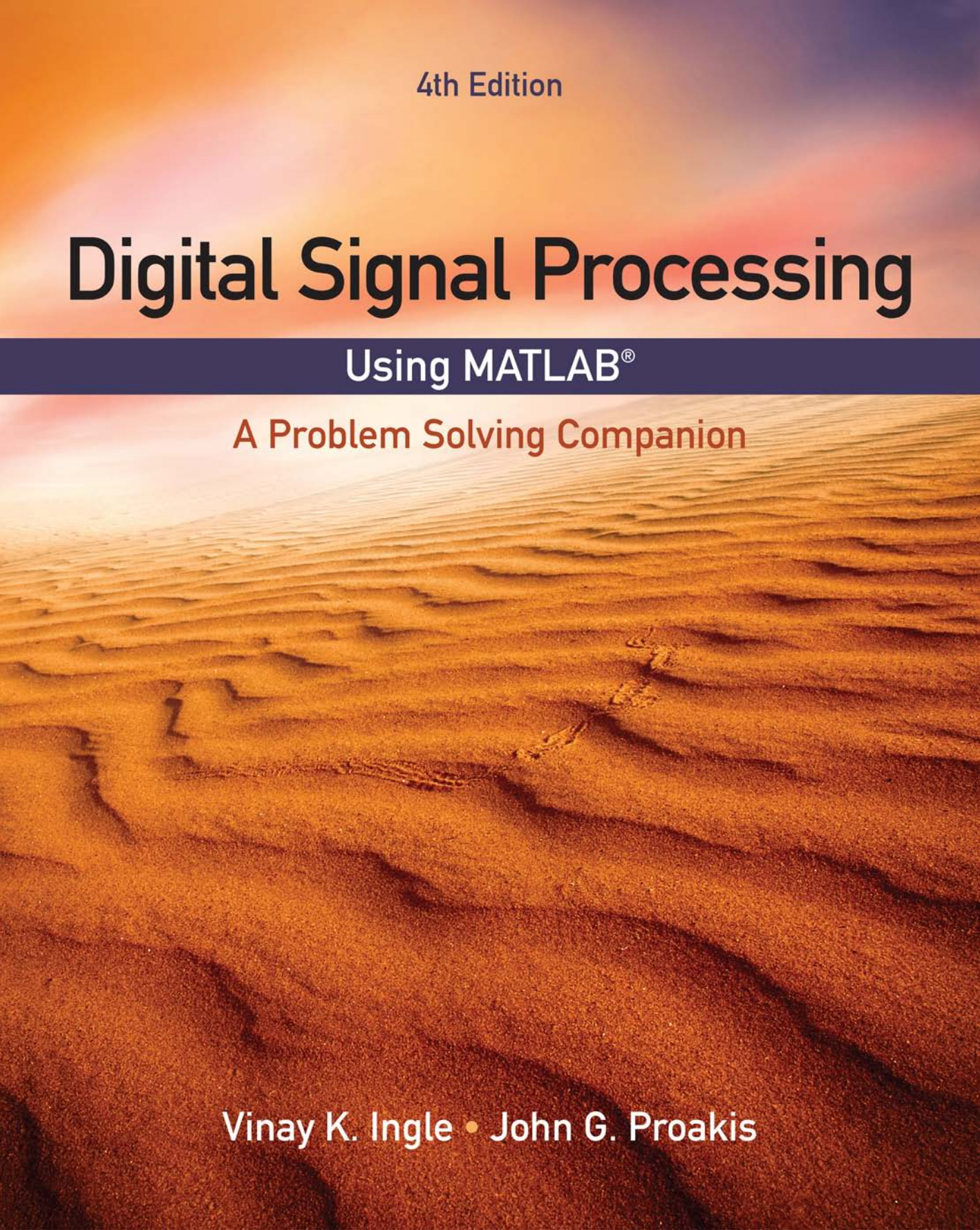
4th Edition

Digital Signal Processing

Using MATLAB®

A Problem Solving Companion



Vinay K. Ingle • John G. Proakis

Digital Signal Processing Using MATLAB®

A Problem Solving Companion

Fourth Edition

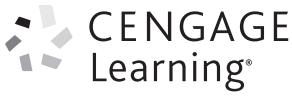
Digital Signal Processing Using MATLAB®

A Problem Solving Companion

Fourth Edition

Vinay K. Ingle
John G. Proakis
Northeastern University

Australia • Brazil • Mexico • Singapore • United Kingdom • United States



***Digital Signal Processing Using MATLAB®: A Problem Solving Companion*, Fourth Edition**
Vinay K. Ingle and John G. Proakis

