Understanding Digital Signal Processing

Third Edition

This page intentionally left blank

Understanding Digital Signal Processing

Third Edition

Richard G. Lyons

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data
Lyons, Richard G., 1948-
Understanding digital signal processing / Richard G. Lyons.—3rd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-702741-9 (hardcover : alk. paper)
1. Signal processing—Digital techniques. I. Title.
TK5102.9.L96 2011
621.382'2—dc22 2010035407

ISBN-13: 978-0-13-702741-5
ISBN-10: 0-13-702741-9

Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.
Fourth printing, August 2012

I dedicate this book to my daughters, Julie and Meredith—I wish I could go with you; to my mother, Ruth, for making me finish my homework; to my father, Grady, who didn't know what he started when he built that workbench in the basement; to my brother Ray for improving us all; to my brother Ken who succeeded where I failed; to my sister Nancy for running interference for us; and to the skilled folks on the USENET newsgroup comp.dsp. They ask the good questions and provide the best answers. Finally, to Sigi Pardula (Batgirl): Without your keeping the place running, this book wouldn't exist.

This page intentionally left blank

Contents

Preface

This book is an expansion of previous editions of *Understanding Digital Signal Processing*. Like those earlier editions, its goals are (1) to help beginning students understand the theory of digital signal processing (DSP) and (2) to provide practical DSP information, not found in other books, to help working engineers/scientists design and test their signal processing systems. Each chapter of this book contains new information beyond that provided in earlier editions.

It's traditional at this point in the preface of a DSP textbook for the author to tell readers why they should learn DSP. I don't need to tell you how important DSP is in our modern engineering world. You already know that. I'll just say that the future of electronics is DSP, and with this book you will not be left behind.

FOR INSTRUCTORS

This third edition is appropriate as the text for a one- or two-semester undergraduate course in DSP. It follows the DSP material I cover in my corporate training activities and a signal processing course I taught at the University of California Santa Cruz Extension. To aid students in their efforts to learn DSP, this third edition provides additional explanations and examples to increase its tutorial value. To test a student's understanding of the material, homework problems have been included at the end of each chapter. (For qualified instructors, a Solutions Manual is available from Prentice Hall.)

FOR PRACTICING ENGINEERS

To help working DSP engineers, the changes in this third edition include, but are not limited to, the following:

- Practical guidance in building discrete differentiators, integrators, and matched filters
- Descriptions of statistical measures of signals, variance reduction by way of averaging, and techniques for computing real-world signal-to-noise ratios (SNRs)
- A significantly expanded chapter on sample rate conversion (multirate systems) and its associated filtering
- Implementing fast convolution (FIR filtering in the frequency domain)
- IIR filter scaling
- Enhanced material covering techniques for analyzing the behavior and performance of digital filters
- Expanded descriptions of industry-standard binary number formats used in modern processing systems
- Numerous additions to the popular "Digital Signal Processing Tricks" chapter

FOR STUDENTS

Learning the fundamentals, and how to speak the language, of digital signal processing does not require profound analytical skills or an extensive background in mathematics. All you need is a little experience with elementary algebra, knowledge of what a sinewave is, this book, and enthusiasm. This may sound hard to believe, particularly if you've just flipped through the pages of this book and seen figures and equations that look rather complicated. The content here, you say, looks suspiciously like material in technical journals and textbooks whose meaning has eluded you in the past. Well, this is not just another book on digital signal processing.

In this book I provide a gentle, but thorough, explanation of the theory and practice of DSP. The text is not written so that you *may* understand the material, but so that you *must* understand the material. I've attempted to avoid the traditional instructor–student relationship and have tried to make reading this book seem like talking to a friend while walking in the park. I've used just enough mathematics to help you develop a fundamental understanding of DSP theory and have illustrated that theory with practical examples.

I have designed the homework problems to be more than mere exercises that assign values to variables for the student to plug into some equation in order to compute a result. Instead, the homework problems are designed to be as educational as possible in the sense of expanding on and enabling further investigation of specific aspects of DSP topics covered in the text. Stated differently, the homework problems are not designed to induce "death by algebra," but rather to improve your understanding of DSP. Solving the problems helps you become proactive in your own DSP education instead of merely being an inactive recipient of DSP information.

THE JOURNEY

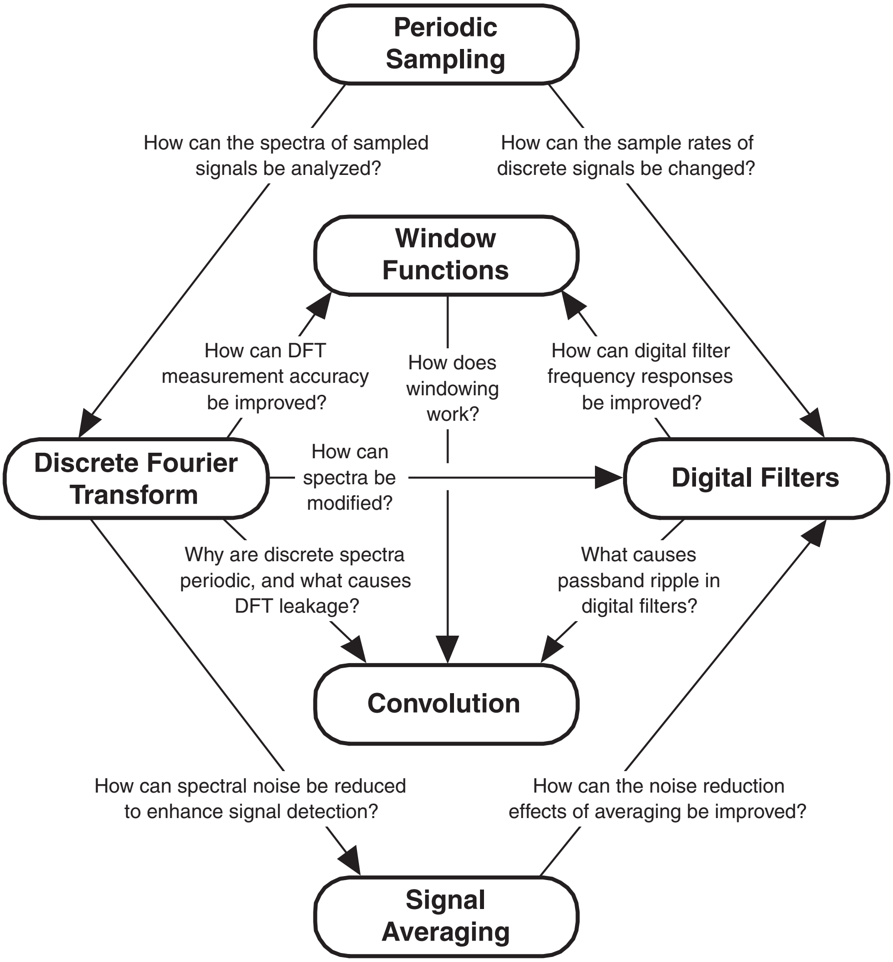
Learning digital signal processing is not something you accomplish; it's a journey you take. When you gain an understanding of one topic, questions arise that cause you to investigate some other facet of digital signal processing.[†] Armed with more knowledge, you're likely to begin exploring further aspects of digital signal processing much like those shown in the diagram on page xviii. This book is your tour guide during the first steps of your journey.

You don't need a computer to learn the material in this book, but it would certainly help. DSP simulation software allows the beginner to verify signal processing theory through the time-tested trial and error process.[‡] In particular, software routines that plot signal data, perform the fast Fourier transforms, and analyze digital filters would be very useful.

As you go through the material in this book, don't be discouraged if your understanding comes slowly. As the Greek mathematician Menaechmus curtly remarked to Alexander the Great, when asked for a quick explanation of mathematics, "There is no royal road to mathematics." Menaechmus was confident in telling Alexander the only way to learn mathematics is through careful study. The same applies to digital signal processing. Also, don't worry if you need to read some of the material twice. While the concepts in this book are not as complicated as quantum physics, as mysterious as the lyrics of the song "Louie Louie," or as puzzling as the assembly instructions of a metal shed, they can become a little involved. They deserve your thoughtful attention. So, go slowly and read the material twice if necessary; you'll be glad you did. If you show persistence, to quote Susan B. Anthony, "Failure is impossible."

[†]"You see I went on with this research just the way it led me. This is the only way I ever heard of research going. I asked a question, devised some method of getting an answer, and got—a fresh question. Was this possible, or that possible? You cannot imagine what this means to an investigator, what an intellectual passion grows upon him. You cannot imagine the strange colourless delight of these intellectual desires" (Dr. Moreau—infamous physician and vivisectionist from H.G. Wells' *Island of Dr. Moreau,* 1896).

[‡]"One must learn by doing the thing; for though you think you know it, you have no certainty until you try it" (Sophocles, 496–406 B.C.).



COMING ATTRACTIONS

Chapter 1 begins by establishing the notation used throughout the remainder of the book. In that chapter we introduce the concept of discrete signal sequences, show how they relate to continuous signals, and illustrate how those sequences can be depicted in both the time and frequency domains. In addition, Chapter 1 defines the operational symbols we'll use to build our signal processing system block diagrams. We conclude that chapter with a brief introduction to the idea of linear systems and see why linearity enables us to use a number of powerful mathematical tools in our analysis.

Chapter 2 introduces the most frequently misunderstood process in digital signal processing, periodic sampling. Although the concept of sampling a continuous signal is not complicated, there are mathematical subtleties in the process that require thoughtful attention. Beginning gradually with simple examples of lowpass sampling, we then proceed to the interesting subject of bandpass sampling. Chapter 2 explains and quantifies the frequency-domain ambiguity (aliasing) associated with these important topics.

Chapter 3 is devoted to one of the foremost topics in digital signal processing, the discrete Fourier transform (DFT) used for spectrum analysis. Coverage begins with detailed examples illustrating the important properties of the DFT and how to interpret DFT spectral results, progresses to the topic of windows used to reduce DFT leakage, and discusses the processing gain afforded by the DFT. The chapter concludes with a detailed discussion of the various forms of the transform of rectangular functions that the reader is likely to encounter in the literature.

Chapter 4 covers the innovation that made the most profound impact on the field of digital signal processing, the fast Fourier transform (FFT). There we show the relationship of the popular radix 2 FFT to the DFT, quantify the powerful processing advantages gained by using the FFT, demonstrate why the FFT functions as it does, and present various FFT implementation structures. Chapter 4 also includes a list of recommendations to help the reader use the FFT in practice.

Chapter 5 ushers in the subject of digital filtering. Beginning with a simple lowpass finite impulse response (FIR) filter example, we carefully progress through the analysis of that filter's frequency-domain magnitude and phase response. Next, we learn how window functions affect, and can be used to design, FIR filters. The methods for converting lowpass FIR filter designs to bandpass and highpass digital filters are presented, and the popular Parks-McClellan (Remez) Exchange FIR filter design technique is introduced and illustrated by example. In that chapter we acquaint the reader with, and take the mystery out of, the process called convolution. Proceeding through several simple convolution examples, we conclude Chapter 5 with a discussion of the powerful convolution theorem and show why it's so useful as a qualitative tool in understanding digital signal processing.

Chapter 6 is devoted to a second class of digital filters, infinite impulse response (IIR) filters. In discussing several methods for the design of IIR filters, the reader is introduced to the powerful digital signal processing analysis tool called the z-transform. Because the z-transform is so closely related to the continuous Laplace transform, Chapter 6 starts by gently guiding the reader from the origin, through the properties, and on to the utility of the Laplace transform in preparation for learning the z-transform. We'll see how IIR filters are designed and implemented, and why their performance is so different from that of FIR filters. To indicate under what conditions these filters should be used, the chapter concludes with a qualitative comparison of the key properties of FIR and IIR filters.

