Dr. Alan A. Desrochers of Rensselaer Polytechnic Institute, Troy, New York; Dr. Omar Farooq of Aligarh Muslim University, India; Dr. Rajendra S. Gad of Goa University, India; Dr. Vikram M. Gadre of Indian Institute of Technology, Bombay; Dr. E. Gopinathan of National Institute of Technology, Calicut, India; Dr. Lim Heng-Siong of Multimedia University, Malayasia; Dr. Hamid Gholam Hosseini of Auckland University of Technology, New Zealand; Dr. J. Abdul Jaleel of Thangal Kunju Musaliar College of Engineering, Kerala, India: Dr. Sabira Khatun of University Putra Malaysia; Dr. Ju Liu of Shandong University, China; Dr. Wagdy H. Mahmoud of University of the District of Columbia; Dr. Ashutosh Marathe of Vishwakarma Institute of Technology, India; Dr. S.N. Merchant of Indian Institute of Technology, Bombay; Dr. Muhammad Javed Mirza of Riphah International University, Pakistan; Dr. Ravinder Nath of National Institute of Technology, Hamirpur, India; Dr. K.M.M. Prabhu of Indian Institute of Technology, Madras; Dr. S.M. Sameer of National Institute of Technology, Calicut, India; Dr. O.P. Sahu of National Institute of Technology, Kurukshetra, India; Dr. Mansour Tahernezhadi of Northern Illinois University; Dr. Nisachon Tangsangiumvisai of Chulalongkorn University, Thailand; and Dr. Imran A. Tasadduq of FAST-National University of Computer & Emerging Sciences, Karachi, Pakistan.

The manuscript of the fourth edition was reviewed by Dr. Wolfgang F.G. Mecklenbräuker of Technical University of Vienna, Austria, and Dr. Truong Nguyen of University of California, San Diego. Some parts of the manuscript of the fourth edition were reviewed by Dr. Suhash Dutta Roy of Indian Institute of Technology, New Delhi.

I thank all of them for their valuable comments, which have improved the book tremendously.

Many of my former research students reviewed various portions of the manuscript of all editions and tested a number of the MATLAB programs. In particular, I would like to thank Drs. Charles D. Creusere, Rajeev Gandhi, Gabriel Gomes, Serkan Hatipoglu, Zhihai He, Hsin-Han Ho, Michael Lightstone, Ing-Song Lin, Luca Lucchese, Michael Moore, Debargha Mukherjee, Norbert Strobel, Stefan Thurnhofer, and Mylene Queiroz de Farias, and Mr. Eric Leipnik. I am also indebted to all former students in my ECE 158 and ECE 258A classes at the University of California, Santa Barbara, and all former students in my class EE 483 at the University of Southern California, Los Angeles for their feedback over the years, which helped refine the book.

I thank Goutam K. Mitra and Alicia Rodriguez for the cover design of the book. Finally, I thank Patricia Monohon for her outstanding assistance in the preparation of the LaTeX files of the fourth edition.

Supplements

All MATLAB programs included in this book are in the CD accompanying this book and are also available from the Internet site **www.ece.ucsb.edu/Faculty/Mitra/Book4e**.

A solutions manual prepared by Hsin-Han Ho, Travis Smith, and Martin Gawecki and containing the solutions to all problems and MATLAB exercises is available to instructors from the publisher. PowerPoint slides of most materials of this book are available to instructors from the author.

Table of Contents

Preface ix
Acknowledgements xiii

1 Signals and Signal Processing 1

1.1 Characterization and Classification of Signals 1
1.2 Typical Signal Processing Operations 4
1.3 Examples of Typical Signals 13
1.4 Typical Signal Processing Applications 21
1.5 Why Digital Signal Processing? 37

2 Discrete-Time Signals in the Time Domain 41

2.1 Time-Domain Representation 42
2.2 Operations on Sequences 46
2.3 Operations on Finite-Length Sequences 55
2.4 Typical Sequences and Sequence Representation 62
2.5 The Sampling Process 72
2.6 Correlation of Signals 74
2.7 Random Signals 80
2.8 Summary 81
2.9 Problems 81
2.10 MATLAB Exercises 87