Product Director, Global Engineering: Timothy L. Anderson

Media Assistant: Ashley Kaupert

Product Assistant: Teresa Versaggi

Marketing Manager: Kristin Stine

Director, Content and Media Production: Sharon L. Smith

Senior Content Project Manager: Jennifer Risden

Production Service: MPS Limited

Copyeditor: Richard Camp

Proofreader: Jennifer Grubba

Indexer: Larry Sweazy

Compositor: MPS Limited

Senior Art Director: Michelle Kunkler

Internal Designer: Carmela Periera

Cover Designer: Jennifer Wahi

Cover Image: Zeljko Radojko/Shutterstock.com

Intellectual Property
Analyst: Christine Myaskovsky
Project Manager: Sarah Shainwald

Text and Image Permissions Researcher: Kristiina Paul

Manufacturing Planner: Doug Wilke

Library of Congress Control Number: 2015944167

ISBN: 978-1-305-63512-8

Cengage Learning
20 Channel Center Street
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com**.

Cengage Learning products are represented in Canada by Nelson Education Ltd.

To learn more about Cengage Learning Solutions, visit **www.cengage.com/engineering**.

Purchase any of our products at your local college store or at our preferred online store **www.cengagebrain.com**.

MATLAB is a registered trademark of The MathWorks, 3 Apple Hill Drive, Natick, MA.

Printed in the United States of America
Print Number: 01 Print Year: 2015

Contents

*Chapters 13–15 are available for download from the Instructor's Companion Website at www.cengage.com/login.

*Chapters 13–15 are available for download from the Instructor's Companion Website at www.cengage.com/login.

Preface

Since the beginning of the 1980s, we have witnessed a revolution in computer technology and an explosion in user-friendly applications. This revolution is still continuing today, with low-cost laptop systems that rival the performance of expensive workstations. This technological prowess should be brought to bear on the educational process and, in particular, on effective teaching that can result in enhanced learning. This problem-solving companion book on digital signal processing (DSP) makes a contribution toward reaching that goal. The fourth edition continues our innovative approach of blending MATLAB®-based learning with traditional teaching to advanced DSP topics such as optimal and adaptive filters.

The teaching methods in signal processing have changed over the years from the simple "lecture-only" format to a more integrated "lecture-laboratory" environment in which practical hands-on issues are taught using DSP hardware. However, for effective teaching of DSP, the lecture component must also make extensive use of computer-based explanations, examples, and exercises. For the past three decades, the MATLAB software developed by *The MathWorks, Inc.* has established itself as the de facto standard for numerical computation in the signal-processing community and as a platform of choice for algorithm development. There are several reasons for this development, but the most important reason is that MATLAB is available on practically all computing platforms. In this book, we have made an attempt at integrating MATLAB with traditional topics in DSP so that it can be used to explore difficult topics and solve problems to gain insight. Many problems or design algorithms in DSP require considerable amount of computation. It is for these that MATLAB provides a convenient tool so that multiple scenarios can be tried with ease. Such an approach can enhance the learning process.

SCOPE OF THE BOOK

This book is primarily intended for use as a problem-solving companion book in senior-level undergraduate or first-year graduate courses on DSP. Although we assume that the student (or user) is familiar with the fundamentals of MATLAB, we have provided a brief introduction to MATLAB in Chapter 1. This book is not written as a textbook in DSP, because of the ready availability of excellent textbooks. What we have tried to do is to provide enough depth to the material augmented by MATLAB functions and examples so that the presentation is consistent, logical, and enjoyable. Therefore, this book can also be used as a self-study guide by anyone interested in DSP.

WHAT IS NEW IN THE FOURTH EDITION

- A new Chapter 13 provides a review on random variables and random processes, including bandpass processes. Extensive use of MATLAB examples makes these topics easier to understand.
- A new Chapter 14 discusses linear prediction and optimal (or Wiener) filters, preparing students for graduate studies.
- A new Chapter 15 deals with theory and applications of adaptive filters. This chapter contains easy-to-understand LMS and RLS algorithms with an extensive set of practical applications, including system identification, echo and noise cancellation, and adaptive arrays. All algorithms and applications are explained and analyzed using MATLAB.
- The coverage of lattice/ladder filters has moved from Chapter 6 to Chapter 14 for a more logical presentation of information.
- All MATLAB functions and scripts have been tested and updated so that they can execute on MATLAB-2014b version and later. Similarly, all MATLAB plots have been recreated with superior graphic elements.
- We have trimmed many included MATLAB scripts from their plotting commands to streamline their appearance and to reduce unnecessary printing. However, all scripts and functions will be made available in their entirety on the book website.

