The Digital Signal Processing Handbook

SECOND EDITION

Digital Signal Processing Fundamentals

EDITOR-IN-CHIEF
Vijay K. Madisetti

CRC Press
Taylor & Francis Group

The Digital Signal Processing Handbook

SECOND EDITION

Digital Signal Processing Fundamentals

EDITOR-IN-CHIEF

Vijay K. Madisetti

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

The Electrical Engineering Handbook Series

Series Editor

Richard C. Dorf
University of California, Davis

Titles Included in the Series

The Handbook of Ad Hoc Wireless Networks, Mohammad Ilyas
The Avionics Handbook, Second Edition, Cary R. Spitzer
The Biomedical Engineering Handbook, Third Edition, Joseph D. Bronzino
The Circuits and Filters Handbook, Second Edition, Wai-Kai Chen
The Communications Handbook, Second Edition, Jerry Gibson
The Computer Engineering Handbook, Vojin G. Oklobdzija
The Control Handbook, William S. Levine
The CRC Handbook of Engineering Tables, Richard C. Dorf
The Digital Avionics Handbook, Second Edition Cary R. Spitzer
The Digital Signal Processing Handbook, Second Edition, Vijay K. Madisetti
The Electrical Engineering Handbook, Second Edition, Richard C. Dorf
The Electric Power Engineering Handbook, Second Edition, Leonard L. Grigsby
The Electronics Handbook, Second Edition, Jerry C. Whitaker
The Engineering Handbook, Third Edition, Richard C. Dorf
The Handbook of Formulas and Tables for Signal Processing, Alexander D. Poularikas
The Handbook of Nanoscience, Engineering, and Technology, Second Edition
William A. Goddard, III, Donald W. Brenner, Sergey E. Lyshevski, and Gerald J. Iafrate
The Handbook of Optical Communication Networks, Mohammad Ilyas and
Hussein T. Mouftah
The Industrial Electronics Handbook, J. David Irwin
The Measurement, Instrumentation, and Sensors Handbook, John G. Webster
The Mechanical Systems Design Handbook, Osita D.I. Nwokah and Yidirim Hurmuzlu
The Mechatronics Handbook, Second Edition, Robert H. Bishop
The Mobile Communications Handbook, Second Edition, Jerry D. Gibson
The Ocean Engineering Handbook, Ferial El-Hawary
The RF and Microwave Handbook, Second Edition, Mike Golio
The Technology Management Handbook, Richard C. Dorf
The Transforms and Applications Handbook, Second Edition, Alexander D. Poularikas
The VLSI Handbook, Second Edition, Wai-Kai Chen

The Digital Signal Processing Handbook, Second Edition

Digital Signal Processing Fundamentals
Video, Speech, and Audio Signal Processing and Associated Standards
Wireless, Networking, Radar, Sensor Array Processing, and Nonlinear Signal Processing

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number: 978-1-4200-4606-9 (Hardback)

Library of Congress Cataloging-in-Publication Data

Digital signal processing fundamentals / editor, Vijay K. Madisetti.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4200-4606-9 (alk. paper)
1. Signal processing--Digital techniques. I. Madisetti, V. (Vijay)

TK5102.5.D4485 2009
621.382'2--dc22 2009022327

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Preface

Digital signal processing (DSP) is concerned with the theoretical and practical aspects of representing information-bearing signals in a digital form and with using computers, special-purpose hardware and software, or similar platforms to extract information, process it, or transform it in useful ways. Areas where DSP has made a significant impact include telecommunications, wireless and mobile communications, multimedia applications, user interfaces, medical technology, digital entertainment, radar and sonar, seismic signal processing, and remote sensing, to name just a few.

Given the widespread use of DSP, a need developed for an authoritative reference, written by the top experts in the world, that would provide information on both theoretical and practical aspects in a manner that was suitable for a broad audience—ranging from professionals in electrical engineering, computer science, and related engineering and scientific professions to managers involved in technical marketing, and to graduate students and scholars in the field. Given the abundance of basic and introductory texts on DSP, it was important to focus on topics that were useful to engineers and scholars without overemphasizing those topics that were already widely accessible. In short, the DSP handbook was created to be relevant to the needs of the engineering community.

A task of this magnitude could only be possible through the cooperation of some of the foremost DSP researchers and practitioners. That collaboration, over 10 years ago, produced the first edition of the successful DSP handbook that contained a comprehensive range of DSP topics presented with a clarity of vision and a depth of coverage to inform, educate, and guide the reader. Indeed, many of the chapters, written by leaders in their field, have guided readers through a unique vision and perception garnered by the authors through years of experience.

The second edition of the DSP handbook consists of volumes on *Digital Signal Processing Fundamentals*; *Video, Speech, and Audio Signal Processing and Associated Standards*; and *Wireless, Networking, Radar, Sensor Array Processing, and Nonlinear Signal Processing* to ensure that each part is dealt with in adequate detail, and that each part is then able to develop its own individual identity and role in terms of its educational mission and audience. I expect each part to be frequently updated with chapters that reflect the changes and new developments in the technology and in the field. The distribution model for the DSP handbook also reflects the increasing need by professionals to access content in electronic form anywhere and at anytime.

Digital Signal Processing Fundamentals, as the name implies, provides a comprehensive coverage of the basic foundations of DSP and includes the following parts: Signals and Systems; Signal Representation and Quantization; Fast Algorithms and Structures; Digital Filtering; Statistical Signal Processing; Adaptive Filtering; Inverse Problems and Signal Reconstruction; and Time–Frequency and Multirate Signal Processing.

I look forward to suggestions on how this handbook can be improved to serve you better.

MATLAB® is a registered trademark of The MathWorks, Inc. For product information, please contact:

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098 USA
Tel: 508 647 7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web: www.mathworks.com

Editor

Vijay K. Madisetti is a professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology in Atlanta. He teaches graduate and undergraduate courses in digital signal processing and computer engineering, and leads a strong research program in digital signal processing, telecommunications, and computer engineering.

Dr. Madisetti received his BTech (Hons) in electronics and electrical communications engineering in 1984 from the Indian Institute of Technology, Kharagpur, India, and his PhD in electrical engineering and computer sciences in 1989 from the University of California at Berkeley.

He has authored or edited several books in the areas of digital signal processing, computer engineering, and software systems, and has served extensively as a consultant to industry and the government. He is a fellow of the IEEE and received the 2006 Frederick Emmons Terman Medal from the American Society of Engineering Education for his contributions to electrical engineering.

Contributors

Joseph Arrowood
IvySys Technologies, LLC
Arlington, Virginia

Bruce W. Bomar
Department of Electrical and Computer Engineering
University of Tennessee Space Institute
Tullahoma, Tennessee

C. Sidney Burrus
Department of Electrical and Computer Engineering
Rice University
Houston, Texas

Zhi Ding
Department of Electrical and Computer Engineering
University of California
Davis, California

Petar M. Djurić
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, New York

John F. Doherty
Department of Electrical Engineering
The Pennsylvania State University
University Park, Pennsylvania

Scott C. Douglas
Department of Electrical Engineering
Southern Methodist University
Dallas, Texas

Pierre Duhamel
CNRS
Gif sur Yvette, France

Kevin R. Farrell
T-NETIX, Inc.
Englewood, Colorado

Ephraim Feig
Innovations-to-Market
San Diego, California

Georgios B. Giannakis
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, Minnesota

Cormac Herley
Microsoft Research
Redmond, Washington

Gabor T. Herman
Department of Computer Science
City University of New York
New York, New York

Alfred Hero
Department of Electrical Engineering and Computer Sciences
University of Michigan
Ann Arbor, Michigan

W. Kenneth Jenkins
Department of Electrical Engineering
The Pennsylvania State University
University Park, Pennsylvania

Thomas Kailath
Department of Electrical Engineering
Stanford University
Stanford, California

Ton Kalker
HP Labs
Palo Alto, California

Lina J. Karam
Department of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, Arizona

Aggelos K. Katsaggelos
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois

Steven M. Kay
Department of Electrical, Computer, and Biomedical Engineering
University of Rhode Island
Kingston, Rhode Island

Stephen Kosonocky
Advanced Micro Devices
Fort Collins, Colorado

Jelena Kovačević
Lucent Technologies
Bell Laboratories
Murray Hill, New Jersey

Vic Larson
Science Applications International Corporation
Arlington, Virginia

B.P. Lathi
Department of Electrical Engineering
California State University
Sacramento, California

Vijay K. Madisetti
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia

Richard J. Mammone
Department of Electrical and Computer Engineering
Rutgers University
Piscataway, New Jersey

Daniel F. Marshall
Raytheon Company
Lexington, Massachusetts

James H. McClellan
Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia

Jerry M. Mendel
Department of Electrical Engineering
University of Southern California
Los Angeles, California

Kambiz Nayebi
Beena Vision Systems Inc.
Roswell, Georgia

Christine Podilchuk
CAIP
Rutgers University
Piscataway, New Jersey

K. Venkatesh Prasad
Ford Motor Company
Detroit, Michigan

Ricardo L. de Queiroz
Engenharia Eletrica
Universidade de Brasilia
Brasília, Brazil

C. Radhakrishnan
Department of Electrical Engineering
The Pennsylvania State University
University Park, Pennsylvania

Ravi P. Ramachandran
Department of Electrical and Computer Engineering
Rowan University
Glassboro, New Jersey

Tami Randolph
Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia

Markus Rupp
Mobile Communications Department
Technical University of Vienna
Vienna, Austria

Ali H. Sayed
Department of Electrical Engineering
University of California at Los Angeles
Los Angeles, California

Ivan W. Selesnick
Department of Electrical and Computer Engineering
Polytechnic University
Brooklyn, New York

Mark J.T. Smith
Department of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana

Iraj Sodagar
PacketVideo
San Diego, California

Clay Stewart
Science Applications International Corporation
Arlington, Virginia

A.C. Surendran
Lucent Technologies
Bell Laboratories
Murray Hill, New Jersey

Charles W. Therrien
Naval Postgraduate School
Monterey, California

Jitendra K. Tugnait
Department of Electrical and Computer Engineering
Auburn University
Auburn, Alabama

Martin Vetterli
École Polytechnique
Lausanne, Switzerland

Douglas B. Williams
Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia

Geoffrey A. Williamson
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, Illinois

Peter Xiao
NeoParadigm Labs. Inc.
San Jose, California

Andrew E. Yagle
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan

Jun Zhang
Department of Electrical Engineering and Computer Science
University of Milwaukee
Milwaukee, Wisconsin

Xiaoyu Zhang
CAIP
Rutgers University
Piscataway, New Jersey

I

Signals and Systems

Vijay K. Madisetti
Georgia Institute of Technology

Douglas B. Williams
Georgia Institute of Technology

THE STUDY OF "SIGNALS AND SYSTEMS" has formed a cornerstone for the development of digital signal processing and is crucial for all of the topics discussed in this book. While the reader is assumed to be familiar with the basics of signals and systems, a small portion is reviewed in this section with an emphasis on the transition from continuous time to discrete time. The reader wishing more background may find in it any of the many fine textbooks in this area, for example [1–6].

In Chapter 1, many important Fourier transform concepts in continuous and discrete time are presented. The discrete Fourier transform, which forms the backbone of modern digital signal processing as its most common signal analysis tool, is also described, together with an introduction to the fast Fourier transform algorithms.

In Chapter 2, the author, B.P. Lathi, presents a detailed tutorial of differential and difference equations and their solutions. Because these equations are the most common structures for both implementing and

modeling systems, this background is necessary for the understanding of many of the later topics in this book. Of particular interest are a number of solved examples that illustrate the solutions to these formulations.

While most software based on workstations and PCs is executed in single or double precision arithmetic, practical realizations for some high throughput digital signal processing applications must be implemented in fixed point arithmetic. These low cost implementations are still of interest to a wide community in the consumer electronics arena. Chapter 3 describes basic number representations, fixed and floating point errors, roundoff noise, and practical considerations for realizations of digital signal processing applications, with a special emphasis on filtering.

References

1. Jackson, L.B., *Signals, Systems, and Transforms*, Addison-Wesley, Reading, MA, 1991.
2. Kamen, E.W. and Heck, B.S., *Fundamentals of Signals and Systems Using MATLAB*, Prentice-Hall, Upper Saddle River, NJ, 1997.
3. Oppenheim, A.V. and Willsky, A.S., with Nawab, S.H., *Signals and Systems*, 2nd ed., Prentice-Hall, Upper Saddle River, NJ, 1997.
4. Strum, R.D. and Kirk, D.E., *Contemporary Linear Systems Using MATLAB*, PWS Publishing, Boston, MA, 1994.
5. Proakis, J.G. and Manolakis, D.G., *Introduction to Digital Signal Processing*, Macmillan, New York; Collier Macmillan, London, UK, 1988.
6. Oppenheim, A.V. and Schafer, R.W., *Discrete Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

1

Fourier Methods for Signal Analysis and Processing

W. Kenneth Jenkins
The Pennsylvania State University

1.1 Introduction

The Fourier transform is a mathematical tool that is used to expand signals into a spectrum of sinusoidal components to facilitate signal representation and the analysis of system performance. In certain applications the Fourier transform is used for spectral analysis, and while in others it is used for spectrum shaping that adjusts the relative contributions of different frequency components in the filtered result. In certain applications the Fourier transform is used for its ability to decompose the input signal into uncorrelated components, so that signal processing can be more effectively implemented on the individual spectral components. Different forms of the Fourier transform, such as the continuous-time (CT) Fourier series, the CT Fourier transform, the discrete-time Fourier transform (DTFT), the discrete

Fourier transform (DFT), and the fast Fourier transform (FFT) are applicable in different circumstances. One goal of this chapter is to clearly define the various Fourier transforms, to discuss their properties, and to illustrate how each form is related to the others in the context of a family tree of Fourier signal processing methods.

Classical Fourier methods such as the Fourier series and the Fourier integral are used for CT signals and systems, i.e., systems in which the signals are defined at all values of t on the continuum $-\infty < t < \infty$. A more recently developed set of discrete Fourier methods, including the DTFT and the DFT, are extensions of basic Fourier concepts for discrete-time (DT) signals and systems. A DT signal is defined only for integer values of n in the range $-\infty < n < \infty$. The class of DT Fourier methods is particularly useful as a basis for digital signal processing (DSP) because it extends the theory of classical Fourier analysis to DT signals and leads to many effective algorithms that can be directly implemented on general computers or special purpose DSP devices.

1.2 Classical Fourier Transform for Continuous-Time Signals

A CT signal $s(t)$ and its Fourier transform $S(j\omega)$ form a transform pair that are related by Equations 1.1a and b for any $s(t)$ for which the integral (Equation 1.1a) converges:

$$S(j\omega) = \int_{-\infty}^{\infty} s(t)\mathrm{e}^{-j\omega t}\mathrm{d}t \tag{1.1a}$$

$$s(t) = \frac{1}{2\Pi}\int_{-\infty}^{\infty} S(j\omega)\mathrm{e}^{j\omega t}\mathrm{d}\omega. \tag{1.1b}$$

In most literature Equation 1.1a is simply called the Fourier transform, whereas Equation 1.1b is called the Fourier integral. The relationship $S(j\omega) = F\{s(t)\}$ denotes the Fourier transformation of $s(t)$, where $F\{\cdot\}$ is a symbolic notation for the integral operator, and where ω is the continuous frequency variable expressed in rad/s. A transform pair $s(t) \leftrightarrow S(j\omega)$ represents a one-to-one invertible mapping as long as $s(t)$ satisfies conditions which guarantee that the Fourier integral converges.

In the following discussion the symbol $\delta(t)$ is used to denote a CT impulse function that is defined to be zero for all $t \neq 0$, undefined for $t = 0$, and has unit area when integrated over the range $-\infty < t < \infty$. From Equation 1.1a it is found that $F\{\delta(t - t_0)\} = \mathrm{e}^{-j\omega t_0}$ due to the well known sifting property of $\delta(t)$. Similarly, from Equation 1.1b we find that $F^{-1}\{2\pi\delta(\omega - \omega_0)\} = \mathrm{e}^{j\omega_0 t}$, so that $\delta(t - t_0) \leftrightarrow \mathrm{e}^{-j\omega t_0}$ and $\mathrm{e}^{j\omega_0 t} \leftrightarrow 2\pi\delta(\omega - \omega_0)$ are Fourier transform pairs. Using these relationships it is easy to establish the Fourier transforms of $\cos(\omega_0 t)$ and $\sin(\omega_0 t)$, as well as many other useful waveforms, many of which are listed in Table 1.1.

The CT Fourier transform is useful in the analysis and design of CT systems, i.e., systems that process CT signals. Fourier analysis is particularly applicable to the design of CT filters which are characterized by Fourier magnitude and phase spectra, i.e., by $|H(j\omega)|$ and $\arg H(j\omega)$, where $H(j\omega)$ is commonly called the frequency response of the filter.

1.2.1 Properties of the Continuous-Time Fourier Transform

The CT Fourier transform has many properties that make it useful for the analysis and design of linear CT systems. Some of the more useful properties are summarized in this section, while a more complete list of the CT Fourier transform properties is given in Table 1.2. Proofs of these properties are found in Oppenheim et al. (1983) and Bracewell (1986). Note that $F\{\cdot\}$ denotes the Fourier transform operation, $F^{-1}\{\cdot\}$ denotes the inverse Fourier transform operation, and "$\star$" denotes the linear convolution operation defined as

TABLE 1.1 CT Fourier Transform Pairs

Single	Fourier Transform	Fourier Series Coefficients (If Periodic)
$\sum_{k=-\infty}^{+\infty} a_k e^{\alpha\omega\delta}$	$2\pi \sum_{k=-\infty}^{+\infty} a_k \delta(\omega_k - \omega_0)$	a_k
$e^{j\omega_0 t}$	$2\pi\delta(\omega - \omega_0)$	$a_1 = 1$ $a_k = 0$, otherwise
$\cos\ \omega_0 t$	$\pi[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$	$a_1 = a_{-1} = 1/2$ $a_k = 0$, otherwise
$\sin\ \omega_0 t$	$\frac{\pi}{j}[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$	$a_1 = -a_{-1} = 1/2j$ $a_k = 0$, otherwise
$x(t) = 1$	$2\pi\delta(\omega)$	$a_0 = 1, a_k = 0, k \neq 0$ (has this Fourier series representation for any choice of $T_0 > 0$)
Periodic square wave $x(t) = \begin{cases} 1, \lvert t\rvert < T_1 \\ 0, T_1 < \lvert t\rvert \leq \frac{T_0}{2} \end{cases}$ and $x(t + T_0) = x(t)$	$\sum_{k=-\infty}^{+\infty} \frac{2\sin k\omega_0 T_1}{k} \delta(\omega_k \omega_0)$	$\frac{\omega_0 T_1}{\pi} \sin c\left(\frac{k\omega_0 T_1}{\pi}\right) = \frac{\sin k\omega_0 T_1}{k\pi}$
$\sum_{n=-\infty}^{+\infty} \delta(t - nT)$	$\frac{2\pi}{T} \sum_{k=-\infty}^{+\infty} k = -\infty\delta\left(\omega - \frac{2\pi k}{T}\right)$	$a_k = \frac{1}{T}$ for all k
$x(t) = \begin{cases} 1, \lvert t\rvert < T_1 \\ 0, \lvert t\rvert > T_1 \end{cases}$	$2T_1 \sin c\left(\frac{\omega T_1}{\pi}\right) = \frac{2\sin \omega T_1}{\omega}$	—
$\frac{W}{\pi} \sin c\left(\frac{Wt}{\pi}\right) = \frac{\sin Wt}{\pi t}$	$X(\omega) = \begin{cases} 1, \lvert\omega\rvert < W \\ 0, \lvert\omega\rvert > W \end{cases}$	—
$\delta(t)$	1	—
$u(t)$	$\frac{1}{j\omega} + \pi\delta(\omega)$	—
$\delta(t - t_0)$	$e^{-j\omega r_0}$	—
$e^{-ar}u(t), \mathrm{Re}\{a\} > 0$	$\frac{1}{a + j\omega}$	—
$te^{-at}u(t), \mathrm{Re}\{a\} > 0$	$\frac{1}{(a + j\omega)^2}$	—
$\frac{t^{n-1}}{(n-1)} e^{-at}u(t),$ $\mathrm{Re}\{a\} > 0$	$\frac{1}{(a + j\omega)^n}$	—

Source: Oppenheim, A.V. et al., *Signals and Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1983. With permission.

$$f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(t) f_2(t - \tau) d\tau.$$

1. Linearity (a and b are complex constants)	$F\{af_1(t) + bf_2(t)\} = aF\{f_1(t)\} + bF\{f_2(t)\}$
2. Time-shifting	$F\{f(t - t_0)\} = e^{-j\omega t_0} F\{f(t)\}$
3. Frequency-shifting	$e^{j\omega_0 t} F^{-1}\{F\{j(\omega - \omega_0)\}$
4. Time-domain convolution	$F\{f_1(t) * f_2(t)\} = F\{f_1(t)\} \cdot F\{f_2(t)\}$
5. Frequency-domain convolution	$F\{f_1(t) \cdot f_2(t)\} = \frac{1}{2\Pi} F\{f_1(t)\} * F\{f_2(t)\}$
6. Time-differentiation	$-j\omega F(j\omega) = F\{d[f(t)]/dt\}$
7. Time-integration	$F\left\{\int_{-\infty}^{t} f(\tau) d\tau\right\} = \frac{1}{j\omega} F(j\omega) + \pi F(0)\delta(\omega)$

TABLE 1.2 Properties of the CT Fourier Transform

Name	If $\mathcal{F}\, f(t) = F(j\omega)$, then:
Definition	$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$
	$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega$
Superposition	$\mathcal{F}[af_1(t) + bf_2(t)] = aF_1(j\omega) + bF_2(j\omega)$
Simplification if:	
(a) $f(t)$ is even	$F(j\omega) = 2\int_0^{\infty} f(t)\cos\omega t\ dt$
(b) $f(t)$ is odd	$F(j\omega) = 2j\int_0^{\infty} f(t)\sin\omega t\ dt$
Negative t	$\mathcal{F}\, f(-t) = F^*(j\omega)$
Scaling:	
(a) Time	$Ff(at) = \frac{1}{\lvert a\rvert}F\left(\frac{j\omega}{a}\right)$
(b) Magnitude	$\mathcal{F}af(t) = aF(j\omega)$
Differentiation	$\mathcal{F}\left[\frac{d^n}{dt^n}f(t)\right] = (j\omega)^n F(j\omega)$
Integration	$\mathcal{F}\left[\int_{-\infty}^{t} f(x)dx\right] = \frac{1}{j\omega}F(j\omega) + \pi F(0)\delta(\omega)$
Time shifting	$\mathcal{F}\, f(t-a) = F(j\omega)e^{-j\omega a}$
Modulation	$\mathcal{F}\, f(t)e^{j\omega_0 t} = F[j(\omega - \omega_0)]$
	$\mathcal{F}\, f(t)\cos\omega_0 t = \frac{1}{2}\{F[j(\omega-\omega_0)] + F[j(\omega+\omega_0)]\}$
	$\mathcal{F}\, f(t)\sin\omega_0 t = \frac{1}{2}j\{F[j(\omega-\omega_0)] + F[j(\omega+\omega_0)]\}$
Time convolution	$\mathcal{F}^{-1}[F_1(j\omega)F_2(j\omega)] = \int_{-\infty}^{\infty} f_1(\tau)f_2(t-\tau)d\tau$
Frequency convolution	$\mathcal{F}[f_1(t)f_2(t)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} F_1(j\lambda)F_2[j(\omega-\lambda)]d\lambda$

Source: Van Valkinburg, M.E., *Network Analysis*, 3rd ed., Prentice Hall, Englewood Cliffs, NJ, 1974. With permission.

The above properties are particularly useful in CT system analysis and design, especially when the system characteristics are easily specified in the frequency domain, as in linear filtering. Note that properties 1, 6, and 7 are useful for solving differential or integral equations. Property 4 (time-domain convolution) provides the basis for many signal processing algorithms, since many systems can be specified directly by their impulse or frequency response. Property 3 (frequency-shifting) is useful for analyzing the performance of communication systems where different modulation formats are commonly used to shift spectral energy among different frequency bands.

1.2.2 Sampling Models for Continuous- and Discrete-Time Signals

The relationship between the CT and the DT domains is characterized by the operations of sampling and reconstruction. If $s_a(t)$ denotes a signal $s(t)$ that has been uniformly sampled every T seconds, then the mathematical representation of $s_a(t)$ is given by

$$s_a(t) = \sum_{n=-\infty}^{n=\infty} s(t)\delta(t - nT), \tag{1.2a}$$

where $\delta(t)$ is the CT impulse function defined previously. Since the only places where the product $s(t)\delta(t - nT)$ is not identically equal to zero are at the sampling instances, $s(t)$ in Equation 1.2a can be replaced with $s(nT)$ without changing the overall meaning of the expression. Hence, an alternate expression for $s_a(t)$ that is often useful in Fourier analysis is

$$s_a(t) = \sum_{n=-\infty}^{n=\infty} s(nT)\delta(t - nT). \tag{1.2b}$$

The CT sampling model $s_a(t)$ consists of a sequence of CT impulse functions uniformly spaced at intervals of T seconds and weighted by the values of the signal $s(t)$ at the sampling instants, as depicted in Figure 1.1. Note that $s_a(t)$ is not defined at the sampling instants because the CT impulse function itself is not defined at $t = 0$. However, the values of $s(t)$ at the sampling instants are imbedded as "area under the curve" of $s_a(t)$, and as such represent a useful mathematical model of the sampling process. In the DT domain, the sampling model is simply the sequence defined by taking the values of $s(t)$ at the sampling instants, i.e.,

$$s[n] = s(t)|_{t=nT}. \tag{1.3}$$

In contrast to $s_a(t)$, which is not defined at the sampling instants, $s[n]$ is well defined at the sampling instants, as illustrated in Figure 1.2. From this discussion it is now clear that $s_a(t)$ and $s[n]$ are different but equivalent models of the sampling process in the CT and DT domains, respectively. They are both useful for signal analysis in their corresponding domains. It will be shown later that their equivalence is established by the fact that they have equal spectra in the Fourier domain, and that the underlying CT signal from which $s_a(t)$ and $s[n]$ are derived can be recovered from either sampling representation provided that a sufficiently high sampling rate is used in the sampling operation.