3 Discrete-Time Signals in the Frequency Domain 89

3.1 The Continuous-Time Fourier Transform 89
3.3 Discrete-Time Fourier Transform Theorems 105
3.4 Energy Density Spectrum of a Discrete-Time Sequence 111
3.5 Band-Limited Discrete-Time Signals 112
3.6 DTFT Computation Using MATLAB 113
3.7 The Unwrapped Phase Function 113
3.8 Digital Processing of Continuous-Time Signals 115

3.9 Sampling of Bandpass Signals 129
3.10 Effect of Sample-and-Hold Operation 131
3.11 Summary 132
3.12 Problems 133
3.13 MATLAB Exercises 142

4 Discrete-Time Systems 143

4.1 Discrete-Time System Examples 143
4.2 Classification of Discrete-Time Systems 149
4.3 Impulse and Step Responses 153
4.4 Time-Domain Characterization of LTI Discrete-Time Systems 154
4.5 Simple Interconnection Schemes 161
4.6 Finite-Dimensional LTI Discrete-Time Systems 164
4.7 Classification of LTI Discrete-Time Systems 172
4.8 Frequency-Domain Representations of LTI Discrete-Time Systems 175
4.9 Phase and Group Delays 185
4.10 Summary 189
4.11 Problems 190
4.12 MATLAB Exercises 198

5 Finite-Length Discrete Transforms 199

5.1 Orthogonal Transforms 199
5.2 The Discrete Fourier Transform 201
5.3 Relation Between the DTFT and the DFT and Their Inverses 205
5.4 Circular Convolution 211
5.5 Classifications of Finite-Length Sequences 216
5.6 DFT Symmetry Relations 221
5.7 Discrete Fourier Transform Theorems 224
5.8 Fourier-Domain Filtering 230
5.9 Computation of the DFT of Real Sequences 232
5.10 Linear Convolution Using the DFT 234
5.11 Short-Time Fourier Transform 245
5.12 Discrete Cosine Transform 249
5.13 The Haar Transform 256
5.14 Energy Compaction Properties 259
5.15 Summary 262
5.16 Problems 262

5.17 MATLAB Exercises 275

6 z-Transform 277

6.1 Definition 277
6.2 Rational z-Transforms 281
6.3 Region of Convergence of a Rational z-Transform 283
6.4 The Inverse z-Transform 289
6.5 z-Transform Theorems 297
6.6 Computation of the Convolution Sum of Finite-Length Sequences 305
6.7 The Transfer Function 308
6.8 Summary 320
6.9 Problems 320
6.10 MATLAB Exercises 332

7 LTI Discrete-Time Systems in the Transform Domain 333

7.1 Transfer Function Classification Based on Magnitude Characteristics 333
7.2 Transfer Function Classification Based on Phase Characteristics 342
7.3 Types of Linear-Phase FIR Transfer Functions 349
7.4 Simple Digital Filters 360
7.5 Complementary Transfer Functions 379
7.6 Inverse Systems 385
7.7 System Identification 389
7.8 Digital Two-Pairs 392
7.9 Algebraic Stability Test 394
7.10 Summary 399
7.11 Problems 400
7.12 MATLAB Exercises 414

8 Digital Filter Structures 417

8.1 Block Diagram Representation 418
8.2 Equivalent Structures 421
8.3 Basic FIR Digital Filter Structures 422
8.4 Basic IIR Digital Filter Structures 427
8.5 Realization of Basic Structures Using MATLAB 433
8.6 Allpass Filters 436
8.7 Parametrically Tunable Low-Order IIR Digital Filter Pairs 445
8.8 IIR Tapped Cascaded Lattice Structures 447

8.9 FIR Cascaded Lattice Structures 452
8.10 Parallel Allpass Realization of IIR Transfer Functions 460
8.11 Tunable High-Order Digital Filters 465
8.12 Computational Complexity of Digital Filter Structures 472
8.13 Summary 472
8.14 Problems 473
8.15 MATLAB Exercises 487