ORGANIZATION OF THE BOOK

The first ten chapters of this book discuss traditional material typically covered in an introductory course on DSP. The next two chapters are

presented as applications in DSP, with emphasis on MATLAB-based projects. The last three chapters deal with advanced material in DSP and are intended for graduate studies. In order to keep the size and the cost of the book down, we have provided these last three chapters through our book's instructor website. Information on how to obtain these chapters is provided in the next section. The following is a list of chapters and a brief description of their contents.

Chapter 1, Introduction: This chapter introduces readers to the discipline of signal processing and presents several applications of digital signal processing, including musical sound processing, echo generation, echo removal, and digital reverberation. A brief introduction to MATLAB is also provided.

Chapter 2, Discrete-Time Signals and Systems: This chapter provides a brief review of discrete-time signals and systems in the time domain. Appropriate use of MATLAB functions is demonstrated.

Chapter 3, The Discrete-Time Fourier Analysis: This chapter discusses discrete-time signal and system representation in the frequency domain. Sampling and reconstruction of analog signals are also presented.

Chapter 4, The z-Transform: This chapter provides signal and system description in the complex frequency domain. MATLAB techniques are introduced to analyze z-transforms and to compute inverse z-transforms. Solutions of difference equations using the z-transform and MATLAB are provided.

Chapter 5, The Discrete Fourier Transform: This chapter is devoted to the computation of the Fourier transform and its efficient implementation. The discrete Fourier series is used to introduce the discrete Fourier transform, and several of its properties are demonstrated using MATLAB. Topics such as fast convolution and fast Fourier transform are thoroughly discussed.

Chapter 6, Implementation of Discrete-Time Filters: This chapter discusses several structures for the implementation of digital filters. Several useful MATLAB functions are developed for the determination and implementation of these structures. In addition to considering various filter structures, we also treat quantization effects when finite-precision arithmetic is used in the implementation of IIR and FIR filters.

Chapter 7, FIR Filter Design: This chapter and the next introduce the important topic of digital filter design. Three important design techniques for FIR filters—namely, window design, frequency sampling design, and the equiripple filter design—are discussed. Several design examples are provided using MATLAB.

Chapter 8, IIR Filter Design: Included in this chapter are techniques used in IIR filter design. The chapter begins with the treatment of some

basic filter types—namely, digital resonators, notch filters, comb filters, allpass filters, and digital sinusoidal oscillators. This is followed by a brief description of the characteristics of three widely used analog filters. Transformations are described for converting these prototype analog filters into different frequency-selective digital filters. The chapter concludes with several IIR filter designs using MATLAB.

Chapter 9, Sampling Rate Conversion: This chapter treats the important problem of sampling rate conversion in digital signal processing. Topics treated include decimation and interpolation by integer factors, sampling rate conversion by a rational factor, and polyphase filter structures for sampling rate conversion.

Chapter 10, Round-Off Effects in Digital Filters: The focus of this chapter is on the effects of finite-precision arithmetic to the filtering aspects in signal processing. Quantization noise introduced in analog-to-digital conversion is characterized statistically, and the quantization effects in finite-precision multiplication and additions are also modeled statistically. The effects of these errors in the filter output are characterized as correlated errors, called limit cycles, and as uncorrelated errors, called round-off noise.

Chapter 11, Applications in Adaptive Filtering: This chapter is the first of two chapters on projects using MATLAB. Included is an introduction to the theory and implementation of adaptive FIR filters with projects in system identification, interference suppression, narrowband frequency enhancement, and adaptive equalization.

Chapter 12, Applications in Communications: This chapter focuses on several projects dealing with waveform representation and coding and with digital communications. Included is a description of pulse-code modulation (PCM), differential PCM (DPCM) and adaptive DPCM (ADPCM), delta modulation (DM) and adaptive DM (ADM), linear predictive coding (LPC), generation and detection of dual-tone multifrequency (DTMF) signals, and a description of signal detection applications in binary communications and spread-spectrum communications.