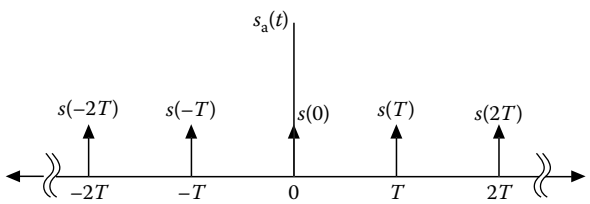

FIGURE 1.1 CT model of a sampled CT signal.

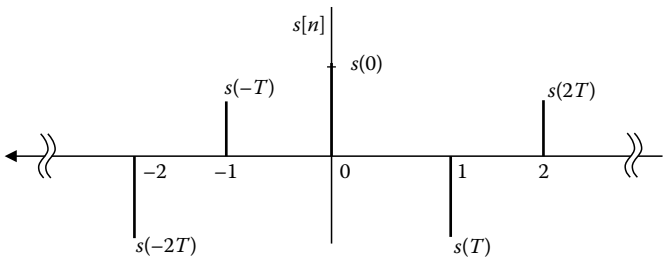

FIGURE 1.2 DT model of a sampled CT signal.

1.2.3 Fourier Spectrum of a Continuous Time Sampled Signal

The operation of uniformly sampling a CT signal $s(t)$ at every T seconds is characterized by Equations 1.2a and b, where $\delta(t)$ is the CT time impulse function defined earlier:

$$s_a(t) = \sum_{n=-\infty}^{\infty} s_a(t)\delta(t - nT) = \sum_{n=-\infty}^{\infty} s_a(nT)\delta(t - nT).$$

Since $s_a(t)$ is a CT signal it is appropriate to apply the CT Fourier transform to obtain an expression for the spectrum of the sampled signal:

$$F\{s_a(t)\} = F\left\{\sum_{n=-\infty}^{\infty} s_a(nT)\delta(t - nT)\right\} = \sum_{n=-\infty}^{\infty} s_a(nT)[e^{j\omega T}]^{-n}. \tag{1.4}$$

Since the expression on the right-hand side of Equation 1.4 is a function of $e^{j\omega T}$ it is customary to express the transform as $F(e^{j\omega T}) = F\{s_a(t)\}$. If ω is replaced with a normalized frequency $\omega' = \omega/T$, so that $-\pi < \omega' < \pi$, then the right-hand side of Equation 1.4 becomes identical to the DTFT that is defined directly for the sequence $s[n] = s_a(nT)$.

1.2.4 Generalized Complex Fourier Transform

The CT Fourier transform characterized by Equation 1.1 can be generalized by considering the variable $j\omega$ to be the special case of $u = \sigma + j\omega$ with $\sigma = 0$, writing Equation 1.1 in terms of u, and interpreting u as a complex frequency variable. The resulting complex Fourier transform pair is given by Equations 1.5a and b (Bracewell 1986):

$$s(t) = \frac{1}{2\Pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} S(u)e^{jut}du \tag{1.5a}$$

$$S(u) = \int_{-\infty}^{\infty} s(t)e^{-jut}dt. \tag{1.5b}$$

The set of all values of u for which the integral of Equation 1.5b converges is called the region of convergence, denoted ROC. Since the transform $S(u)$ is defined only for values of u within the ROC, the path of integration in Equation 1.5a must be defined so the entire path lies within the ROC. In some

literature this transform pair is called the bilateral Laplace transform because it is the same result obtained by including both the negative and positive portions of the time axis in the classical Laplace transform integral. The complex Fourier transform (bilateral Laplace transform) is not often used in solving practical problems, but its significance lies in the fact that it is the most general form that represents the place where Fourier and Laplace transform concepts merge together. Identifying this connection reinforces the observation that Fourier and Laplace transform concepts share common properties because they result from placing different constraints on the same parent form.

1.3 Fourier Series Representation of Continuous Time Periodic Signals

The classical Fourier series representation of a periodic time domain signal $s(t)$ involves an expansion of $s(t)$ into an infinite series of terms that consist of sinusoidal basis functions, each weighted by a complex constant (Fourier coefficient) that provides the proper contribution of that frequency component to the complete waveform. The conditions under which a periodic signal $s(t)$ can be expanded in a Fourier series are known as the Dirichlet conditions. They require that in each period $s(t)$ has a finite number of discontinuities, a finite number of maxima and minima, and satisfies the absolute convergence criterion of Equation 1.6 (VanValkenburg 1974):

$$\int_{-T/2}^{T/2} |s(t)|\mathrm{d}t < \infty. \tag{1.6}$$

It is assumed throughout the following discussion that the Dirichlet conditions are satisfied by all functions that will be represented by a Fourier series.

1.3.1 Exponential Fourier Series

If $s(t)$ is a CT periodic signal with period T the exponential Fourier series expansion of $s(t)$ is given by

$$s(t) = \sum_{n=-\infty}^{\infty} a_n \mathrm{e}^{jn\omega_0 t}, \tag{1.7a}$$

where $\omega_0 = 2\pi/T$. The a_n's are the complex Fourier coefficients given by

$$a_n = \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} s(t)\mathrm{e}^{-jn\omega_0 t}\mathrm{d}t \quad -\infty < n < \infty. \tag{1.7b}$$

For every value of t where $s(t)$ is continuous the right-hand side of Equation 1.7a converges to $s(t)$. At values of t where $s(t)$ has a finite jump discontinuity, the right-hand side of Equation 1.7a converges to the average of $s(t^-)$ and $s(t^+)$, where $s(t^-) = \lim_{\varepsilon\to 0}(t-\varepsilon)$ and $s(t^+) = \lim_{\varepsilon\to 0}(t+\varepsilon)$.

For example, the Fourier series expansion of the sawtooth waveform illustrated in Figure 1.3 is characterized by $T = 2\pi, \omega_0 = 1, a_0 = 0$, and $a_n = a_{-n} = A\cos(n\pi)/(jn\pi)$ for $n = 1, 2, \ldots$. The coefficients of the exponential Fourier series given by Equation 1.5b can be interpreted as a spectral representation of $s(t)$, since the a_nth coefficient represents the contribution of the $(n\omega_0)$th frequency

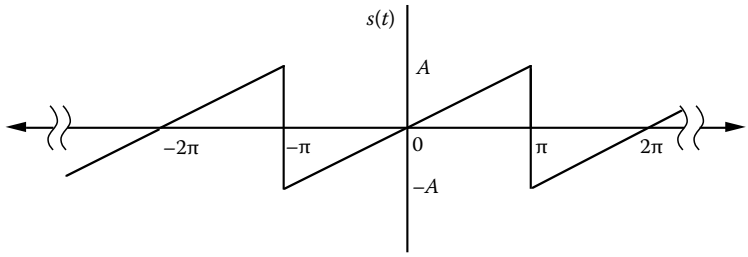

FIGURE 1.3 Periodic CT signal used in Fourier series Example 1.

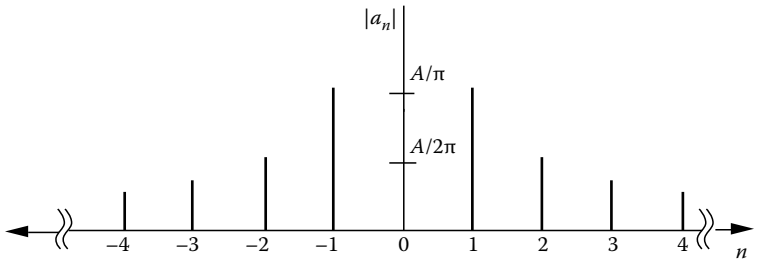

FIGURE 1.4 Magnitude of the Fourier coefficients for Example 1.

component to the complete waveform. Since the a_n's are complex valued, the Fourier domain (spectral) representation has both magnitude and phase spectra. For example, the magnitudes of the a_n's are plotted in Figure 1.4 for the saw tooth waveform of Figure 1.3 (Example 1). The fact that the a_n's constitute a discrete set is consistent with the fact that a periodic signal has a spectrum that contains only integer multiples of the fundamental frequency ω_0. The equation pair given by Equations 1.5a and b can be interpreted as a transform pair that is similar to the CT Fourier transform for periodic signals. This leads to the observation that the classical Fourier series can be interpreted as a special transform that provides a one-to-one invertible mapping between the discrete-spectral domain and the CT domain.

1.3.2 Trigonometric Fourier Series

Although the complex form of the Fourier series expansion is useful for complex periodic signals, the Fourier series can be more easily expressed in terms of real-valued sine and cosine functions for real-valued periodic signals. In the following discussion it is assumed that the signal $s(t)$ is real-valued. When $s(t)$ is periodic and real-valued it is convenient to replace the complex exponential Fourier series with a trigonometric expansion that contains $\sin(\omega_0 t)$ and $\cos(\omega_0 t)$ terms with corresponding real-valued coefficients (VanValkenburg 1974). The trigonometric form of the Fourier series for a real-valued signal $s(t)$ is given by

$$s(t) = \sum_{n=0}^{\infty} b_n \cos(n\omega_0) + \sum_{n=1}^{\infty} c_n \sin(n\omega_0), \tag{1.8a}$$

where $\omega_0 = 2\pi/T$. In Equation 1.8a the b_n's and c_n's are real-valued Fourier coefficients determined by

$$b_0 = \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} s(t)\mathrm{d}t$$

$$b_n = \frac{2}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} s(t)\cos(n\omega_0 t)\mathrm{d}t, \quad n = 1, 2, \ldots \tag{1.8b}$$

$$\text{and} \quad c_n = \frac{2}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} s(t)\sin(n\omega_0 t)\mathrm{d}t, \quad n = 1, 2, \ldots.$$

An arbitrary real-valued signal $s(t)$ can be expressed as a sum of even and odd components, $s(t) = s_{\text{even}}(t) + s_{\text{odd}}(t)$, where $s_{\text{even}}(t) = s_{\text{even}}(-t)$ and $s_{\text{odd}}(t) = -s_{\text{odd}}(-t)$, and where $s_{\text{even}}(t) = [s(t) + s(-t)]/2$ and $s_{\text{odd}}(t) = [s(t) - s(-t)]/2$. For the trigonometric Fourier series, it can be shown that $s_{\text{even}}(t)$ is represented by the (even) cosine terms in the infinite series, $s_{\text{odd}}(t)$ is represented by the (odd) sine terms, and b_0 is the DC level of the signal. Therefore, if it can be determined by inspection that a signal has a DC level, or if it is even or odd, then the correct form of the trigonometric series can be chosen to simplify the analysis. For example, it is easily seen that the signal shown in Figure 1.5 (Example 2) is an even signal with a zero DC level, and therefore, can be accurately represented by the cosine series with $b_n = 2A\sin(\pi n/2)/(\pi n/2), n = 1, 2, \ldots$, as shown in Figure 1.6. In contrast note that the sawtooth waveform used in the previous example is an odd signal with zero DC level, so that it can be completely specified by the sine terms of the trigonometric series. This result can be demonstrated by pairing each positive frequency component from the exponential series with its conjugate partner, i.e., $c_n = \sin(n\omega_0 t) = a_n e^{jn\omega}o^t + a_{-n}e^{-jn\omega}o^t$, whereby it is found that $c_n = 2A\cos(n\pi)/(n\pi)$ for this example. In general it is found that $a_n = (b_n - jc_n)/2$ for $n = 1, 2, \ldots, a_0 = b_0$, and $a_{-n} = a_n^{\star}$. The trigonometric

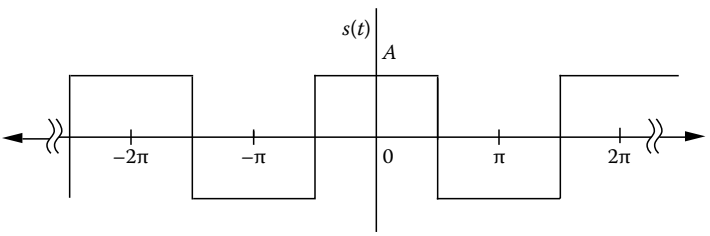

FIGURE 1.5 Periodic CT signal used in Fourier series Example 2.

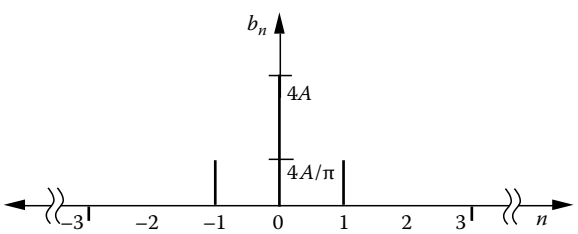

FIGURE 1.6 Fourier coefficients for example of Figure 1.5.

Fourier series is common in the signal processing literature because it replaces complex coefficients with real ones and often results in a simpler and more intuitive interpretation of the results.

1.3.3 Convergence of the Fourier Series

The Fourier series representation of a periodic signal is an approximation that exhibits mean squared convergence to the true signal. If $s(t)$ is a periodic signal of period T, and $s'(t)$ denotes the Fourier series approximation of $s(t)$, then $s(t)$ and $s'(t)$ are equal in the mean square sense if

$$\text{mse} = \int_{\frac{-T}{2}}^{\frac{T}{2}} |s(t) - s\prime(t)|^2 \mathrm{d}t = 0. \tag{1.9}$$

Even with Equation 1.9 is satisfied, mean square error convergence does not guarantee that $s(t) = s'(t)$ at every value of t. In particular, it is known that at values of t where $s(t)$ is discontinuous the Fourier series converges to the average of the limiting values to the left and right of the discontinuity. For example if t_0 is a point of discontinuity, then $s'(t_0) = [s(t_0^-) + s(t_0^+)]/2$, where $s(t_0^-)$ and $s(t_0^+)$ were defined previously (note that at points of continuity, this condition is also satisfied by the very definition of continuity). Since the Dirichlet conditions require that $s(t)$ have at most a finite number of points of discontinuity in one period, the set S_t such that $s(t) \neq s'(t)$ within one period contains a finite number of points, and S_t is a set of measure zero in the formal mathematical sense. Therefore $s(t)$ and its Fourier series expansion $s'(t)$ are equal almost everywhere, and $s(t)$ can be considered identical to $s'(t)$ for analysis in most practical engineering problems.

The condition of convergence almost everywhere is satisfied only in the limit as an infinite number of terms are included in the Fourier series expansion. If the infinite series expansion of the Fourier series is truncated to a finite number of terms, as it must always be in practical applications, then the approximation will exhibit an oscillatory behavior around the discontinuity, known as the Gibbs phenomenon (VanValkenburg 1974). Let $s'_N(t)$ denote a truncated Fourier series approximation of $s(t)$, where only the terms in Equation 1.7a from $n = -N$ to $n = N$ are included if the complex Fourier series representation is used, or where only the terms in Equation 1.8a from $n = 0$ to $n = N$ are included if the trigonometric form of the Fourier series is used. It is well known that in the vicinity of a discontinuity at t_0 the Gibbs phenomenon causes $s'_N(t)$ to be a poor approximation to $s(t)$. The peak magnitude of the Gibbs oscillation is 13% of the size of the jump discontinuity $s(t_0^-) - s(t_0^+)$ regardless of the number of terms used in the approximation. As N increases, the region that contains the oscillation becomes more concentrated in the neighborhood of the discontinuity, until, in the limit as N approaches infinity, the Gibbs oscillation is squeezed into a single point of mismatch at t_0. The Gibbs phenomenon is illustrated in Figure 1.7 where an ideal lowpass frequency response is approximated by impulse response

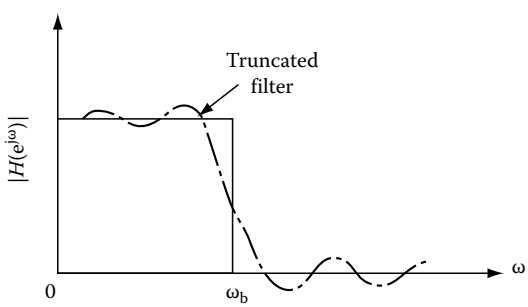

FIGURE 1.7 Gibbs phenomenon in a lowpass digital filter caused by truncating the impulse response to N terms.

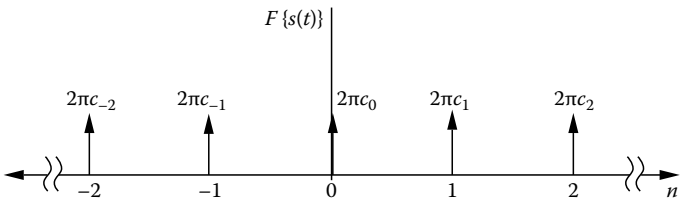

FIGURE 1.8 Spectrum of the Fourier representation of a periodic signal.

function that has been limited to having only N nonzero coefficients, and hence the Fourier series expansion contains only a finite number of terms.

An important property of the Fourier series is that the exponential basis functions $e^{jn\omega}o^{t}$ (or $\sin(n\omega_0 t)$ and $\cos(n\omega_0 t)$ for the trigonometric form) for $n = 0, \pm 1, \pm 2, \ldots$ (or $n = 0, 1, 2, \ldots$ for the trigonometric form) constitute an "orthonormal set," i.e., $t_{nk} = 1$ for $n = k$, and $t_{nk} = 0$ for $n \neq k$, where

$$t_{nk} = \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} (e^{-jn\omega_0 t})(e^{jk\omega_0 t})dt.$$

As terms are added to the Fourier series expansion, the orthogonality of the basis functions guarantees that the approximation error decreases in the mean square sense, i.e., that mse_N decreases monotonically as N is increased, where

$$\text{mse}_N = \int_{\frac{-T}{2}}^{\frac{T}{2}} \left|s(t) - s'_N(t)\right|^2 dt.$$

Therefore, when applying Fourier series analysis including more terms always improves the accuracy of the signal representation.

1.3.4 Fourier Transform of Periodic Continuous Time Signals

For a periodic signal $s(t)$ the CT Fourier transform can then be applied to the Fourier series expansion of $s(t)$ to produce a mathematical expression for the "line spectrum" that is characteristic of periodic signals:

$$F\{s(t)\} = F\left\{\sum_{n=-\infty}^{\infty} a_n e^{jn\omega_0 t}\right\} = 2\pi \sum_{n=-\infty}^{\infty} a_n \delta(\omega - \omega_0). \tag{1.10}$$

The spectrum is shown in Figure 1.8. Note the similarity between the spectral representation of Figure 1.8 and the plots of the Fourier coefficients in Figures 1.4 and 1.6, which were heuristically interpreted as a line spectrum. Figures 1.4 and 1.6 are different from Figure 1.8 but they are equivalent representations of the Fourier line spectrum that is characteristic of periodic signals.

1.4 Discrete-Time Fourier Transform

The DTFT is obtained directly in terms of the sequence samples $s[n]$ by taking the relationship obtained in Equation 1.4 to be the definition of the DTFT. Letting $T = 1$ so that the sampling period is removed from the equations and the frequency variable is replaced with a normalized frequency $\omega' = \omega T$, the DTFT pair is defined by Equation 1.11. In order to simplify notation it is not customary to distinguish

between ω and ω′, but rather to rely on the context of the discussion to determine whether ω refers to the normalized ($T = 1$) or the un-normalized ($T \neq 1$) frequency variable:

$$S(e^{j\omega'}) = \sum_{n=-\infty}^{\infty} s[n]e^{-j\omega' n} \tag{1.11a}$$

$$s[n] = \frac{1}{2\Pi} \int_{-\Pi}^{\Pi} S(e^{j\omega'})e^{jn\omega'} d\omega'. \tag{1.11b}$$

The spectrum $S(e^{j\omega'})$ is periodic in ω′ with period 2π. The fundamental period in the range $-\pi < \omega' \leq \pi$, referred to as the baseband, is the useful frequency range of the DT system because frequency components in this range can be represented unambiguously in sampled form (without aliasing error). In much of the signal processing literature the explicit primed notation is omitted from the frequency variable. However, the explicit primed notation will be used throughout this section because there is a potential for confusion when so many related Fourier concepts are discussed within the same framework.

By comparing Equations 1.4 and 1.11a, and noting that $\omega' = \omega T$, it is seen that

$$F\{s_a(t)\} = \text{DTFT}\{s[n]\},$$

where $s[n] = s_a(t)|_{t=nT}$. This demonstrates that the spectrum of $s_a(t)$ as calculated by the CT Fourier transform is identical to the spectrum of $s[n]$ as calculated by the DTFT. Therefore although $s_a(t)$ and $s[n]$ are quite different sampling models, they are equivalent in the sense that they have the same Fourier domain representation. A list of common DTFT pairs is presented in Table 1.3. Just as the CT Fourier

TABLE 1.3 Some Basic DTFT Pairs

Sequence	Fourier Transform
1. $\delta[n]$	1
2. $\delta[n - n_0]$	$e^{-j\omega n_0}$
3. $1(-\infty < n < \infty)$	$\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega + 2\pi k)$
4. $a^n u[n]$ $(\lvert a\rvert < 1)$	$\frac{1}{1 - ae^{-j\omega}}$
5. $u[n]$	$\frac{1}{1 - e^{-j\omega}} + \sum_{k=-\infty}^{\infty} \pi\delta(\omega + 2\pi k)$
6. $(n+1)a^n u[n]$ $(\lvert a\rvert < 1)$	$\frac{1}{(1 - ae^{-j\omega})^2}$
7. $\frac{r^n \sin \omega_p(n+1)}{\sin \omega_p} u[n]$ $(\lvert r\rvert < 1)$	$\frac{1}{1 - 2r\cos\omega_p e^{-j\omega} + r^2 e^{-j2\omega}}$
8. $\frac{\sin \omega_c n}{\pi n}$	$X(e^{j\omega}) = \begin{cases} 1, & \lvert\omega\rvert < \omega_c, \\ 0, & \omega_c < \lvert\omega\rvert \leq \pi \end{cases}$
9. $x[n] = \begin{cases} 1, & 0 \leq n \leq M \\ 0, & \text{otherwise} \end{cases}$	$\frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)} = e^{-j\omega M/2}$
10. $e^{j\omega_0 n}$	$\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega - \omega_0 + 2\pi k)$
11. $\cos(\omega_0 n + \varphi)$	$\pi \sum_{k=-\infty}^{\infty} \left[e^{j\varphi}\delta(\omega - \omega_0 + 2\pi k) + e^{j\varphi\delta}(\omega + \omega_0 + 2\pi k)\right]$

Source: Oppenheim, A.V. and Schafer, R.W., *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989. With permission.

transform is useful in CT signal system analysis and design, the DTFT is equally useful for DT system analysis and design.

In the same way that the CT Fourier transform was found to be a special case of the complex Fourier transform (or bilateral Laplace transform), the DTFT is a special case of the bilateral z-transform with $z = e^{j\omega' t}$. The more general bilateral z-transform is given by

$$S(z) = \sum_{n=-\infty}^{\infty} s[n] z^{-n} \tag{1.12a}$$

$$s[n] = \frac{1}{2\pi j} \oint_C S(z) z^{n-1} \mathrm{d}z, \tag{1.12b}$$

where C is a counterclockwise contour of integration which is a closed path completely contained within the region of convergence of $S(z)$. Recall that the DTFT was obtained by taking the CT Fourier transform of the CT sampling model $s_a(t)$. Similarly, the bilateral z-transform results by taking the bilateral Laplace transform of $s_a(t)$. If the lower limit on the summation of Equation 1.12a is taken to be $n = 0$, then Equations 1.12a and b become the one-sided z-transform, which is the DT equivalent of the one-sided Laplace transform for CT signals.

1.4.1 Properties of the Discrete-Time Fourier Transform

Since the DTFT is a close relative of the classical CT Fourier transform, it should come as no surprise that many properties of the DTFT are similar to those of the CT Fourier transform. In fact, for many of the properties presented earlier there is an analogous property for the DTFT. The following list parallels the list that was presented earlier for the CT Fourier transform, to the extent that the same properties exist (a more complete list of DTFT properties is given in Table 1.4). Note that $F\{\cdot\}$ denotes the DTFT

TABLE 1.4 Properties of the DTFT

Sequence	Fourier
$x[n]$	$X(e^{j\omega})$
$y[n]$	$Y(e^{j\omega})$
1. $ax[n] + by[n]$	$aX(e^{j\omega}) + bY(e^{j\omega})$
2. $x[n - n_d]$ (n_d an integer)	$e^{-j\omega n_d} X(e^{j\omega})$
3. $e^{j\omega_0 n} x[n]$	$X\left[e^{j(\omega-\omega_0)}\right]$
4. $x[-n]$	$X(e^{-j\omega})$ if $x[n]$ real
	$X^*(e^{j\omega})$
5. $nx[n]$	$j\dfrac{\mathrm{d}X(e^{j\omega})}{\mathrm{d}\omega}$
6. $x[n] = y[n]$	$X(e^{j\omega})Y(e^{j\omega})$
7. $x[n]y[n]$	$\frac{1}{2\pi}\int_{-x}^{x} X(e^{j\theta})Y\left[e^{j(\omega-\theta)}\right]\mathrm{d}\theta$
Parseval's theorem	
8. $\sum_{n=-\infty}^{\infty} \lvert x[n]\rvert^2 = \frac{1}{2\pi}\int_{-x}^{x} \lvert X(e^{j\omega})\rvert^2 \mathrm{d}\omega$	
9. $\sum_{n=-\infty}^{\infty} x[n]y^*[n] = \frac{1}{2\pi}\int_{-x}^{x} X(e^{j\omega})Y^*(e^{j\omega})\,\mathrm{d}\omega$	

Source: Oppenheim, A.V. and Schafer, R.W., *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989. With permission.

operation, $F^{-1}\{\cdot\}$ denotes the inverse DTFT operation, and "$\star$" denotes the DT convolution operation defined as

$$f_1[n] \star f_2[n] = \sum_{k=-\infty}^{+\infty} f_1[n] f_2[n-k].$$

1. Linearity (a and b are complex constants)	$\text{DTFT}\{af_1[n] + bf_2[n]\} = a \cdot \text{DTFT}\{f_1[n]\} + b \cdot \text{DTFT}\{f_2[n]\}$
2. Index-shifting	$\text{DTFT}\{f[n-n_0]\} = \text{e}^{-j\omega n_0}\text{DTFT}\{f[n]\}$
3. Frequency-shifting	$\text{e}^{j\omega_0 n} f[n] = \text{DTFT}^{-1}\{F(j(\omega-\omega_0))\}$
4. Time-domain convolution	$\text{DTFT}\{f_1[n] \star f_2[n]\} = F\{f_1[n]\} \cdot F\{f_2[n]\}$
5. Frequency-domain convolution	$\text{DTFT}\{f_1[n] \cdot f_2[n]\} = \frac{1}{2\Pi}\text{DTFT}\{f_1[n]\} \star \text{DTFT}\{f_2[n]\}$
6. Frequency-differentiation	$nf[n] = \text{DTFT}^{-1}\{\text{d}F(j\omega)/\text{d}\omega\}$

Note that the time-differentiation and time-integration properties of the CT Fourier transform do not have analogous counterparts in the DTFT because time domain differentiation and integration are not defined for DT signals. When working with DT systems practitioners must often manipulate difference equations in the frequency domain. For this purpose the properties of linearity and index-shifting are very important. As with the CT Fourier transform time-domain convolution is also important for DT systems because it allows engineers to work with the frequency response of the system in order to achieve proper shaping of the input spectrum, or to achieve frequency selective filtering for noise reduction or signal detection.