9 IIR Digital Filter Design 489

9.1 Preliminary Considerations 489
9.2 Bilinear Transformation Method of IIR Filter Design 494
9.3 Design of Lowpass IIR Digital Filters 499
9.4 Design of Highpass, Bandpass, and Bandstop IIR Digital Filters 501
9.5 Spectral Transformations of IIR Filters 505
9.6 IIR Digital Filter Design Using MATLAB 512
9.7 Computer-Aided Design of IIR Digital Filters 515
9.8 Summary 519
9.9 Problems 519
9.10 MATLAB Exercises 525

10 FIR Digital Filter Design 527

10.1 Preliminary Considerations 527
10.2 FIR Filter Design Based on Windowed Fourier Series 531
10.3 Computer-Aided Design of Equiripple Linear-Phase FIR Filters 546
10.4 Design of Minimum-Phase FIR Filters 555
10.5 FIR Digital Filter Design Using MATLAB 556
10.6 Design of Computationally Efficient FIR Digital Filters 573
10.7 Summary 586
10.8 Problems 587
10.9 MATLAB Exercises 594

11 DSP Algorithm Implementation 599

11.1 Basic Issues 599
11.2 Structure Simulation and Verification Using MATLAB 610
11.3 Computation of the Discrete Fourier Transform 617
11.4 Fast DFT Algorithms Based on Index Mapping 632
11.5 DFT and IDFT Computation Using MATLAB 640

11.6 Sliding Discrete Fourier Transform 642
11.7 DFT Computation over a Narrow Frequency Band 642
11.8 Number Representation 647
11.9 Handling of Overflow 652
11.10 Summary 653
11.11 Problems 653
11.12 MATLAB Exercises 661

12 Analysis of Finite Wordlength Effects 663

12.1 The Quantization Process and Errors 664
12.2 Quantization of Fixed-Point Numbers 665
12.3 Quantization of Floating-Point Numbers 668
12.4 Analysis of Coefficient Quantization Effects 668
12.5 A/D Conversion Noise Analysis 681
12.6 Analysis of Arithmetic Round-Off Errors 691
12.7 Dynamic Range Scaling 695
12.8 Signal-to-Noise Ratio in Low-Order IIR Filters 706
12.9 Low-Sensitivity Digital Filters 710
12.10 Reduction of Product Round-Off Noise Using Error Feedback 716
12.11 Limit Cycles in IIR Digital Filters 719
12.12 Round-Off Errors in FFT Algorithms 727
12.13 Summary 730
12.14 Problems 731
12.15 MATLAB Exercises 736

13 Multirate Digital Signal Processing Fundamentals 739

13.1 The Basic Sampling Rate Alteration Devices 740
13.2 Multirate Structures for Sampling Rate Conversion 750
13.3 Multistage Design of Decimator and Interpolator 758
13.4 The Polyphase Decomposition 760
13.5 Arbitrary-Rate Sampling Rate Converter 771
13.6 Nyquist Filters 783
13.7 CIC Decimators and Interpolators 792
13.8 Summary 796
13.9 Problems 797
13.10 MATLAB Exercises 805

14 Multirate Filter Banks and Wavelets 807

14.1 Digital Filter Banks 807
14.2 Two-Channel Quadrature-Mirror Filter Bank 813
14.3 Perfect Reconstruction Two-Channel FIR Filter Banks 823
14.4 *L*-Channel QMF Banks 832
14.5 Multilevel Filter Banks 840
14.6 Discrete Wavelet Transform 844
14.7 Summary 853
14.8 Problems 853
14.9 MATLAB Exercises 861

A Analog Lowpass Filter Design 863

A.1 Analog Filter Specifications 863
A.2 Butterworth Approximation 865
A.3 Chebyshev Approximation 867
A.4 Elliptic Approximation 870
A.5 Linear-Phase Approximation 871
A.6 Analog Filter Design Using MATLAB 872
A.7 Analog Lowpass Filter Design Examples 875
A.8 A Comparison of the Filter Types 877
A.9 Anti-Aliasing Filter Design 880
A.10 Reconstruction Filter Design 882

B Design of Analog Highpass, Bandpass, and Bandstop Filters 887

B.1 Analog Highpass Filter Design 887
B.2 Analog Bandpass Filter Design 889
B.3 Analog Bandstop Filter Design 892