Chapter 7 introduces specialized networks known as *digital differentiators*, *integrators*, and *matched filters*. In addition, this chapter covers two specialized digital filter types that have not received their deserved exposure in traditional DSP textbooks. Called *interpolated FIR* and *frequency sampling* filters, providing improved lowpass filtering computational efficiency, they belong in our arsenal of filter design techniques. Although these are FIR filters, their introduction is delayed to this chapter because familiarity with the z-transform (in Chapter 6) makes the properties of these filters easier to understand.

Chapter 8 presents a detailed description of quadrature signals (also called *complex* signals). Because quadrature signal theory has become so important in recent years, in both signal analysis and digital communications implementations, it deserves its own chapter. Using three-dimensional illustrations, this chapter gives solid physical meaning to the mathematical notation, processing advantages, and use of quadrature signals. Special emphasis is given to quadrature sampling (also called *complex down-conversion*).

Chapter 9 provides a mathematically gentle, but technically thorough, description of the Hilbert transform—a process used to generate a quadrature (complex) signal from a real signal. In this chapter we describe the properties, behavior, and design of practical Hilbert transformers.

Chapter 10 presents an introduction to the fascinating and useful process of sample rate conversion (changing the effective sample rate of discrete data sequences through decimation or interpolation). Sample rate conversion—so useful in improving the performance and reducing the computational complexity of many signal processing operations—is essentially an exercise in lowpass filter design. As such, polyphase and cascaded integrator-comb filters are described in detail in this chapter.

Chapter 11 covers the important topic of signal averaging. There we learn how averaging increases the accuracy of signal measurement schemes by reducing measurement background noise. This accuracy enhancement is called *processing gain*, and the chapter shows how to predict the processing gain associated with averaging signals in both the time and frequency domains. In addition, the key differences between coherent and incoherent averaging techniques are explained and demonstrated with examples. To complete that chapter the popular scheme known as *exponential averaging* is covered in some detail.

Chapter 12 presents an introduction to the various binary number formats the reader is likely to encounter in modern digital signal processing. We establish the precision and dynamic range afforded by these formats along with the inherent pitfalls associated with their use. Our exploration of the critical subject of binary data word width (in bits) naturally leads to a discussion of the numerical resolution limitations of analog-to-digital (A/D) converters and how to determine the optimum A/D converter word size for a given application. The problems of data value overflow roundoff errors are covered along with a statistical introduction to the two most popular remedies for overflow, truncation and rounding. We end that chapter by covering the interesting subject of floating-point binary formats that allow us to overcome most of the limitations induced by fixed-point binary formats, particularly in reducing the ill effects of data overflow.

Chapter 13 provides the literature's most comprehensive collection of *tricks of the trade* used by DSP professionals to make their processing algorithms more efficient. These techniques are compiled into a chapter at the end of the book for two reasons. First, it seems wise to keep our collection of tricks in one chapter so that we'll know where to find them in the future. Second, many of these clever schemes require an understanding of the material from the previous chapters, making the last chapter an appropriate place to keep our arsenal of clever tricks. Exploring these techniques in detail verifies and reiterates many of the important ideas covered in previous chapters.

The appendices include a number of topics to help the beginner understand the nature and mathematics of digital signal processing. A comprehensive description of the arithmetic of complex numbers is covered in Appendix A, and Appendix B derives the often used, but seldom explained, closed form of a geometric series. The subtle aspects and two forms of time reversal in discrete systems (of which zero-phase digital filtering is an application) are explained in Appendix C. The statistical concepts of mean, variance, and standard deviation are introduced and illustrated in Appendix D, and Appendix E provides a discussion of the origin and utility of the logarithmic decibel scale used to improve the magnitude resolution of spectral representations. Appendix F, in a slightly different vein, provides a glossary of the terminology used in the field of digital filters. Appendices G and H provide supplementary information for designing and analyzing specialized digital filters. Appendix I explains the computation of Chebyshev window sequences.

ACKNOWLEDGMENTS

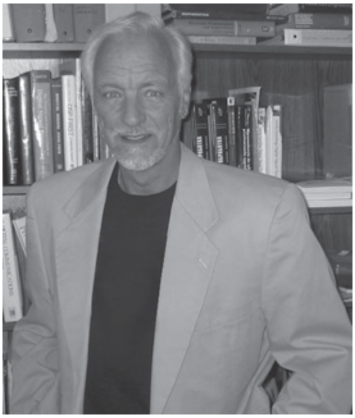
Much of the new material in this edition is a result of what I've learned from those clever folk on the USENET newsgroup comp.dsp. (I could list a dozen names, but in doing so I'd make 12 friends and 500 enemies.) So, I say thanks to my DSP pals on comp.dsp for teaching me so much signal processing theory.

In addition to the reviewers of previous editions of this book, I thank Randy Yates, Clay Turner, and Ryan Groulx for their time and efforts to help me improve the content of this book. I am especially indebted to my eagle-eyed mathematician friend Antoine Trux for his relentless hard work to both enhance this DSP material and create a homework Solutions Manual.

As before, I thank my acquisitions editor, Bernard Goodwin, for his patience and guidance, and his skilled team of production people, project editor Elizabeth Ryan in particular, at Prentice Hall.

If you're still with me this far in this Preface, I end by saying I had a ball writing this book and sincerely hope you benefit from reading it. If you have any comments or suggestions regarding this material, or detect any errors no matter how trivial, please send them to me at R.Lyons@ieee.org. I promise I will reply to your e-mail.

About the Author

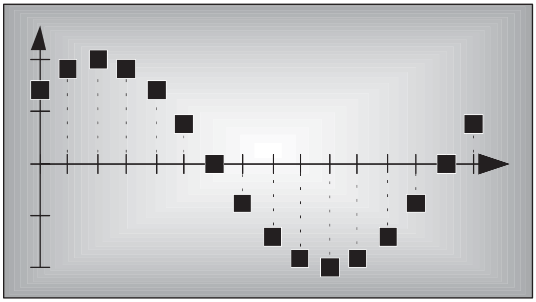


Richard Lyons is a consulting systems engineer and lecturer with Besser Associates in Mountain View, California. He has been the lead hardware engineer for numerous signal processing systems for both the National Security Agency (NSA) and Northrop Grumman Corp. Lyons has taught DSP at the University of California Santa Cruz Extension and authored numerous articles on DSP. As associate editor for the *IEEE Signal Processing Magazine* he created, edits, and contributes to the magazine's "DSP Tips & Tricks" column.

This page intentionally left blank

CHAPTER ONE

Discrete Sequences and Systems



Digital signal processing has never been more prevalent or easier to perform. It wasn't that long ago when the fast Fourier transform (FFT), a topic we'll discuss in Chapter 4, was a mysterious mathematical process used only in industrial research centers and universities. Now, amazingly, the FFT is readily available to us all. It's even a built-in function provided by inexpensive spreadsheet software for home computers. The availability of more sophisticated commercial signal processing software now allows us to analyze and develop complicated signal processing applications rapidly and reliably. We can perform spectral analysis, design digital filters, develop voice recognition, data communication, and image compression processes using software that's interactive both in the way algorithms are defined and how the resulting data are graphically displayed. Since the mid-1980s the same integrated circuit technology that led to affordable home computers has produced powerful and inexpensive hardware development systems on which to implement our digital signal processing designs.† Regardless, though, of the ease with which these new digital signal processing development systems and software can be applied, we still need a solid foundation in understanding the basics of digital signal processing. The purpose of this book is to build that foundation.

In this chapter we'll set the stage for the topics we'll study throughout the remainder of this book by defining the terminology used in digital signal process-

† During a television interview in the early 1990s, a leading computer scientist stated that had automobile technology made the same strides as the computer industry, we'd all have a car that would go a half million miles per hour and get a half million miles per gallon. The cost of that car would be so low that it would be cheaper to throw it away than pay for one day's parking in San Francisco.

ing, illustrating the various ways of graphically representing discrete signals, establishing the notation used to describe sequences of data values, presenting the symbols used to depict signal processing operations, and briefly introducing the concept of a linear discrete system.

1.1 DISCRETE SEQUENCES AND THEIR NOTATION

In general, the term *signal processing* refers to the science of analyzing time-varying physical processes. As such, signal processing is divided into two categories, analog signal processing and digital signal processing. The term *analog* is used to describe a waveform that's continuous in time and can take on a continuous range of amplitude values. An example of an analog signal is some voltage that can be applied to an oscilloscope, resulting in a continuous display as a function of time. Analog signals can also be applied to a conventional spectrum analyzer to determine their frequency content. The term *analog* appears to have stemmed from the analog computers used prior to 1980. These computers solved linear differential equations by means of connecting physical (electronic) differentiators and integrators using old-style telephone operator patch cords. That way, a continuous voltage or current in the actual circuit was *analogous* to some variable in a differential equation, such as speed, temperature, air pressure, etc. (Although the flexibility and speed of modern-day digital computers have since made analog computers obsolete, a good description of the short-lived utility of analog computers can be found in reference [1].) Because present-day signal processing of continuous radio-type signals using resistors, capacitors, operational amplifiers, etc., has nothing to do with analogies, the term *analog* is actually a misnomer. The more correct term is *continuous signal processing* for what is today so commonly called analog signal processing. As such, in this book we'll minimize the use of the term *analog signals* and substitute the phrase *continuous signals* whenever appropriate.

The term *discrete-time signal* is used to describe a signal whose independent time variable is quantized so that we know only the value of the signal at discrete instants in time. Thus a discrete-time signal is not represented by a continuous waveform but, instead, a sequence of values. In addition to quantizing time, a discrete-time signal quantizes the signal amplitude. We can illustrate this concept with an example. Think of a continuous sinewave with a peak amplitude of 1 at a frequency f_o described by the equation

$$x(t) = \sin(2\pi f_o t). \tag{1–1}$$

The frequency f_o is measured in hertz (Hz). (In physical systems, we usually measure frequency in units of hertz. One Hz is a single oscillation, or cycle, per second. One kilohertz (kHz) is a thousand Hz, and a megahertz (MHz) is

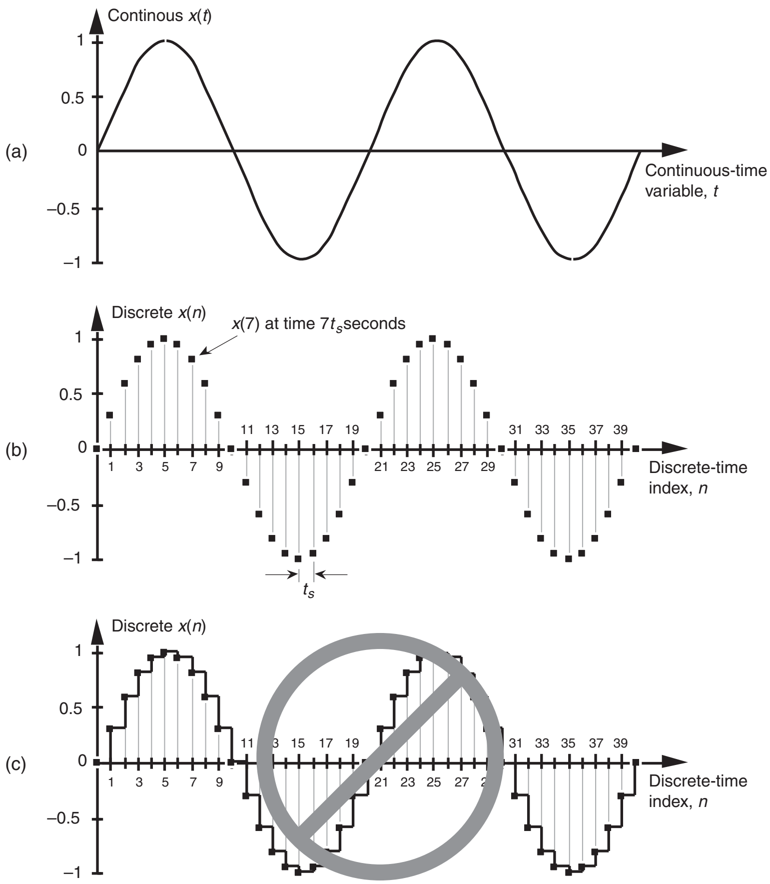


Figure 1–1 A time-domain sinewave: (a) continuous waveform representation; (b) discrete sample representation; (c) discrete samples with connecting lines.

one million Hz.†) With t in Eq. 1–1 representing time in seconds, the f_ot factor has dimensions of cycles, and the complete $2\pi f_ot$ term is an angle measured in radians.