1.4.2 Relationship between the CT and DT Spectra

Since DT signals often originate by sampling a CT signal, it is important to develop the relationship between the original spectrum of the CT signal and the spectrum of the DT signal that results. First, the CT Fourier transform is applied to the CT sampling model, and the properties are used to produce the following result:

$$F\{s_\text{a}(t)\} = F\left\{ s_\text{a}(t) \sum_{n=-\infty}^{\infty} \delta(t-nT) \right\} = \frac{1}{2\pi} \text{S}_\text{a}(j\omega) F\left\{ \sum_{n=-\infty}^{\infty} \delta(t-nT) \right\}. \tag{1.13}$$

Since the sampling function (summation of shifted impulses) on the right-hand side of Equation 1.13 is periodic with period T it can be replaced with a CT Fourier series expansion and the frequency-domain convolution property of the CT Fourier transform can be applied to yield two equivalent expressions for the DT spectrum:

$$S(\text{e}^{j\omega T}) = \frac{1}{T} \sum_{n=-\infty}^{\infty} S_\text{a}(j[\omega - n\omega_\text{s}]) \quad \text{or} \quad S(\text{e}^{j\omega'}) = \frac{1}{T} \sum_{n=-\infty}^{\infty} S_\text{a}(j[\omega' - n2\pi/T]). \tag{1.14}$$

In Equation 1.14 $\omega_\text{s} = (2\pi/T)$ is the sampling frequency and $\omega' = \omega T$ is the normalized DT frequency axis expressed in radians. Note that $S(\text{e}^{j\omega T}) = S(\text{e}^{j\omega'})$ consists of an infinite number of replicas of the CT spectrum $S(j\omega)$, positioned at intervals of $(2\pi/T)$ on the ω-axis (or at intervals of 2π on the ω'-axis), as illustrated in Figure 1.9. Note that if $S(j\omega)$ is band-limited with a bandwidth ω_c, and if T is chosen sufficiently small so that $\omega_\text{s} > 2\omega_\text{c}$, then the DT spectrum is a copy of $S(j\omega)$ (scaled by $1/T$) in the baseband. The limiting case of $\omega_\text{s} = 2\omega_\text{c}$ is called the Nyquist sampling frequency. Whenever a CT signal

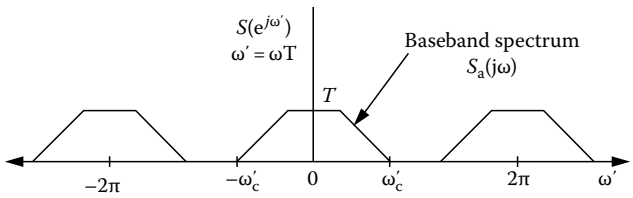

FIGURE 1.9 Relationship between the CT and DT spectra.

is sampled at or above the Nyquist rate, no aliasing distortion occurs (i.e., the baseband spectrum does not overlap with the higher order replicas) and the CT signal can be exactly recovered from its samples by extracting the baseband spectrum of $S(e^{j\omega'})$ with an ideal lowpass filter that recovers the original CT spectrum by removing all spectral replicas outside the baseband and scaling the baseband by a factor of T.

1.5 Discrete Fourier Transform

To obtain the DFT the continuous-frequency domain of the DTFT is sampled at N points uniformly spaced around the unit circle in the z-plane, i.e., at the points $\omega_k = (2\pi k/N), k = 0, 1, \ldots, N-1$. The result is the DFT transform pair defined by Equations 1.15a and b:

$$S[k] = \sum_{n=0}^{N-1} s[n] e^{-j\frac{2\pi kn}{N}}, \quad k = 0, 1, \ldots, N-1 \tag{1.15a}$$

$$s[k] = \frac{1}{N} \sum_{k=0}^{N-1} S[k] e^{j\frac{2\pi kn}{N}}, \quad n = 0, 1, \ldots, N-1, \tag{1.15b}$$

The signal $s[n]$ is either a finite length sequence of length N, or it is a periodic sequence with period N. Regardless of whether $s[n]$ is a finite length or periodic sequence, the DFT treats the N samples of $s[n]$ as though they are one period of a periodic sequence. This is a peculiar feature of the DFT, and one that must be handled properly in signal processing to prevent the introduction of artifacts.

1.5.1 Properties of the DFT

Important properties of the DFT are summarized in Table 1.5. The notation $[k]_N$ denotes k modulo N, and $R_N[n]$ is a rectangular window such that $R_N[n] = 1$ for $n = 0, \ldots, N-1$, and $R_N[n] = 0$ for $n < 0$ and $n \geq N$. The transform relationship given by Equations 1.15a and 1.15b is also valid when $s[n]$ and $S[k]$ are periodic sequences, each of period N. In this case n and k are permitted to range over the complete set of real integers, and $S[k]$ is referred to as the discrete Fourier series (DFS). In some cases the DFS is developed as a distinct transform pair in its own right (Jenkins and Desai 1986). Whether or not the DFT and the DFS are considered identical or distinct is not important in this discussion. The important point to be emphasized here is that the DFT treats $s[n]$ as though it were a single period of a periodic sequence, and all signal processing done with the DFT will inherit the consequences of this assumed periodicity.

Most of the properties listed in Table 1.5 for the DFT are similar to those of the z-transform and the DTFT, although there are important differences. For example, Property 5 (time-shifting property), holds for circular shifts of the finite length sequence $s[n]$, which is consistent with the notion that the DFT treats $s[n]$ as one period of a periodic sequence. Also, the multiplication of two DFTs results in the circular convolution of the corresponding DT sequences, as specified by Property 7. This later property is quite different from the linear convolution property of the DTFT. Circular convolution is simply a linear

TABLE 1.5 Properties of the DFT

Finite-Length Sequence (Length N)	N-Point DFT (Length N)
1. $x[n]$	$X[k]$
2. $x_1[n], x_2[n]$	$X_1[k], X_2[k]$
3. $ax_1[n] + bx_2[n]$	$aX_1[k] + bX_2[k]$
4. $X[n]$	$Nx[(-k)_N]$
5. $x[(n-m)_N]$	$W_N^{km}X[k]$
6. $W_N^{-\ell n}x[n]$	$X[(k-\ell)_N]$
7. $\sum_{m=0}^{N-1} x_1(m)x_2[(n-m)_N]$	$X_1[k]X_2[k]$
8. $x_1[n]x_2[n]$	$\frac{1}{N}\sum_{\ell=0}^{N-1} X_1(\ell)X_2[(k-\ell)_N]$
9. $x^*[n]$	$X^*[(-k)_N]$
10. $x^*[(-n)_N]$	$X^*[k]$
11. $\mathrm{Re}\{x[n]\}$	$X_{\mathrm{ep}}[k] = \frac{1}{2}\{X[(k)_N] + X^*[(-k)_N]\}$
12. $j\mathrm{Im}\{x[n]\}$	$X_{\mathrm{op}}[k] = \frac{1}{2}\{X[(k)_N] - X^*[(-k)_N]\}$
13. $x_{\mathrm{ep}}[n] = \frac{1}{2}\{x[n] + x^*[(-n)_N]\}$	$\mathrm{Re}\{X[k]\}$
14. $x_{\mathrm{op}}[n] = \frac{1}{2}\{x[n] - x^*[(-n)_N]\}$	$j\mathrm{Im}\{X[k]\}$
Properties 15–17 apply only when $x[n]$ is real	
15. Symmetry properties	$\begin{cases} X[k] = X^*[(-k)_N] \\ \mathrm{Re}\{X[k]\} = \mathrm{Re}\{X[(-k)_N]\} \\ \mathrm{Im}\{X[k]\} = -\mathrm{Im}\{X[(-k)_N]\} \\ \lvert X[k]\rvert = \lvert X[(-k)_N]\rvert \\ \measuredangle X[k] = -\measuredangle\{X[(-k)_N]\}. \end{cases}$
16. $x_{\mathrm{ep}}[n] = \frac{1}{2}\{x[n] + x[(-n)_N]\}$	$\mathrm{Re}\{X[k]\}$
17. $x_{\mathrm{op}}[n] = \frac{1}{2}\{x[n] - x[(-n)_N]\}$	$j\mathrm{Im}\{X[k]\}$

Source: Oppenheim, A.V. and Schafer, R.W., *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989. With permission.

convolution of the periodic extensions of the finite sequences being convolved, where each of the finite sequences of length N defines the structure of one period of the periodic extensions.

For example, suppose it is desired to implement a digital filter with finite impulse response (FIR) $h[n]$. The output in response to $s[n]$ is

$$y[n] = \sum_{k=0}^{N-1} h[k]s[n-k] \tag{1.16}$$

which is obtained by transforming $h[n]$ and $s[n]$ into $H[k]$ and $S[k]$ using the DFT, multiplying the transforms point-wise to obtain $Y[k] = H[k]S[k]$, and then using the inverse DFT to obtain $y[n] = \mathrm{DFT}^{-1}\{Y[k]\}$. If $s[n]$ is a finite sequence of length M, then the results of the circular convolution implemented by the DFT will correspond to the desired linear convolution if and only if the block length of the DFT, N_{DFT}, is chosen sufficiently large so that $N_{\mathrm{DFT}} > N + (M-1)$ and both $h[n]$ and $s[n]$ are padded with zeros to form blocks of length N_{DFT}.

1.5.2 Fast Fourier Transform Algorithms

The DFT is typically implemented in practice with one of the common forms of the FFT algorithm. The FFT is not a Fourier transform in its own right, but rather it is simply a computationally efficient

algorithm that reduces the complexity of the computing DFT from Order $\{N^2\}$ to Order $\{N \log_2 N\}$. When N is large, the computational savings provided by the FFT algorithm is so great that the FFT makes real-time DFT analysis practical in many situations which would be entirely impractical without it. There are numerous FFT algorithms, including decimation-in-time (D-I-T) algorithms, decimation-in-frequency (D-I-F) algorithms, bit-reversed algorithms, normally ordered algorithms, mixed-radix algorithms (for block lengths that are not powers-of-2 [PO2]), prime factor algorithms, and Winograd algorithms [Blahut 1985]. The D-I-T and the D-I-F radix-2 FFT algorithms are the most widely used in practice. Detailed discussions of various FFT algorithms can be found in Brigham (1974) and Oppenheim and Schafer (1975).

The FFT is easily understood by examining the simple example of $N = 8$. There are numerous ways to develop the FFT algorithm, all of which deal with a nested decomposition of the summation operator of Equation 1.20a. The development presented here is called an algebraic development of the FFT because it follows straightforward algebraic manipulation. First, each of the summation indices (k, n) in Equation 1.15a is expressed as explicit binary integers, $k = 4k_2 + 2k_1 + k_0$ and $n = 4n_2 + 2n_1 + n_0$, where k_i and n_i are bits that take on the values of either 0 or 1. If these expressions are substituted into Equation 1.20a, all terms in the exponent that contain the factor $N = 8$ can be deleted because $e^{-j2\pi l} = 1$ for any integer l. Upon deleting such terms and re-grouping the remaining terms, the product nk can be expressed in either of two ways:

$$nk = (4k_0)n_2 + (4k_1 + 2k_0)n_1 + (4k_2 + 2k_1 + k_0)n_0 \tag{1.17a}$$

$$nk = (4n_0)k_2 + (4n_1 + 2n_0)k_1 + (4n_2 + 2n_1 + n_0)k_0. \tag{1.17b}$$

Substituting Equation 1.17a into Equation 1.15a leads to the D-I-T FFT, whereas substituting Equation 1.25b leads to the D-I-F FFT. Only the D-I-T FFT is discussed further here. The D-I-F and various related forms are treated in detail in Oppenheim and Schafer (1975).

The D-I-T FFT decomposes into $\log_2 N$ stages of computation, plus a stage of bit reversal,

$$x_1[k_0, n_1, n_0] = \sum_{n_2=0}^{n_2=1} s[n_2, n_1, n_0] W_8^{4k_0n_2} \quad \text{(stage 1)} \tag{1.18a}$$

$$x_2[k_0, k_1, n_0] = \sum_{n_1=0}^{n_1=1} x_1[k_0, n_1, n_0] W_8^{(4k_1+2k_0)n_1} \quad \text{(stage 2)} \tag{1.18b}$$

$$x_3[k_0, k_1, k_2] = \sum_{n_0=0}^{n_0=1} x_2[k_0, k_1, n_0] W_8^{(4k_2+2k_1+k_0)n_0} \quad \text{(stage 3)} \tag{1.18c}$$

$$s(k_2, k_1, k_0) = x_3(k_0, k_1, k_2) \quad \text{(bit reversal)}. \tag{1.18d}$$

In each summation above, one of the n_i's is summed out of the expression, while at the same time a new k_i is introduced. The notation is chosen to reflect this. For example, in stage 3, n_0 is summed out, k_2 is introduced as a new variable, and n_0 is replaced by k_2 in the result. The last operation, called bit reversal, is necessary to correctly locate the frequency samples $X[k]$ in the memory. It is easy to show that if the samples are paired correctly, an in-place computation can be done by a sequence of butterfly operations. The term in-place means that each time a butterfly is to be computed, a pair of data samples is read from memory, and the new data pair produced by the butterfly calculation is written back into the memory locations where the original pair was stored, thereby overwriting the original data. An in-place algorithm is designed so that each data pair is needed for only one butterfly, and so the new results can be immediately stored on top of the old in order to minimize memory requirements.

For example, in stage 3 the $k = 6$ and $k = 7$ samples should be paired, yielding a "butterfly" computation that requires one complex multiply, one complex add, and one subtract:

$$x_3(1,1,0) = x_2(1,1,0) + W_8^3 x_2(1,1,1) \tag{1.19a}$$

$$x_3(1,1,1) = x_2(1,1,0) - W_8^3 x_2(1,1,1) \tag{1.19b}$$

Samples $x_2(6)$ and $x_2(7)$ are read from the memory, the butterfly is executed on the pair, and $x_3(6)$ and $x_3(7)$ are written back to the memory, overwriting the original values of $x_2(6)$ and $x_2(7)$. In general, there are $N/2$ butterflies per stage and $\log_2 N$ stages, so the total number of butterflies is $(N/2)\log_2 N$. Since there is at most one complex multiplication per butterfly, the total number of multiplications is bounded by $(N/2)\log_2 N$ (some of the multiplies involve factors of unity and should not be counted).

Figure 1.10 shows the signal flow graph of the D-I-T FFT for $N = 8$. This algorithm is referred to as an in-place FFT with normally ordered input samples and bit-reversed outputs. Minor variations that include bit-reversed inputs and normally ordered outputs, and non-in-place algorithms with normally ordered inputs and outputs are possible. Also, when N is not a PO2, a mixed-radix algorithm can be used to reduce computation. The mixed-radix FFT is most efficient when N is highly composite, i.e., $N = p_1^{r_1} p_2^{r_2} \cdots p_L^{r_L}$, where the p_i's are small prime numbers and the r_i's are positive integers. It can be shown that the order of complexity of the mixed-radix FFT is Order $\{N[r_1(p_1 - 1) + r_2(p_2 - 1) + \cdots + r_L(p^L - 1)]\}$. Because of the lack of uniformity of structure among stages, this algorithm has not received much attention for hardware implementation. However, the mixed-radix FFT is often used in software applications, especially for processing data recorded in laboratory experiments where it is not convenient to restrict the block lengths to be PO2. Many advanced FFT algorithms, such as higher radix forms, the mixed-radix form, prime-factor algorithm, and the Winograd algorithm are described in Blahut (1985). Algorithms specialized for real-valued data reduce the computational cost by a factor of 2.

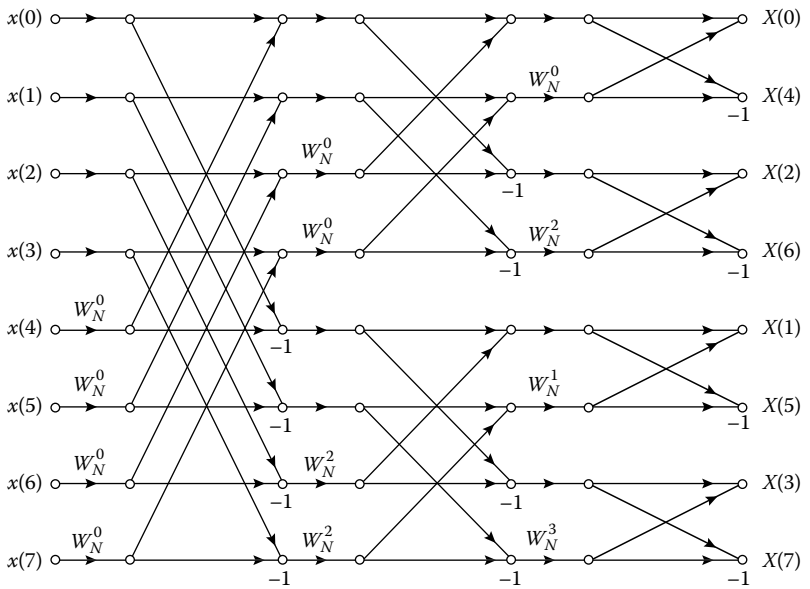

FIGURE 1.10 D-I-T FFT algorithm with normally ordered inputs and bit-reversed outputs.

1.6 Family Tree of Fourier Transforms

Figure 1.11 illustrates the functional relationships among the various forms of CT Fourier transform and DTFT that have been discussed in the previous sections. The family of CT Fourier transforms is shown on the left side of Figure 1.11, whereas the right side of the figure shows the hierarchy of DTFTs. Note that the most general, and consequently the most powerful, Fourier transform is the classical complex Fourier transform (or equivalently, the bilateral Laplace transform). Note also that the complex Fourier transform is identical to the bilateral Laplace transform, and it is at this level that the classical Laplace transform techniques and Fourier transform techniques become identical. Each special member of the CT Fourier family is obtained by impressing certain constraints on the general form, thereby producing special transforms that are simpler and more useful in practical problems where the constraints are met. In Figure 1.11 it is seen that the bilateral z-transform is analogous to the complex Fourier transform, the unilateral z-transform is analogous to the classical (one-sided) Laplace transform, the DTFT is analogous to the classical Fourier (CT) transform, and the DFT is analogous to the classical (CT) Fourier series.

1.6.1 Walsh–Hadamard Transform

The Walsh–Hadamard transform (WHT) is a computationally attractive orthogonal transform that is structurally related to the DFT, and which can be implemented in practical applications without

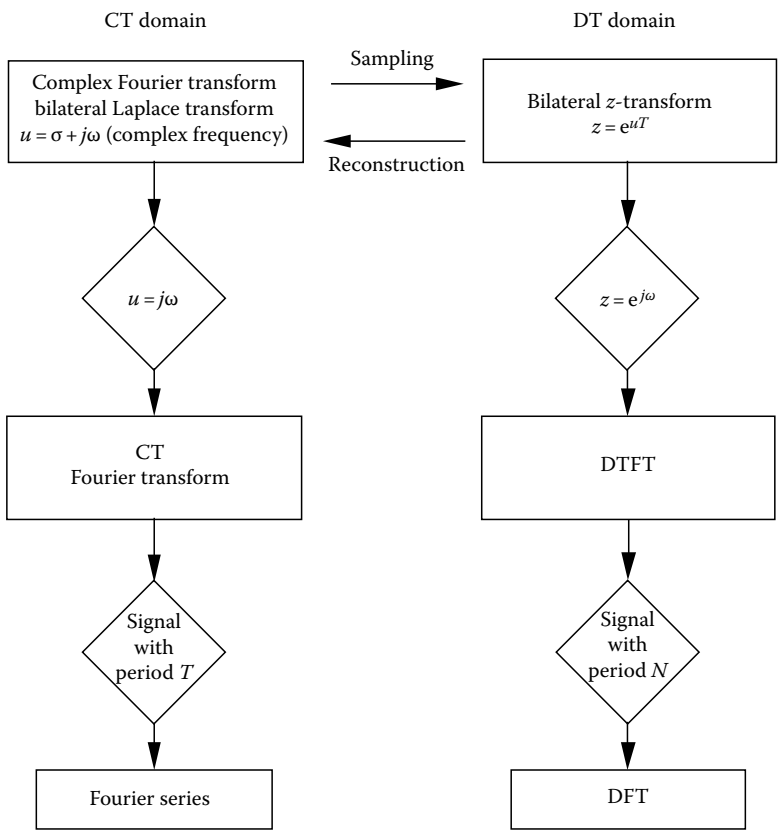

FIGURE 1.11 Functional relationships among various forms of the Fourier transform.

multiplication, and with a computational complexity for addition that is on the same order of complexity as that of an FFT. The t_{mk}th element of the WHT matrix $\mathbf{T}_{\text{WHT}}$ is given by

$$t_{mk} = \frac{1}{\sqrt{N}} \prod_{\ell=0}^{p-1} (-1)^{b_l(m)b_{p-1-\ell}(k)}, \quad m \text{ and } k = 0, \ldots, N-1,$$

where $b_\ell(m)$ is the ℓth order bit in the binary representation of m, and $N = 2^p$. The WHT is defined only when N is a PO2. Note that the columns of $\mathbf{T}_{\text{WHT}}$ form a set of orthogonal basis vectors whose elements are all 1's or -1's, so that the calculation of the matrix-vector product $\mathbf{T}_{\text{WHT}}\mathbf{X}$ can be accomplished with only additions and subtractions. It is well known that $\mathbf{T}_{\text{WHT}}$ of dimension $(N \times N)$, for N a PO2, can be computed recursively according to

$$\mathbf{T}_k = \begin{bmatrix} \mathbf{T}_{k/2} & \mathbf{T}_{k/2} \\ \mathbf{T}_{k/2} & -\mathbf{T}_{k/2} \end{bmatrix} \quad \text{for } K = 4, \ldots, N \text{ (even)} \quad \text{and} \quad \mathbf{T}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The above relationship provides a convenient way of quickly constructing the Walsh–Hadamard matrix for any arbitrary (even) size N.

Due to structural similarities between the DFT and the WHT matrices, the WHT transform can be implemented using a modified FFT algorithm. The core of any FFT program is a butterfly calculation that is characterized by a pair of coupled equations that have the following form:

$$X_{i+1}(\ell, m) = X_i(\ell, m) + \mathrm{e}^{j\theta(\ell,m,k,s)} X_i(k, s)$$
$$X_{i+1}(\ell, m) = X_i(\ell, m) - \mathrm{e}^{j\theta(\ell,m,k,s)} X_i(k, s).$$

If the exponential factor in the butterfly calculation is replaced by a "1," so the "modified butterfly" calculation becomes

$$X_{i+1}(\ell, m) = X_i(\ell, m) + X_i(k, s)$$
$$X_{i+1}(\ell, m) = X_i(\ell, m) - X_i(k, s),$$

the modified FFT program will in fact perform a WHT on the input vector. This property not only provides a quick and convenient way to implement the WHT, but is also establishes clearly that in addition to the WHT requiring no multiplication, the number of additions required has order of complexity of $(N/2)\log_2 N$, i.e., the same as the that of the FFT.

The WHT is used in many applications that require signals to be decomposed in real time into a set of orthogonal components. A typical application in which the WHT has been used in this manner is in code division multiple access (CDMA) wireless communication systems. A CDMA system requires spreading of each user's signal spectrum using a PN sequence. In addition to the PN spreading codes, a set of length-64 mutually orthogonal codes, called the Walsh codes, are used to ensure orthogonality among the signals for users received from the same base station. The length $N = 64$ Walsh codes can be thought of as the orthogonal column vectors from a (64×64) Walsh–Hadamard matrix, and the process of demodulation in the receiver can be interpreted as performing a WHT on the complex input signal containing all the modulated user's signals so they can be separated for accurate detection.

1.7 Selected Applications of Fourier Methods

1.7.1 DFT (FFT) Spectral Analysis

An FFT program is often used to perform spectral analysis on signals that are sampled and recorded as part of laboratory experiments, or in certain types of data acquisition systems. There are several issues to

be addressed when spectral analysis is performed on (sampled) analog waveforms that are observed over a finite interval of time.

1.7.1.1 Windowing

The FFT treats the block of data as though it were one period of a periodic sequence. If the underlying waveform is not periodic, then harmonic distortion may occur because the periodic waveform created by the FFT may have sharp discontinuities at the boundaries of the blocks. This effect is minimized by removing the mean of the data (it can always be reinserted) and by windowing the data so the ends of the block are smoothly tapered to zero. A good rule of thumb is to taper 10% of the data on each end of the block using either a cosine taper or one of the other common windows (e.g., Hamming, Von Hann, Kaiser windows, etc.). An alternate interpretation of this phenomenon is that the finite length observation has already windowed the true waveform with a rectangular window that has large spectral sidelobes. Hence, applying an additional window results in a more desirable window that minimizes frequency-domain distortion.

1.7.1.2 Zero-Padding

An improved spectral analysis is achieved if the block length of the FFT is increased. This can be done by (1) taking more samples within the observation interval, (2) increasing the length of the observation interval, or (3) augmenting the original data set with zeros. First, it must be understood that the finite observation interval results in a fundamental limit on the spectral resolution, even before the signals are sampled. The CT rectangular window has a $(\sin x)/x$ spectrum, which is convolved with the true spectrum of the analog signal. Therefore, the frequency resolution is limited by the width of the mainlobe in the $(\sin x)/x$ spectrum, which is inversely proportional to the length of the observation interval. Sampling causes a certain degree of aliasing, although this effect can be minimized by using a sufficiently high sampling rate. Therefore, lengthening the observation interval increases the fundamental resolution limit, while taking more samples within the observation interval minimizes aliasing distortion and provides a better definition (more sample points) on the underlying spectrum.