Chapter 13, Random Processes: This is the first of the last three chapters that are available online through the website for the book. In this chapter, we provide a brief review of analytical concepts in random signals that model waveform variations and provide sound techniques to calculate the response of linear filters to random signals. We begin by defining probability functions and statistical averages, and continue with pairs of random variables. These concepts are extended to random signals, in terms of second-order statistics, and then delve into stationary and ergodic processes, correlation functions, and power spectra. We apply this theory to processing of random signals through LTI systems

using both the time and frequency domains. Finally, we discuss a few representative random processes, including Gaussian, Markov, white noise, and filtered noise processes.

Chapter 14, Linear Prediction and Optimum Linear Filters: In this chapter, we treat the problem of optimum filter design from a statistical viewpoint. The filters are constrained to be linear, and the optimization criterion is based on the minimization of the mean square error. We discuss the design of optimum filters for linear prediction, which has applications in speech signal processing, image processing, and noise suppression in communication systems. This design technique requires the solution of a set of linear equations with special symmetry. We describe two algorithms, Levinson–Durbin and Schur, which provide the solution to the equations through computationally efficient procedures that exploit the symmetry properties. The last section of this chapter treats an important class of optimum filters called Wiener filters, which are widely used in many applications involving the estimation of signals corrupted with additive noise.

Chapter 15, Adaptive Filters: The focus of this chapter is on *adaptive filters*, which have adjustable coefficients for use in applications in which the filter coefficients cannot be designed a priori due to unknown or changing statistics. We begin with several practical applications in which adaptive filters have been successfully used in the estimation of signals corrupted by noise and other interference. Adaptive filters incorporate algorithms that allow the filter coefficients to adapt to changes in the signal statistics. We describe two basic algorithms: the least-mean-square (LMS) algorithm, which is based on gradient optimization for determining the coefficients, and the class of recursive least-squares (RLS) algorithms.

ABOUT THE ONLINE RESOURCES

This book is an outgrowth of our teaching of a MATLAB-based undergraduate DSP course over several years. Most of the MATLAB functions discussed in this book were developed for this course. These functions are collected in the book toolbox called DSPUM_v4 and are available online on the book's companion website. Many examples in the book contain MATLAB scripts. Similarly, many figure plots were created using MATLAB scripts. All these scripts are made available at the companion website for the benefit of students and instructors. Students should study these scripts to gain insight into MATLAB procedures. We will appreciate any comments, corrections, or compact coding of these functions

and scripts. Solutions to problems and the associated script files will be made available to instructors adopting the book through the companion website.

To access the book's companion website and all additional course materials, please visit www.cengage.com/login. After signing in, search for the ISBN of your title (from the back cover of your book) using the search box at the top of the page. This will take you to the companion site where these resources can be found.

Further information about MATLAB and related publications may be obtained from:

The MathWorks, Inc.
Natick, MA 01760
Phone: (508) 647-7000 Fax: (508) 647-7001
E-mail: info@mathworks.com
WWW: http://www.mathworks.com

ACKNOWLEDGMENTS

We are indebted to numerous students in our undergraduate DSP course at Northeastern University who provided us a forum to test teaching ideas using MATLAB and who endured our constant emphasis on MATLAB. Many efficient MATLAB functions used in this book were developed by some of these students. We are also indebted to reviewers of the original edition, whose constructive criticism resulted in a better presentation of the material: Abeer A. H. Alwan, University of California, Los Angeles; Steven Chin, Catholic University; Prof. Huaichen, Xidian University, P. R. China; and Joel Trussel, North Carolina State University. The following reviewers provided additional encouragement, numerous refinements, and useful comments for the second edition: Jyotsna Bapat, Fairleigh Dickinson University; David Clark, California State Polytechnic University; Artyom Grigoryan, University of Texas, San Antonio; Tao Li, University of Florida; and Zixiang Xiong, Texas A & M University. Based on their use of the second edition, the following reviewers provided several suggestions, changes, and modifications that led to the third edition: Kalyan Mondal, Fairleigh Dickinson University; Artyom M. Grigoryan, University of Texas at San Antonio; A. David Salvia, Pennsylvania State University; Matthew Valenti, West Virginia University; and Christopher J. James, University of Southampton, UK. Finally, the fourth edition was motivated by the constructive feedback and comments provided by Wasfy B. Mikhael, University of Central Florida; Hongbin Li, Stevens

Institute of Technology; Robert Paz, New Mexico State University; and Ramakrishnan Sundaram, Gannon University. We sincerely thank all of them.