Plotting Eq. (1–1), we get the venerable continuous sinewave curve shown in Figure 1–1(a). If our continuous sinewave represents a physical volt-

† The dimension for frequency used to be *cycles/second*; that's why the tuning dials of old radios indicate frequency as kilocycles/second (kcps) or megacycles/second (Mcps). In 1960 the scientific community adopted hertz as the unit of measure for frequency in honor of the German physicist Heinrich Hertz, who first demonstrated radio wave transmission and reception in 1887.

age, we could *sample* it once every t_s seconds using an analog-to-digital converter and represent the sinewave as a sequence of discrete values. Plotting those individual values as dots would give us the discrete waveform in Figure 1–1(b). We say that Figure 1–1(b) is the "discrete-time" version of the continuous signal in Figure 1–1(a). The independent variable t in Eq. (1–1) and Figure 1–1(a) is continuous. The independent *index* variable n in Figure 1–1(b) is discrete and can have only integer values. That is, index n is used to identify the individual elements of the discrete sequence in Figure 1–1(b).

Do not be tempted to draw lines between the dots in Figure 1–1(b). For some reason, people (particularly those engineers experienced in working with continuous signals) want to connect the dots with straight lines, or the stair-step lines shown in Figure 1–1(c). Don't fall into this innocent-looking trap. Connecting the dots can mislead the beginner into forgetting that the $x(n)$ sequence is nothing more than a list of numbers. Remember, $x(n)$ is a discrete-time sequence of individual values, and each value in that sequence plots as a single dot. It's not that we're ignorant of what lies between the dots of $x(n)$; there *is* nothing between those dots.

We can reinforce this discrete-time sequence concept by listing those Figure 1–1(b) sampled values as follows:

$$
\begin{array}{ll}
x(0) = 0 & \text{(1st sequence value, index } n = 0\text{)} \\
x(1) = 0.31 & \text{(2nd sequence value, index } n = 1\text{)} \\
x(2) = 0.59 & \text{(3rd sequence value, index } n = 2\text{)} \\
x(3) = 0.81 & \text{(4th sequence value, index } n = 3\text{)} \\
\quad\dots & \qquad\qquad\dots \\
 & \text{and so on,}
\end{array}
\tag{1–2}
$$

where n represents the time index integer sequence 0, 1, 2, 3, etc., and t_s is some constant time period between samples. Those sample values can be represented collectively, and concisely, by the discrete-time expression

$$x(n) = \sin(2\pi f_o n t_s). \tag{1–3}$$

(Here again, the $2\pi f_o n t_s$ term is an angle measured in radians.) Notice that the index n in Eq. (1–2) started with a value of 0, instead of 1. There's nothing sacred about this; the first value of n could just as well have been 1, but we start the index n at zero out of habit because doing so allows us to describe the sinewave starting at time zero. The variable $x(n)$ in Eq. (1–3) is read as "the sequence x of n." Equations (1–1) and (1–3) describe what are also referred to as *time-domain* signals because the independent variables, the continuous time t in Eq. (1–1), and the discrete-time nt_s values used in Eq. (1–3) are measures of time.

With this notion of a discrete-time signal in mind, let's say that a discrete system is a collection of hardware components, or software routines, that operate on a discrete-time signal sequence. For example, a discrete system could be a process that gives us a discrete output sequence $y(0)$, $y(1)$, $y(2)$, etc., when a discrete input sequence of $x(0)$, $x(1)$, $x(2)$, etc., is applied to the system input as shown in Figure 1–2(a). Again, to keep the notation concise and still keep track of individual elements of the input and output sequences, an abbreviated notation is used as shown in Figure 1–2(b) where n represents the integer sequence 0, 1, 2, 3, etc. Thus, $x(n)$ and $y(n)$ are general variables that represent two separate sequences of numbers. Figure 1–2(b) allows us to describe a system's output with a simple expression such as

$$y(n) = 2x(n) - 1. \tag{1–4}$$

Illustrating Eq. (1–4), if $x(n)$ is the five-element sequence $x(0) = 1$, $x(1) = 3$, $x(2) = 5$, $x(3) = 7$, and $x(4) = 9$, then $y(n)$ is the five-element sequence $y(0) = 1$, $y(1) = 5$, $y(2) = 9$, $y(3) = 13$, and $y(4) = 17$.

Equation (1–4) is formally called a *difference equation.* (In this book we won't be working with differential equations or partial differential equations. However, we will, now and then, work with partially difficult equations.)

The fundamental difference between the way time is represented in continuous and discrete systems leads to a very important difference in how we characterize frequency in continuous and discrete systems. To illustrate, let's reconsider the continuous sinewave in Figure 1–1(a). If it represented a voltage at the end of a cable, we could measure its frequency by applying it to an oscilloscope, a spectrum analyzer, or a frequency counter. We'd have a problem, however, if we were merely given the list of values from Eq. (1–2) and asked to determine the frequency of the waveform they represent. We'd graph those discrete values, and, sure enough, we'd recognize a single sinewave as in Figure 1–1(b). We can say that the sinewave repeats every 20 samples, but there's no way to determine the exact sinewave frequency from the discrete sequence values alone. You can probably see the point we're leading to here. If we knew the time between samples—the sample period t_s—we'd be able to determine the absolute frequency of the discrete sinewave.

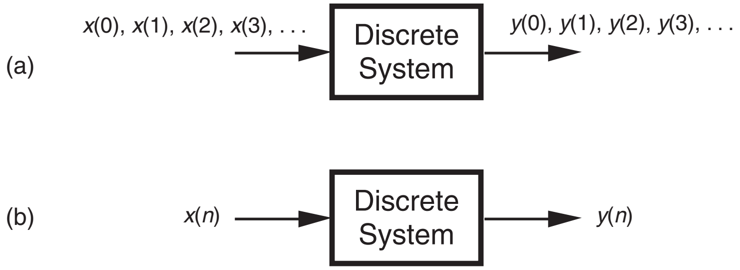


Figure 1–2 With an input applied, a discrete system provides an output: (a) the input and output are sequences of individual values; (b) input and output using the abbreviated notation of $x(n)$ and $y(n)$.

Given that the t_s sample period is, say, 0.05 milliseconds/sample, the period of the sinewave is

$$\text{sinewave period} = \frac{20 \text{ samples}}{\text{period}} \cdot \frac{0.05 \text{ milliseconds}}{\text{sample}} = 1 \text{ millisecond.} \quad (1\text{–}5)$$

Because the frequency of a sinewave is the reciprocal of its period, we now know that the sinewave's absolute frequency is 1/(1 ms), or 1 kHz. On the other hand, if we found that the sample period was, in fact, 2 milliseconds, the discrete samples in Figure 1–1(b) would represent a sinewave whose period is 40 milliseconds and whose frequency is 25 Hz. The point here is that when dealing with discrete systems, absolute frequency determination in Hz is dependent on the sampling frequency

$$f_s = 1/t_s. \quad (1\text{–}5')$$

We'll be reminded of this dependence throughout the remainder of this book.

In digital signal processing, we often find it necessary to characterize the frequency content of discrete time-domain signals. When we do so, this frequency representation takes place in what's called the *frequency domain*. By

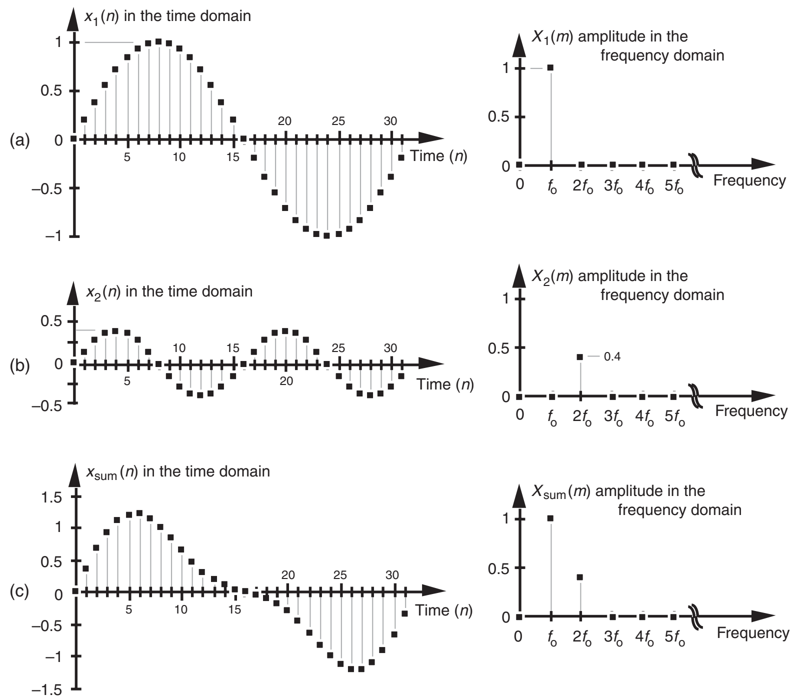


Figure 1–3 Time- and frequency-domain graphical representations: (a) sinewave of frequency f_o; (b) reduced amplitude sinewave of frequency $2f_o$; (c) sum of the two sinewaves.

way of example, let's say we have a discrete sinewave sequence $x_1(n)$ with an arbitrary frequency f_o Hz as shown on the left side of Figure 1–3(a). We can also characterize $x_1(n)$ by showing its spectral content, the $X_1(m)$ sequence on the right side of Figure 1-3(a), indicating that it has a single spectral component, and no other frequency content. Although we won't dwell on it just now, notice that the frequency-domain representations in Figure 1–3 are themselves discrete.

To illustrate our time- and frequency-domain representations further, Figure 1–3(b) shows another discrete sinewave $x_2(n)$, whose peak amplitude is 0.4, with a frequency of $2f_o$. The discrete sample values of $x_2(n)$ are expressed by the equation

$$x_2(n) = 0.4 \cdot \sin(2\pi 2 f_o n t_s). \tag{1–6}$$

When the two sinewaves, $x_1(n)$ and $x_2(n)$, are added to produce a new waveform $x_{\text{sum}}(n)$, its time-domain equation is

$$x_{\text{sum}}(n) = x_1(n) + x_2(n) = \sin(2\pi f_o n t_s) + 0.4 \cdot \sin(2\pi 2 f_o n t_s), \tag{1–7}$$

and its time- and frequency-domain representations are those given in Figure 1–3(c). We interpret the $X_{\text{sum}}(m)$ frequency-domain depiction, the *spectrum*, in Figure 1–3(c) to indicate that $x_{\text{sum}}(n)$ has a frequency component of f_o Hz and a reduced-amplitude frequency component of $2f_o$ Hz.