Padding the data with zeros and computing a longer FFT does give more frequency domain points (improved spectral resolution), but it does not improve the fundamental limit, nor does it alter the effects of aliasing error. The resolution limits are established by the observation interval and the sampling rate. No amount of zero padding can improve these basic limits. However, zero padding is a useful tool for providing more spectral definition, i.e., it enables one to get a better look at the (distorted) spectrum that results once the observation and sampling effects have occurred.

1.7.1.3 Leakage and the Picket-Fence Effect

An FFT with block length N can accurately resolve only frequencies $w_k = (2\pi/N)k, k = 0, \ldots, N-1$ that are integer multiples of the fundamental $w_1 = (2\pi/N)$. An analog waveform that is sampled and subjected to spectral analysis may have frequency components between the harmonics. For example, a component at frequency $w_{k+1/2} = (2\pi/N)(k + 1/2)$ will appear scattered throughout the spectrum. The effect is illustrated in Figure 1.12 for a sinusoid that is observed through a rectangular window and then sampled a N points. The "picket-fence effect" means that not all frequencies can be seen by the FFT. Harmonic components are seen accurately, but other components "slip through the picket fence" while their energy is "leaked" into the harmonics. These effects produce artifacts in the spectral domain that must be carefully monitored to assure that an accurate spectrum is obtained from FFT processing.

1.7.2 FIR Digital Filter Design

A common method for designing FIR digital filters is by use of windowing and FFT analysis. In general, window designs can be carried out with the aid of a hand calculator and a table of well-known window

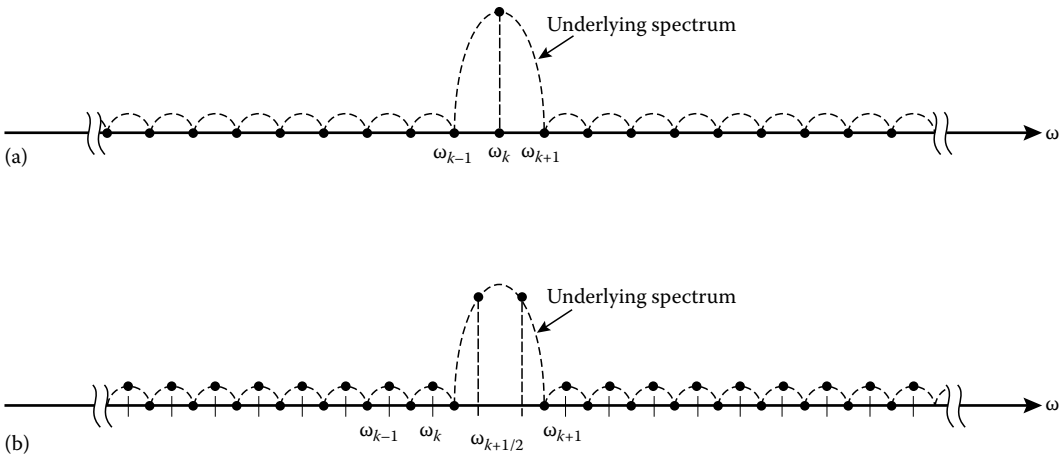

FIGURE 1.12 Illustration of leakage and the picket fence effects. (a) FFT of a windowed sinusoid with frequency $\omega_k = 2\pi k/N$ and (b) leakage for a nonharmonic sinusoidal component.

functions. Let $h[n]$ be the impulse response that corresponds to some desired frequency response, $H(e^{j\omega})$. If $H(e^{j\omega})$ has sharp discontinuities then $h[n]$ will represent an infinite impulse response function. The objective is to time-limit $h[n]$ in such a way as to not distort $H(e^{j\omega})$ any more than necessary. If $h[n]$ is simply truncated, a ripple (Gibbs phenomenon) occurs around the discontinuities in the spectrum, resulting in a distorted filter, as was earlier illustrated in Figure 1.7.

Suppose that $w[n]$ is a window function that time-limits $h[n]$ to create an FIR approximation, $h'[n]$; i.e., $h'[n] = w[n]h[n]$. Then if $W(e^{j\omega})$ is the DTFT of $w[n]$, $h'[n]$ will have a Fourier transform given by $H'(e^{j\omega}) = W(e^{j\omega}) * H(e^{j\omega})$, where $*$ denotes convolution. From this it can be seen that the ripples in $H'(e^{j\omega})$ result from the sidelobes of $W(e^{j\omega})$. Ideally, $W(e^{j\omega})$ should be similar to an impulse so that $H'(e^{j\omega})$ is approximately equal to $H(e^{j\omega})$.

1.7.2.1 Special Case

Let $h[n] = \cos n\omega_0$, for all n. Then $h[n] = w[n]\cos n\omega_0$, and

$$H'(e^{j\omega}) = (1/2)W(e^{j(\omega+\tilde{\omega})}) + (1/2)W(e^{j(\omega-\tilde{\omega})}) \tag{1.20}$$

as illustrated in Figure 1.13. For this simple class, the center frequency of the passband is controlled by ω_0, and both the shape of the passband and the sidelobe structure are strictly determined by the choice of the window. While this simple class of FIRs does not allow for very flexible designs, it is a simple technique for determining quite useful lowpass, bandpass, and highpass FIR filters.

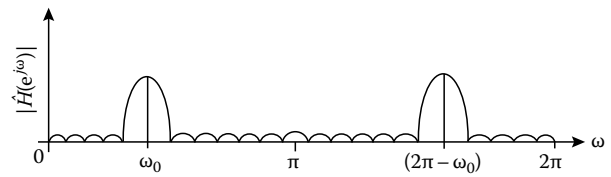

FIGURE 1.13 Design of a simple bandpass FIR filter by windowing.

1.7.2.2 General Case

Specify an ideal frequency response, $H(e^{j\omega})$, and choose samples at selected values of w. Use a long inverse FFT of length N' to find $h'[n]$, an approximation to $h[n]$, where if N is the desired length of the final filter, then $N' \gg N$. Then use a carefully selected window to truncate $h'[n]$ to obtain $h[n]$ by letting $h[n] = w[n]h'[n]$. Finally, use an FFT of length N' to find $H'(e^{j\omega})$. If $H'(e^{j\omega})$ is a satisfactory approximation to $H(e^{j\omega})$, the design is finished. If not, choose a new $H(e^{j\omega})$, or a new $w[n]$ and repeat. Throughout the design procedure it is important to choose $N' = kN$, with k an integer that is typically in the range $[4, \ldots, 10]$. Since this design technique is a trial-and-error procedure, the quality of the result depends to some degree on the skill and experience of the designer.

1.7.3 Fourier Block Processing in Real-Time Filtering Applications

In some practical applications, either the value of M is too large for the memory available, or $s[n]$ may not actually be finite in length, but rather a continual stream of data samples that must be processed by a filter at real time rates. Two well-known algorithms are available that partition $s[n]$ into smaller blocks and process the individual blocks with a smaller-length DFT: (1) overlap-save partitioning and (2) overlap-add partitioning. Each of these algorithms is summarized below (Burrus and Parks 1985, Jenkins 2002).

1.7.3.1 Overlap-Save Processing

In this algorithm, N_{DFT} is chosen to be some convenient value with $N_{\text{DFT}} > N$. The signal, $s[n]$, is partitioned into blocks which are of length N_{DFT} and which overlap by $N-1$ data points. Hence, the kth block is $s_k[n] = s[n + k(N_{\text{DFT}} - N + 1)], n = 0, \ldots, N_{\text{DFT}} - 1$. The filter impulse response $h[n]$ is augmented with $N_{\text{DFT}} - N$ zeros to produce

$$h_{\text{pad}}[n] = \begin{bmatrix} h[n], & n = 0, \ldots, N-1 \\ 0, & n = N, \ldots, N_{\text{DFT}} - 1 \end{bmatrix}. \tag{1.21}$$

The DFT is then used to obtain $Y_{\text{pad}}[n] = \text{DFT}\{h_{\text{pad}}[n]\} \cdot \text{DFT}\{s_k[n]\}$, and $y_{\text{pad}}[n] = \text{IDFT}\{Y_{\text{pad}}[n]\}$. From the $y_{\text{pad}}[n]$ array the values that correctly correspond to the linear convolution are saved; values that are erroneous due to wraparound error caused by the circular convolution of the DFT are discarded. The kth block of the filtered output is obtained by

$$y_k[n] = \begin{bmatrix} y_{\text{pad}}[n], & n = 0, \ldots, N-1 \\ 0, & n = N, \ldots, N_{\text{DFT}} - 1 \end{bmatrix}. \tag{1.22}$$

For the overlap-save algorithm, each time a block is processed there are $N_{\text{DFT}} - N + 1$ points saved and $N-1$ points discarded. Each block moves forward by $N_{\text{DFT}} - N + 1$ data points and overlaps the previous block by $N-1$ points.

1.7.3.2 Overlap-Add Processing

This algorithm is similar to the previous one except that the kth input block is defined to be

$$s_k[n] = \begin{bmatrix} s[n], & n = 0, \ldots, L-1 \\ 0, & n = L, \ldots, N_{\text{DFT}} - 1 \end{bmatrix}, \tag{1.23}$$

where $L = N_{\text{DFT}} - N + 1$. The filter function $h_{\text{pad}}[n]$ is augmented with zeros, as before, to create $h_{\text{pad}}[n]$, and the DFT processing is executed as before. In each block $y_{\text{pad}}[n]$ that is obtained at the output, the first $N-1$ points are erroneous, the last $N-1$ points are erroneous, and the middle $N_{\text{DFT}} - 2(N-1)$ points correctly correspond to the linear convolution. However, if the last $N-1$ points from block k are overlapped with the first $N-1$ points from block $k+1$ and added pairwise, correct results corresponding

to linear convolution are obtained from these positions, too. Hence, after this addition the number of correct points produced per block is $N_{\mathrm{DFT}} - N + 1$, which is the same as that for the overlap-save algorithm. The overlap-add algorithm requires approximately the same amount of computation as the overlap-save algorithm, although the addition of the overlapping portions of blocks is extra. This feature, together with the extra delay of waiting for the next block to be finished before the previous one is complete, has resulted in more popularity for the overlap-save algorithm in practical applications.

Block filtering algorithms make it possible to efficiently filter continual data streams in real time because the FFT algorithm can be used to implement the DFT, thereby minimizing the total computation time and permits reasonably high overall data rates. However, block filtering generates data in bursts, i.e., there is a delay during which no filtered data appears, and then suddenly an entire block is generated. In real-time systems, buffering must be used. The block algorithms are particularly effective for filtering very long sequences of data that are pre-recorded on magnetic tape or disk.

1.7.4 Fourier Domain Adaptive Filtering

A transform domain adaptive filter (TDAF) is a generalization of the well-known least mean square (LMS) adaptive filter in which the input signal is passed through a linear transformation in order to decompose it into a set of orthogonal components and to optimize the adaptive step size for each component and thereby maximize the learning rate of the adaptive filter (Jenkins et al. 1996). The LMS algorithm is an approximation to the steepest descent optimization strategy. For a length N FIR filter with the input expressed as a column vector $\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^{\mathrm{T}}$, the filter output $y(n)$ is expressed as

$$y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n),$$

where

$\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]^{\mathrm{T}}$ is the time varying vector of filter coefficients (tap weights) and superscript "T" denotes the vector transpose

The output error is formed as the difference between the filter output and a training signal $d(n)$, i.e. $e(n) = d(n) - y(n)$. Strategies for obtaining an appropriate $d(n)$ vary from one application to another. In many cases the availability of a suitable training signal determines whether an adaptive filtering solution will be successful in a particular application. The ideal cost function is defined by the mean squared error (MSE) criterion, $E\{|e(n)|^2\}$. The LMS algorithm is derived by approximating the ideal cost function by the instantaneous squared error, resulting in $J_{\mathrm{LMS}}(n) = |e(n)|^2$. While the LMS seems to make a rather crude approximation at the very beginning, the approximation results in an unbiased estimator. In many applications the LMS algorithm is quite robust and is able to converge rapidly to a small neighborhood of the Wiener solution.

When a steepest descent optimization strategy is combined with a gradient approximation formed using the LMS cost function $J_{\mathrm{LMS}}(n) = |e(n)|^2$, the conventional LMS adaptive algorithm results

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu e(n)\mathbf{x}(n), \\ e(n) &= d(n) - y(n), \end{aligned} \tag{1.24}$$

and

$$y(n) = \mathbf{x}(n)^{\mathrm{T}}\mathbf{w}(n).$$

The convergence behavior of the LMS algorithm, as applied to a direct form FIR filter structure, is controlled by the autocorrelation matrix $\mathbf{R}_x$ of the input process, where

$$\mathbf{R}_x \equiv E[\mathbf{x}^*(n)\mathbf{x}^{\mathrm{T}}(n)]. \tag{1.25}$$

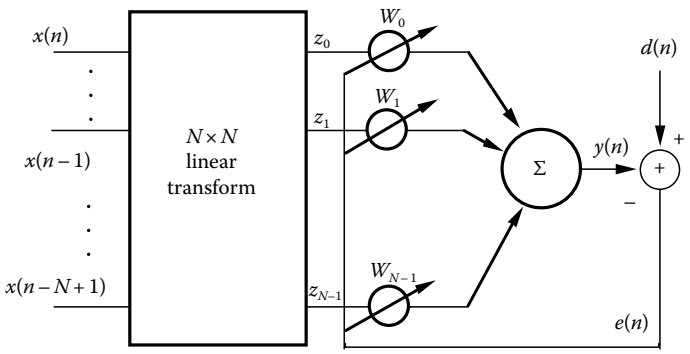

FIGURE 1.14 TDAF structure. (From Jenkins, W. K., Marshall, D. F., Kreidle, J. R., and Murphy, J. J., *IEEE Trans. Circuits Sys.*, 36(4), 474, 1989. With permission.)

The autocorrelation matrix $\mathbf{R}_x$ is usually positive definite, which is one of the conditions necessary to guarantee convergence to the Wiener solution. Another necessary condition for convergence is $0 < m < 1/l_{\max}$, where $l_{\max}$ is the largest eigenvalue of $\mathbf{R}_x$. It is well established that the convergence of this algorithm is directly related to the eigenvalue spread of $\mathbf{R}_x$. The eigenvalue spread is measured by the condition number of $\mathbf{R}_x$, defined as $\dot{k} = l_{\max}/l_{\min}$, where $l_{\min}$ is the minimum eigenvalue of $\mathbf{R}_x$. Ideal conditioning occurs when $\dot{k} = 1$ (white noise); as this ratio increases, slower convergence results. The eigenvalue spread (condition number) depends on the spectral distribution of the input signal, and is related to the maximum and minimum values of the input power spectrum. From this line of reasoning it becomes clear that white noise is the ideal input signal for rapidly training an LMS adaptive filter. The adaptive process is slower and requires more computation for input signals that are colored.

The TDAF structure is shown in Figure 1.14. The input $x(n)$ and the desired signal $d(n)$ are assumed to be zero mean and jointly stationary. The input to the filter is a vector of N current and past input samples, defined in the previous section and denoted as $\mathbf{x}(n)$. This vector is processed by a unitary transform, such as the DFT. Once the filter order N is fixed, the transform is simply an $N \times N$ matrix $\mathbf{T}$, which is in general complex, with orthonormal rows. The transformed outputs form a vector $\mathbf{v}(n)$ which is given by

$$\mathbf{z}(n) = [v_0(n), v_1(n), \ldots, v_{N-1}(n)]^{\mathrm{T}} = \mathbf{T}\mathbf{x}(n).$$

With an adaptive tap vector defined as $\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]^{\mathrm{T}}$, the filter output is given by

$$y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{v}(n) = \mathbf{W}^{\mathrm{T}}(n)\mathbf{T}\mathbf{x}(n). \tag{1.26}$$

The instantaneous output error is then formed and used to update the adaptive filter taps using a modified form of the LMS algorithm (Jenkins et al. 1996):

$$\begin{aligned} \mathbf{W}(n+1) &= \mathbf{W}(n) + \mu e(n)\Lambda^{-2}\mathbf{v}^*(n) \\ \Lambda^2 &\equiv \mathrm{diag}\left[\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2\right], \end{aligned} \tag{1.27}$$

where $\sigma_i^2 = E[|v_i(n)|^2]$.

The power estimates σ_i^2 can be developed on-line by computing an exponentially weighted average of past samples according to

$$\sigma_i^2(n) = \alpha\sigma_\iota^2(n-1) + |v_i(n)|^2, \quad 0 < a < 1. \tag{1.28}$$

If σ_i^2 becomes too small due to an insufficient amount of energy in the *i*th channel, the update mechanism becomes ill-conditioned due to a very large effective step size. In some cases the process will become unstable and register overflow will cause the adaptation to catastrophically fail. So the algorithm given by Equation 1.27 should have the update mechanism disabled for the *i*th orthogonal channel if σ_i^2 falls below a critical threshold.

The motivation for using the TDAF adaptive system instead of a simpler LMS based system is to achieve rapid convergence of the filters coefficients when the input signal is not white, while maintaining a reasonably low computational complexity requirement. The optimal decorrelating transform is composed of the orthonormal eigenvectors of the input autocorrelation matrix, and is known as the Karhunen–Loéve transform (KLT). The KLT is signal dependent and usually cannot be easily computed in real time. Throughout the literature the DFT, discrete cosine transform (DCT), and WHT have received considerable attention as possible candidates for use in TDAF.

Figure 1.15 shows learning characteristics for computer generated TDAF examples using six different orthogonal transforms to decorrelate the input signal. The examples presented are for system identification experiments, where the desired signal was derived by passing the input through an 8-tap FIR filter that is the "unknown system" to be identified. The filter input was generated by filtering white pseudo-noise with a 32-tap linear phase FIR coloring filter to produce an input autocorrelation matrix with a condition number (eigenvalue ratio) of 681. Examples were then produced using the DFT, DCT, WHT, discrete Hartley transform (DHT), and a specially designed computationally efficient PO2 transform. The condition numbers that result from transform processing with each of these transforms are also shown in Figure 1.15. Note that all of the transforms used in this example are able to reduce the input condition number and greatly improve convergence rates, although some transforms are seen to be more effective than others for the coloring chosen for these examples.

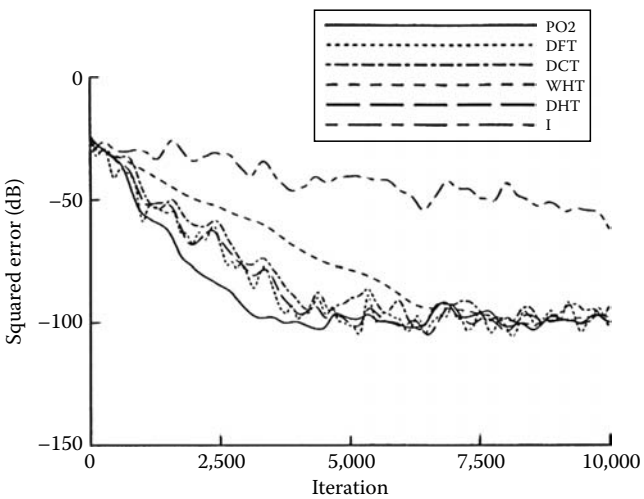

FIGURE 1.15 Comparison of (smoothed) learning curves for five different transforms operating on a colored noise input signal with condition number 681 fault in any of the coefficients. When *R* redundant coefficients are added as many as *R* coefficients can fail to adjust without any adverse effect on the filter's ability to achieve the minimum MSE condition. (From Jenkins, W. K., Marshall, D. F., Kreidle, J. R., and Murphy, J. J., *IEEE Trans. Circuits Sys.*, 36(4), 474, 1989. With permission.)

Transform	Effective Input Correlation Matrix Eigenvalue Ratio
Identity (I)	681
DFT	210
DCT	200
WHT	216
DHT	218
PO2 transform	128

1.7.5 Adaptive Fault Tolerance via Fourier Domain Adaptive Filtering

Adaptive systems adjust their parameters to minimize a specified error criterion under normal operating conditions. Fixed errors or Hardware faults would prevent the system to minimize the error criterion, but at the same time the system will adapt the parameters such that the best possible solution is reached. In adaptive fault tolerance the inherent learning ability of the adaptive system is used to compensate for failure of the adaptive coefficients. This mechanism can be used with specially designed structures whose redundant coefficients have the ability to compensate for the adjustment failures of other coefficients [Jenkins et al. 1996].

The FFT-based transform domain fault tolerant adaptive filter (FTAF) is described by the following equations:

$$\begin{aligned} \mathbf{x}[n] &= [\mathbf{x}_{\text{in}}[n], 0\ 0 \cdots 0] \\ \mathbf{x}_{\text{T}}[n] &= \mathbf{T}\mathbf{x}[n] \\ y[n] &= \mathbf{w}_{\text{T}}^{t}[n]\mathbf{x}_{\text{T}}[n] \\ e[n] &= y[n] - d[n], \end{aligned} \tag{1.29}$$

where

$\mathbf{x}_{\text{in}}[n] = [x[n], x[n-1], \ldots, x[n-N+1]]$ is the vector of the current input and $N-1$ past inputs samples
$\mathbf{x}[n]$ is $\mathbf{x}_{\text{in}}[n]$ zero-padded with R zeros
$\mathbf{T}$ is the $M \times M$ DFT matrix where $M = N + R$
$\mathbf{w}_{\text{T}}[n]$ is the vector of M adaptive coefficients in the transform domain
$d[n]$ is the desired response
$e[n]$ is the output error

The FFT-based transform domain FTAF is similar to a standard TDAF except that the input data vector is zero-padded with R zeros before it is multiplied by the transform matrix. Since the input data vector is zero padded the transform domain FTAF maintains a length N impulse response and has R redundant coefficients in the transform domain. When used with the zero padding strategy described above, this structure possesses a property called full fault tolerance, where each redundant coefficient is sufficient to compensate for a single "stuck at" fault condition in any of the coefficients. When R redundant coefficients are added as many as R coefficients can fail without any adverse effect on the filter's ability to achieve the minimum MSE condition.

An example of a transform domain FTAF with one redundant filter tap ($R = 1$) is demonstrated below for the identification of a 64-tap FIR lowpass "unknown" system. The training signal is Gaussian white

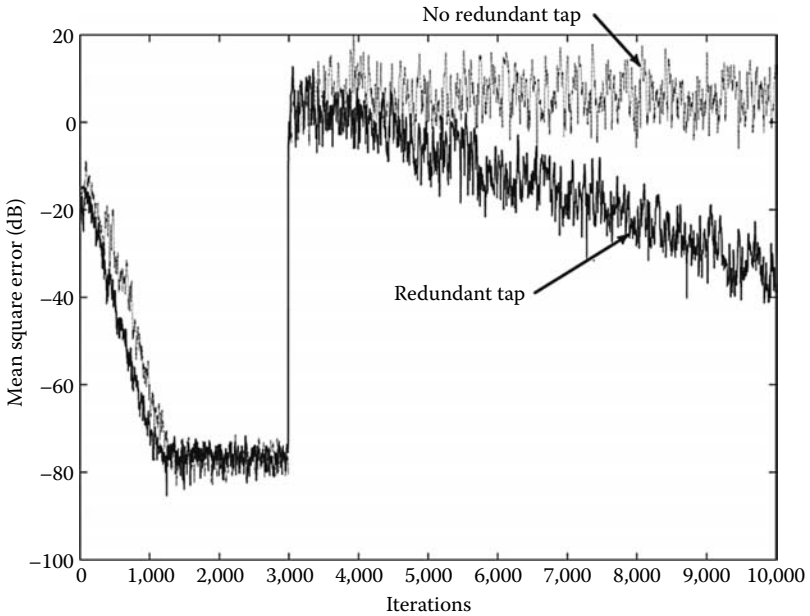

FIGURE 1.16 Learning curve demonstrating post-fault behavior both with and without a redundant tap.

noise with a unit variance and a noise floor of −60 dB. A fixed fault is introduced at iteration 3000 by setting an arbitrary filter coefficient to a random fixed value. Simulated learning curves are shown in Figure 1.16 both demonstrated that the redundant tap allows the filter to re-converge after the occurrence of the fault, although the post-fault convergence rate slowed somewhat due to an increased condition number of the post-fault autocorrelation matrix [Jenkins et al. 1996].

1.8 Summary

Numerous Fourier transform concepts have been presented for both CT and DT signals and systems. Emphasis was placed on illustrating how various forms of the Fourier transform relate to one another, and how they are all derived from more general complex transforms, the complex Fourier (or bilateral Laplace) transform for CT, and the bilateral z-transform for DT. It was shown that many of these transforms have similar properties that are inherited from their parent forms, and that there is a parallel hierarchy among Fourier transform concepts in the CT and DT domains. Both CT and DT sampling models were introduced as a means of representing sampled signals in these two different domains and it was shown that the models are equivalent by virtue of having the same Fourier spectra when transformed into the Fourier domain with the appropriate Fourier transform. It was shown how Fourier analysis properly characterizes the relationship between the spectra of a CT signal and its DT counterpart obtained by sampling, and the classical reconstruction formula was obtained as a result of this analysis. Finally, the DFT, the backbone for much of modern DSP, was obtained from more classical forms of the Fourier transform by simultaneously discretizing the time and frequency domains. The DFT, together with the remarkable computational efficiency provided by the FFT algorithm, has contributed to the resounding success that engineers and scientists have had in applying DSP to many practical scientific problems.

References

Blahut, R. E., *Fast Algorithms for Digital Signal Processing*, Reading, MA: Addison-Wesley Publishing Co., 1985.

Bracewell, R. N., *The Fourier Transform*, 2nd edition, New York: McGraw-Hill, 1986.

Brigham, E. O., *The Fast Fourier Transform*, Englewood Cliffs, NJ: Prentice-Hall, 1974.

Burrus, C. S. and Parks, T. W., *DFT/FFT and Convolution Algorithms*, New York: John Wiley and Sons, 1985.

Jenkins, W. K., Discrete-time signal processing, in *Reference Data for Engineers: Radio, Electronics, Computers, and Communications*, Wendy M. Middleton (editor-in-chief), 9th edition, Carmel, MA: Newnes (Butterworth-Heinemann), 2002, Chapter 28.

Jenkins, W. K. and Desai, M. D., The discrete-frequency Fourier transform, *IEEE Transactions on Circuits and Systems*, CAS-33(7), 732–734, July 1986.

Jenkins, W. K. et al., *Advanced Concepts in Adaptive Signal Processing*, Boston, MA: Kluwer Academic Publishers, 1996.

Oppenheim, A. V. and Schafer, R. W., *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1975.

Oppenheim, A. V. and Schafer, R. W., *Discrete-Time Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.