C Discrete-Time Random Signals 893

C.1 Statistical Properties of a Random Variable 893
C.2 Statistical Properties of a Random Signal 895
C.3 Wide-Sense Stationary Random Signal 896
C.4 Concept of Power in a Random Signal 897
C.5 Ergodic Signal 898
C.6 Transform-Domain Representations of Random Signals 899
C.7 White Noise 901
C.8 Discrete-Time Processing of Random Signals 901

Bibliography 907
Index 927

Chapter 1
Signals and Signal Processing

Signals play an important role in our daily lives. Examples of signals that we encounter frequently are speech, music, picture, and video signals. A signal is a function of independent variables such as time, distance, position, temperature, and pressure. For example, speech and music signals represent air pressure as a function of time at a point in space. A black-and-white picture is a representation of light intensity as a function of two spatial coordinates. The video signal in television consists of a sequence of images, called frames, and is a function of three variables: two spatial coordinates and time.

Most signals we encounter are generated by natural means. However, a signal can also be generated synthetically or by computer simulation. A signal carries information, and the objective of signal processing is to extract useful information carried by the signal. The method of information extraction depends on the type of signal and the nature of the information being carried by the signal. Thus, roughly speaking, signal processing is concerned with the mathematical representation of the signal and the algorithmic operation carried out on it to extract the information present. The representation of the signal can be in terms of basis functions in the domain of the original independent variable(s), or it can be in terms of basis functions in a transformed domain. Likewise, the information extraction process may be carried out in the original domain of the signal or in a transformed domain. This book is concerned with discrete-time representation of signals and their discrete-time processing.

This chapter provides an overview of signals and signal processing methods. The mathematical characterization of the signal is first discussed along with a classification of signals. Next, some typical signals are discussed in detail, and the type of information carried by them is described. Then, a review of some commonly used signal processing operations is provided and illustrated through examples. A brief review of some typical signal processing applications is discussed next. Finally, the advantages and disadvantages of digital processing of signals are discussed.

1.1 Characterization and Classification of Signals

Depending on the nature of the independent variables and the value of the function defining the signal, various types of signals can be defined. For example, independent variables can be continuous or discrete. Likewise, the signal can either be a continuous or a discrete function of the independent variables. Moreover, the signal can be either a real-valued function or a complex-valued function.

A signal can be generated by a single source or by multiple sources. In the former case, it is a scalar signal, and in the latter case, it is a vector signal, often called a multichannel signal. A one-dimensional (1-D) signal is a function of a single independent variable. A two-dimensional (2-D) signal is a function of two independent variables. A multidimensional (M-D) signal is a function of more than one variable. The speech signal is an example of a 1-D signal where the independent variable is time. An image signal, such

as a photograph, is an example of a 2-D signal where the two independent variables are the two spatial variables. Each frame of a black-and-white video signal is a 2-D image signal that is a function of two discrete spatial variables, with each frame occurring sequentially at discrete instants of time. Hence, the black-and-white video signal can be considered an example of a three-dimensional (3-D) signal where the three independent variables are the two spatial variables and time. A color video signal is a three-channel signal composed of three 3-D signals representing the three primary colors: red, green, and blue (RGB). For transmission purposes, the RGB television signal is transformed into another type of three-channel signal composed of a luminance component and two chrominance components.

The value of the signal at a specific value of the independent variable is called its *amplitude*. The variation of the amplitude as a function of the independent variable is called its *waveform*.

For a 1-D signal, the independent variable is usually labeled as *time*. If the independent variable is continuous, the signal is called a *continuous-time signal*. If the independent variable is discrete, the signal is called a *discrete-time signal*. A continuous-time signal is defined at every instant of time. On the other hand, a discrete-time signal takes certain numerical values at specified discrete instants of time, and between these specified instants of time, the signal is not defined. Hence, a discrete-time signal is basically a sequence of numbers.