Notice three things in Figure 1–3. First, time sequences use lowercase variable names like the "x" in $x_1(n)$, and uppercase symbols for frequency-domain variables such as the "X" in $X_1(m)$. The term $X_1(m)$ is read as "the spectral sequence X sub one of m." Second, because the $X_1(m)$ frequency-domain representation of the $x_1(n)$ time sequence is itself a sequence (a list of numbers), we use the index "m" to keep track of individual elements in $X_1(m)$. We can list frequency-domain sequences just as we did with the time sequence in Eq. (1–2). For example, $X_{\text{sum}}(m)$ is listed as

$$\begin{array}{ll}
X_{\text{sum}}(0) = 0 & (\text{1st } X_{\text{sum}}(m) \text{ value, index } m = 0) \\
X_{\text{sum}}(1) = 1.0 & (\text{2nd } X_{\text{sum}}(m) \text{ value, index } m = 1) \\
X_{\text{sum}}(2) = 0.4 & (\text{3rd } X_{\text{sum}}(m) \text{ value, index } m = 2) \\
X_{\text{sum}}(3) = 0 & (\text{4th } X_{\text{sum}}(m) \text{ value, index } m = 3) \\
\quad \cdots & \qquad \cdots \\
& \text{and so on,}
\end{array}$$

where the frequency index m is the integer sequence 0, 1, 2, 3, etc. Third, because the $x_1(n) + x_2(n)$ sinewaves have a phase shift of zero degrees relative to each other, we didn't really need to bother depicting this phase relationship in $X_{\text{sum}}(m)$ in Figure 1–3(c). In general, however, phase relationships in frequency-domain sequences are important, and we'll cover that subject in Chapters 3 and 5.

A key point to keep in mind here is that we now know three equivalent ways to describe a discrete-time waveform. Mathematically, we can use a time-domain equation like Eq. (1–6). We can also represent a time-domain waveform graphically as we did on the left side of Figure 1–3, and we can depict its corresponding, discrete, frequency-domain equivalent as that on the right side of Figure 1–3.

As it turns out, the discrete time-domain signals we're concerned with are not only quantized in time; their amplitude values are also quantized. Because we represent all digital quantities with binary numbers, there's a limit to the resolution, or granularity, that we have in representing the values of discrete numbers. Although signal amplitude quantization can be an important consideration—we cover that particular topic in Chapter 12—we won't worry about it just now.

1.2 SIGNAL AMPLITUDE, MAGNITUDE, POWER

Let's define two important terms that we'll be using throughout this book: *amplitude* and *magnitude*. It's not surprising that, to the layman, these terms are typically used interchangeably. When we check our thesaurus, we find that they are synonymous.† In engineering, however, they mean two different things, and we must keep that difference clear in our discussions. The amplitude of a variable is the measure of how far, and in what direction, that variable differs from zero. Thus, signal amplitudes can be either positive or negative. The time-domain sequences in Figure 1–3 presented the sample value amplitudes of three different waveforms. Notice how some of the individual discrete amplitude values were positive and others were negative.

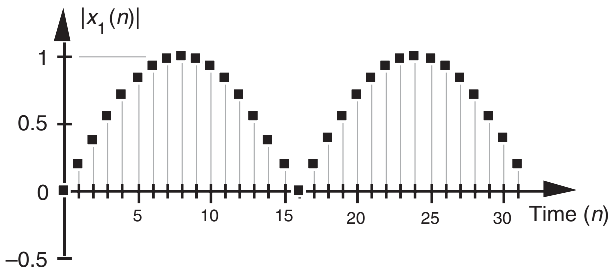


Figure 1–4 Magnitude samples, $|x_1(n)|$, of the time waveform in Figure 1–3(a).

† Of course, laymen are "other people." To the engineer, the brain surgeon is the layman. To the brain surgeon, the engineer is the layman.

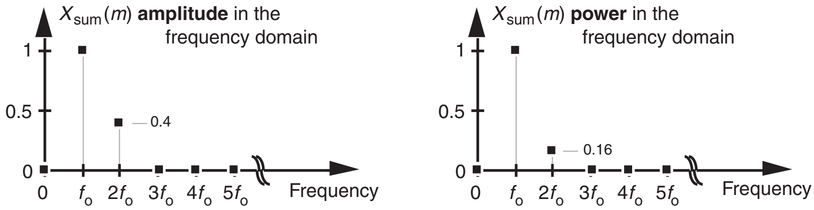


Figure 1–5 Frequency-domain amplitude and frequency-domain power of the $x_{sum}(n)$ time waveform in Figure 1–3(c).

The magnitude of a variable, on the other hand, is the measure of how far, regardless of direction, its quantity differs from zero. So magnitudes are always positive values. Figure 1–4 illustrates how the magnitude of the $x_1(n)$ time sequence in Figure 1–3(a) is equal to the amplitude, but with the sign always being positive for the magnitude. We use the modulus symbol (| |) to represent the magnitude of $x_1(n)$. Occasionally, in the literature of digital signal processing, we'll find the term *magnitude* referred to as the *absolute value*.

When we examine signals in the frequency domain, we'll often be interested in the power level of those signals. The power of a signal is proportional to its amplitude (or magnitude) squared. If we assume that the proportionality constant is one, we can express the power of a sequence in the time or frequency domains as

$$x_{pwr}(n) = |x(n)|^2, \tag{1–8}$$

or

$$X_{pwr}(m) = |X(m)|^2. \tag{1–8′}$$

Very often we'll want to know the difference in power levels of two signals in the frequency domain. Because of the squared nature of power, two signals with moderately different amplitudes will have a much larger difference in their relative powers. In Figure 1–3, for example, signal $x_1(n)$'s amplitude is 2.5 times the amplitude of signal $x_2(n)$, but its power level is 6.25 that of $x_2(n)$'s power level. This is illustrated in Figure 1–5 where both the amplitude and power of $X_{sum}(m)$ are shown.

Because of their squared nature, plots of power values often involve showing both very large and very small values on the same graph. To make these plots easier to generate and evaluate, practitioners usually employ the decibel scale as described in Appendix E.

1.3 SIGNAL PROCESSING OPERATIONAL SYMBOLS

We'll be using block diagrams to graphically depict the way digital signal processing operations are implemented. Those block diagrams will comprise an assortment of fundamental processing symbols, the most common of which are illustrated and mathematically defined in Figure 1–6.

Figure 1–6(a) shows the addition, element for element, of two discrete sequences to provide a new sequence. If our sequence index n begins at 0, we say that the first output sequence value is equal to the sum of the first element of the b sequence and the first element of the c sequence, or $a(0) = b(0) + c(0)$. Likewise, the second output sequence value is equal to the sum of the second

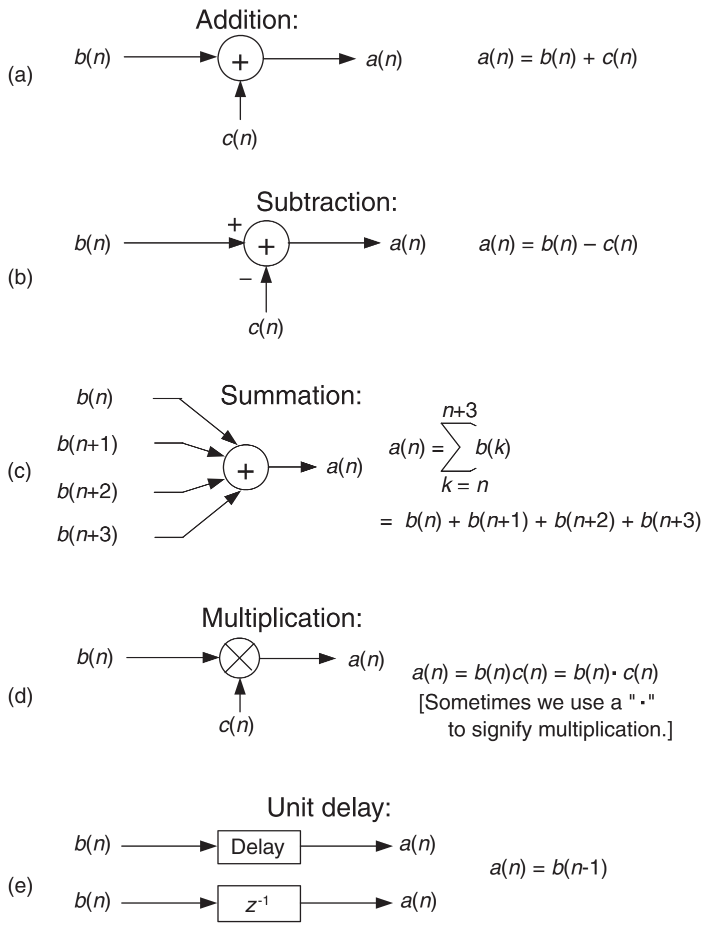


Figure 1–6 Terminology and symbols used in digital signal processing block diagrams.

element of the b sequence and the second element of the c sequence, or $a(1) = b(1) + c(1)$. Equation (1–7) is an example of adding two sequences. The subtraction process in Figure 1–6(b) generates an output sequence that's the element-for-element difference of the two input sequences. There are times when we must calculate a sequence whose elements are the sum of more than two values. This operation, illustrated in Figure 1–6(c), is called *summation* and is very common in digital signal processing. Notice how the lower and upper limits of the summation index k in the expression in Figure 1–6(c) tell us exactly which elements of the b sequence to sum to obtain a given $a(n)$ value. Because we'll encounter summation operations so often, let's make sure we understand their notation. If we repeat the summation equation from Figure 1–6(c) here, we have

$$a(n) = \sum_{k=n}^{n+3} b(k). \tag{1–9}$$

This means that

$$\begin{array}{ll}
\text{when } n = 0\text{, index } k \text{ goes from 0 to 3, so} & a(0) = b(0) + b(1) + b(2) + b(3) \\
\text{when } n = 1\text{, index } k \text{ goes from 1 to 4, so} & a(1) = b(1) + b(2) + b(3) + b(4) \\
\text{when } n = 2\text{, index } k \text{ goes from 2 to 5, so} & a(2) = b(2) + b(3) + b(4) + b(5) \\
\text{when } n = 3\text{, index } k \text{ goes from 3 to 6, so} & a(3) = b(3) + b(4) + b(5) + b(6) \\
\quad \cdots & \quad \cdots \\
\text{and so on.} &
\end{array} \tag{1–10}$$

We'll begin using summation operations in earnest when we discuss digital filters in Chapter 5.

The multiplication of two sequences is symbolized in Figure 1–6(d). Multiplication generates an output sequence that's the element-for-element product of two input sequences: $a(0) = b(0)c(0)$, $a(1) = b(1)c(1)$, and so on. The last fundamental operation that we'll be using is called the *unit delay* in Figure 1–6(e). While we don't need to appreciate its importance at this point, we'll merely state that the unit delay symbol signifies an operation where the output sequence $a(n)$ is equal to a delayed version of the $b(n)$ sequence. For example, $a(5) = b(4)$, $a(6) = b(5)$, $a(7) = b(6)$, etc. As we'll see in Chapter 6, due to the mathematical techniques used to analyze digital filters, the unit delay is very often depicted using the term z^{-1}.