Oppenheim, A. V., Willsky, A. S., and Young, I.T., *Signals and Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

VanValkenburg, M. E., *Network Analysis*, 3rd edition, Englewood Cliffs, NJ: Prentice-Hall, 1974.

2

Ordinary Linear Differential and Difference Equations

B.P. Lathi
California State University

2.1 Differential Equations

A function containing variables and their derivatives is called a differential expression, and an equation involving differential expressions is called a differential equation. A differential equation is an ordinary differential equation if it contains only one independent variable; it is a partial differential equation if it contains more than one independent variable. We shall deal here only with ordinary differential equations.

In the mathematical texts, the independent variable is generally x, which can be anything such as time, distance, velocity, pressure, and so on. In most of the applications in control systems, the independent variable is time. For this reason we shall use here independent variable t for time, although it can stand for any other variable as well.

The following equation

$$\left(\frac{\mathrm{d}^2y}{\mathrm{d}t^2}\right)^4 + 3\frac{\mathrm{d}y}{\mathrm{d}t} + 5y^2(t) = \sin t$$

is an ordinary differential equation of second order because the highest derivative is of the second order. An nth-order differential equation is linear if it is of the form

$$a_n(t)\frac{\mathrm{d}^n y}{\mathrm{d}t^n} + a_{n-1}(t)\frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}} + \cdots + a_1(t)\frac{\mathrm{d}y}{\mathrm{d}t} + a_0(t)y(t) = r(t) \tag{2.1}$$

where the coefficients $a_i(t)$ are not functions of $y(t)$. If these coefficients (a_i) are constants, the equation is linear with constant coefficients. Many engineering (as well as nonengineering) systems can be modeled by these equations. Systems modeled by these equations are known as linear time-invariant (LTI)

systems. In this chapter we shall deal exclusively with linear differential equations with constant coefficients. Certain other forms of differential equations are dealt with elsewhere in this book.

2.1.1 Role of Auxiliary Conditions in Solution of Differential Equations

We now show that a differential equation does not, in general, have a unique solution unless some additional constraints (or conditions) on the solution are known. This fact should not come as a surprise. A function $y(t)$ has a unique derivative dy/dt, but for a given derivative dy/dt there are infinite possible functions $y(t)$. If we are given dy/dt, it is impossible to determine $y(t)$ uniquely unless an additional piece of information about $y(t)$ is given. For example, the solution of a differential equation

$$\frac{dy}{dt} = 2 \tag{2.2}$$

obtained by integrating both sides of the equation is

$$y(t) = 2t + c \tag{2.3}$$

for any value of c. Equation 2.2 specifies a function whose slope is 2 for all t. Any straight line with a slope of 2 satisfies this equation. Clearly the solution is not unique, but if we place an additional constraint on the solution $y(t)$, then we specify a unique solution.

For example, suppose we require that $y(0) = 5$; then out of all the possible solutions available, only one function has a slope of 2 and an intercept with the vertical axis at 5. By setting $t = 0$ in Equation 2.3 and substituting $y(0) = 5$ in the same equation, we obtain $y(0) = 5 = c$ and

$$y(t) = 2t + 5$$

which is the unique solution satisfying both Equation 2.2 and the constraint $y(0) = 5$.

In conclusion, differentiation is an irreversible operation during which certain information is lost. To reverse this operation, one piece of information about $y(t)$ must be provided to restore the original $y(t)$. Using a similar argument, we can show that, given d^2y/dt^2, we can determine $y(t)$ uniquely only if two additional pieces of information (constraints) about $y(t)$ are given. In general, to determine $y(t)$ uniquely from its nth derivative, we need n additional pieces of information (constraints) about $y(t)$. These constraints are also called auxiliary conditions. When these conditions are given at $t = 0$, they are called initial conditions.

We discuss here two systematic procedures for solving linear differential equations of the form in Equation 2.1. The first method is the classical method, which is relatively simple, but restricted to a certain class of inputs. The second method (the convolution method) is general and is applicable to all types of inputs. A third method (Laplace transform) is discussed elsewhere in this book. Both the methods discussed here are classified as time-domain methods because with these methods we are able to solve the above equation directly, using t as the independent variable. The method of Laplace transform (also known as the frequency-domain method), on the other hand, requires transformation of variable t into a frequency variable s.

In engineering applications, the form of linear differential equation that occurs most commonly is given by

$$\begin{aligned}\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_1\frac{dy}{dt} + a_0 y(t) \\ = b_m\frac{d^m f}{dt^m} + b_{m-1}\frac{d^{m-1}f}{dt^{m-1}} + \cdots + b_1\frac{df}{dt} + b_0 f(t)\end{aligned} \tag{2.4a}$$

where all the coefficients a_i and b_i are constants. Using operational notation D to represent d/dt, this equation can be expressed as

$$\begin{aligned}&\left(D^n + a_{n-1}D^{n-1} + \cdots + a_1 D + a_0\right)y(t)\\ &\quad = \left(b_m D^m + b_{m-1}D^{m-1} + \cdots + b_1 D + b_0\right)f(t)\end{aligned} \tag{2.4b}$$

or

$$Q(D)y(t) = P(D)f(t) \tag{2.4c}$$

where the polynomials $Q(D)$ and $P(D)$, respectively, are

$$\begin{aligned}Q(D) &= D^n + a_{n-1}D^{n-1} + \cdots + a_1 D + a_0\\ P(D) &= b_m D^m + b_{m-1}D^{m-1} + \cdots + b_1 D + b_0\end{aligned}$$

Observe that this equation is of the form of Equation 2.1, where $r(t)$ is in the form of a linear combination of $f(t)$ and its derivatives. In this equation, $y(t)$ represents an output variable, and $f(t)$ represents an input variable of an LTI system. Theoretically, the powers m and n in the above equations can take on any value. Practical noise considerations, however, require [1] $m \leq n$.

2.1.2 Classical Solution

When $f(t) \equiv 0$, Equation 2.4 is known as the homogeneous (or complementary) equation. We shall first solve the homogeneous equation. Let the solution of the homogeneous equation be $y_c(t)$, that is,

$$Q(D)y_c(t) = 0$$

or

$$\left(D^n + a_{n-1}D^{n-1} + \cdots + a_1 D + a_0\right)y_c(t) = 0$$

We first show that if $y_p(t)$ is the solution of Equation 2.4, then $y_c(t) + y_P(t)$ is also its solution. This follows from the fact that

$$Q(D)y_c(t) = 0$$

If $y_P(t)$ is the solution of Equation 2.4, then

$$Q(D)y_P(t) = P(D)f(t)$$

Addition of these two equations yields

$$Q(D)[y_c(t) + y_P(t)] = P(D)f(t)$$

Thus, $y_c(t) + y_P(t)$ satisfies Equation 2.4 and therefore is the general solution of Equation 2.4. We call $y_c(t)$ the complementary solution and $y_P(t)$ the particular solution. In system analysis parlance, these components are called the natural response and the forced response, respectively.

2.1.2.1 Complementary Solution (the Natural Response)

The complementary solution $y_c(t)$ is the solution of

$$Q(D)y_c(t) = 0 \tag{2.5a}$$

or

$$\left(D^n + a_{n-1}D^{n-1} + \cdots + a_1 D + a_0\right)y_c(t) = 0 \tag{2.5b}$$

A solution to this equation can be found in a systematic and formal way. However, we will take a short cut by using heuristic reasoning. Equation 2.5b shows that a linear combination of $y_c(t)$ and its n successive derivatives is zero, not at some values of t, but for all t. This is possible if and only if $y_c(t)$ and all its n successive derivatives are of the same form. Otherwise their sum can never add to zero for all values of t. We know that only an exponential function $e^{\lambda t}$ has this property. So let us assume that

$$y_c(t) = ce^{\lambda t}$$

is a solution to Equation 2.5b. Now

$$\begin{aligned} Dy_c(t) &= \frac{dy_c}{dt} = c\lambda e^{\lambda t} \\ D^2y_c(t) &= \frac{d^2y_c}{dt^2} = c\lambda^2 e^{\lambda t} \\ &\vdots \\ D^ny_c(t) &= \frac{d^ny_c}{dt^n} = c\lambda^n e^{\lambda} t \end{aligned}$$

Substituting these results in Equation 2.5b, we obtain

$$c\left(\lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_1\lambda + a_0\right)e^{\lambda t} = 0$$

For a nontrivial solution of this equation,

$$\lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_1\lambda + a_0 = 0 \tag{2.6a}$$

This result means that $ce^{\lambda t}$ is indeed a solution of Equation 2.5 provided that λ satisfies Equation 2.6a. Note that the polynomial in Equation 2.6a is identical to the polynomial $Q(D)$ in Equation 2.5b, with λ replacing D. Therefore, Equation 2.6a can be expressed as

$$Q(\lambda) = 0 \tag{2.6b}$$

When $Q(\lambda)$ is expressed in factorized form, Equation 2.6b can be represented as

$$Q(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n) = 0 \tag{2.6c}$$

Clearly λ has n solutions: $\lambda_1, \lambda_2, \ldots, \lambda_n$. Consequently, Equation 2.5 has n possible solutions: $c_1e^{\lambda_1 t}$, $c_2e^{\lambda_2 t}, \ldots, c_ne^{\lambda_n t}$, with $c_1, c_2, \ldots, c_n$ as arbitrary constants. We can readily show that a general solution is given by the sum of these n solutions,* so that

$$y_c(t) = c_1e^{\lambda_1 t} + c_2e^{\lambda_2 t} + \cdots + c_ne^{\lambda_n t} \tag{2.7}$$

where $c_1, c_2, \ldots, c_n$ are arbitrary constants determined by n constraints (the auxiliary conditions) on the solution.

The polynomial $Q(\lambda)$ is known as the characteristic polynomial. The equation

$$Q(\lambda) = 0 \tag{2.8}$$

is called the characteristic or auxiliary equation. From Equation 2.6c, it is clear that $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the roots of the characteristic equation; consequently, they are called the characteristic roots. The terms characteristic values, eigenvalues, and natural frequencies are also used for characteristic roots.† The expotentials $e^{\lambda_i t} (i = 1, 2, \ldots, n)$ in the complementary solution are the characteristic modes (also known as modes or natural modes). There is a characteristic mode for each characteristic root, and the complementary solution is a linear combination of the characteristic modes.

2.1.2.2 Repeated Roots

The solution of Equation 2.5 as given in Equation 2.7 assumes that the characteristic roots $\lambda_1, \lambda_2, \ldots, \lambda_n$ are distinct. If there are repeated roots (same root occurring more than once), the form of the solution is modified slightly. By direct substitution we can show that the solution of the equation

$$(D - \lambda)^2 y_c(t) = 0$$

is given by

$$y_c(t) = (c_1 + c_2 t)e^{\lambda t}$$

In this case the root λ repeats twice. Observe that the characteristic modes in this case are $e^{\lambda t}$ and $te^{\lambda t}$. Continuing this pattern, we can show that for the differential equation

$$(D - \lambda)^r y_c(t) = 0 \tag{2.9}$$

the characteristic modes are $e^{\lambda t}$, $te^{\lambda t}$, $t^2e^{\lambda t}$, …, $t^{r-1}e^{\lambda t}$, and the solution is

$$y_c(t) = \left(c_1 + c_2 t + \cdots + c_r t^{r-1}\right)e^{\lambda t} \tag{2.10}$$

---

* To prove this fact, assume that $y_1(t), y_2(t), \ldots, y_n(t)$ are all solutions of Equation 2.5. Then

$$\begin{aligned} Q(D)y_1(t) &= 0 \\ Q(D)y_2(t) &= 0 \\ &\vdots \\ Q(D)y_n(t) &= 0 \end{aligned}$$

Multiplying these equations by $c_1, c_2, \ldots, c_n$, respectively, and adding them together yields

$$Q(D)[c_1y_1(t) + c_2y_n(t)] = 0$$

This result shows that $c_1y_1(t) + c_2y_2(t) + \cdots + c_ny_n(t)$ is also a solution of the homogeneous equation (Equation 2.5).

† The term *eigenvalue* is German for characteristic value.

Consequently, for a characteristic polynomial

$$Q(\lambda) = (\lambda - \lambda_1)^r(\lambda - \lambda_{r+1})\dots(\lambda - \lambda_n)$$

the characteristic modes are $e^{\lambda_1 t}$, $te^{\lambda_1 t}, \dots, t^{r-1}e^{\lambda t}$, $e^{\lambda_{r+1} t}, \dots, e^{\lambda_n t}$ and the complementary solution is

$$y_c(t) = \left(c_1 + c_2 t + \cdots + c_r t^{r-1}\right)e^{\lambda t} + c_{r+1}e^{\lambda_{r+1} t} + \cdots + c_n e^{\lambda_n t}$$

2.1.2.3 Particular Solution (the Forced Response): Method of Undetermined Coefficients

The particular solution $y_p(t)$ is the solution of

$$Q(D)y_p(t) = P(D)f(t) \tag{2.11}$$

It is a relatively simple task to determine $y_p(t)$ when the input $f(t)$ is such that it yields only a finite number of independent derivatives. Inputs having the form $e^{\zeta t}$ or t^r fall into this category. For example, $e^{\zeta t}$ has only one independent derivative; the repeated differentiation of $e^{\zeta t}$ yields the same form, that is, $e^{\zeta t}$. Similarly, the repeated differentiation of t^r yields only r independent derivatives. The particular solution to such an input can be expressed as a linear combination of the input and its independent derivatives. Consider, for example, the input $f(t) = at^2 + bt + c$. The successive derivatives of this input are $2at + b$ and $2a$. In this case, the input has only two independent derivatives. Therefore the particular solution can be assumed to be a linear combination of $f(t)$ and its two derivatives. The suitable form for $y_p(t)$ in this case is therefore

$$y_p(t) = \beta_2 t^2 + \beta_1 t + \beta_0$$

The undetermined coefficients β_0, β_1, and β_2 are determined by substituting this expression for $y_p(t)$ in Equation 2.11 and then equating coefficients of similar terms on both sides of the resulting expression.

Although this method can be used only for inputs with a finite number of derivatives, this class of inputs includes a wide variety of the most commonly encountered signals in practice. Table 2.1 shows a variety of such inputs and the form of the particular solution corresponding to each input. We shall demonstrate this procedure with an example.

Note: By definition, $y_p(t)$ cannot have any characteristic mode terms. If any term $p(t)$ shown in the right-hand column for the particular solution is also a characteristic mode, the correct form of the forced response must be modified to $t^i p(t)$, where i is the smallest possible integer that can be used and still can prevent $t^i p(t)$ from having characteristic mode term. For example, when the input is $e^{\zeta t}$, the forced response (right-hand column) has the form $\beta e^{\zeta t}$. But if $e^{\zeta t}$ happens to be a characteristic mode, the correct form of the particular solution is $\beta te^{\zeta t}$ (see Pair 2). If $te^{\zeta t}$ also happens to be characteristic mode, the correct form of the particular solution is $\beta t^2 e^{\zeta t}$, and so on.

TABLE 2.1 Inputs and Responses for Commonly Encountered Signals

No.	Input $f(t)$	Forced Response
1	$e^{\zeta t}$ $\zeta \neq \lambda_i (i = 1, 2, \dots, n)$	$\beta e^{\zeta t}$
2	$e^{\zeta t}$ $\zeta \neq \lambda_i$	$\beta te^{\zeta t}$
3	k (a constant)	β (a constant)
4	$\cos(\omega t + \theta)$	$(\beta \cos(\omega t + \varphi)$
5	$(t^r + \alpha_{r-1}t^{r-1} + \cdots + \alpha_1 t + \alpha_0)e^{\zeta t}$	$(\beta_r t^r + \beta_{r-1}t^{r-1} + \cdots + \beta_1 t + \beta_0)e^{\zeta t}$

Example 2.1

Solve the differential equation

$$(D^2 + 3D + 2)y(t) = Df(t) \tag{2.12}$$

if the input

$$f(t) = t^2 + 5t + 3$$

and the initial conditions are $y(0^+) = 2$ and $\dot{y}(0^+) = 3$.

The characteristic polynomial is

$$\lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2)$$

Therefore the characteristic modes are e^{-t} and e^{-2t}. The complementary solution is a linear combination of these modes, so that

$$y_c(t) = c_1 e^{-t} + c_2 e^{-2t} \quad t \geq 0$$

Here the arbitrary constants c_1 and c_2 must be determined from the given initial conditions.

The particular solution to the input $t^2 + 5t + 3$ is found from Table 2.1 (Pair 5 with $\zeta = 0$) to be

$$y_p(t) = \beta_2 t^2 + \beta_1 t + \beta_0$$

Moreover, $y_p(t)$ satisfies Equation 2.11, that is,

$$(D^2 + 3D + 2)y_p(t) = Df(t) \tag{2.13}$$

Now

$$Dy_p(t) = \frac{d}{dt}\left(\beta_2 t^2 + \beta_1 t + \beta_0\right) = 2\beta_2 t + \beta_1$$

$$D^2 y_p(t) = \frac{d^2}{dt^2}\left(\beta_2 t^2 + \beta_1 t + \beta_0\right) = 2\beta_2$$

and

$$Df(t) = \frac{d}{dt}[t^2 + 5t + 3] = 2t + 5$$

Substituting these results in Equation 2.13 yields

$$2\beta_2 + 3(2\beta_2 t + \beta_1) + 2(\beta_2 t^2 + \beta_1 t + \beta_0) = 2t + 5$$

or

$$2\beta_2 t^2 + (2\beta_1 + 6\beta_2)t + (2\beta_0 + 3\beta_1 + 2\beta_2) = 2t + 5$$

Equating coefficients of similar powers on both sides of this expression yields

$$\begin{aligned} 2\beta_2 &= 0 \\ 2\beta_1 + 6\beta_2 &= 2 \\ 2\beta_0 + 3\beta_1 + 2\beta_2 &= 5 \end{aligned}$$

Solving these three equations for their unknowns, we obtain $\beta_0 = 1$, $\beta_1 = 1$, and $\beta_2 = 0$. Therefore,

$$y_p(t) = t + 1 \quad t > 0$$

The total solution $y(t)$ is the sum of the complementary and particular solutions. Therefore,

$$\begin{aligned} y(t) &= y_c(t) + y_p(t) \\ &= c_1 e^{-t} + c_2 e^{-2t} + t + 1 \quad t > 0 \end{aligned}$$

so that

$$\dot{y}(t) = -c_1 e^{-t} - 2c_2 e^{-2t} + 1$$

Setting $t = 0$ and substituting the given initial conditions $y(0) = 2$ and $\dot{y}(0) = 3$ in these equations, we have

$$\begin{aligned} 2 &= c_1 + c_2 + 1 \\ 3 &= -c_1 - 2c_2 + 1 \end{aligned}$$

The solution to these two simultaneous equations is $c_1 = 4$ and $c_2 = -3$. Therefore,

$$y(t) = 4e^{-t} - 3e^{-2t} + t + 1 \quad t \geq 0$$

2.1.2.4 The Exponential Input $e^{\zeta t}$

The exponential signal is the most important signal in the study of LTI systems. Interestingly, the particular solution for an exponential input signal turns out to be very simple. From Table 2.1 we see that the particular solution for the input $e^{\zeta t}$ has the form $\beta e^{\zeta t}$. We now show that $\beta = Q(\zeta)/P(\zeta)$.* To determine the constant β, we substitute $y_p(t) = \beta e^{\zeta t}$ in Equation 2.11, which gives us

$$Q(D)\left[\beta e^{\zeta t}\right] = P(D)e^{\zeta t} \tag{2.14a}$$

Now observe that

$$\begin{aligned} De^{\zeta t} &= \frac{d}{dt}(e^{\zeta t}) = \zeta e^{\zeta t} \\ D^2 e^{\zeta t} &= \frac{d^2}{dt^2}(e^{\zeta t}) = \zeta^2 e^{\zeta t} \\ &\vdots \\ D^r e^{\zeta t} &= \zeta^r e^{\zeta t} \end{aligned}$$

* This is true only if ζ is not a characteristic root.

Consequently,

$$Q(D)e^{\zeta t} = Q(\zeta)e^{\zeta t} \quad \text{and} \quad P(D)e^{\zeta t} = P(\zeta)e^{\zeta t}$$

Therefore, Equation 2.14a becomes

$$\beta Q(\zeta)e^{\zeta t} = P(\zeta)e^{\zeta t} \tag{2.14b}$$

and

$$\beta = \frac{P(\zeta)}{Q(\zeta)}$$

Thus, for the input $f(t) = e^{\zeta t}$, the particular solution is given by

$$y_p(t) = H(\zeta)e^{\zeta t} \quad t > 0 \tag{2.15a}$$

where

$$H(\zeta) = \frac{P(\zeta)}{Q(\zeta)} \tag{2.15b}$$

This is an interesting and significant result. It states that for an exponential input $e^{\zeta t}$ the particular solution $y_p(t)$ is the same exponential multiplied by $H(\zeta) = P(\zeta)/Q(\zeta)$. The total solution $y(t)$ to an exponential input $e^{\zeta t}$ is then given by

$$y(t) = \sum_{j=1}^{n} c_j e^{\lambda_j t} + H(\zeta)e^{\zeta t}$$

where the arbitrary constants $c_1, c_2, \ldots, c_n$ are determined from auxiliary conditions.

Recall that the exponential signal includes a large variety of signals, such as a constant ($\zeta = 0$), a sinusoid ($\zeta = \pm j\omega$), and an exponentially growing or decaying sinusoid ($\zeta = \sigma \pm j\omega$). Let us consider the forced response for some of these cases.

2.1.2.5 The Constant Input $f(t) = C$

Because $C = Ce^{0t}$, the constant input is a special case of the exponential input $Ce^{\zeta t}$ with $\zeta = 0$. The particular solution to this input is then given by

$$\begin{aligned} y_p(t) &= CH(\zeta)e^{\zeta t} \quad \text{with } \zeta = 0 \\ &= CH(0) \end{aligned} \tag{2.16}$$

2.1.2.6 The Complex Exponential Input $e^{j\omega t}$

Here $\zeta = j\omega$, and

$$y_p(t) = H(j\omega)e^{j\omega t} \tag{2.17}$$

2.1.2.7 The Sinusoidal Input $f(t) = \cos \omega_0 t$

We know that the particular solution for the input $e^{\pm j\omega t}$ is $H(\pm j\omega)e^{\pm j\omega t}$. Since $\cos \omega t = (e^{j\omega t} + e^{-j\omega t})/2$, the particular solution to $\cos \omega t$ is

$$y_p(t) = \frac{1}{2}\left[H(j\omega)e^{j\omega t} + H(-j\omega)e^{-j\omega t}\right]$$

Because the two terms on the right-hand side are conjugates,

$$y_p(t) = \mathrm{Re}\left[H(j\omega)e^{j\omega t}\right]$$

But

$$H(j\omega) = |H(j\omega)|e^{j\angle H(j\omega)}$$

so that

$$\begin{aligned} y_p(t) &= \mathrm{Re}\left\{|H(j\omega)|e^{j[\omega t + \angle H(j\omega)]}\right\} \\ &= |H(j\omega)|\cos[\omega t + \angle H(j\omega)] \end{aligned} \tag{2.18}$$

This result can be generalized for the input $f(t) = \cos(\omega t + \theta)$. The particular solution in this case is

$$y_p(t) = |H(j\omega)|\cos[\omega t + \theta + \angle H(j\omega)] \tag{2.19}$$

Example 2.2

Solve Equation 2.12 for the following inputs:

(a) $10e^{-3t}$ (b) 5 (c) e^{-2t} (d) $10\cos(3t + 30°)$.

The initial conditions are $y(0^+) = 2$, $\dot{y}(0^+) = 3$.

The complementary solution for this case is already found in Example 2.1 as

$$y_c(t) = c_1e^{-t} + c_2e^{-2t} \quad t \geq 0$$

For the exponential input $f(t) = e^{\zeta t}$, the particular solution, as found in Equation 2.15 is $H(\zeta)e^{\zeta t}$, where

$$H(\zeta) = \frac{P(\zeta)}{Q(\zeta)} = \frac{\zeta}{\zeta^2 + 3\zeta + 2}$$

(a) For input $f(t) = 10e^{-3t}$, $\zeta = -3$, and

$$\begin{aligned} y_p(t) &= 10H(-3)e^{-3t} \\ &= 10\left[\frac{-3}{(-3)^2 + 3(-3) + 2}\right]e^{-3t} \\ &= -15e^{-3t} \quad t > 0 \end{aligned}$$

The total solution (the sum of the complementary and particular solutions) is

$$y(t) = c_1e^{-t} + c_2e^{-2t} - 15e^{-3t} \quad t \geq 0$$

and

$$\dot{y}(t) = -c_1 e^{-t} - 2c_2 e^{-2t} + 45e^{-3t} \quad t \geq 0$$

The initial conditions are $y(0^+) = 2$ and $\dot{y}(0^+) = 3$. Setting $t = 0$ in the above equations and substituting the initial conditions yields

$$c_1 + c_2 - 15 = 2 \quad \text{and} \quad -c_1 - 2c_2 + 45 = 3$$

Solution of these equations yields $c_1 = -8$ and $c_2 = 25$. Therefore,

$$y(t) = -8e^{-t} + 25e^{-2t} - 15e^{-3t} \quad t \geq 0$$

(b) For input $f(t) = 5 = 5e^{0t}$, $\zeta = 0$, and

$$y_p(t) = 5H(0) = 0 \quad t > 0$$

The complete solution is $y(t) = y_c(t) + y_p(t) = c_1 e^{-t} + c_2 e^{-2t}$. We then substitute the initial conditions to determine c_1 and c_2 as explained in (a).

(c) Here $\zeta = -2$, which is also a characteristic root. Hence (see Pair 2, Table 2.1, or the comment at the bottom of the table),

$$y_p(t) = \beta t e^{-2t}$$

To find β, we substitute $y_p(t)$ in Equation 2.11, giving us

$$(D^2 + 3D + 2)y_p(t) = Df(t)$$

or

$$(D^2 + 3D + 2)\left[\beta t e^{-2t}\right] = De^{-2t}$$

But

$$D[\beta t e^{-2t}] = \beta(1 - 2t)e^{-2t}$$
$$D^2[\beta t e^{-2t}] = 4\beta(t - 1)e^{-2t}$$
$$De^{-2t} = -2e^{-2t}$$

Consequently,

$$\beta(4t - 4 + 3 - 6t + 2t)e^{-2t} = -2e^{-2t}$$

or

$$-\beta e^{-2t} = -2e^{-2t}$$

This means that $\beta = 2$, so that

$$y_p(t) = 2te^{-2t}$$

The complete solution is $y(t) = y_c(t) + y_p(t) = c_1 e^{-t} + c_2 e^{-2t} + 2te^{-2t}$. We then substitute the initial conditions to determine c_1 and c_2 as explained in (a).