We would also like to take this opportunity to acknowledge several people at Cengage Learning without whom this project would not have been possible. We thank the Product Director, Timothy Anderson, who oversees the Global Engineering publishing program at Cengage, for encouraging the fourth edition. Media Assistant Ashley Kaupert worked on the revisions and helped see the fourth edition through development to production. This project could not have been completed within time limits without their constant push. Senior Content Project Manager Jennifer Risden oversaw the book's production process. We thank them all for their professional help. Finally, we express our sincere gratitude to Richard Camp for his diligent copy editing and everyone at Cengage Learning who aided in the development of this edition.

Vinay K. Ingle
John G. Proakis
Boston, Massachusetts

CHAPTER 1

Introduction

During the past several decades, the field of digital signal processing (DSP) has grown to be important, both theoretically and technologically. A major reason for its success in industry is the development and use of low-cost software and hardware. New technologies and applications in various fields are now taking advantage of DSP algorithms. This will lead to a greater demand for electrical and computer engineers with a background in DSP. Therefore, it is necessary to make DSP an integral part of any electrical engineering curriculum.

Three decades ago an introductory course on DSP was given mainly at the graduate level. It was supplemented by computer exercises on filter design, spectrum estimation, and related topics using mainframe (or mini) computers. However, considerable advances in personal computers and software during the past three decades have made it necessary to introduce a DSP course to undergraduates. Since DSP applications are primarily algorithms that are implemented either on a DSP processor [36] or in software, a fair amount of programming is required. Using interactive software, such as MATLAB, it is now possible to place more emphasis on learning new and difficult concepts than on programming algorithms. Interesting practical examples can be discussed, and useful problems can be explored.

With this philosophy in mind, we have developed this book as a *companion book* (to traditional textbooks like [71, 79]) in which MATLAB is an integral part in the discussion of topics and concepts. We have chosen MATLAB as the programming tool primarily because of its wide availability on computing platforms in many universities across the world. Furthermore, a low-cost student version of MATLAB has been available for several years, placing it among the least expensive software products

for educational purposes. We have treated MATLAB as a computational and programming toolbox containing several tools (sort of a super calculator with several keys) that can be used to explore and solve problems and, thereby, enhance the learning process.

This book is written at an introductory level in order to introduce undergraduate students to the exciting and practical field of DSP. We emphasize that this is not a textbook in the traditional sense but a companion book in which more attention is given to problem solving and hands-on experience with MATLAB. Similarly, it is not a tutorial book in MATLAB. We assume that the student is familiar with MATLAB and is currently taking a course in DSP. The book provides basic analytical tools needed to process real-world signals (a.k.a. analog signals) using digital techniques. We deal mostly with discrete-time signals and systems, which are analyzed in both the time and the frequency domains. The analysis and design of processing structures called *filters* and *spectrum analyzers* are among some of the most important aspects of DSP and are treated in great detail in this book. Similarly, the topics of finite word-length effects on filter performance as well as on filter output and the sampling-rate conversion between two DSP systems are of practical significance. These are also treated extensively in this book. To further our philosophy of MATLAB-based learning to advanced topics taught in graduate courses, we have also included some material from statistical and adaptive signal processing areas such as random signals, linear prediction, optimal filters, and adaptive filters.

In this chapter, we provide a brief overview of DSP and an introduction to MATLAB.

1.1 OVERVIEW OF DIGITAL SIGNAL PROCESSING

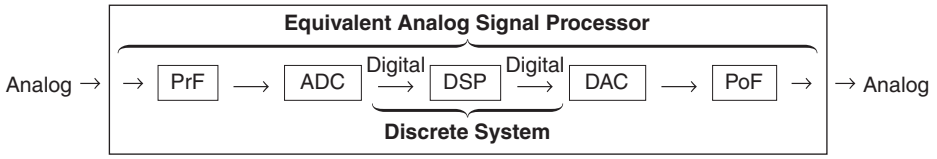
In this modern world, we are surrounded by all kinds of signals in various forms. Some of the signals are natural, but most of the signals are man-made. Some signals are necessary (speech), some are pleasant (music), while many are unwanted or unnecessary in a given situation. In an engineering context, signals are carriers of information, both useful and unwanted. Therefore, extracting or enhancing the useful information from a mix of conflicting information is the simplest form of signal processing. More generally, signal processing is an operation designed for extracting, enhancing, storing, and transmitting useful information. The distinction between useful and unwanted information is often subjective as well as objective. Hence signal processing tends to be application dependent.