The symbols in Figure 1–6 remind us of two important aspects of digital signal processing. First, our processing operations are always performed on sequences of individual discrete values, and second, the elementary operations themselves are very simple. It's interesting that, regardless of how complicated they appear to be, the vast majority of digital signal processing algorithms can be performed using combinations of these simple operations. If we think of a digital signal processing algorithm as a recipe, then the symbols in Figure 1–6 are the ingredients.

1.4 INTRODUCTION TO DISCRETE LINEAR TIME-INVARIANT SYSTEMS

In keeping with tradition, we'll introduce the subject of linear time-invariant (LTI) systems at this early point in our text. Although an appreciation for LTI systems is not essential in studying the next three chapters of this book, when we begin exploring digital filters, we'll build on the strict definitions of linearity and time invariance. We need to recognize and understand the notions of linearity and time invariance not just because the vast majority of discrete systems used in practice are LTI systems, but because LTI systems are very accommodating when it comes to their analysis. That's good news for us because we can use straightforward methods to predict the performance of any digital signal processing scheme as long as it's linear and time invariant. Because linearity and time invariance are two important system characteristics having very special properties, we'll discuss them now.

1.5 DISCRETE LINEAR SYSTEMS

The term *linear* defines a special class of systems where the output is the superposition, or sum, of the individual outputs had the individual inputs been applied separately to the system. For example, we can say that the application of an input $x_1(n)$ to a system results in an output $y_1(n)$. We symbolize this situation with the following expression:

$$x_1(n) \xrightarrow{\text{results in}} y_1(n). \tag{1–11}$$

Given a different input $x_2(n)$, the system has a $y_2(n)$ output as

$$x_2(n) \xrightarrow{\text{results in}} y_2(n). \tag{1–12}$$

For the system to be linear, when its input is the sum $x_1(n) + x_2(n)$, its output must be the sum of the individual outputs so that

$$x_1(n)+x_2(n) \xrightarrow{\text{results in}} y_1(n)+y_2(n). \tag{1–13}$$

One way to paraphrase expression (1–13) is to state that a linear system's output is the sum of the outputs of its parts. Also, part of this description of linearity is a proportionality characteristic. This means that if the inputs are scaled by constant factors c_1 and c_2, then the output sequence parts are also scaled by those factors as

$$c_1x_1(n)+c_2x_2(n) \xrightarrow{\text{results in}} c_1y_1(n)+c_2y_2(n). \tag{1–14}$$

In the literature, this proportionality attribute of linear systems in expression (1–14) is sometimes called the *homogeneity property*. With these thoughts in mind, then, let's demonstrate the concept of system linearity.

1.5.1 Example of a Linear System

To illustrate system linearity, let's say we have the discrete system shown in Figure 1–7(a) whose output is defined as

$$y(n) = \frac{-x(n)}{2}, \tag{1–15}$$

that is, the output sequence is equal to the negative of the input sequence with the amplitude reduced by a factor of two. If we apply an $x_1(n)$ input sequence representing a 1 Hz sinewave sampled at a rate of 32 samples per cycle, we'll have a $y_1(n)$ output as shown in the center of Figure 1–7(b). The frequency-domain spectral amplitude of the $y_1(n)$ output is the plot on the

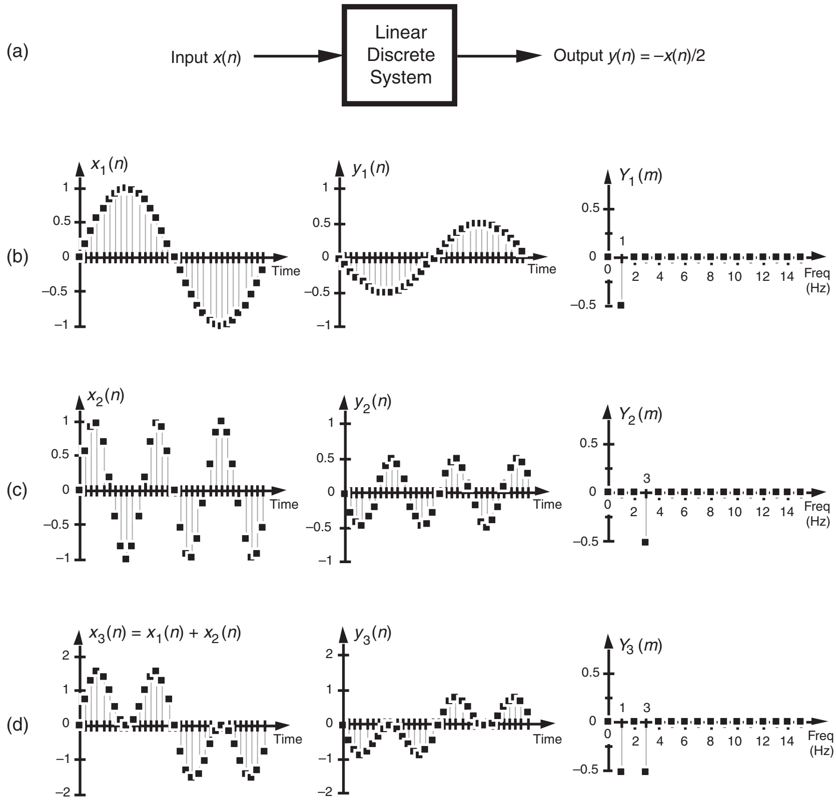


Figure 1–7 Linear system input-to-output relationships: (a) system block diagram where $y(n) = -x(n)/2$; (b) system input and output with a 1 Hz sinewave applied; (c) with a 3 Hz sinewave applied; (d) with the sum of 1 Hz and 3 Hz sinewaves applied.

right side of Figure 1–7(b), indicating that the output comprises a single tone of peak amplitude equal to –0.5 whose frequency is 1 Hz. Next, applying an $x_2(n)$ input sequence representing a 3 Hz sinewave, the system provides a $y_2(n)$ output sequence, as shown in the center of Figure 1–7(c). The spectrum of the $y_2(n)$ output, $Y_2(m)$, confirming a single 3 Hz sinewave output is shown on the right side of Figure 1–7(c). Finally—here's where the linearity comes in—if we apply an $x_3(n)$ input sequence that's the sum of a 1 Hz sinewave and a 3 Hz sinewave, the $y_3(n)$ output is as shown in the center of Figure 1–7(d). Notice how $y_3(n)$ is the sample-for-sample sum of $y_1(n)$ and $y_2(n)$. Figure 1–7(d) also shows that the output spectrum $Y_3(m)$ is the sum of $Y_1(m)$ and $Y_2(m)$. That's linearity.

1.5.2 Example of a Nonlinear System

It's easy to demonstrate how a nonlinear system yields an output that is not equal to the sum of $y_1(n)$ and $y_2(n)$ when its input is $x_1(n) + x_2(n)$. A simple example of a nonlinear discrete system is that in Figure 1–8(a) where the output is the square of the input described by

$$y(n) = [x(n)]^2. \tag{1–16}$$

We'll use a well-known trigonometric identity and a little algebra to predict the output of this nonlinear system when the input comprises simple sinewaves. Following the form of Eq. (1–3), let's describe a sinusoidal sequence, whose frequency f_o = 1 Hz, by

$$x_1(n) = \sin(2\pi f_o nt_s) = \sin(2\pi \cdot 1 \cdot nt_s). \tag{1–17}$$

Equation (1–17) describes the $x_1(n)$ sequence on the left side of Figure 1–8(b). Given this $x_1(n)$ input sequence, the $y_1(n)$ output of the nonlinear system is the square of a 1 Hz sinewave, or

$$y_1(n) = [x_1(n)]^2 = \sin(2\pi \cdot 1 \cdot nt_s) \cdot \sin(2\pi \cdot 1 \cdot nt_s). \tag{1–18}$$

We can simplify our expression for $y_1(n)$ in Eq. (1–18) by using the following trigonometric identity:

$$\sin(\alpha) \cdot \sin(\beta) = \frac{\cos(\alpha - \beta)}{2} - \frac{\cos(\alpha + \beta)}{2}. \tag{1–19}$$

Using Eq. (1–19), we can express $y_1(n)$ as

$$\begin{aligned} y_1(n) &= \frac{\cos(2\pi \cdot 1 \cdot nt_s - 2\pi \cdot 1 \cdot nt_s)}{2} - \frac{\cos(2\pi \cdot 1 \cdot nt_s + 2\pi \cdot 1 \cdot nt_s)}{2} \\ &= \frac{\cos(0)}{2} - \frac{\cos(4\pi \cdot 1 \cdot nt_s)}{2} = \frac{1}{2} - \frac{\cos(2\pi \cdot 2 \cdot nt_s)}{2}, \end{aligned} \tag{1–20}$$

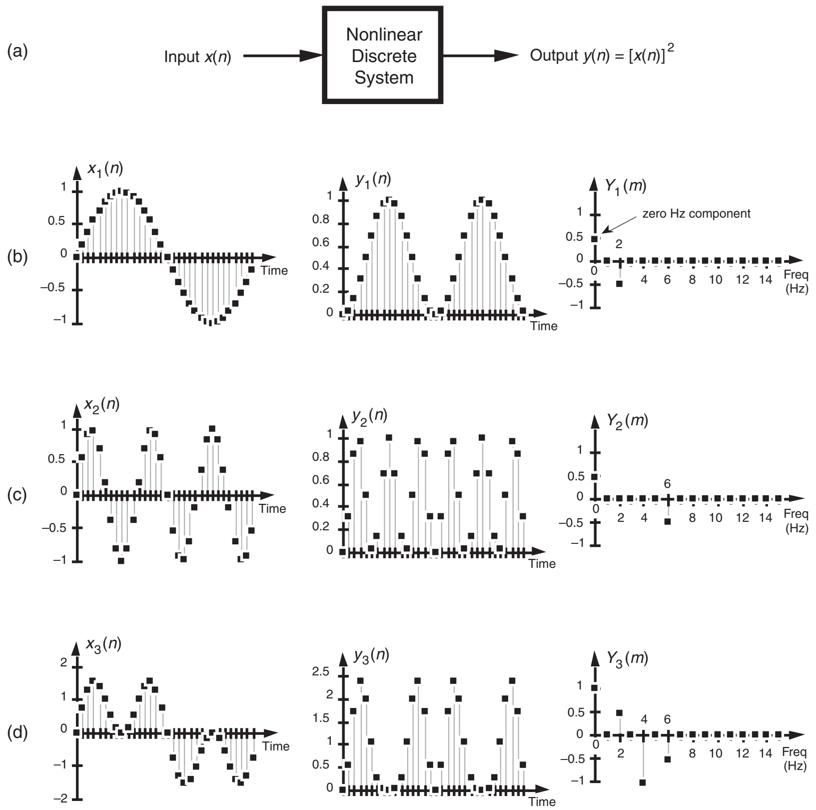


Figure 1–8 Nonlinear system input-to-output relationships: (a) system block diagram where $y(n) = (x(n))^2$; (b) system input and output with a 1 Hz sinewave applied; (c) with a 3 Hz sinewave applied; (d) with the sum of 1 Hz and 3 Hz sinewaves applied.

which is shown as the all-positive sequence in the center of Figure 1–8(b). Because Eq. (1–19) results in a frequency sum ($\alpha + \beta$) and frequency difference ($\alpha - \beta$) effect when multiplying two sinusoids, the $y_1(n)$ output sequence will be a cosine wave of 2 Hz and a peak amplitude of –0.5, added to a constant value of 1/2. The constant value of 1/2 in Eq. (1–20) is interpreted as a zero Hz frequency component, as shown in the $Y_1(m)$ spectrum in Figure 1–8(b). We could go through the same algebraic exercise to determine that when a 3 Hz sinewave $x_2(n)$ sequence is applied to this nonlinear system, the output $y_2(n)$ would contain a zero Hz component and a 6 Hz component, as shown in Figure 1–8(c).