(d) For the input $f(t) = 10\cos(3t + 30°)$, the particular solution (see Equation 2.19) is

$$y_p(t) = 10|H(j3)|\cos[3t + 30° + \angle H(j3)]$$

where

$$H(j3) = \frac{P(j3)}{Q(j3)} = \frac{j3}{(j3)^2 + 3(j3) + 2}$$
$$= \frac{j3}{-7 + j9} = \frac{27 - j21}{130} = 0.263e^{-j37.9°}$$

Therefore,

$$|H(j3)| = 0.263, \quad \angle H(j3) = -37.9°$$

and

$$y_p(t) = 10(0.263)\cos(3t + 30° - 37.9°)$$
$$= 2.63\cos(3t - 7.9°)$$

The complete solution is $y(t) = y_c(t) + y_p(t) = c_1 e^{-t} + c_2 e^{-2t} + 2.63\cos(3t - 7.9°)$. We then substitute the initial conditions to determine c_1 and c_2 as explained in (a).

2.1.3 Method of Convolution

In this method, the input $f(t)$ is expressed as a sum of impulses. The solution is then obtained as a sum of the solutions to all the impulse components. The method exploits the superposition property of the linear differential equations. From the sampling (or sifting) property of the impulse function, we have

$$f(t) = \int_0^t f(x)\delta(t - x)dx \quad t \geq 0 \tag{2.20}$$

The right-hand side expresses $f(t)$ as a sum (integral) of impulse components. Let the solution of Equation 2.4 be $y(t) = h(t)$ when $f(t) = \delta(t)$ and all the initial conditions are zero. Then use of the linearity property yields the solution of Equation 2.4 to input $f(t)$ as

$$y(t) = \int_0^t f(x)h(t - x)dx \tag{2.21}$$

For this solution to be general, we must add a complementary solution. Thus, the general solution is given by

$$y(t) = \sum_{j=1}^{n} c_j e^{\lambda_j t} + \int_0^t f(x)h(t - x)dx \tag{2.22}$$

The first term on the right-hand side consists of a linear combination of natural modes and should be appropriately modified for repeated roots. For the integral on the right-hand side, the lower limit 0 is understood to be 0^- in order to ensure that impulses, if any, in the input $f(t)$ at the origin are accounted for. The integral on the right-hand side of Equation 2.22 is well known in the literature as the convolution integral. The function $h(t)$ appearing in the integral is the solution of Equation 2.4 for the impulsive input $[f(t) = \delta(t)]$. It can be shown that [2]

$$h(t) = P(D)[y_o(t)u(t)] \tag{2.23}$$

where $y_o(t)$ is a linear combination of the characteristic modes subject to initial conditions

$$\begin{aligned} &y_o^{(n-1)}(0) = 1 \\ &y_o(0) = y_o^{(1)}(0) = \cdots = y_o^{(n-2)}(0) = 0 \end{aligned} \tag{2.24}$$

The function $u(t)$ appearing on the right-hand side of Equation 2.23 represents the unit step function, which is unity for $t \geq 0$ and is 0 for $t < 0$.

The right-hand side of Equation 2.23 is a linear combination of the derivatives of $y_o(t)u(t)$. Evaluating these derivatives is clumsy and inconvenient because of the presence of $u(t)$. The derivatives will generate an impulse and its derivatives at the origin [recall that $\frac{d}{dt}u(t) = \delta(t)$]. Fortunately when $m \leq n$ in Equation 2.4, the solution simplifies to

$$h(t) = b_n\delta(t) + [P(D)y_o(t)]u(t) \tag{2.25}$$

Example 2.3

Solve Example 2.2(a) using the method of convolution.

We first determine $h(t)$. The characteristic modes for this case, as found in Example 2.1, are e^{-t} and e^{-2t}. Since $y_o(t)$ is a linear combination of the characteristic modes

$$y_o(t) = K_1e^{-t} + K_2e^{-2t} \quad t \geq 0$$

Therefore,

$$\dot{y}_o(t) = -K_1e^{-t} - 2K_2e^{-2t} \quad t \geq 0$$

The initial conditions according to Equation 2.24 are $\dot{y}_o(0) = 1$ and $y_o(0) = 0$. Setting $t = 0$ in the above equations and using the initial conditions, we obtain

$$K_1 + K_2 = 0 \quad \text{and} \quad -K_1 - 2K_2 = 1$$

Solution of these equations yields $K_1 = 1$ and $K_2 = -1$. Therefore,

$$y_o(t) = e^{-t} - e^{-2t}$$

Also in this case the polynomial $P(D) = D$ is of the first-order, and $b_2 = 0$. Therefore, from Equation 2.25

$$\begin{aligned} h(t) &= [P(D)y_o(t)]u(t) = [Dy_o(t)]u(t) \\ &= \left[\frac{d}{dt}\left(e^{-t} - e^{-2t}\right)\right]u(t) \\ &= (-e^{-t} + 2e^{-2t})u(t) \end{aligned}$$

and

$$\int_0^t f(x)h(t-x)\mathrm{d}x = \int_0^t 10\mathrm{e}^{-3x}\left[-\mathrm{e}^{-(t-x)} + 2\mathrm{e}^{-2(t-x)}\right]\mathrm{d}x$$
$$= -5\mathrm{e}^{-t} + 20\mathrm{e}^{-2t} - 15\mathrm{e}^{-3t}$$

The total solution is obtained by adding the complementary solution $y_c(t) = c_1\mathrm{e}^{-t} + c_2\mathrm{e}^{-2t}$ to this component. Therefore,

$$y(t) = c_1\mathrm{e}^{-t} + c_2\mathrm{e}^{-2t} - 5\mathrm{e}^{-t} + 20\mathrm{e}^{-2t} - 15\mathrm{e}^{-3t}$$

Setting the conditions $y(0^+) = 2$ and $y(0^+) = 3$ in this equation (and its derivative), we obtain $c_1 = -3$, $c_2 = 5$ so that

$$y(t) = -8\mathrm{e}^{-t} + 25\mathrm{e}^{-2t} - 15\mathrm{e}^{-3t} \quad t \geq 0$$

which is identical to the solution found by the classical method.

2.1.3.1 Assessment of the Convolution Method

The convolution method is more laborious compared to the classical method. However, in system analysis, its advantages outweigh the extra work. The classical method has a serious drawback because it yields the total response, which cannot be separated into components arising from the internal conditions and the external input. In the study of systems it is important to be able to express the system response to an input $f(t)$ as an explicit function of $f(t)$. This is not possible in the classical method. Moreover, the classical method is restricted to a certain class of inputs; it cannot be applied to any input.*

If we must solve a particular linear differential equation or find a response of a particular LTI system, the classical method may be the best. In the theoretical study of linear systems, however, it is practically useless. General discussion of differential equations can be found in numerous texts on the subject [1].

2.2 Difference Equations

The development of difference equations is parallel to that of differential equations. We consider here only linear difference equations with constant coefficients. An n th-order difference equation can be expressed in two different forms; the first form uses delay terms such as $y[k-1]$, $y[k-2]$, $f[k-1]$, $f[k-2]$, etc., and the alternative form uses advance terms such as $y[k+1]$, $y[k+2]$, etc. Both forms are useful. We start here with a general nth-order difference equation, using advance operator form.

$$y[k+n] + a_{n-1}y[k+n-1] + \cdots + a_1y[k+1] + a_0y[k]$$
$$= b_mf[k+m] + b_{m-1}f[k+m-1] + \cdots + b_1f[k+1] + b_0f[k] \tag{2.26}$$

* Another minor problem is that because the classical method yields total response, the auxiliary conditions must be on the total response, which exists only for $t \geq 0^+$. In practice we are most likely to know the conditions at $t = 0^-$ (before the input is applied). Therefore, we need to derive a new set of auxiliary conditions at $t = 0^+$ from the known conditions at $t = 0^-$. The convolution method can handle both kinds of initial conditions. If the conditions are given at $t = 0^-$, we apply these conditions only to $y_c(t)$ because by its definition the convolution integral is 0 at $t = 0^-$.

2.2.1 Causality Condition

The left-hand side of Equation 2.26 consists of values of $y[k]$ at instants $k+n$, $k+n-1$, $k+n-2$, and so on. The right-hand side of Equation 2.26 consists of the input at instants $k+m$, $k+m-1$, $k+m-2$, and so on. For a casual equation, the solution cannot depend on future input values. This show that when the equation is in the advance operator form of Equation 2.26, casuality requires $m \leq n$. For a general casual case, $m = n$, and Equation 2.26 becomes

$$\begin{aligned} &y[k+n] + a_{n-1}y[k+n-1] + \cdots + a_1 y[k+1] + a_0 y[k] \\ &\quad = b_n f[k+n] + b_{n-1} f[k+n-1] + \cdots + b_1 f[k+1] + b_0 f[k] \end{aligned} \tag{2.27a}$$

where some of the coefficients on both sides can be zero. However, the coefficient of $y[k+n]$ is normalized to unity. Equation 2.27a is valid for all values of k. Therefore, the equation is still valid if we replace k by $k-n$ throughout the equation. This yields the alternative form (the delay operator form) of Equation 2.27a

$$\begin{aligned} &y[k] + a_{n-1}y[k-1] + \cdots + a_1 y[k-n+1] + a_0 y[k-n] \\ &\quad = b_n f[k] + b_{n-1} f[k-1] + \cdots + b_1 f[k-n+1] + b_0 f[k-n] \end{aligned} \tag{2.27b}$$

We designate the form of Equation 2.27a the advance operator form, and the form of Equation 2.27b the delay operator form.

2.2.2 Initial Conditions and Iterative Solution

Equation 2.27b can be expressed as

$$\begin{aligned} y[k] = &- a_{n-1}y[k-1] - a_{n-2}y[k-2] - \cdots - a_0 y[k-n] + b_n f[k] \\ &+ b_{n-1} f[k-1] + \cdots + b_0 f[k-n] \end{aligned} \tag{2.27c}$$

This equation shows that $y[k]$, the solution at the k th instant, is computed from $2n+1$ pieces of information. These are the past n values of $y[k]$: $y[k-1], y[k-2], \ldots, y[k-n]$ and the present and past n values of the input: $f[k], f[k-1], f[k-2], \ldots, f[k-n]$. If the input $f[k]$ is known for $k = 0, 1, 2, \ldots$, then the values of $y[k]$ for $k = 0, 1, 2, \ldots$ can be computed from the $2n$ initial conditions $y[-1], y[-2], \ldots, y[-n]$ and $f[-1], f[-2], \ldots, f[-n]$. If the input is causal, that is, if $f[k] = 0$ for $k < 0$, then $f[-1] = f[-2] = \cdots = f[-n] = 0$, and we need only n initial conditions $y[-1]$, $y[-2], \ldots$, $y[-n]$. This allows us to compute iteratively or recursively the values $y[0]$, $y[1]$, $y[2]$, $y[3], \ldots$, and so on.* For instance, to find $y[0]$ we set $k = 0$ in Equation 2.27c. The left-hand side is $y[0]$, and the right-hand side contains terms $y[-1], y[-2], \ldots,$ $y[-n]$, and the inputs $f[0], f[-1], f[-2], \ldots, f[-n]$.

Therefore, to begin with, we must know the n initial conditions $y[-1]$, $y[-2], \ldots$, $y[-n]$. Knowing these conditions and the input $f[k]$, we can iteratively find the response $y[0]$, $y[1]$, $y[2], \ldots$, and so on. The following example demonstrates this procedure. This method basically reflects the manner in which a computer would solve a difference equation, given the input and initial conditions.

---

* For this reason Equation 2.27 is called a *recursive difference equation*. However, in Equation 2.27 if $a_0 = a_1 = a_2 = \cdots = a_{n-1} = 0$, then it follows from Equation 2.27c that determination of the present value of $y[k]$ does not require the past values $y[k-1]$, y $[k-2]$, etc. For this reason when $a_i = 0$ $(i = 0, 1, \ldots, n-1)$, the difference Equation 2.27 is *nonrecursive*. This classification is important in designing and realizing digital filters. In this discussion, however, this classification is not important. The analysis techniques developed here apply to general recursive and nonrecursive equations. Observe that a nonrecursive equation is a special case of recursive equation with $a_0 = a_1 = \cdots = a_{n-1} = 0$.

Example 2.4

Solve iteratively

$$y[k] - 0.5y[k-1] = f[k] \tag{2.28a}$$

with initial condition $y[-1] = 16$ and the input $f[k] = k^2$ (starting at $k = 0$). This equation can be expressed as

$$y[k] = 0.5y[k-1] + f[k] \tag{2.28b}$$

If we set $k = 0$ in this equation, we obtain

$$\begin{aligned} y[0] &= 0.5y[-1] + f[0] \\ &= 0.5(16) + 0 = 8 \end{aligned}$$

Now, setting $k = 1$ in Equation 2.28b and using the value $y[0] = 8$ (computed in the first step) and $f[1] = (1)^2 = 1$, we obtain

$$y[1] = 0.5(8) + (1)^2 = 5$$

Next, setting $k = 2$ in Equation 2.28b and using the value $y[1] = 5$ (computed in the previous step) and $f[2] = (2)^2$, we obtain

$$y[2] = 0.5(5) + (2)^2 = 6.5$$

Continuing in this way iteratively, we obtain

$$\begin{aligned} y[3] &= 0.5(6.5) + (3)^2 = 12.25 \\ y[4] &= 0.5(12.25) + (4)^2 = 22.125 \end{aligned}$$

and so on.

This iterative solution procedure is available only for difference equations; it cannot be applied to differential equations. Despite the many uses of this method, a closed-form solution of a difference equation is far more useful in the study of system behavior and its dependence on the input and the various system parameters. For this reason we shall develop a systematic procedure to obtain a closed-form solution of Equation 2.27.

2.2.3 Operational Notation

In difference equations it is convenient to use operational notation similar to that used in differential equations for the sake of compactness and convenience. For differential equations, we use the operator D to denote the operation of differentiation. For difference equations, we use the operator E to denote the operation for advancing the sequence by one time interval. Thus,

$$\begin{aligned} Ef[k] &\equiv f[k+1] \\ E^2 f[k] &\equiv f[k+2] \\ &\vdots \\ E^n f[k] &\equiv f[k+n] \end{aligned} \tag{2.29}$$

A general n th-order difference Equation 2.27a can be expressed as

$$\begin{aligned}&\left(E^n + a_{n-1}E^{n-1} + \cdots + a_1E + a_0\right)y[k]\\&\quad = \left(b_nE^n + b_{n-1}E^{n-1} + \cdots + b_1E + b_0\right)f[k]\end{aligned} \tag{2.30a}$$

or

$$Q[E]y[k] = P[E]f[k] \tag{2.30b}$$

where $Q[E]$ and $P[E]$ are n th-order polynomial operators, respectively,

$$Q[E] = E^n + a_{n-1}E^{n-1} + \cdots + a_1E + a_0 \tag{2.31a}$$

$$P[E] = b_nE^n + b_{n-1}E^{n-1} + \cdots + b_1E + b_0 \tag{2.31b}$$

2.2.4 Classical Solution

Following the discussion of differential equations, we can show that if $y_p[k]$ is a solution of Equation 2.27 or Equation 2.30, that is,

$$Q[E]y_p[k] = P[E]f[k] \tag{2.32}$$

then $y_p[k] + y_c[k]$ is also a solution of Equation 2.30, where $y_c[k]$ is a solution of the homogeneous equation

$$Q[E]y_c[k] = 0 \tag{2.33}$$

As before, we call $y_p[k]$ the particular solution and $y_c[k]$ the complementary solution.

2.2.4.1 Complementary Solution (the Natural Response)

By definition

$$Q[E]y_c[k] = 0 \tag{2.33a}$$

or

$$(E^n + a_{n-1}E^{n-1} + \cdots + a_1E + a_0)y_c[k] = 0 \tag{2.33b}$$

or

$$y_c[k+n] + a_{n-1}y_c[k+n-1] + \cdots + a_1y_c[k+1] + a_0y_c[k] = 0 \tag{2.33c}$$

We can solve this equation systematically, but even a cursory examination of this equation points to its solution. This equation states that a linear combination of $y_c[k]$ and delayed $y_c[k]$ is zero not for some values of k, but for all k. This is possible if and only if $y_c[k]$ and delayed $y_c[k]$ have the same form. Only an exponential function γ^k has this property as seen from the equation

$$\gamma^{k-m} = \gamma^{-m}\gamma^k$$

This shows that the delayed γ^k is a constant times γ^k. Therefore, the solution of Equation 2.33 must be of the form

$$y_c[k] = c\gamma^k \tag{2.34}$$

To determine c and γ, we substitute this solution in Equation 2.33. From Equation 2.34, we have

$$\begin{aligned} Ey_c[k] &= y_c[k+1] = c\gamma^{k+1} = (c\gamma)\gamma^k \\ E^2y_c[k] &= y_c[k+2] = c\gamma^{k+2} = (c\gamma^2)\gamma^k \\ &\vdots \\ E^ny_c[k] &= y_c[k+n] = c\gamma^{k+n} = (c\gamma^n)\gamma^k \end{aligned} \tag{2.35}$$

Substitution of this in Equation 2.33 yields

$$c\left(\gamma^n + a_{n-1}\gamma^{n-1} + \cdots + a_1\gamma + a_0\right)\gamma^k = 0 \tag{2.36}$$

For a nontrivial solution of this equation

$$\left(\gamma^n + a_{n-1}\gamma^{n-1} + \cdots + a_1\gamma + a_0\right) = 0 \tag{2.37a}$$

or

$$Q[\gamma] = 0 \tag{2.37b}$$

Our solution $c\gamma^k$ (Equation 2.34) is correct, provided that γ satisfies Equation 2.37. Now, $Q[\gamma]$ is an nth-order polynomial and can be expressed in the factorized form (assuming all distinct roots):

$$(\gamma - \gamma_1)(\gamma - \gamma_2)\cdots(\gamma - \gamma_n) = 0 \tag{2.37c}$$

Clearly γ has n solutions $\gamma_1, \gamma_2, \ldots, \gamma_n$ and, therefore, Equation 2.33 also has n solutions $c_1\gamma_1^k, c_2\gamma_2^k, \ldots, c_n\gamma_n^k$. In such a case we have shown that the general solution is a linear combination of the n solutions. Thus,

$$y_c[k] = c_1\gamma_1^k + c_2\gamma_2^k + \cdots + c_n\gamma_n^k \tag{2.38}$$

where $\gamma_1, \gamma_2, \ldots, \gamma_n$ are the roots of Equation 2.37 and $c_1, c_2, \ldots, c_n$ are arbitrary constants determined from n auxiliary conditions. The polynomial $Q[\gamma]$ is called the characteristic polynomial, and

$$Q[\gamma] = 0 \tag{2.39}$$

is the characteristic equation. Moreover, $\gamma_1, \gamma_2, \ldots, \gamma_n$ the roots of the characteristic equation, are called characteristic roots or characteristic values (also eigenvalues). The exponentials $\gamma_i^k (i = 1, 2, \ldots, n)$ are the characteristic modes or natural modes. A characteristic mode corresponds to each characteristic root, and the complementary solution is a linear combination of the characteristic modes of the system.

2.2.4.2 Repeated Roots

For repeated roots, the form of characteristic modes is modified. It can be shown by direct substitution that if a root γ repeats r times (root of multiplicity r), the characteristic modes corresponding to this root are γ^k, $k\gamma^k$, $k^2\gamma^k, \ldots, k^{r-1}\gamma^k$. Thus, if the characteristic equation is

$$Q[\gamma] = (\gamma - \gamma_1)^r(\gamma - \gamma_{r+1})(\gamma - \gamma_{r+2}) \ldots (\gamma - \gamma_n) \tag{2.40}$$

the complementary solution is

$$\begin{aligned} y_c[k] = &\left(c_1 + c_2 k + c_3 k^2 + \cdots + c_r k^{r-1}\right)\gamma_1^k \\ &+ c_{r+1}\gamma_{r+1}^k + c_{r+2}\gamma_{r+2}^k + \cdots + c_n\gamma_n^k \end{aligned} \tag{2.41}$$

2.2.4.3 Particular Solution

The particular solution $y_p[k]$ is the solution of

$$Q[E]y_p[k] = p[E]f[k] \tag{2.42}$$

We shall find the particular solution using the method of undetermined coefficients, the same method used for differential equations. Table 2.2 lists the inputs and the corresponding forms of solution with undetermined coefficients. These coefficients can be determined by substituting $y_p[k]$ in Equation 2.42 and equating the coefficients of similar terms.

Note: By definition, $y_p[k]$ cannot have any characteristic mode terms. If any term $p[k]$ shown in the right-hand column for the particular solution should also be a characteristic mode, the correct form of the particular solution must be modified to $k^i p[k]$, where i is the smallest integer that will prevent $k^i p[k]$ from having a characteristic mode term. For example, when the input is r^k, the particular solution in the right-hand column is of the form cr^k. But if r^k happens to be a natural mode, the correct form of the particular solution is $\beta k r^k$ (see Pair 2).

Example 2.5

Solve

$$(E^2 - 5E + 6)y[k] = (E - 5)f[k] \tag{2.43}$$

if the input $f[k] = (3k + 5)u[k]$ and the auxiliary conditions are $y[0] = 4$, $y[1] = 13$.

The characteristic equation is

$$\gamma^2 - 5\gamma + 6 = (\gamma - 2)(\gamma - 3) = 0$$

TABLE 2.2 Inputs and Forms of Solution

No.	Input $f[k]$	Forced Response $y_p[k]$
1	r^k $r \neq \gamma_i$ $(i = 1, 2, \ldots, n)$	βr^k
2	r^k $r = \gamma_i$	$\beta k r^k$
3	$\cos(\Omega k + \theta)$	$\beta \cos(\Omega k + \varphi)$
4	$\left(\sum_{i=0}^{m} a_i k^i\right) r^k$	$\left(\sum_{i=0}^{m} \beta_i k^i\right) r^k$

Therefore, the complementary solution is

$$y_c[k] = c_1(2)^k + c_2(3)^k$$

To find the form of $y_p[k]$ we use Table 2.2, Pair 4 with $r = 1$, $m = 1$. This yields

$$y_p[k] = \beta_1 k + \beta_0$$

Therefore,

$$y_p[k+1] = \beta_1(k+1) + \beta_0 = \beta_1 k + \beta_1 + \beta_0$$
$$y_p[k+2] = \beta_1(k+2) + \beta_0 = \beta_1 k + 2\beta_1 + \beta_0$$

Also,

$$f[k] = 3k + 5$$

and

$$f[k+1] = 3(k+1) + 5 = 3k + 8$$

Substitution of the above results in Equation 2.43 yields

$$\beta_1 k + 2\beta_1 + \beta_0 - 5(\beta_1 k + \beta_1 + \beta_0) + 6(\beta_1 k + \beta_0) = 3k + 8 - 5(3k + 5)$$

or

$$2\beta_1 k - 3\beta_1 + 2\beta_0 = -12k - 17$$

Comparison of similar terms on two sides yields

$$\left.\begin{aligned} 2\beta_1 &= -12 \\ -3\beta_1 + 2\beta_0 &= -17 \end{aligned}\right\} \Rightarrow \begin{aligned} \beta_1 &= -6 \\ \beta_2 &= -\tfrac{35}{2} \end{aligned}$$

This means

$$y_p[k] = -6k - \frac{35}{2}$$

The total response is

$$\begin{aligned} y[k] &= y_c[k] + y_p[k] \\ &= c_1(2)^k + c_2(3)^k - 6k - \frac{35}{2} \quad k \geq 0 \end{aligned} \tag{2.44}$$

To determine arbitrary constants c_1 and c_2 we set $k = 0$ and 1 and substitute the auxiliary conditions $y[0] = 4$, $y[1] = 13$, to obtain

$$\left.\begin{aligned} 4 &= c_1 + c_2 - \tfrac{35}{2} \\ 13 &= 2c_1 + 3c_2 - \tfrac{47}{2} \end{aligned}\right\} \Rightarrow \begin{aligned} c_1 &= 28 \\ c_2 &= \tfrac{-13}{2} \end{aligned}$$

Therefore,

$$y_c[k] = 28(2)^k - \frac{13}{2}(3)^k \tag{2.45}$$

and

$$y[k] = \underbrace{28(2)^k - \frac{13}{2}(3)^k}_{y_c[k]} \underbrace{- 6k - \frac{35}{2}}_{y_p[k]} \tag{2.46}$$

2.2.4.4 A Comment on Auxiliary Conditions

This method requires auxiliary conditions $y[0], y[1], \ldots, y[n-1]$, because the total solution is valid only for $k \geq 0$. But if we are given the initial conditions $y[-1], y[-2], \ldots, y[-n]$, we can derive the conditions $y[0], y[1], \ldots, y[n-1]$, using the iterative procedure discussed earlier.

2.2.4.5 Exponential Input

As in the case of differential equations, we can show that for the equation

$$Q[E]y[k] = P[E]f[k] \tag{2.47}$$

the particular solution for the exponential input $f[k] = r^k$ is given by

$$y_p[k] = H[r]r^k \quad r \neq \gamma_i \tag{2.48}$$

where

$$H[r] = \frac{P[r]}{Q[r]} \tag{2.49}$$

The proof follows from the fact that if the input $f[k] = r^k$, then from Table 2.2 (Pair 4), $y_p[k] = \beta r^k$. Therefore,

$$E^i f[k] = f[k+i] = r^{k+i} = r^i r^k \quad \text{and} \quad P[E]f[k] = P[r]r^k$$
$$E^j y_p[k] = \beta r^{k+j} = \beta r^j r^k \quad \text{and} \quad Q[E]y[k] = \beta Q[r]r^k$$

so that Equation 2.47 reduces to

$$\beta Q[r]r^k = P[r]r^k$$

which yields $\beta = P[r]/Q[r] = H[r]$.

This result is valid only if r is not a characteristic root. If r is a characteristic root, the particular solution is $\beta k r^k$ where β is determined by substituting $y_p[k]$ in Equation 2.47 and equating coefficients of similar terms on the two sides. Observe that the exponential r^k includes a wide variety of signals such as a constant C, a sinusoid $\cos(\Omega k + \theta)$, and an exponentially growing or decaying sinusoid $|\gamma|^k \cos(\Omega k + \theta)$.