Speech Demo 1

A continuous-time signal with a continuous amplitude is usually called an *analog signal*. A speech signal is an example of an analog signal. Analog signals are commonly encountered in our daily lives and are usually generated by natural means. A discrete-time signal with discrete-valued amplitudes represented by a finite number of digits is referred to as a *digital signal*. An example of a digital signal is the digitized music signal stored in a CD-ROM disk. A discrete-time signal with continuous-valued amplitudes is called a *sampled-data signal* This last type of signal occurs in switched-capacitor (SC) circuits. A digital signal is thus a quantized sampled-data signal. Finally, a continuous-time signal with discrete-valued amplitudes has been referred to as a *quantized boxcar signal* [Ste93]. The latter type of signals occurs in digital electronic circuits where the signal is kept at fixed level (usually one of two values) between two instants of clocking. Figure 1.1 illustrates the four types of signals.

The functional dependence of a signal in its mathematical representation is often explicitly shown. For a continuous-time 1-D signal, the continuous independent variable is usually denoted by t, whereas for a discrete-time 1-D signal, the discrete independent variable is usually denoted by n. For example, $u(t)$ represents a continuous-time 1-D signal and $\{v[n]\}$ represents a discrete-time 1-D signal. Each member, $v[n]$, of a discrete-time signal is called a *sample*. In many applications, a discrete-time signal is generated from a parent continuous-time signal by sampling the latter at uniform intervals of time. If the discrete instants of time at which a discrete-time signal is defined are uniformly spaced, the independent discrete variable n can be normalized to assume integer values.

Image Demo 1
Video Demo 1

In the case of a continuous-time 2-D signal, the two independent variables are usually the spatial coordinates, which are usually denoted by x and y. For example, the intensity of a black-and-white image can be expressed as $u(x, y)$. A color image $u(x, y)$, is composed of three signals representing the three primary colors, red, green, and blue:

$$\mathbf{u}(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}.$$

On the other hand, a digitized image is a 2-D discrete signal, and its two independent variables are discretized spatial variables often denoted by m and n. Hence, a digitized image can be represented as $v[m, n]$. Likewise, a black-and-white video sequence is a 3-D signal and can be represented as $u(x, y, t)$, where x and y denote the two spatial variables and t denotes the temporal variable time. A color video