1.1.1 HOW ARE SIGNALS PROCESSED?

The signals that we encounter in practice are mostly analog signals. These signals, which vary continuously in time and amplitude, are processed using electrical networks containing active and passive circuit elements. This approach is known as analog signal processing (ASP)—for example, radio and television receivers.

Analog signal: $x_a(t) \longrightarrow$ [Analog signal processor] $\longrightarrow y_a(t)$:Analog signal

They can also be processed using digital hardware containing adders, multipliers, and logic elements or using special-purpose microprocessors. However, one needs to convert analog signals into a form suitable for digital hardware. This form of the signal is called a digital signal. It takes one of the finite number of values at specific instances in time, and hence it can be represented by binary numbers, or bits. The processing of digital signals is called DSP; in block diagram form it is represented by



The various block elements are discussed as follows.

PrF: This is a prefilter or an antialiasing filter, which conditions the analog signal to prevent aliasing.

ADC: This is an analog-to-digital converter, which produces a stream of binary numbers from analog signals.

Digital Signal Processor: This is the heart of DSP and can represent a general-purpose computer or a special-purpose processor, or digital hardware, and so on.

DAC: This is the inverse operation to the ADC, called a digital-to-analog converter, which produces a staircase waveform from a sequence of binary numbers, a first step toward producing an analog signal.

PoF: This is a postfilter to smooth out staircase waveform into the desired analog signal.

It appears from the above two approaches to signal processing, analog and digital, that the DSP approach is the more complicated, containing more components than the "simpler looking" ASP. Therefore, one might ask, Why process signals digitally? The answer lies in the many advantages offered by DSP.

1.1.2 ADVANTAGES OF DSP OVER ASP

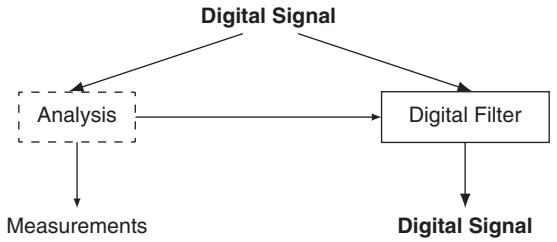
A major drawback of ASP is its limited scope for performing complicated signal-processing applications. This translates into nonflexibility in processing and complexity in system designs. All of these generally lead to expensive products. On the other hand, using a DSP approach, it is possible to convert an inexpensive personal computer into a powerful signal processor. Some important advantages of DSP are these:

1. Systems using the DSP approach can be developed using software running on a general-purpose computer. Therefore, DSP is relatively convenient to develop and test, and the software is portable.
2. DSP operations are based solely on additions and multiplications, leading to extremely stable processing capability—for example, stability independent of temperature.
3. DSP operations can easily be modified in real time, often by simple programming changes or by reloading of registers.
4. DSP has lower cost due to VLSI technology, which reduces costs of memories, gates, microprocessors, and so forth.

The principal disadvantage of DSP is the limited speed of operations due to the DSP hardware, especially at very high frequencies. Primarily because of its advantages, DSP is now becoming a first choice in many technologies and applications, such as consumer electronics, communications, wireless telephones, and medical imaging.

1.1.3 TWO IMPORTANT CATEGORIES OF DSP

Most DSP operations can be categorized as being either signal *analysis* tasks or signal *filtering* tasks:



Signal analysis This task deals with the measurement of signal properties. It is generally a frequency-domain operation. Some of its applications are

- spectrum (frequency and/or phase) analysis,
- speech recognition,
- speaker verification, and
- target detection.

Signal filtering This task is characterized by the signal-in signal-out situation. The systems that perform this task are generally called *filters*. It is usually (but not always) a time-domain operation. Some of the applications are

- removal of unwanted background noise,
- removal of interference,
- separation of frequency bands, and
- shaping of the signal spectrum.

In some applications, such as voice synthesis, a signal is first analyzed to study its characteristics, which are then used in digital filtering to generate a synthetic voice.

1.2 A BRIEF INTRODUCTION TO MATLAB

MATLAB is an interactive, matrix-based system for scientific and engineering numeric computation and visualization. Its strength lies in the fact that complex numerical problems can be solved easily and in a fraction of the time required by a programming language such as Fortran or C. It is also powerful in the sense that, with its relatively simple programming capability, MATLAB can be easily extended to create new commands and functions.