2.2.4.6 A Constant Input $f(k) = C$

This is a special case of exponential Cr^k with $r = 1$. Therefore, from Equation 2.48 we have

$$y_p[k] = C\frac{P[1]}{Q[1]}(1)^k = CH[1] \tag{2.50}$$

2.2.4.7 A Sinusoidal Input

The input $e^{j\Omega k}$ is an exponential r^k with $r = e^{j\Omega}$. Hence,

$$y_p[k] = H[e^{j\Omega}]e^{j\Omega k} = \frac{P[e^{j\Omega}]}{Q[e^{j\Omega}]}e^{j\Omega k}$$

Similarly for the input $e^{-j\Omega k}$

$$y_p[k] = H[e^{-j\Omega}]e^{-j\Omega k}$$

Consequently, if the input

$$f[k] = \cos \Omega k = \frac{1}{2}(e^{j\Omega k} + e^{-j\Omega k})$$
$$y_p[k] = \frac{1}{2}\{H[e^{j\Omega}]e^{j\Omega k} + H[e^{-j\Omega}]e^{-j\Omega k}\}$$

Since the two terms on the right-hand side are conjugates

$$y_p[k] = \mathrm{Re}\{H[e^{j\Omega}]e^{j\Omega k}\}$$

If

$$H[e^{j\Omega}] = |H[e^{j\Omega}]|e^{j\angle H[e^{j\Omega}]}$$

then

$$\begin{aligned} y_p[k] &= \mathrm{Re}\{|H[e^{j\Omega}]|e^{j(\Omega k + \angle H[e^{j\Omega}])}\} \\ &= |H[e^{j\Omega}]|\cos(\Omega k + \angle H[e^{j\Omega}]) \end{aligned} \tag{2.51}$$

Using a similar argument, we can show that for the input

$$\begin{aligned} f[k] &= \cos(\Omega k + \theta) \\ y_p[k] &= |H[e^{j\Omega}]|\cos(\Omega k + \theta + \angle H[e^{j\Omega}]) \end{aligned} \tag{2.52}$$

Example 2.6

Solve

$$(E^2 - 3E + 2)y[k] = (E + 2)f[k]$$

for $f[k] = (3)^k u[k]$ and the auxiliary conditions $y[0] = 2, y[1] = 1$.

In this case

$$H[r] = \frac{P[r]}{Q[r]} = \frac{r+2}{r^2 - 3r + 2}$$

and the particular solution to input $(3)^k u[k]$ is $H[3](3)^k$, that is,

$$y_p[k] = \frac{3+2}{(3)^2 - 3(3) + 2}(3)^k = \frac{5}{2}(3)^k$$

The characteristic polynomial is $(\gamma^2 - 3\gamma + 2) = (\gamma - 1)(\gamma - 2)$. The characteristic roots are 1 and 2. Hence, the complementary solution is $y_c[k] = c_1 + c_2(2)^k$ and the total solution is

$$y[k] = c_1(1)^k + c_2(2)^k + \frac{5}{2}(3)^k$$

Setting $k = 0$ and 1 in this equation and substituting auxiliary conditions yields

$$2 = c_1 + c_2 + \frac{5}{2} \quad \text{and} \quad 1 = c_1 + 2c_2 + \frac{15}{2}$$

Solution of these two simultaneous equations yields $c_1 = 5.5, c_2 = -5$. Therefore,

$$y[k] = 5.5 - 6(2)^k + \frac{5}{2}(3)^k \quad k \geq 0$$

2.2.5 Method of Convolution

In this method, the input $f[k]$ is expressed as a sum of impulses. The solution is then obtained as a sum of the solutions to all the impulse components. The method exploits the superposition property of the linear difference equations. A discrete-time unit impulse function $\delta[k]$ is defined as

$$\delta[k] = \begin{cases} 1 & k = 0(94) \\ 0 & k \neq 0 \end{cases} \tag{2.53}$$

Hence, an arbitrary signal $f[k]$ can be expressed in terms of impulse and delayed impulse functions as

$$f[k] = f[0]\delta[k] + f[1]\delta[k-1] + f[2]\delta[k-2] + \cdots + f[k]\delta[0] + \cdots \quad k \geq 0 \tag{2.54}$$

The right-hand side expresses $f[k]$ as a sum of impulse components. If $h[k]$ is the solution of Equation 2.30 to the impulse input $f[k] = \delta[k]$, then the solution to input $\delta[k-m]$ is $h[k-m]$. This follows from the fact that because of constant coefficients, Equation 2.30 has time invariance property. Also, because Equation 2.30 is linear, its solution is the sum of the solutions to each of the impulse components of $f[k]$ on the right-hand side of Equation 2.54 Therefore,

$$\begin{aligned} y[k] = {} & f[0]h[k] + f[1]h[k-1] + f[2]h[k-2] \\ & + \cdots + f[k]h[0] + f[k+1]h[-1] + \cdots \end{aligned}$$

All practical systems with time as the independent variable are causal, that is, $h[k] = 0$ for $k < 0$. Hence, all the terms on the right-hand side beyond $f[k]h[0]$ are zero. Thus,

$$\begin{aligned} y[k] &= f[0]h[k] + f[1]h[k-1] + f[2]h[k-2] + \cdots + f[k]h[0] \\ &= \sum_{m=0}^{k} f[m]h[k-m] \end{aligned} \tag{2.55}$$

The first term on the right-hand side consists of a linear combination of natural modes and should be appropriately modified for repeated roots. The general solution is obtained by adding a complementary solution to the above solution. Therefore, the general solution is given by

$$y[k] = \sum_{j=1}^{n} c_j \gamma_j^k + \sum_{m=0}^{k} f[m]h[k-m] \tag{2.56}$$

The last sum on the right-hand side is known as the convolution sum of $f[k]$ and $h[k]$.

The function $h[k]$ appearing in Equation 2.56 is the solution of Equation 2.30 for the impulsive input ($f[k] = \delta[k]$) when all initial conditions are zero, that is, $h[-1] = h[-2] = \cdots = h[-n] = 0$. It can be shown that [2] $h[k]$ contains an impulse and a linear combination of characteristic modes as

$$h[k] = \frac{b_0}{a_0}\delta[k] + A_1\gamma_1^k + A_2\gamma_2^k + \cdots + A_n\gamma_n^k \tag{2.57}$$

where the unknown constants A_i are determined from n values of $h[k]$ obtained by solving the equation $Q[E]h[k] = P[E]\delta[k]$ iteratively.

Example 2.7

Solve Example 2.5 using convolution method. In other words solve

$$(E^2 - 3E + 2)y[k] = (E + 2)f[k]$$

for $f[k] = (3)^k u[k]$ and the auxiliary conditions $y[0] = 2, y[1] = 1$.

The unit impulse solution $h[k]$ is given by Equation 2.57. In this case $a_0 = 2$ and $b_0 = 2$. Therefore,

$$h[k] = \delta[k] + A_1(1)^k + A_2(2)^k \tag{2.58}$$

To determine the two unknown constants A_1 and A_2 in Equation 2.58, we need two values of $h[k]$, for instance $h[0]$ and $h[1]$. These can be determined iteratively by observing that $h[k]$ is the solution of $(E^2 - 3E + 2)h[k] = (E + 2)\delta[k]$, that is,

$$h[k+2] - 3h[k+1] + 2h[k] = \delta[k+1] + 2\delta[k] \tag{2.59}$$

subject to initial conditions $h[-1] = h[-2] = 0$. We now determine $h[0]$ and $h[1]$ iteratively from Equation 2.59. Setting $k = -2$ in this equation yields

$$h[0] - 3(0) + 2(0) = 0 + 0 \Rightarrow h[0] = 0$$

Next, setting $k = -1$ in Equation 2.59 and using $h[0] = 0$, we obtain

$$h[1] - 3(0) + 2(0) = 1 + 0 \Rightarrow h[1] = 1$$

Setting $k = 0$ and 1 in Equation 2.58 and substituting $h[0] = 0, h[1] = 1$ yields

$$0 = 1 + A_1 + A_2 \quad \text{and} \quad 1 = A_1 + 2A_2$$

Solution of these two equations yields $A_1 = -3$ and $A_2 = 2$. Therefore,

$$h[k] = \delta[k] - 3 + 2(2)^k$$

and from Equation 2.56

$$\begin{aligned} y[k] &= c_1 + c_2(2)^k + \sum_{m=0}^{k} (3)^m \left[\delta[k-m] - 3 + 2(2)^{k-m}\right] \\ &= c_1 + c_2(2)^k + 1.5 - 4(2)^k + 2.5(3)^k \end{aligned}$$

The sums in the above expression are found by using the geometric progression sum formula

$$\sum_{m=0}^{k} r^m = \frac{r^{k+1} - 1}{r - 1} r \neq 1$$

Setting $k = 0$ and 1 and substituting the given auxiliary conditions $y[0] = 2, y[1] = 1$, we obtain

$$2 = c_1 + c_2 + 1.5 - 4 + 2.5 \quad \text{and} \quad 1 = c_1 + 2c_2 + 1.5 - 8 + 7.5$$

Solution of these equations yields $c_1 = 4$ and $c_2 = -2$. Therefore,

$$y[k] = 5.5 - 6(2)^k + 2.5(3)^k$$

which confirms the result obtained by the classical method.

2.2.5.1 Assessment of the Classical Method

The earlier remarks concerning the classical method for solving differential equations also apply to difference equations. General discussion of difference equations can be found in texts on the subject [3].

References

1. Birkhoff, G. and Rota, G.C., *Ordinary Differential Equations*, 3rd edn., John Wiley & Sons, New York, 1978.
2. Lathi, B.P., *Signal Processing and Linear Systems*, Berkeley-Cambridge Press, Carmichael, CA, 1998.
3. Goldberg, S., *Introduction to Difference Equations*, John Wiley & Sons, New York, 1958.

3

Finite Wordlength Effects

Bruce W. Bomar
University of Tennessee Space Institute

3.1 Introduction

Practical digital filters must be implemented with finite precision numbers and arithmetic. As a result, both the filter coefficients and the filter input and output signals are in discrete form. This leads to four types of finite wordlength effects.

Discretization (quantization) of the filter coefficients has the effect of perturbing the location of the filter poles and zeros. As a result, the actual filter response differs slightly from the ideal response. This deterministic frequency response error is referred to as **coefficient quantization error**.

The use of finite precision arithmetic makes it necessary to quantize filter calculations by rounding or truncation. **Roundoff noise** is that error in the filter output that results from rounding or truncating calculations within the filter. As the name implies, this error looks like low-level noise at the filter output.

Quantization of the filter calculations also renders the filter slightly nonlinear. For large signals this nonlinearity is negligible and roundoff noise is the major concern. However, for recursive filters with a zero or constant input, this nonlinearity can cause spurious oscillations called **limit cycles**.

With fixed-point arithmetic it is possible for filter calculations to overflow. The term **overflow oscillation**, sometimes also called **adder overflow limit cycle**, refers to a high-level oscillation that can exist in an otherwise stable filter due to the nonlinearity associated with the overflow of internal filter calculations.

In this chapter, we examine each of these finite wordlength effects. Both fixed-point and floating-point number representations are considered.

3.2 Number Representation

In digital signal processing, $(B+1)$-bit fixed-point numbers are usually represented as two's-complement signed fractions in the format

$$b_0 b_{-1} b_{-2} \ldots b_{-B}$$

The number represented is then

$$X = -b_0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-B}2^{-B} \tag{3.1}$$

where b_0 is the sign bit and the number range is $-1 \le X < 1$. The advantage of this representation is that the product of two numbers in the range from -1 to 1 is another number in the same range.

Floating-point numbers are represented as

$$X = (-1)^s m 2^c \tag{3.2}$$

where
- s is the sign bit
- m is the **mantissa**
- c is the characteristic or exponent

To make the representation of a number unique, the mantissa is normalized so that $0.5 \le m < 1$.

Although floating-point numbers are always represented in the form of Equation 3.2, the way in which this representation is actually stored in a machine may differ. Since $m \ge 0.5$, it is not necessary to store the 2^{-1}-weight bit of m, which is always set. Therefore, in practice numbers are usually stored as

$$X = (-1)^s (0.5 + f) 2^c \tag{3.3}$$

where f is an unsigned fraction, $0 \le f < 0.5$.

Most floating-point processors now use the IEEE Standard 754 32-bit floating-point format for storing numbers. According to this standard the exponent is stored as an unsigned integer p where

$$p = c + 126 \tag{3.4}$$

Therefore, a number is stored as

$$X = (-1)^s (0.5 + f) 2^{p-126} \tag{3.5}$$

where
- s is the sign bit
- f is a 23-bit unsigned fraction in the range $0 \le f < 0.5$
- p is an 8-bit unsigned integer in the range $0 \le p \le 255$

The total number of bits is $1 + 23 + 8 = 32$. For example, in IEEE format 3/4 is written $(-1)^0(0.5 + 0.25)2^0$ so $s = 0$, $p = 126$, and $f = 0.25$. The value $X = 0$ is a unique case and is represented by all bits zero (i.e., $s = 0$, $f = 0$, and $p = 0$). Although the 2^{-1}-weight mantissa bit is not actually stored, it does exist so the mantissa has 24 bits plus a sign bit.

3.3 Fixed-Point Quantization Errors

In fixed-point arithmetic, a multiply doubles the number of significant bits. For example, the product of the two 5-bit numbers 0.0011 and 0.1001 is the 10-bit number 00.00011011. The extra bit to the left of the decimal point can be discarded without introducing any error. However, the least significant four of the remaining bits must ultimately be discarded by some form of quantization so that the result can be stored to 5 bits for use in other calculations. In the example above this results in 0.0010 (quantization by rounding) or 0.0001 (quantization by truncating). When a sum of products calculation is performed, the quantization can be performed either after each multiply or after all products have been summed with double-length precision.

We will examine three types of fixed-point quantization: rounding, truncation, and magnitude truncation. If X is an exact value, then the rounded value will be denoted $Q_r(X)$, the truncated value $Q_t(X)$, and the magnitude truncated value $Q_{mt}(X)$. If the quantized value has B bits to the right of the decimal point, the quantization step size is

$$\Delta = 2^{-B} \tag{3.6}$$

Since rounding selects the quantized value nearest the unquantized value, it gives a value which is never more than $\pm\Delta/2$ away from the exact value. If we denote the rounding error by $\pm\Delta/2$ away from the exact value. If we denote the rounding error by

$$\varepsilon_r = Q_r(X) - X \tag{3.7}$$

then

$$-\frac{\Delta}{2} \leq \varepsilon_r \frac{\Delta}{2} \tag{3.8}$$

Truncation simply discards the low-order bits, giving a quantized value that is always less than or equal to the exact value so

$$-\Delta < \varepsilon_t \leq 0 \tag{3.9}$$

Magnitude truncation chooses the nearest quantized value that has a magnitude less than or equal to the exact value so

$$-\Delta < \varepsilon_{mt} \leq \Delta \tag{3.10}$$

The error resulting from quantization can be modeled as a random variable uniformly distributed over the appropriate error range. Therefore, calculations with roundoff error can be considered error-free calculations that have been corrupted by additive white noise. The mean of this noise for rounding is

$$m_{\varepsilon_r} = E\{\varepsilon_r\} = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} \varepsilon_r d\varepsilon_r = 0 \tag{3.11}$$

where $E\{\}$ represents the operation of taking the expected value of a random variable. Similarly, the variance of the noise for rounding is

$$\sigma_{\varepsilon_r}^2 = E\left\{(\varepsilon_r - m_{\varepsilon_r})^2\right\} = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2}(\varepsilon_r - m_{\varepsilon_r})^2 d\varepsilon_r = \frac{\Delta^2}{12} \tag{3.12}$$

Likewise, for truncation,

$$\begin{aligned} m_{\varepsilon_t} &= E\{\varepsilon_t\} = -\frac{\Delta}{2} \\ \sigma_{\varepsilon_t}^2 &= E\left\{(\varepsilon_t - m_{\varepsilon_t})^2\right\} = \frac{\Delta^2}{12} \end{aligned} \tag{3.13}$$

and, for magnitude truncation,

$$m_{\varepsilon_t} = E\left\{(\varepsilon_{mt} - \boldsymbol{m}_{\varepsilon_{mt}})^2\right\} = \frac{\Delta^2}{3} \tag{3.14}$$

3.4 Floating-Point Quantization Errors

With floating-point arithmetic, it is necessary to quantize after both multiplications and additions. The addition quantization arises because, prior to addition, the mantissa of the smaller number in the sum is shifted right until the exponent of both numbers is the same. In general, this gives a sum mantissa that is too long and so must be quantized.

We will assume that quantization in floating-point arithmetic is performed by rounding. Because of the exponent in floating-point arithmetic, it is the relative error that is important. The relative error is defined as

$$\varepsilon_r = \frac{Q_r(X) - X}{X} = \frac{\varepsilon_r}{X} \tag{3.15}$$

Since $X = (-1)^s\, m2^c$, $Q_r(X) = (-1)^s Q_r(m)2^c$ and

$$\varepsilon_r = \frac{Q_r(m) - m}{m} = \frac{\varepsilon}{m} \tag{3.16}$$

If the quantized mantissa has B bits to the right of the decimal point, $|\varepsilon| < \Delta/2$ where, as before, $\Delta = 2^{-B}$. Therefore, since $0.5 \le m < 1$,

$$|\varepsilon_r| < \Delta \tag{3.17}$$

If we assume that ε is uniformly distributed over the range from $-\Delta/2$ to $\Delta/2$ and m is uniformly distributed over 0.5–1, then

$$m_{\varepsilon_r} = E\left\{\frac{\varepsilon}{m}\right\} = 0$$

and

$$\sigma_{\varepsilon_r}^2 = E\left\{\left(\frac{\varepsilon}{m}\right)^2\right\} = \frac{2}{\Delta}\int_{1/2}^{1}\int_{-\Delta/2}^{\Delta/2}\frac{\varepsilon^2}{m^2}\,d\varepsilon dm$$
$$= \frac{\Delta^2}{6} = (0.167)2^{-2B} \tag{3.18}$$

In practice, the distribution of m is not exactly uniform. Actual measurements of roundoff noise in [1] suggested that

$$\sigma_{\varepsilon_r}^2 \approx 0.23\Delta^2 \tag{3.19}$$

while a detailed theoretical and experimental analysis in [2] determined

$$\sigma_{\varepsilon_r}^2 \approx 0.18\Delta^2 \tag{3.20}$$

From Equation 3.15, we can represent a quantized floating-point value in terms of the unquantized value and the random variable ε_r using

$$Q_r(X) = X(1 + \varepsilon_r) \tag{3.21}$$

Therefore, the finite-precision product X_1X_2 and the sum $X_1 + X_2$ can be written as

$$fl(X_1X_2) = X_1X_2(1 + \varepsilon_r) \tag{3.22}$$

and

$$fl(X_1 + X_2) = (X_1 + X_2)(1 + \varepsilon_r) \tag{3.23}$$

where ε_r is zero-mean with the variance of Equation 3.20.

3.5 Roundoff Noise

To determine the roundoff noise at the output of a digital filter, we will assume that the noise due to a quantization is stationary, white, and uncorrelated with the filter input, output, and internal variables. This assumption is good if the filter input changes from sample to sample in a sufficiently complex manner. It is not valid for zero or constant inputs for which the effects of rounding are analyzed from a limit-cycle perspective.

To satisfy the assumption of a sufficiently complex input, roundoff noise in digital filters is often calculated for the case of a zero-mean white noise filter input signal $x(n)$ of variance σ_x^2. This simplifies calculation of the output roundoff noise because expected values of the form $E\{x(n)x(n-k)\}$ are zero for $k \neq 0$ and give σ_x^2 when $k = 0$. This approach to analysis has been found to give estimates of the output roundoff noise that are close to the noise actually observed for other input signals.

Another assumption that will be made in calculating roundoff noise is that the product of two quantization errors is zero. To justify this assumption, consider the case of a 16-bit fixed-point processor. In this case, a quantization error is of the order 2^{-15}, while the product of two quantization errors is of the order 2^{-30}, which is negligible by comparison.

If a linear system with impulse response $g(n)$ is excited by white noise with mean m_x and variance σ_x^2, the output is noise of mean [3, pp. 788–790]

$$m_y = m_x \sum_{n=-\infty}^{\infty} g(n) \tag{3.24}$$

and variance

$$\sigma_y^2 = \sigma_x^2 \sum_{n=-\infty}^{\infty} g^2(n) \tag{3.25}$$

Therefore, if $g(n)$ is the impulse response from the point where a roundoff takes place to the filter output, the contribution of that roundoff to the variance (mean-square value) of the output roundoff noise is given by Equation 3.25 with σ_x^2 replaced with the variance of the roundoff. If there is more than one source of roundoff error in the filter, it is assumed that the errors are uncorrelated so the output noise variance is simply the sum of the contributions from each source.

3.5.1 Roundoff Noise in FIR Filters

The simplest case to analyze is a finite impulse response (FIR) filter realized via the convolution summation

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{3.26}$$

When fixed-point arithmetic is used and quantization is performed after each multiply, the result of the N multiplies is N-times the quantization noise of a single multiply. For example, rounding after each multiply gives, from Equations 3.6 and 3.12, an output noise variance of

$$\sigma_o^2 = N\frac{2^{-2B}}{12} \tag{3.27}$$

Virtually all digital signal processor integrated circuits contain one or more double-length accumulator registers which permit the sum-of-products in Equation 3.26 to be accumulated without quantization. In this case only a single quantization is necessary following the summation and

$$\sigma_o^2 = \frac{2^{-2B}}{12} \tag{3.28}$$

For the floating-point roundoff noise case we will consider Equation 3.26 for $N = 4$ and then generalize the result to other values of N. The finite-precision output can be written as the exact output plus an error term $e(n)$. Thus,

$$\begin{aligned} y(n) + e(n) = (\{&[h(0)x(n)[1+\varepsilon_1(n)] \\ &+ h(1)x(n-1)[1+\varepsilon_2(n)]][1+\varepsilon_3(n)] \\ &+ h(2)x(n-2)[1+\varepsilon_4(n)]\}\{1+\varepsilon_5(n)\} \\ &+ h(3)x(n-3)[1+\varepsilon_6(n)])[1+\varepsilon_7(n)] \end{aligned} \tag{3.29}$$

In Equation 3.29, $\varepsilon_1(n)$ represents the error in the first product, $\varepsilon_2(n)$ the error in the second product, $\varepsilon_3(n)$ the error in the first addition, etc. Notice that it has been assumed that the products are summed in the order implied by the summation of Equation 3.26.

Expanding Equation 3.29, ignoring products of error terms, and recognizing $y(n)$ gives

$$\begin{aligned} e(n) = {} & h(0)x(n)[\varepsilon_1(n) + \varepsilon_3(n) + \varepsilon_5(n) + \varepsilon_7(n)] \\ & + h(1)x(n-1)[\varepsilon_2(n) + \varepsilon_3(n) + \varepsilon_5(n) + \varepsilon_7(n)] \\ & + h(2)x(n-2)[\varepsilon_4(n) + \varepsilon_5(n) + \varepsilon_7(n)] \\ & + h(3)x(n-3)[\varepsilon_6(n) + \varepsilon_7(n)] \end{aligned} \tag{3.30}$$

Assuming that the input is white noise of variance σ_x^2 so that $E\{x(n)x(n-k)\}$ is zero for $k \neq 0$, and assuming that the errors are uncorrelated,

$$E\{e^2(n)\} = \left[4h^2(0) + 4h^2(1) + 3h^2(2) + 2h^2(3)\right]\sigma_x^2\sigma_{\varepsilon_r}^2 \tag{3.31}$$

In general, for any N,

$$\sigma_o^2 = E\{e^2(n)\} = \left[Nh^2(0) + \sum_{k=1}^{N-1}(N+1-k)h^2(k)\right]\sigma_x^2\sigma_{\varepsilon_r}^2 \tag{3.32}$$

Notice that if the order of summation of the product terms in the convolution summation is changed, then the order in which the $h(k)$'s appear in Equation 3.32 changes. If the order is changed so that the $h(k)$ with smallest magnitude is first, followed by the next smallest, etc., then the roundoff noise variance is minimized. However, performing the convolution summation in nonsequential order greatly complicates data indexing and so may not be worth the reduction obtained in roundoff noise.

3.5.2 Roundoff Noise in Fixed-Point IIR Filters

To determine the roundoff noise of a fixed-point infinite impulse response (IIR) filter realization, consider a causal first-order filter with impulse response

$$h(n) = a^n u(n) \tag{3.33}$$

realized by the difference equation

$$y(n) = ay(n-1) + x(n) \tag{3.34}$$

Due to roundoff error, the output actually obtained is

$$\hat{y}(n) = Q\{ay(n-1) + x(n)\} = ay(n-1) + x(n) + e(n) \tag{3.35}$$

where $e(n)$ is a random roundoff noise sequence. Since $e(n)$ is injected at the same point as the input, it propagates through a system with impulse response $h(n)$. Therefore, for fixed-point arithmetic with rounding, the output roundoff noise variance from Equations 3.6, 3.12, 3.25, and 3.33 is

$$\sigma_o^2 = \frac{\Delta^2}{12}\sum_{n=-\infty}^{\infty} h^2(n) = \frac{\Delta^2}{12}\sum_{n=0}^{\infty} a^{2n} = \frac{2^{-2B}}{12}\frac{1}{1-a^2} \tag{3.36}$$

With fixed-point arithmetic there is the possibility of overflow following addition. To avoid overflow it is necessary to restrict the input signal amplitude. This can be accomplished by either placing a scaling multiplier at the filter input or by simply limiting the maximum input signal amplitude. Consider the case of the first-order filter of Equation 3.34. The transfer function of this filter is

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = \frac{1}{e^{j\omega} - a} \tag{3.37}$$

so

$$\left|H(e^{j\omega})\right|^2 = \frac{1}{1 + a^2 - 2a\cos(\omega)} \tag{3.38}$$

and

$$\left|H(e^{j\omega})\right|_{\max} = \frac{1}{1 - |a|} \tag{3.39}$$

The peak gain of the filter is $1/(1 - |a|)$ so limiting input signal amplitudes to $|x(n)| \leq 1 - |a|$ will make overflows unlikely.

