\sim}{B}}(x, y) = 0.5$ (iv) $\mu_{\underset{\sim}{A}\to\underset{\sim}{B}}(x, y) = 0.5$

12.11 $z^* = 6.76$

12.12 1857.28

12.13 $\mu_{\underset{\sim}{C}'}(z) = \max\left\{\min\left(\frac{2}{3}, \mu_{\underset{\sim}{C}_1}(z)\right), \min\left(\frac{1}{3}, \mu_{\underset{\sim}{C}_2}(z)\right)\right\}$

$z^*_{COA} = 4.7$

12.14 (i) 0.25, 0.62

(ii) $\mu_{agg}(z) = \max\left\{\min\left(0.25, \mu_{\underset{\sim}{PL}}(z)\right), \min\left(0.62, \mu_{\underset{\sim}{PM}}(z)\right)\right\}$

(iii) $z^* = 53.18$

12.15 34.40

12.16 Rules 2, 3, and 4 are active; $u^* = 2.663$

12.17 67.4

12.18 2.3

12.22 2.974

12.23 12.04

Index