MATLAB is available in a number of computing environments: PCs running all flavors of Windows, Apple Macs running OS-X, UNIX/Linux workstations, and parallel computers. The basic MATLAB program is further enhanced by the availability of numerous toolboxes (collections of specialized functions in specific topics) over the years. The information in this book generally applies to all these environments. In addition to the basic MATLAB product, the Signal Processing toolbox (SP toolbox) is required for this book. The original development of the book was done using the professional version 3.5 running under DOS. The MATLAB scripts and functions described in the book were later extended and made compatible with the present version of MATLAB. Furthermore, through the services of www.cengagebrain.com, every effort will be made to preserve this compatibility under future versions of MATLAB.

In this section, we will undertake a brief review of MATLAB. The scope and power of MATLAB go far beyond the few topics discussed in this section. For more detailed tutorial-based discussion, students and readers new to MATLAB should also consult several excellent reference books available in the literature, including [29], [35], and [76]. The information given in all these references, along with the online MATLAB's `help` facility, usually is sufficient to enable readers to use this book.

The best approach to become familiar with MATLAB is to open a MATLAB session and experiment with various operators, functions, and commands until their use and capabilities are understood. Then one can progress to writing simple MATLAB scripts and functions to execute a sequence of instructions to accomplish an analytical goal.

1.2.1 GETTING STARTED

The interaction with MATLAB is through the command window of its graphical user interface (GUI). In the command window, the user types MATLAB instructions, which are executed instantaneously, and the results are displayed in the window. In the MATLAB command window, the characters "`>>`" indicate the prompt that is waiting for the user to type a command to be executed. For example,

```
>> command;
```

means an instruction `command` has been issued at the MATLAB prompt. If a semicolon (;) is placed at the end of a command, then all output from that command is suppressed. Multiple commands can be placed on the same line, separated by semicolons. Comments are marked by the percent sign (`%`), in which case MATLAB ignores anything to the right of the sign. The comments allow the reader to follow code more easily. The integrated help manual provides help for every command through the fragment

```
>> help command;
```

which will provide information on the inputs, outputs, usage, and functionality of the command. A complete listing of commands sorted by functionality can be obtained by typing `help` at the prompt.

There are three basic elements in MATLAB: numbers, variables, and operators. In addition, punctuation marks (`,`, `;`, `:`, etc.) have special meanings.

Numbers MATLAB is a high-precision numerical engine and can handle all types of numbers—that is, integers, real numbers, complex numbers, among others—with relative ease. For example, the real number 1.23 is represented as simply `1.23`, while the real number 4.56×10^7 can be written as `4.56e7`. The imaginary number $\sqrt{-1}$ is denoted either by `1i` or `1j`, although in this book we will use the symbol `1j`. Hence the complex number whose real part is 5 and whose imaginary part is 3 will be written as `5+1j*3`. Other constants preassigned by MATLAB are `pi` for π, `inf` for ∞, and `NaN` for not a number (e.g., $0/0$). These preassigned constants are very important and, to avoid confusion, should not be redefined by users.

Variables In MATLAB, which stands for MATrix LABoratory, the basic variable is a matrix, or an array. Hence, when MATLAB operates on this variable, it operates on all its elements. This is what makes it a powerful and an efficient engine. MATLAB now supports multidimensional arrays; we will discuss only up to two-dimensional arrays of numbers.

1. **Matrix:** A matrix is a two-dimensional set of numbers arranged in rows and columns. Numbers can be real- or complex-valued.
2. **Array:** This is another name for matrix. However, operations on arrays are treated differently from those on matrices. This difference is very important in implementation.

The following are four types of matrices (or arrays).

- **Scalar:** This is a 1×1 matrix or a single number that is denoted by the *variable* symbol, that is, lowercase italic typeface like

$$a = a_{11}$$

- **Column vector:** This is an $(N \times 1)$ matrix or a vertical arrangement of numbers. It is denoted by the *vector* symbol, that is, lowercase bold typeface like

$$\mathbf{x} = [x_{i1}]_{i:1,\ldots,N} = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{N1} \end{bmatrix}$$

A typical vector in linear algebra is denoted by the column vector.

- **Row vector:** This is a $(1 \times M)$ matrix or a horizontal arrangement of numbers. It is also denoted by the vector symbol, that is,

$$\mathbf{y} = [y_{1j}]_{j=1,\ldots,M} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1M} \end{bmatrix}$$

A one-dimensional discrete-time signal is typically represented by an array as a row vector.