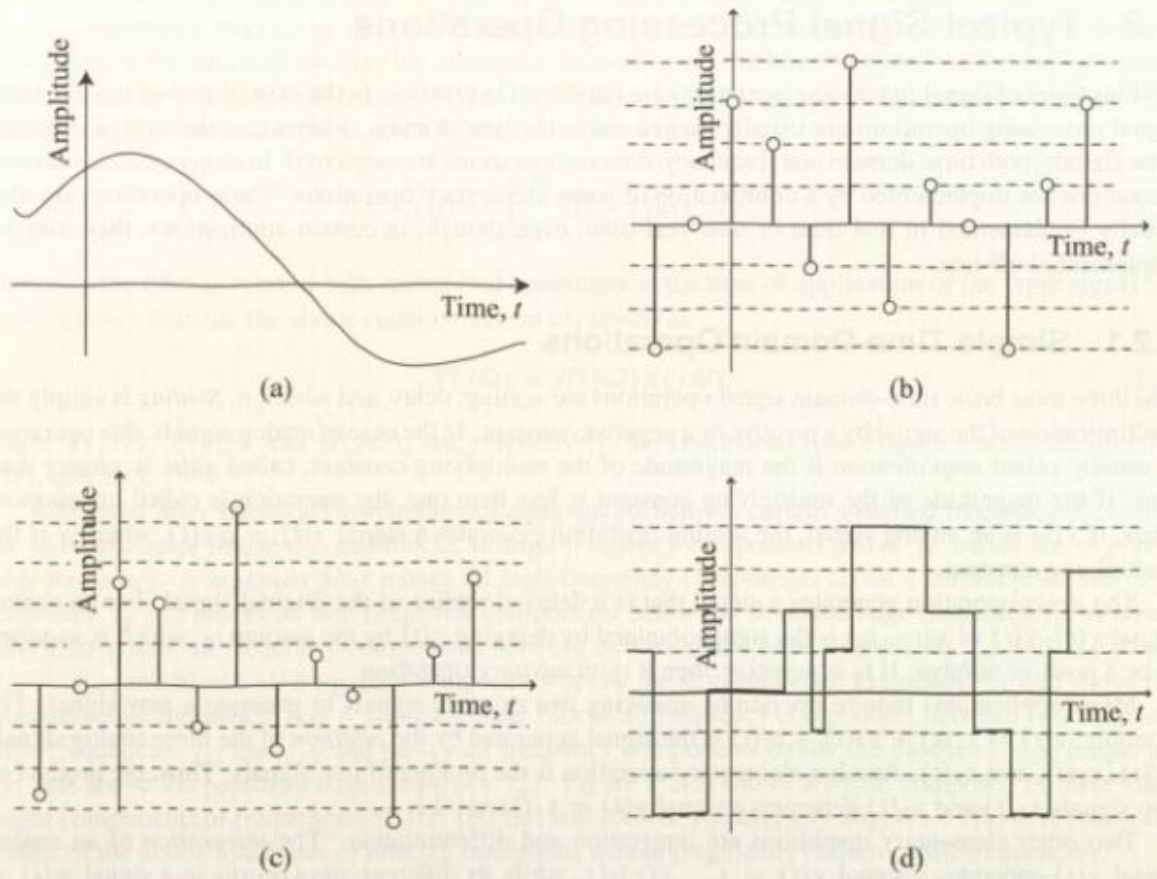


Figure 1.1: (a) An analog signal, (b) a digital signal, (c) a sampled-data signal, and (d) a quantized boxcar signal.

signal is a vector signal composed of three video signals representing the three primary colors, red, green, and blue.

There is another classification of signals that depends on the certainty by which the signal can be uniquely described. A signal that can be uniquely determined by a well-defined process such as a mathematical expression or rule, or table look-up, is called a *deterministic signal*. A signal that is generated in a random fashion and cannot be predicted ahead of time is called a *random signal*. In this text, we are primarily concerned with the processing of discrete-time deterministic signals. However, since practical discrete-time systems employ finite wordlengths for the storing of signals and the implementation of the signal processing algorithms, it is necessary to develop tools for the analysis of finite wordlength effects on the performance of discrete-time systems. To this end, it has been found convenient to represent certain pertinent signals as random signals and employ statistical techniques for their analysis.

Some typical signal processing operations performed on analog signals are reviewed in the following section.

1.2 Typical Signal Processing Operations

Various types of signal processing operations are employed in practice. In the case of analog signals, most signal processing operations are usually carried out in the time domain, whereas, in the case of discrete-time signals, both time-domain and frequency-domain operations are employed. In either case, the desired operations are implemented by a combination of some elementary operations. These operations are also usually implemented in real-time or near real-time, even though, in certain applications, they may be implemented off-line.

1.2.1 Simple Time-Domain Operations

The three most basic time-domain signal operations are scaling, delay, and addition. *Scaling* is simply the multiplication of the signal by a positive or a negative constant. In the case of analog signals, this operation is usually called *amplification* if the magnitude of the multiplying constant, called *gain*, is greater than one. If the magnitude of the multiplying constant is less than one, the operation is called *attenuation*. Thus, if $x(t)$ is an analog signal, the scaling operation generates a signal $y(t) = \alpha x(t)$, where α is the multiplying constant.

The *delay* operation generates a signal that is a delayed replica of the original signal. For an analog signal $x(t)$, $y(t) = x(t - t_0)$ is the signal obtained by delaying $x(t)$ by the amount t_0, which is assumed to be a positive number. If t_0 is negative, then it is an *advance* operation.

Many applications require operations involving two or more signals to generate a new signal. For example, $y(t) = x_1(t) + x_2(t) - x_3(t)$ is the signal generated by the *addition* of the three analog signals $x_1(t)$, $x_2(t)$, and $x_3(t)$. Another elementary operation is the *product* of two signals. Thus, the product of two signals $x_1(t)$ and $x_2(t)$ generates a signal $y(t) = x_1(t)\,x_2(t)$.

Two other elementary operations are integration and differentiation. The *integration* of an analog signal $x(t)$ generates a signal $y(t) = \int_{-\infty}^{t} x(\tau)\,d\tau$, while its *differentiation* results in a signal $w(t) = dx(t)/dt$.

The first three elementary operations, namely, scaling, delay, and addition, are also carried out on discrete-time signals and are discussed further in later parts of this text. The other two elementary operations, namely, integration and differentiation, are implemented approximately in the discrete-time domain.