System nonlinearity is evident if we apply an $x_3(n)$ sequence comprising the sum of a 1 Hz and a 3 Hz sinewave as shown in Figure 1–8(d). We can predict the frequency content of the $y_3(n)$ output sequence by using the algebraic relationship

$$(a+b)^2 = a^2+2ab+b^2, \tag{1–21}$$

where a and b represent the 1 Hz and 3 Hz sinewaves, respectively. From Eq. (1–19), the a^2 term in Eq. (1–21) generates the zero Hz and 2 Hz output sinusoids in Figure 1–8(b). Likewise, the b^2 term produces in $y_3(n)$ another zero Hz and the 6 Hz sinusoid in Figure 1–8(c). However, the $2ab$ term yields additional 2 Hz and 4 Hz sinusoids in $y_3(n)$. We can show this algebraically by using Eq. (1–19) and expressing the $2ab$ term in Eq. (1–21) as

$$2ab = 2\cdot\sin(2\pi\cdot 1\cdot nt_s)\cdot\sin(2\pi\cdot 3\cdot nt_s)$$

$$= \frac{2\cos(2\pi\cdot 1\cdot nt_s - 2\pi\cdot 3\cdot nt_s)}{2} - \frac{2\cos(2\pi\cdot 1\cdot nt_s + 2\pi\cdot 3\cdot nt_s)}{2} \tag{1–22}$$

$$= \cos(2\pi\cdot 2\cdot nt_s) - \cos(2\pi\cdot 4\cdot nt_s).^{\dagger}$$

Equation (1–22) tells us that two additional sinusoidal components will be present in $y_3(n)$ because of the system's nonlinearity, a 2 Hz cosine wave whose amplitude is +1 and a 4 Hz cosine wave having an amplitude of –1. These spectral components are illustrated in $Y_3(m)$ on the right side of Figure 1–8(d).

Notice that when the sum of the two sinewaves is applied to the nonlinear system, the output contained sinusoids, Eq. (1–22), that were not present in either of the outputs when the individual sinewaves alone were applied. Those extra sinusoids were generated by an interaction of the two input sinusoids due to the squaring operation. That's nonlinearity; expression (1–13) was not satisfied. (Electrical engineers recognize this effect of internally generated sinusoids as *intermodulation distortion*.) Although nonlinear systems are usually difficult to analyze, they are occasionally used in practice. References [2], [3], and [4], for example, describe their application in nonlinear digital filters. Again, expressions (1–13) and (1–14) state that a linear system's output resulting from a sum of individual inputs is the superposition (sum) of the individual outputs. They also stipulate that the output sequence $y_1(n)$ depends only on $x_1(n)$ combined with the system characteristics, and not on the other input $x_2(n)$; i.e., there's no interaction between inputs $x_1(n)$ and $x_2(n)$ at the output of a linear system.

† The first term in Eq. (1–22) is $\cos(2\pi \cdot nt_s - 6\pi \cdot nt_s) = \cos(-4\pi \cdot nt_s) = \cos(-2\pi \cdot 2 \cdot nt_s)$. However, because the cosine function is even, $\cos(-\alpha) = \cos(\alpha)$, we can express that first term as $\cos(2\pi \cdot 2 \cdot nt_s)$.

1.6 TIME-INVARIANT SYSTEMS

A time-invariant system is one where a time delay (or shift) in the input sequence causes an equivalent time delay in the system's output sequence. Keeping in mind that n is just an indexing variable we use to keep track of our input and output samples, let's say a system provides an output $y(n)$ given an input of $x(n)$, or

$$x(n) \xrightarrow{\text{results in}} y(n). \tag{1–23}$$

For a system to be time invariant, with a shifted version of the original $x(n)$ input applied, $x'(n)$, the following applies:

$$x'(n) = x(n+k) \xrightarrow{\text{results in}} y'(n) = y(n+k), \tag{1–24}$$

where k is some integer representing k sample period time delays. For a system to be time invariant, Eq. (1–24) must hold true for any integer value of k and any input sequence.

1.6.1 Example of a Time-Invariant System

Let's look at a simple example of time invariance illustrated in Figure 1–9. Assume that our initial $x(n)$ input is a unity-amplitude 1 Hz sinewave sequence with a $y(n)$ output, as shown in Figure 1–9(b). Consider a different input sequence $x'(n)$, where

$$x'(n) = x(n–4). \tag{1–25}$$

Equation (1–25) tells us that the input sequence $x'(n)$ is equal to sequence $x(n)$ shifted to the right by $k = -4$ samples. That is, $x'(4) = x(0)$, $x'(5) = x(1)$, $x'(6) = x(2)$, and so on as shown in Figure 1–9(c). The discrete system is time invariant because the $y'(n)$ output sequence is equal to the $y(n)$ sequence shifted to the right by four samples, or $y'(n) = y(n–4)$. We can see that $y'(4) = y(0)$, $y'(5) = y(1)$, $y'(6) = y(2)$, and so on as shown in Figure 1–9(c). For time-invariant systems, the time shifts in $x'(n)$ and $y'(n)$ are equal. Take careful notice of the minus sign in Eq. (1–25). In later chapters, that is the notation we'll use to algebraically describe a time-delayed discrete sequence.

Some authors succumb to the urge to define a time-invariant system as one whose parameters do not change with time. That definition is incomplete and can get us in trouble if we're not careful. We'll just stick with the formal definition that a time-invariant system is one where a time shift in an input sequence results in an equal time shift in the output sequence. By the way, time-invariant systems in the literature are often called *shift-invariant* systems.†

---

† An example of a discrete process that's not time invariant is the downsampling, or decimation, process described in Chapter 10.

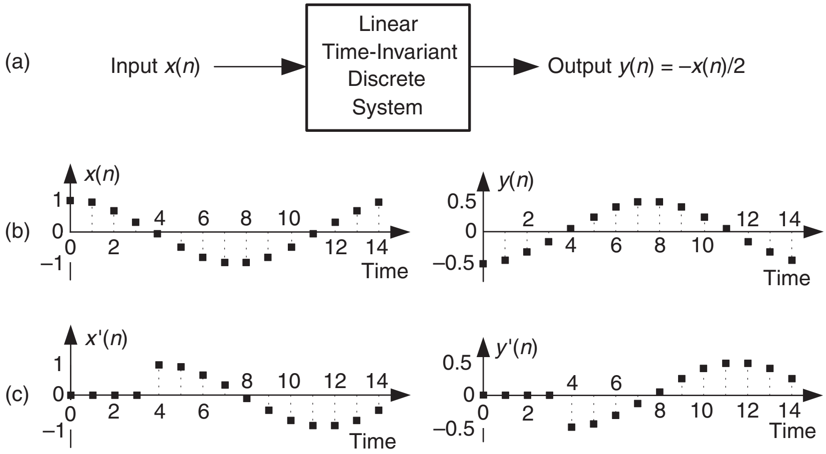


Figure 1–9 Time-invariant system input/output relationships: (a) system block diagram, $y(n) = -x(n)/2$; (b) system input/output with a sinewave input; (c) input/output when a sinewave, delayed by four samples, is the input.

1.7 THE COMMUTATIVE PROPERTY OF LINEAR TIME-INVARIANT SYSTEMS

Although we don't substantiate this fact until we reach Section 6.11, it's not too early to realize that LTI systems have a useful commutative property by which their sequential order can be rearranged with no change in their final output. This situation is shown in Figure 1–10 where two different LTI systems are configured in series. Swapping the order of two cascaded systems does not alter the final output. Although the intermediate data sequences $f(n)$ and $g(n)$ will usually not be equal, the two pairs of LTI systems will have iden-

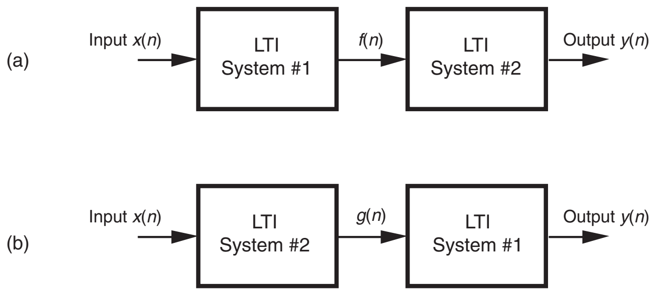


Figure 1–10 Linear time-invariant (LTI) systems in series: (a) block diagram of two LTI systems; (b) swapping the order of the two systems does not change the resultant output $y(n)$.

tical $y(n)$ output sequences. This commutative characteristic comes in handy for designers of digital filters, as we'll see in Chapters 5 and 6.

1.8 ANALYZING LINEAR TIME-INVARIANT SYSTEMS

As previously stated, LTI systems can be analyzed to predict their performance. Specifically, if we know the *unit impulse response* of an LTI system, we can calculate everything there is to know about the system; that is, the system's unit impulse response completely characterizes the system. By "unit impulse response" we mean the system's time-domain output sequence when the input is a single unity-valued sample (unit impulse) preceded and followed by zero-valued samples as shown in Figure 1–11(b).

Knowing the (unit) impulse response of an LTI system, we can determine the system's output sequence for any input sequence because the output is equal to the *convolution* of the input sequence and the system's impulse response. Moreover, given an LTI system's time-domain impulse response, we can find the system's *frequency response* by taking the Fourier transform in the form of a *discrete Fourier transform* of that impulse response[5]. The concepts in the two previous sentences are among the most important principles in all of digital signal processing!

Don't be alarmed if you're not exactly sure what is meant by convolution, frequency response, or the discrete Fourier transform. We'll introduce these subjects and define them slowly and carefully as we need them in later chapters. The point to keep in mind here is that LTI systems can be designed and analyzed using a number of straightforward and powerful analysis techniques. These techniques will become tools that we'll add to

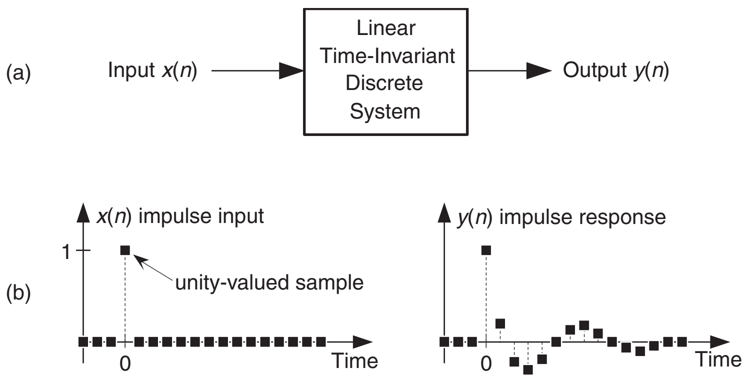


Figure 1–11 LTI system unit impulse response sequences: (a) system block diagram; (b) impulse input sequence $x(n)$ and impulse response output sequence $y(n)$.

our signal processing toolboxes as we journey through the subject of digital signal processing.

In the testing (analyzing) of continuous linear systems, engineers often use a narrow-in-time impulsive signal as an input signal to their systems. Mechanical engineers give their systems a little whack with a hammer, and electrical engineers working with analog-voltage systems generate a very narrow voltage spike as an impulsive input. Audio engineers, who need an impulsive acoustic test signal, sometimes generate an audio impulse by firing a starter pistol.