An expression for the output roundoff noise-to-signal ratio can easily be obtained for the case where the filter input is white noise, uniformly distributed over the interval from $-(1 - |a|)$ to $(1 - |a|)$ [4,5]. In this case,

$$\sigma_x^2 = \frac{1}{2(1 - |a|)} \int_{-(1-|a|)}^{1-|a|} x^2 dx = \frac{1}{3}(1 - |a|)^2 \tag{3.40}$$

so, from Equation 3.25,

$$\sigma_y^2 = \frac{1}{3}\frac{(1 - |a|)^2}{1 - a^2} \tag{3.41}$$

Combining Equations 3.36 and 3.41 then gives

$$\frac{\sigma_o^2}{\sigma_y^2} = \left(\frac{2^{-2B}}{12}\frac{1}{1 - a^2}\right)\left(3\frac{1 - a^2}{(1 - |a|)^2}\right) = \frac{2^{-2B}}{12}\frac{3}{(1 - |a|)^2} \tag{3.42}$$

Notice that the noise-to-signal ratio increases without bound as $|a| \to 1$.

Similar results can be obtained for the case of the causal second-order filter realized by the difference equation

$$y(n) = 2r\cos(\theta)y(n - 1) - r^2 y(n - 2) + x(n) \tag{3.43}$$

This filter has complex-conjugate poles at $re^{\pm j\theta}$ and impulse response

$$h(n) = \frac{1}{\sin(\theta)} r^n \sin[(n + 1)\theta]u(n) \tag{3.44}$$

Due to roundoff error, the output actually obtained is

$$\hat{y}(n) = 2r\cos(\theta)y(n-1) - r^2y(n-2) + x(n) + e(n) \tag{3.45}$$

There are two noise sources contributing to $e(n)$ if quantization is performed after each multiply, and there is one noise source if quantization is performed after summation. Since

$$\sum_{n=-\infty}^{\infty} h^2(n) = \frac{1+r^2}{1-r^2}\,\frac{1}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \tag{3.46}$$

the output roundoff noise is

$$\sigma_o^2 = \nu\frac{2^{-2B}}{12}\,\frac{1+r^2}{1-r^2}\,\frac{1}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \tag{3.47}$$

where $\nu = 1$ for quantization after summation, and $\nu = 2$ for quantization after each multiply.

To obtain an output noise-to-signal ratio we note that

$$H(e^{j\omega}) = \frac{1}{1 - 2r\cos(\theta)e^{-j\omega} + r^2e^{-j2\omega}} \tag{3.48}$$

and, using the approach of [6],

$$\left|H(e^{j\omega})\right|^2_{\max} = \frac{1}{4r^2\left\{\left[\text{sat}\left(\frac{1+r^2}{2r}\cos(\theta)\right) - \frac{1+r^2}{2r}\cos(\theta)\right]^2 + \left[\frac{1-r^2}{2r}\sin(\theta)\right]^2\right\}} \tag{3.49}$$

where

$$\text{sat}(\mu) = \begin{cases} 1 & \mu > 1 \\ \mu & -1 \le \mu \le 1 \\ -1 & \mu < -1 \end{cases} \tag{3.50}$$

Following the same approach as for the first-order case then gives

$$\frac{\sigma_o^2}{\sigma_y^2} = \nu\frac{2^{-2B}}{12}\,\frac{1+r^2}{1-r^2}\,\frac{3}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \times \frac{1}{4r^2\left\{\left[\text{sat}\left(\frac{1+r^2}{2r}\cos(\theta)\right) - \frac{1+r^2}{2r}\cos(\theta)\right]^2 + \left[\frac{1-r^2}{2r}\sin(\theta)\right]^2\right\}} \tag{3.51}$$

Figure 3.1 is a contour plot showing the noise-to-signal ratio of Equation 3.51 for $\nu = 1$ in units of the noise variance of a single quantization, $2^{-2B}/12$. The plot is symmetrical about $\theta = 90°$, so only the range from 0° to 90° is shown. Notice that as $r \to 1$, the roundoff noise increases without bound. Also notice that the noise increases as $\theta \to 0°$.

It is possible to design state-space filter realizations that minimize fixed-point roundoff noise [7–10]. Depending on the transfer function being realized, these structures may provide a roundoff noise level that is orders-of-magnitude lower than for a nonoptimal realization. The price paid for this reduction in roundoff noise is an increase in the number of computations required to implement the filter. For an

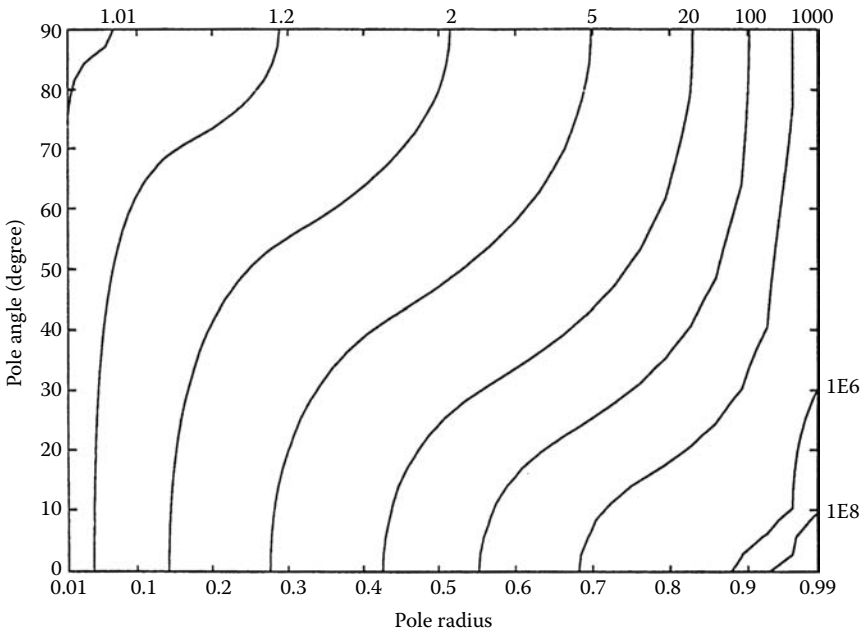

FIGURE 3.1 Normalized fixed-point roundoff noise variance.

Nth-order filter the increase is from roughly $2N$ multiplies for a direct form realization to roughly $(N+1)^2$ for an optimal realization. However, if the filter is realized by the parallel or cascade connection of first- and second-order optimal subfilters, the increase is only to about $4N$ multiplies. Furthermore, near-optimal realizations exist that increase the number of multiplies to only about $3N$ [10].

3.5.3 Roundoff Noise in Floating-Point IIR Filters

For floating-point arithmetic it is first necessary to determine the injected noise variance of each quantization. For the first-order filter this is done by writing the computed output as

$$y(n) + e(n) = [ay(n-1)(1+\varepsilon_1(n)) + x(n)](1+\varepsilon_2(n)) \tag{3.52}$$

where

$\varepsilon_1(n)$ represents the error due to the multiplication
$\varepsilon_2(n)$ represents the error due to the addition

Neglecting the product of errors, Equation 3.52 becomes

$$y(n) + e(n) \approx ay(n-1) + x(n) + ay(n-1)\varepsilon_1(n) + ay(n-1)\varepsilon_2(n) + x(n)\varepsilon_2(n) \tag{3.53}$$

Comparing Equations 3.34 and 3.53, it is clear that

$$e(n) = ay(n-1)\varepsilon_1(n) + ay(n-1)\varepsilon_2(n) + x(n)\varepsilon_2(n) \tag{3.54}$$

Taking the expected value of $e^2(n)$ to obtain the injected noise variance then gives

$$\begin{aligned} E\{e^2(n)\} = {} & a^2E\{y^2(n-1)\}E\{\varepsilon_1^2(n)\} + a^2E\{y^2(n-1)\}E\{\varepsilon_2^2(n)\} \\ & + E\{x^2(n)\}E\{\varepsilon_2^2(n)\} + E\{x(n)y(n-1)\}E\{\varepsilon_2^2(n)\} \end{aligned} \tag{3.55}$$

To carry this further it is necessary to know something about the input. If we assume the input is zero-mean white noise with variance σ_x^2, then $E\{x^2(n)\} = \sigma_x^2$ and the input is uncorrelated with past values of the output so $E\{x(n)y(n-1)\} = 0$ giving

$$E\{e^2(n)\} = 2a^2\sigma_y^2\sigma_{\varepsilon_r}^2 + \sigma_x^2\sigma_{\varepsilon_r}^2 \tag{3.56}$$

and

$$\sigma_o^2 = \left(2a^2\sigma_y^2\sigma_{\varepsilon_r}^2 + \sigma_x^2\sigma_{\varepsilon_r}^2\right)\sum_{n=-\infty}^{\infty} h^2(n) = \frac{2a^2\sigma_y^2 + \sigma_x^2}{1-a^2}\sigma_{\varepsilon_r}^2 \tag{3.57}$$

However,

$$\sigma_y^2 = \sigma_x^2\sum_{n=-\infty}^{\infty} h^2(n) = \frac{\sigma_x^2}{1-a^2} \tag{3.58}$$

so

$$\sigma_o^2 = \frac{1+a^2}{(1-a^2)^2}\sigma_{\varepsilon_r}^2\sigma_x^2 = \frac{1+a^2}{1-a^2}\sigma_{\varepsilon_r}^2\sigma_y^2 \tag{3.59}$$

and the output roundoff noise-to-signal ratio is

$$\frac{\sigma_o^2}{\sigma_y^2} = \frac{1+a^2}{1-a^2}\sigma_{\varepsilon_r}^2 \tag{3.60}$$

Similar results can be obtained for the second-order filter of Equation 3.43 by writing

$$y(n) + e(n) = \left(\left[2r\cos(\theta)y(n-1)(1+\varepsilon_1(n)) - r^2y(n-2)(1+\varepsilon_2(n))\right] \times [1+\varepsilon_3(n)] + x(n)\right)(1+\varepsilon_4(n)) \tag{3.61}$$

Expanding with the same assumptions as before gives

$$\begin{aligned} e(n) \approx\ & 2r\cos(\theta)y(n-1)[\varepsilon_1(n) + \varepsilon_3(n) + \varepsilon_4(n)] \\ & - r^2y(n-2)[\varepsilon_2(n) + \varepsilon_3(n) + \varepsilon_4(n)] + x(n)\varepsilon_4(n) \end{aligned} \tag{3.62}$$

and

$$\begin{aligned} E\{e^2(n)\} =\ & 4r^2\cos^2(\theta)\sigma_y^2 3\sigma_{\varepsilon_r}^2 + r^2\sigma_y^2 3\sigma_{\varepsilon_r}^2 \\ & + \sigma_x^2\sigma_{\varepsilon_r}^2 - 8r^3\cos(\theta)\sigma_{\varepsilon_r}^2 E\{y(n-1)y(n-2)\} \end{aligned} \tag{3.63}$$

However,

$$\begin{aligned} E\{y(n-1)y(n-2)\} &= E\left\{\left[2r\ \cos(\theta)y(n-2) - r^2y(n-3) + x(n-1)\right]y(n-2)\right\} \\ &= 2r\ \cos(\theta)E\left\{y^2(n-2)\right\} - r^2E\{y(n-2)y(n-3)\} \\ &= 2r\ \cos(\theta)E\left\{y^2(n-2)\right\} - r^2E\{y(n-1)y(n-2)\} \\ &= \frac{2r\ \cos(\theta)}{1+r^2}\sigma_y^2 \end{aligned} \tag{3.64}$$

so

$$E\{e^2(n)\} = \sigma_{\varepsilon_r}^2 \sigma_x^2 + \left[3r^2 + 12r^2\cos^2(\theta) - \frac{16r^4\cos^2(\theta)}{1+r^2}\right]\sigma_{\varepsilon_r}^2 \sigma_y^2 \tag{3.65}$$

and

$$\sigma_o^2 = E\{e^2(n)\} \sum_{n=-\infty}^{\infty} h^2(n)\xi\left[\sigma_{\varepsilon_r}^2 \sigma_x^2 + \left[3r^4 + 12r^2\cos^2(\theta) - \frac{16r^4\cos^2(\theta)}{1+r^2}\right]\sigma_{\varepsilon_r}^2 \sigma_y^2\right] \tag{3.66}$$

where from Equation 3.46,

$$\xi = \sum_{n=-\infty}^{\infty} h^2(n) = \frac{1+r^2}{1-r^2}\frac{1}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \tag{3.67}$$

Since $\sigma_y^2 = \xi\sigma_x^2$, the output roundoff noise-to-signal ratio is then

$$\frac{\sigma_o^2}{\sigma_y^2} = \xi\left[1 + \xi\left[3r^2 + 12r^2\cos^2(\theta) - \frac{16r^4\cos^2(\theta)}{1+r^2}\right]\right]\sigma_{\varepsilon_r}^2 \tag{3.68}$$

Figure 3.2 is a contour plot showing the noise-to-signal ratio of Equation 3.68 in units of the noise variance of a single quantization $\sigma_{\varepsilon_r}^2$. The plot is symmetrical about $\theta = 90°$, so only the range from 0° to 90° is shown. Notice the similarity of this plot to that of Figure 3.1 for the fixed-point case. It has been observed that filter structures generally have very similar fixed-point and floating-point roundoff characteristics [2]. Therefore, the techniques of [7–10], which were developed for the fixed-point case,

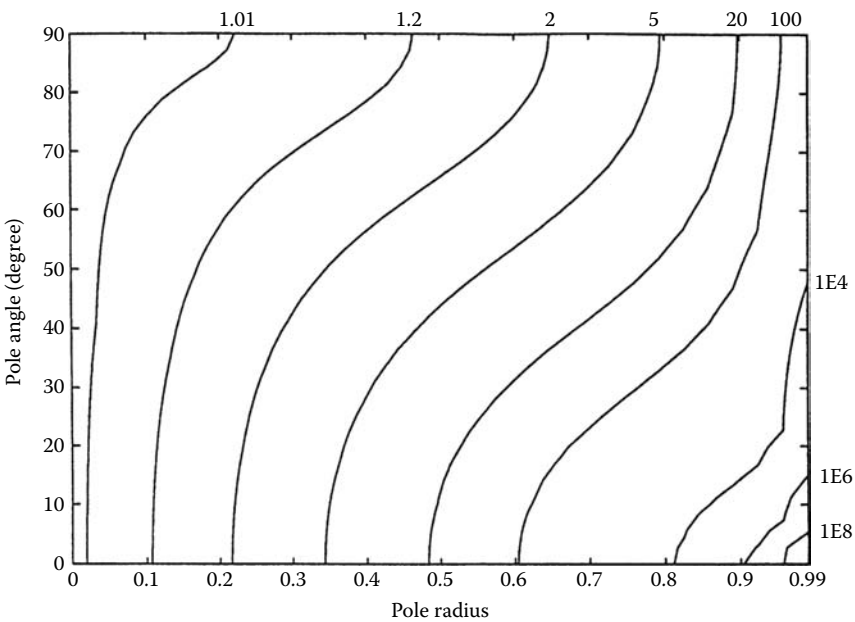

FIGURE 3.2 Normalized floating-point roundoff noise variance.

can also be used to design low-noise floating-point filter realizations. Furthermore, since it is not necessary to scale the floating-point realization, the low-noise realizations need not require significantly more computation than the direct form realization.

3.6 Limit Cycles

A limit cycle, sometimes referred to as a **multiplier roundoff limit cycle**, is a low-level oscillation that can exist in an otherwise stable filter as a result of the nonlinearity associated with rounding (or truncating) internal filter calculations [11]. Limit cycles require recursion to exist and do not occur in nonrecursive FIR filters.

As an example of a limit cycle, consider the second-order filter realized by

$$y(n) = Q_r\left\{\frac{7}{8}y(n-1) - \frac{5}{8}y(n-2) + x(n)\right\} \tag{3.69}$$

where $Q_r\{\}$ represents quantization by rounding. This is stable filter with poles at $0.4375 \pm j0.6585$. Consider the implementation of this filter with 4-bit (3-bit and a sign bit) two's complement fixed-point arithmetic, zero initial conditions ($y(-1) = y(-2) = 0$), and an input sequence $x(n) = \frac{3}{8}\delta(n)$, where $\delta(n)$ is the unit impulse or unit sample. The following sequence is obtained:

$$\begin{aligned}
y(0) &= Q_r\left\{\frac{3}{8}\right\} = \frac{3}{8}\\
y(1) &= Q_r\left\{\frac{21}{64}\right\} = \frac{3}{8}\\
y(2) &= Q_r\left\{\frac{3}{32}\right\} = \frac{1}{8}\\
y(3) &= Q_r\left\{-\frac{1}{8}\right\} = -\frac{1}{8}\\
y(4) &= Q_r\left\{-\frac{3}{16}\right\} = -\frac{1}{8}\\
y(5) &= Q_r\left\{-\frac{1}{32}\right\} = 0\\
y(6) &= Q_r\left\{\frac{5}{64}\right\} = \frac{1}{8}\\
y(7) &= Q_r\left\{\frac{7}{64}\right\} = \frac{1}{8}\\
y(8) &= Q_r\left\{\frac{1}{32}\right\} = 0\\
y(9) &= Q_r\left\{-\frac{5}{64}\right\} = -\frac{1}{8}\\
y(10) &= Q_r\left\{-\frac{7}{64}\right\} = -\frac{1}{8}\\
y(11) &= Q_r\left\{-\frac{1}{32}\right\} = 0\\
y(12) &= Q_r\left\{\frac{5}{64}\right\} = \frac{1}{8}
\end{aligned} \tag{3.70}$$

Notice that while the input is zero except for the first sample, the output oscillates with amplitude 1/8 and period 6.

Limit cycles are primarily of concern in fixed-point recursive filters. As long as floating-point filters are realized as the parallel or cascade connection of first- and second-order subfilters, limit cycles will generally not be a problem since limit cycles are practically not observable in first- and second-order systems implemented with 32-bit floating-point arithmetic [12]. It has been shown that such systems must have an extremely small margin of stability for limit cycles to exist at anything other than underflow levels, which are at an amplitude of less than 10^{-38} [12].

There are at least three ways of dealing with limit cycles when fixed-point arithmetic is used. One is to determine a bound on the maximum limit cycle amplitude, expressed as an integral number of quantization steps [13]. It is then possible to choose a wordlength that makes the limit cycle amplitude acceptably low. Alternately, limit cycles can be prevented by randomly rounding calculations up or down [14]. However, this approach is complicated to implement. The third approach is to properly choose the filter realization structure and then quantize the filter calculations using magnitude truncation [15,16]. This approach has the disadvantage of producing more roundoff noise than truncation or rounding (see Equations 3.12 through 3.14).

3.7 Overflow Oscillations

With fixed-point arithmetic it is possible for filter calculations to overflow. This happens when two numbers of the same sign add to give a value having magnitude greater than one. Since numbers with magnitude greater than one are not representable, the result overflows. For example, the two's complement numbers 0.101 (5/8) and 0.100 (4/8) add to give 1.001 which is the two's complement representation of −7/8.

The overflow characteristic of two's complement arithmetic can be represented as $R\{\}$ where

$$R\{X\} = \begin{cases} X-2 & X \geq 1 \\ X & -1 \leq X < 1 \\ X+2 & X < -1 \end{cases} \tag{3.71}$$

For the example just considered, $R\{9/8\} = -7/8$.

An overflow oscillation, sometimes also referred to as an adder overflow limit cycle, is a high-level oscillation that can exist in an otherwise stable fixed-point filter due to the gross nonlinearity associated with the overflow of internal filter calculations [17]. Like limit cycles, overflow oscillations require recursion to exist and do not occur in nonrecursive FIR filters. Overflow oscillations also do not occur with floating-point arithmetic due to the virtual impossibility of overflow.

As an example of an overflow oscillation, once again consider the filter of Equation 3.69 with 4-bit fixed-point two's complement arithmetic and with the two's complement overflow characteristic of Equation 3.71:

$$y(n) = Q_r\left\{R\left[\frac{7}{8}y(n-1) - \frac{5}{8}y(n-2) + x(n)\right]\right\} \tag{3.72}$$

In this case, we apply the input

$$\begin{aligned} x(n) &= -\frac{3}{4}\delta(n) - \frac{5}{8}\delta(n-1) \\ &= \left\{-\frac{3}{4}, -\frac{5}{8}, 0, 0, \ldots\right\} \end{aligned} \tag{3.73}$$

giving the output sequence

$$\begin{aligned}
y(0) &= Q_r\left\{R\left[-\frac{3}{4}\right]\right\} = Q_r\left\{-\frac{3}{4}\right\} = -\frac{3}{4}\\
y(1) &= Q_r\left\{R\left[-\frac{41}{32}\right]\right\} = Q_r\left\{\frac{23}{32}\right\} = \frac{3}{4}\\
y(2) &= Q_r\left\{R\left[\frac{9}{8}\right]\right\} = Q_r\left\{-\frac{7}{8}\right\} = -\frac{7}{8}\\
y(3) &= Q_r\left\{R\left[-\frac{79}{64}\right]\right\} = Q_r\left\{\frac{49}{64}\right\} = \frac{3}{4}\\
y(4) &= Q_r\left\{R\left[\frac{77}{64}\right]\right\} = Q_r\left\{-\frac{51}{64}\right\} = -\frac{3}{4}\\
y(5) &= Q_r\left\{R\left[-\frac{9}{8}\right]\right\} = Q_r\left\{\frac{7}{8}\right\} = \frac{7}{8}\\
y(6) &= Q_r\left\{R\left[\frac{79}{64}\right]\right\} = Q_r\left\{-\frac{49}{64}\right\} = -\frac{3}{4}\\
y(7) &= Q_r\left\{R\left[-\frac{77}{64}\right]\right\} = Q_r\left\{\frac{51}{64}\right\} = \frac{3}{4}\\
y(8) &= Q_r\left\{R\left[\frac{9}{8}\right]\right\} = Q_r\left\{-\frac{7}{8}\right\} = -\frac{7}{8}
\end{aligned} \tag{3.74}$$

This is a large-scale oscillation with nearly full-scale amplitude.

There are several ways to prevent overflow oscillations in fixed-point filter realizations. The most obvious is to scale the filter calculations so as to render overflow impossible. However, this may unacceptably restrict the filter dynamic range. Another method is to force completed sums-of-products to saturate at ± 1, rather than overflowing [18,19]. It is important to saturate only the completed sum, since intermediate overflows in two's complement arithmetic do not affect the accuracy of the final result. Most fixed-point digital signal processors provide for automatic saturation of completed sums if their saturation arithmetic feature is enabled. Yet another way to avoid overflow oscillations is to use a filter structure for which any internal filter transient is guaranteed to decay to zero [20]. Such structures are desirable anyway, since they tend to have low roundoff noise and be insensitive to coefficient quantization [21].

3.8 Coefficient Quantization Error

Each filter structure has its own finite, generally nonuniform grids of realizable pole and zero locations when the filter coefficients are quantized to a finite wordlength. In general the pole and zero locations desired in filter do not correspond exactly to the realizable locations. The error in filter performance (usually measured in terms of a frequency response error) resulting from the placement of the poles and zeros at the nonideal but realizable locations is referred to as coefficient quantization error.

Consider the second-order filter with complex-conjugate poles

$$\begin{aligned}
\lambda &= re^{\pm j\theta}\\
&= \lambda_r \pm j\lambda_i\\
&= r\cos(\theta) \pm jr\sin(\theta)
\end{aligned} \tag{3.75}$$

and transfer function

$$H(z) = \frac{1}{1 - 2r\cos(\theta)z^{-1} + r^2 z^{-2}} \tag{3.76}$$

realized by the difference equation

$$y(n) = 2r\cos(\theta)y(n-1) - r^2 y(n-2) + x(n) \tag{3.77}$$

Figure 3.3 from [5] shows that quantizing the difference equation coefficients results in a nonuniform grid of realizable pole locations in the z-plane. The grid is defined by the intersection of vertical lines corresponding to quantization of $2\lambda_r$ and concentric circles corresponding to quantization of $-r^2$. The sparseness of realizable pole locations near $Z = \pm 1$ will result in a large coefficient quantization error for poles in this region.

Figure 3.4 gives an alternative structure to Equation 3.77 for realizing the transfer function of Equation 3.76. Notice that quantizing the coefficients of this structure corresponds to quantizing λ_r and λ_i. As shown in Figure 3.5 from [5], this results in a uniform grid of realizable pole locations. Therefore, large coefficient quantization errors are avoided for all pole locations.

It is well established that filter structures with low roundoff noise tend to be robust to coefficient quantization, and visa versa [22–24]. For this reason, the uniform grid structure of Figure 3.4 is also popular because of its low roundoff noise. Likewise, the low-noise realizations of [7–10] can be expected to be relatively insensitive to coefficient quantization, and digital wave filters and lattice filters that are derived from low-sensitivity analog structures tend to have not only low coefficient sensitivity, but also low roundoff noise [25,26].

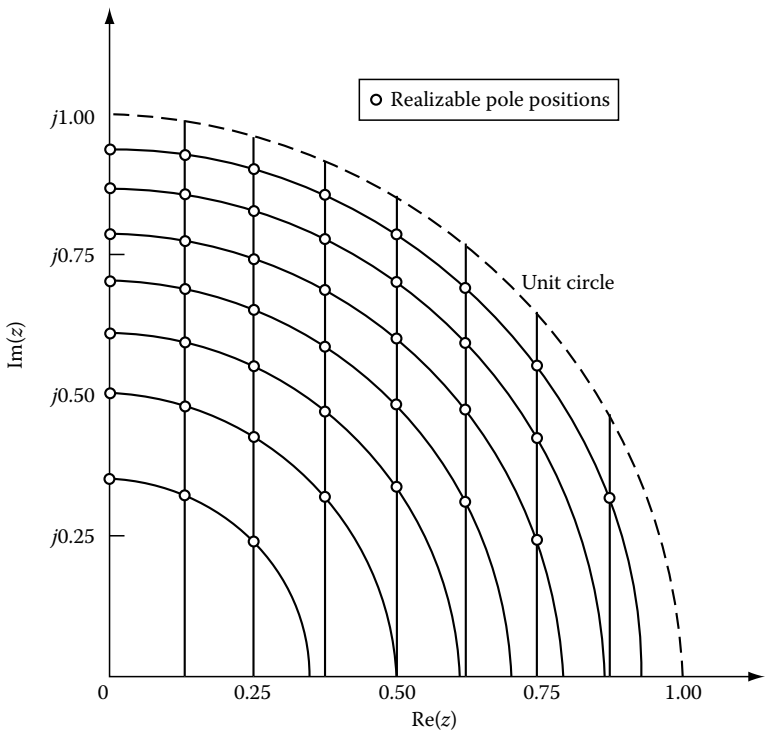

FIGURE 3.3 Realizable pole locations for the difference equation of Equation 3.76.