- **General matrix:** This is the most general case of an $(N \times M)$ matrix and is denoted by the matrix symbol, that is, uppercase bold typeface like

$$\mathbf{A} = [a_{ij}]_{i=1,\ldots,N;j=1,\ldots,m} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}$$

This arrangement is typically used for two-dimensional discrete-time signals or images.

MATLAB does not distinguish between an array and a matrix except for operations. The following assignments denote indicated matrix types in MATLAB:

`a = [3]` is a scalar,
`x = [1,2,3]` is a row vector,
`y = [1;2;3]` is a column vector, and
`A = [1,2,3;4,5,6]` is a matrix.

MATLAB provides many useful functions to create special matrices. These include `zeros(M,N)` for creating a matrix of all zeros, `ones(M,N)` for creating matrix of all ones, `eye(N)` for creating an $N \times N$ identity matrix, and so on. Consult MATLAB's help manual for a complete list.

Operators MATLAB provides several arithmetic and logical operators, some of which follow. For a complete list, MATLAB's help manual should be consulted.

`=`	assignment	`==`	equality
`+`	addition	`-`	subtraction or minus
`*`	multiplication	`.*`	array multiplication
`^`	power	`.^`	array power
`/`	division	`./`	array division
`<>`	relational operators	`&`	logical AND
`|`	logical OR	`~`	logical NOT
`'`	transpose	`.'`	array transpose

We now provide a more detailed explanation on some of these operators.

1.2.2 MATRIX OPERATIONS

Following are the most useful and important operations on matrices.

- **Matrix addition and subtraction:** These are straightforward operations that are also used for array addition and subtraction. Care must be taken that the two matrix operands be *exactly* the same size.
- **Matrix conjugation:** This operation is meaningful only for complex-valued matrices. It produces a matrix in which all imaginary parts are negated. It is denoted by $\mathbf{A}^*$ in analysis and by `conj(A)` in MATLAB.
- **Matrix transposition:** This is an operation in which every row (column) is turned into column (row). Let $\mathbf{X}$ be an $(N \times M)$ matrix. Then

$$\mathbf{X}' = [x_{ji}]; \quad j = 1, \ldots, M, \; i = 1, \ldots, N$$

is an $(M \times N)$ matrix. In MATLAB, this operation has one additional feature. If the matrix is real-valued, then the operation produces the

usual transposition. However, if the matrix is complex-valued, then the operation produces a complex-conjugate transposition. To obtain just the transposition, we use the array operation of conjugation, that is, `A.'` will do just the transposition.

- **Multiplication by a scalar:** This is a simple straightforward operation in which each element of a matrix is scaled by a constant, that is,

$$ab \Rightarrow \texttt{a*b} \text{ (scalar)}$$

$$a\mathbf{x} \Rightarrow \texttt{a*x} \text{ (vector or array)}$$

$$a\mathbf{X} \Rightarrow \texttt{a*X} \text{ (matrix)}$$

  This operation is also valid for an array scaling by a constant.
- **Vector-vector multiplication:** In this operation, one has to be careful about matrix dimensions to avoid invalid results. The operation produces either a scalar or a matrix. Let $\mathbf{x}$ be an $(N \times 1)$ vector and $\mathbf{y}$ be a $(1 \times M)$ vector. Then

$$\texttt{x} * \texttt{y} \Rightarrow \mathbf{xy} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \begin{bmatrix} y_1 & \cdots & y_M \end{bmatrix} = \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_M \\ \vdots & \ddots & \vdots \\ x_N y_1 & \cdots & x_N y_M \end{bmatrix}$$

  produces a matrix. If $M = N$, then

$$\texttt{y} * \texttt{x} \Rightarrow \mathbf{yx} = \begin{bmatrix} y_1 & \cdots & y_M \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix} = x_1 y_1 + \cdots + x_M y_M$$

- **Matrix-vector multiplication:** If the matrix and the vector are compatible (i.e., the number of matrix-columns is equal to the vector-rows), then this operation produces a column vector:

$$\texttt{y = A*x} \Rightarrow \mathbf{y} = \mathbf{Ax} = \begin{bmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NM} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

- **Matrix-matrix multiplication:** Finally, if two matrices are compatible, then their product is well defined. The result is also a matrix with the number of rows equal to that of the first matrix and the number of columns equal to that of the second matrix. Note that the order in matrix multiplication is very important.