In the world of DSP, an impulse sequence called a *unit impulse* takes the form

$$x(n) = \ldots 0, 0, 0, 0, 0, A, 0, 0, 0, 0, 0, \ldots \tag{1–26}$$

The value A is often set equal to one. The leading sequence of zero-valued samples, before the A-valued sample, must be a bit longer than the length of the transient response of the system under test in order to initialize the system to its zero state. The trailing sequence of zero-valued samples, following the A-valued sample, must be a bit longer than the transient response of the system under test in order to capture the system's entire $y(n)$ impulse response output sequence.

Let's further explore this notion of impulse response testing to determine the frequency response of a discrete system (and take an opportunity to start using the operational symbols introduced in Section 1.3). Consider the block diagram of a 4-point moving averager shown in Figure 1–12(a). As the $x(n)$ input samples march their way through the system, at each time index n four successive input samples are averaged to compute a single $y(n)$ output. As we'll learn in subsequent chapters, a *moving averager* behaves like a digital lowpass filter. However, we can quickly illustrate that fact now.

If we apply an impulse input sequence to the system, we'll obtain its $y(n)$ impulse response output shown in Figure 1–12(b). The $y(n)$ output is computed using the following difference equation:

$$y(n) = \frac{1}{4}[x(n) + x(n-1) + x(n-2) + x(n-3)] = \frac{1}{4}\sum_{k=n-3}^{n} x(k). \tag{1–27}$$

If we then perform a discrete Fourier transform (a process we cover in much detail in Chapter 3) on $y(n)$, we obtain the $Y(m)$ frequency-domain information, allowing us to plot the frequency magnitude response of the 4-point moving averager as shown in Figure 1–12(c). So we see that a moving averager indeed has the characteristic of a lowpass filter. That is, the averager attenuates (reduces the amplitude of) high-frequency signal content applied to its input.

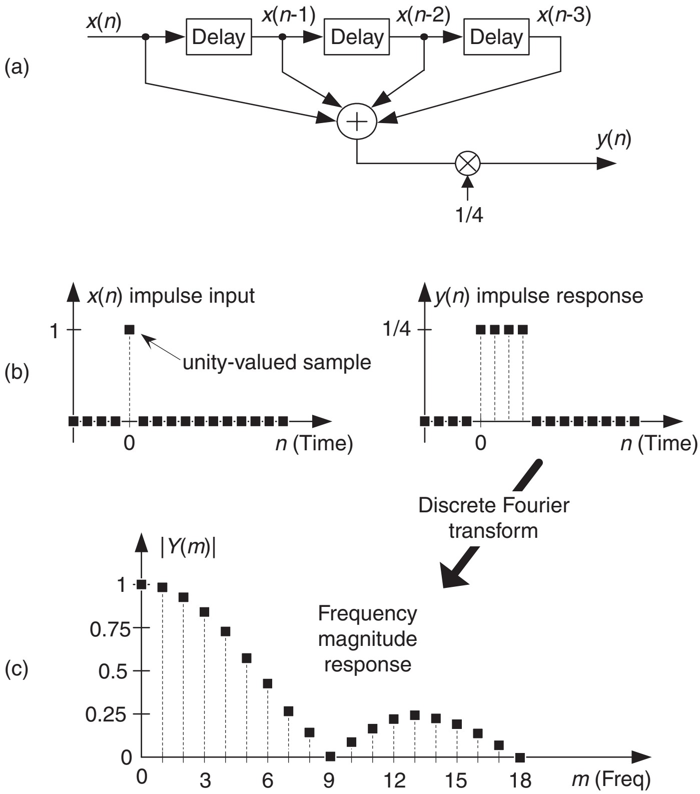


Figure 1–12 Analyzing a moving averager: (a) averager block diagram; (b) impulse input and impulse response; (c) averager frequency magnitude response.

OK, this concludes our brief introduction to discrete sequences and systems. In later chapters we'll learn the details of discrete Fourier transforms, discrete system impulse responses, and digital filters.

REFERENCES

[1] Karplus, W. J., and Soroka, W. W. *Analog Methods*, 2nd ed., McGraw-Hill, New York, 1959, p. 117.

[2] Mikami, N., Kobayashi, M., and Yokoyama, Y. "A New DSP-Oriented Algorithm for Calculation of the Square Root Using a Nonlinear Digital Filter," *IEEE Trans. on Signal Processing*, Vol. 40, No. 7, July 1992.

[3] Heinen, P., and Neuvo, Y. "FIR-Median Hybrid Filters," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, Vol. ASSP-35, No. 6, June 1987.

[4] Oppenheim, A., Schafer, R., and Stockham, T. "Nonlinear Filtering of Multiplied and Convolved Signals," *Proc. IEEE*, Vol. 56, August 1968.

[5] Pickerd, John. "Impulse-Response Testing Lets a Single Test Do the Work of Thousands," *EDN*, April 27, 1995.

CHAPTER 1 PROBLEMS

1.1 This problem gives us practice in thinking about sequences of numbers. For centuries mathematicians have developed clever ways of computing π. In 1671 the Scottish mathematician James Gregory proposed the following very simple series for calculating π:

$$\pi \approx 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} \ldots\right).$$

Thinking of the terms inside the parentheses as a sequence indexed by the variable n, where $n = 0, 1, 2, 3, \ldots, 100$, write Gregory's algorithm in the form

$$\pi \approx 4 \cdot \sum_{n=0}^{100} (-1)^{?} \cdot ?$$

replacing the "?" characters with expressions in terms of index n.

1.2 One of the ways to obtain discrete sequences, for follow-on processing, is to digitize a continuous (analog) signal with an analog-to-digital (A/D) converter. A 6-bit A/D converter's output words (6-bit binary words) can only represent 2^6=64 different numbers. (We cover this digitization, *sampling*, and A/D converters in detail in upcoming chapters.) Thus we say the A/D converter's "digital" output can only represent a finite number of amplitude values. Can you think of a continuous time-domain electrical signal that only has a finite number of amplitude values? If so, draw a graph of that continuous-time signal.

1.3 On the Internet, the author once encountered the following line of C-language code

```
PI = 2*asin(1.0);
```

whose purpose was to define the constant π. In standard mathematical notation, that line of code can be described by

$$\pi = 2 \cdot \sin^{-1}(1).$$

Under what assumption does the above expression correctly define the constant π?

1.4 Many times in the literature of signal processing you will encounter the identity

$$x^0 = 1.$$

That is, x raised to the zero power is equal to one. Using the Laws of Exponents, prove the above expression to be true.

1.5 Recall that for discrete sequences the t_s sample period (the time period between samples) is the reciprocal of the sample frequency f_s. Write the equations, as we did in the text's Eq. (1–3), describing time-domain sequences for unity-amplitude cosine waves whose f_o frequencies are

(a) $f_o = f_s/2$, one-half the sample rate,
(b) $f_o = f_s/4$, one-fourth the sample rate,
(c) $f_o = 0$ (zero) Hz.

1.6 Draw the three time-domain cosine wave sequences, where a sample value is represented by a dot, described in Problem 1.5. The correct solution to Part (a) of this problem is a useful sequence used to convert some lowpass digital filters into highpass filters. (Chapter 5 discusses that topic.) The correct solution to Part (b) of this problem is an important discrete sequence used for *frequency translation* (both for signal *down-conversion* and *up-conversion*) in modern-day wireless communications systems. The correct solution to Part (c) of this problem should convince us that it's perfectly valid to describe a cosine sequence whose frequency is zero Hz.

1.7 Draw the three time-domain sequences of unity-amplitude sinewaves (not cosine waves) whose frequencies are

(a) $f_o = f_s/2$, one-half the sample rate,
(b) $f_o = f_s/4$, one-fourth the sample rate,
(c) $f_o = 0$ (zero) Hz.

The correct solutions to Parts (a) and (c) show us that the two frequencies, 0 Hz and $f_s/2$ Hz, are special frequencies in the world of discrete signal processing. What is *special* about the sinewave sequences obtained from the above Parts (a) and (c)?

1.8 Consider the infinite-length time-domain sequence $x(n)$ in Figure P1–8. Draw the first eight samples of a shifted time sequence defined by

$$x_{\text{shift}}(n) = x(n+1).$$

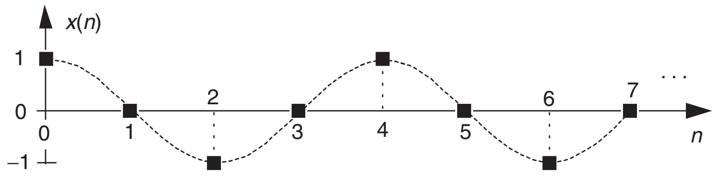


Figure P1–8

1.9 Assume, during your reading of the literature of DSP, you encounter the process shown in Figure P1–9. The $x(n)$ input sequence, whose f_s sample rate is 2500 Hz, is multiplied by a sinusoidal $m(n)$ sequence to produce the $y(n)$ output sequence. What is the frequency, measured in Hz, of the sinusoidal $m(n)$ sequence?

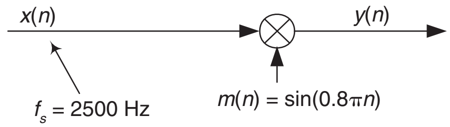


Figure P1–9

1.10 There is a process in DSP called an "N-point running sum" (a kind of digital lowpass filter, actually) that is described by the following equation:

$$y(n) = \sum_{p=0}^{N-1} x(n-p).$$

Write out, giving the indices of all the $x()$ terms, the algebraic expression that describes the computations needed to compute $y(9)$ when N=6.

1.11 A 5-point *moving averager* can be described by the following difference equation:

$$y(n) = \frac{1}{5}[x(n)+x(n-1)+x(n-2)+x(n-3)+x(n-4)] = \frac{1}{5}\sum_{k=n-4}^{n} x(k). \qquad \text{(P1–1)}$$

The averager's signal-flow block diagram is shown in Figure P1–11, where the $x(n)$ input samples flow through the averager from left to right.

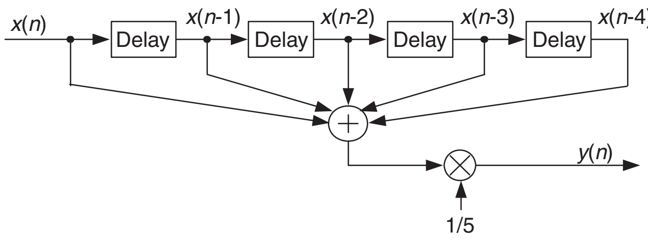


Figure P1–11

Equation (P1–1) is equivalent to

$$y(n) = \frac{x(n)}{5} + \frac{x(n-1)}{5} + \frac{x(n-2)}{5} + \frac{x(n-3)}{5} + \frac{x(n-4)}{5}$$
$$= \sum_{k=n-4}^{n} \frac{x(k)}{5}. \tag{P1–2}$$

(a) Draw the block diagram of the discrete system described by Eq. (P1–2).

(b) The *moving average* processes described by Eqs. (P1–1) and (P1–2) have identical impulse responses. Draw that impulse response.

(c) If you had to implement (using programmable hardware or assembling discrete hardware components) either Eq. (P1–1) or Eq. (P1–2), which would you choose? Explain why.

1.12 In this book we will look at many two-dimensional drawings showing the value of one variable (y) plotted as a function of another variable (x). Stated in different words, we'll graphically display what are the values of a y axis variable for various values of an x axis variable. For example, Figure P1–12(a) plots the weight of a male child as a function of the child's age. The dimension of the x axis is years

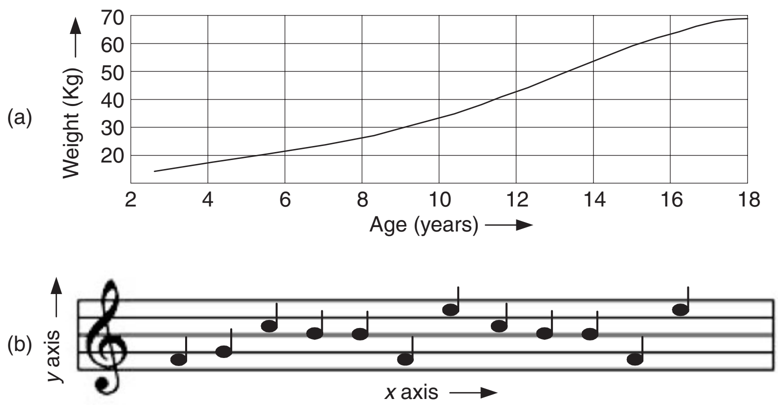


Figure P1–12

and the dimension of the y axis is kilograms. What are the dimensions of the x and y axes of the familiar two-dimensional plot given in Figure P1–12(b)?

1.13 Let's say you are writing software code to generate an $x(n)$ test sequence composed of the sum of two equal-amplitude discrete cosine waves, as

$$x(n) = \cos(2\pi f_o nt_s + \phi) + \cos(2\pi f_o nt_s)$$

where t_s is the time between your $x(n)$ samples, and ϕ is a constant phase shift measured in radians. An example $x(n)$ when $\phi = \pi/2$ is shown in Figure P1–13 where the $x(n)$ sequence, represented by the circular dots, is a single sinusoid whose frequency is f_o Hz.

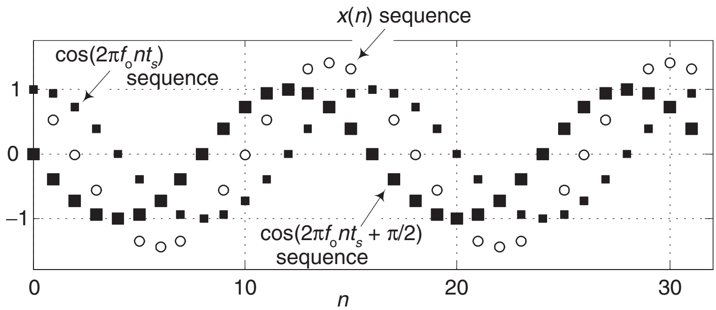


Figure P1–13

Using the trigonometric identity $\cos(\alpha+\beta) + \cos(\alpha-\beta) = 2\cos(\alpha)\cos(\beta)$, derive an equation for $x(n)$ that is of the form

$$x(n) = 2\cos(\alpha)\cos(\beta)$$

where variables α and β are in terms of $2\pi f_o nt_s$ and ϕ.

1.14 In your engineering education you'll often read in some mathematical derivation, or hear someone say, "For small α, $\sin(\alpha) = \alpha$." (In fact, you'll encounter that statement a few times in this book.) Draw two curves defined by

$$x = \alpha, \text{ and } y = \sin(\alpha)$$

over the range of $\alpha = -\pi/2$ to $\alpha = \pi/2$, and discuss why that venerable "For small α, $\sin(\alpha) = \alpha$" statement is valid.

1.15 Considering two continuous (analog) sinusoids, having initial phase angles of α radians at time $t = 0$, replace the following "?" characters with the correct angle arguments:

(a) $\sin(2\pi f_o t + \alpha) = \cos(?)$.
(b) $\cos(2\pi f_o t + \alpha) = \sin(?)$.

1.16 National Instruments Corp. manufactures an A/D converter, Model #NI USB-5133, that is capable of sampling an analog signal at an f_s sample rate of 100 megasamples per second (100 MHz). The A/D converter has internal memory that can store up to 4×10^6 discrete samples. What is the maximum number of cycles of a 25 MHz analog sinewave that can be stored in the A/D converter's memory? Show your work.

1.17 In the first part of the text's Section 1.5 we stated that for a process (or system) to be *linear* it must satisfy a scaling property that we called the *proportionality* characteristic in the text's Eq. (1–14). Determine if the following processes have that proportionality characteristic:

(a) $y_a(n) = x(n-1)/6$,
(b) $y_b(n) = 3 + x(n)$,
(c) $y_c(n) = \sin[x(n)]$.

This problem is *not* "busy work." Knowing if a process (or system) is linear tells us what signal processing principles, and algorithms, can be applied in the analysis of that process (or system).

1.18 There is an often-used process in DSP called *decimation*, and in that process we retain some samples of an $x(n)$ input sequence and discard other $x(n)$ samples. Decimation by a factor of two can be described algebraically by

$$y(m) = x(2n) \tag{P1–3}$$

where index m=0,1,2,3,. . . The decimation defined by Eq. (P1–3) means that $y(m)$ is equal to alternate samples (every other sample) of $x(n)$. For example:

$$y(0) = x(0),\ y(1) = x(2),\ y(2) = x(4),\ y(3) = x(6),\ . . .$$

and so on. Here is the question: Is that decimation process time invariant? Illustrate your answer by decimating a simple sinusoidal $x(n)$ time-domain sequence by a factor of two to obtain $y(m)$. Next, create a shifted-by-one-sample version of $x(n)$ and call it $x_{shift}(n)$. That new sequence is defined by

$$x_{shift}(n) = x(n+1). \tag{P1–4}$$

Finally, decimate $x_{shift}(n)$ according to Eq. (P1–3) to obtain $y_{shift}(m)$. The decimation process is time invariant if $y_{shift}(m)$ is equal to a time-shifted version of $y(m)$. That is, decimation is time invariant if

$$y_{shift}(m) = y(m+1).$$

1.19 In Section 1.7 of the text we discussed the commutative property of linear time-invariant systems. The two networks in Figure P1–19 exhibit that property. Prove this to be true by showing that, given the same $x(n)$ input sequence, outputs $y_1(n)$ and $y_2(n)$ will be equal.

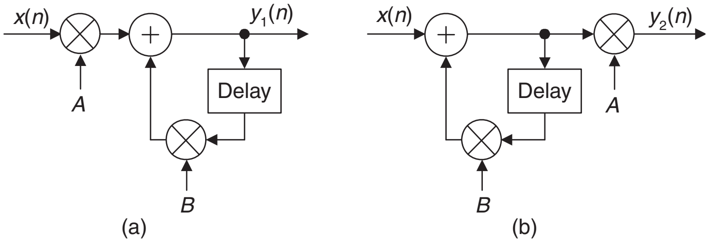


Figure P1–19

1.20 Here we investigate several simple discrete processes that turn out to be useful in a number of DSP applications. Draw the block diagrams, showing their inputs as $x(n)$, of the processes described by the following difference equations:

(a) a *4th-order comb filter*: $y_C(n) = x(n) - x(n-4)$,

(b) an *integrator*: $y_I(n) = x(n) + y_I(n-1)$,

(c) a *leaky integrator*: $y_{LI}(n) = Ax(n) + (1-A)y_{LI}(n-1)$ [the scalar value A is a real-valued constant in the range $0 < A < 1$],

(d) a *differentiator*: $y_D(n) = 0.5x(n) - 0.5x(n\text{-}2)$.

1.21 Draw the unit impulse responses (the output sequences when the input is a unit sample impulse applied at time n=0) of the four processes listed in Problem 1.20. Let A = 0.5 for the leaky integrator. Assume that all sample values within the systems are zero at time n = 0.

1.22 DSP engineers involved in building control systems often need to know what is the *step response* of a discrete system. The step response, $y_{\text{step}}(n)$, can be defined in two equivalent ways. One way is to say that $y_{\text{step}}(n)$ is a system's response to an input sequence of all unity-valued samples. A second definition is that $y_{\text{step}}(n)$ is the cumulative sum (the accumulation, discrete integration) of that system's unit impulse response $y_{\text{imp}}(n)$. Algebraically, this second definition of step response is expressed as

$$y_{\text{step}}(n) = \sum_{k=-\infty}^{n} y_{\text{imp}}(k).$$

In words, the above $y_{\text{step}}(n)$ expression tells us: "The step response at time index n is equal to the sum of all the previous impulse response samples up to and including $y_{\text{imp}}(n)$." With that said, what are the step responses of the four processes listed in Problem 1.20? (Let A = 0.5 for the leaky integrator.) Assume that all sample values within the system are zero at time n=0.

1.23 Thinking about the spectra of signals, the *ideal* continuous (analog) squarewave $s(t)$ in Figure P1–23, whose fundamental frequency is f_o Hz, is equal to the sum of an f_o Hz sinewave and all sinewaves whose frequencies are odd multiples of f_o Hz. We call $s(t)$ "ideal" because we assume the amplitude transitions from plus and minus A occur instantaneously (zero seconds!). Continuous Fourier analysis of the $s(t)$ squarewave allows us to describe this sum of frequencies as the following infinite sum:

$$s(t) = \frac{4A}{\pi}\left[\sin(2\pi f_o t) + \frac{\sin(6\pi f_o t)}{3} + \frac{\sin(10\pi f_o t)}{5} + \frac{\sin(14\pi f_o t)}{7} + \ldots\right].$$

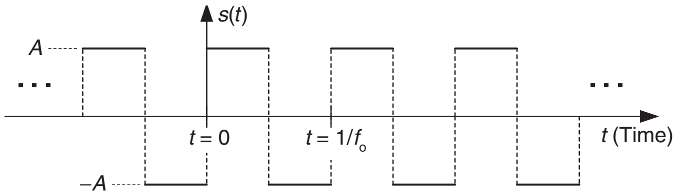


Figure P1–23

Using a summation symbol, we can express squarewave $s(t)$ algebraically as

$$s(t) = \frac{4A}{\pi}\sum_{n=1}^{\infty}\sin(2\pi n f_o t)/n,$$

for n = odd integers only, showing $s(t)$ to be an infinite sum of sinusoids.

(a) Imagine applying $s(t)$ to a filter that completely removes $s(t)$'s lowest-frequency spectral component. Draw the time-domain waveform at the output of such a filter.

(b) Assume $s(t)$ represents a voltage whose f_o fundamental frequency is 1 Hz, and we wish to amplify that voltage to peak amplitudes of ±2A. Over what frequency range must an amplifier operate (that is, what must be the amplifier's *passband width*) in order to exactly double the ideal 1 Hz squarewave's peak-peak amplitude?

1.24 This interesting problem illustrates an *illegal* mathematical operation that we must learn to avoid in our future algebraic activities. The following claims to be a mathematical proof that $4 = 5$. Which of the following steps is illegal? Explain why.

Proof that $4 = 5$:

Step 1: $16 - 36 = 25 - 45$
Step 2: $4^2 - 9 \cdot 4 = 5^2 - 9 \cdot 5$
Step 3: $4^2 - 9 \cdot 4 + 81/4 = 5^2 - 9 \cdot 5 + 81/4$
Step 4: $(4 - 9/2)^2 = (5 - 9/2)^2$
Step 5: $4 - 9/2 = 5 - 9/2$
Step 6: $4 = 5$

This page intentionally left blank

Index

A

C

E

F

I

M

N

O

P

Q

R

S

T

U

V