Next we review some commonly used complex signal processing operations that are implemented by combining two or more of the elementary operations. The characteristics of some of these operations are better understood in the frequency domain by making use of the continuous-time Fourier transform. The continuous-time Fourier transform $X(j\Omega)$ of a continuous-time signal $x(t)$ is given by[1]

$$X(j\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t}\,dt. \tag{1.1}$$

$X(j\Omega)$ is called the spectrum of $x(t)$.

1.2.2 Filtering

One of the most widely used complex signal processing operations is *filtering*, whose main objective is to alter the spectrum according to some given specifications. The system implementing this operation is called a *filter*. For example, the filter may be designed to pass certain frequency components in a signal through the system and to block other frequency components. The range of frequencies of the signal

[1] See Section 3.1 for a review of the continuous-time Fourier transform.

components allowed to pass through the filter is called the *passband*, and the range of frequencies of the signal components blocked by the filter is called the *stopband*. Various types of filters can be defined, depending on the nature of the filtering operation. In most cases, the filtering operation for analog signals is performed by a linear, time-invariant filter. If the filter is characterized by an impulse response $h(t)$, then its output $y(t)$ to an input is given by the convolution integral

$$y(t) = \int_{-\infty}^{\infty} h(t-\tau)x(\tau)\,d\tau, \tag{1.2}$$

assuming the filter is relaxed with zero initial conditions at the time of application of the input signal. In the frequency domain, the above equation can be expressed as

$$Y(j\Omega) = H(j\Omega)X(j\Omega), \tag{1.3}$$

where $Y(j\Omega)$, $X(j\Omega)$, and $H(j\Omega)$, are, respectively, the continuous-time Fourier transforms of $y(t)$, $x(t)$, and $h(t)$.

A *lowpass filter* passes all low-frequency components below a certain specified frequency f_p, called the *passband edge frequency*, and blocks all high-frequency components above f_s, called the *stopband edge frequency*. A *highpass filter* passes all high-frequency components above a certain passband edge frequency f_p and blocks all low-frequency components below the stopband edge frequency f_s. A *bandpass filter* passes all frequency components between two passband edge frequencies f_{p1} and f_{p2} where $f_{p1} < f_{p2}$, and blocks all frequency components below the stopband edge frequency f_{s1} and above the stopband edge frequency f_{s2}. A *bandstop filter* blocks all frequency components between two stopband edge frequencies f_{s1} and f_{s2} and passes all frequency components below the passband edge frequency f_{p1} and above the passband edge frequency f_{p2}. Figure 1.2(a) shows a signal composed of three sinusoidal components of frequencies 50 Hz, 100 Hz, and 200 Hz, respectively. Figures 1.2(b) to (e) show the results of the above four types of filtering operations with appropriately chosen cutoff frequencies.

A bandstop filter designed to block a single frequency component is called a *notch filter*. A *multiband filter* has more than one passband and more than one stopband. A *comb filter* is designed to block frequencies that are integral multiples of a low frequency.

A signal may get corrupted unintentionally by an interfering signal called *interference* or *noise*. In many applications, the desired signal occupies a low-frequency band from dc to some frequency f_L Hz, and it is corrupted by a high-frequency noise with frequency components above f_H Hz with $f_H > f_L$. In such cases, the desired signal can be recovered from the noise-corrupted signal by passing the latter through a lowpass filter with a cutoff frequency f_c where $f_L < f_c < f_H$. In some applications, the noise corrupting the desired signal may be a single-frequency sinusoidal signal. For example, the noise generated by power lines radiating electric and magnetic fields appears as a 60-Hz sinusoidal signal. The desired signal can then be recovered by passing the corrupted signal through a notch filter with a notch frequency at 60 Hz.[2]

1.2.3 Generation of Complex-Valued Signals

As indicated earlier, a signal can be real-valued or complex-valued. For convenience, the former is usually called a *real signal*, while the latter is called a *complex signal*. All naturally generated signals are real-valued. In some applications, it is necessary to develop a complex signal from a real signal having more

[2] In many countries, power lines generate 50-Hz noise.

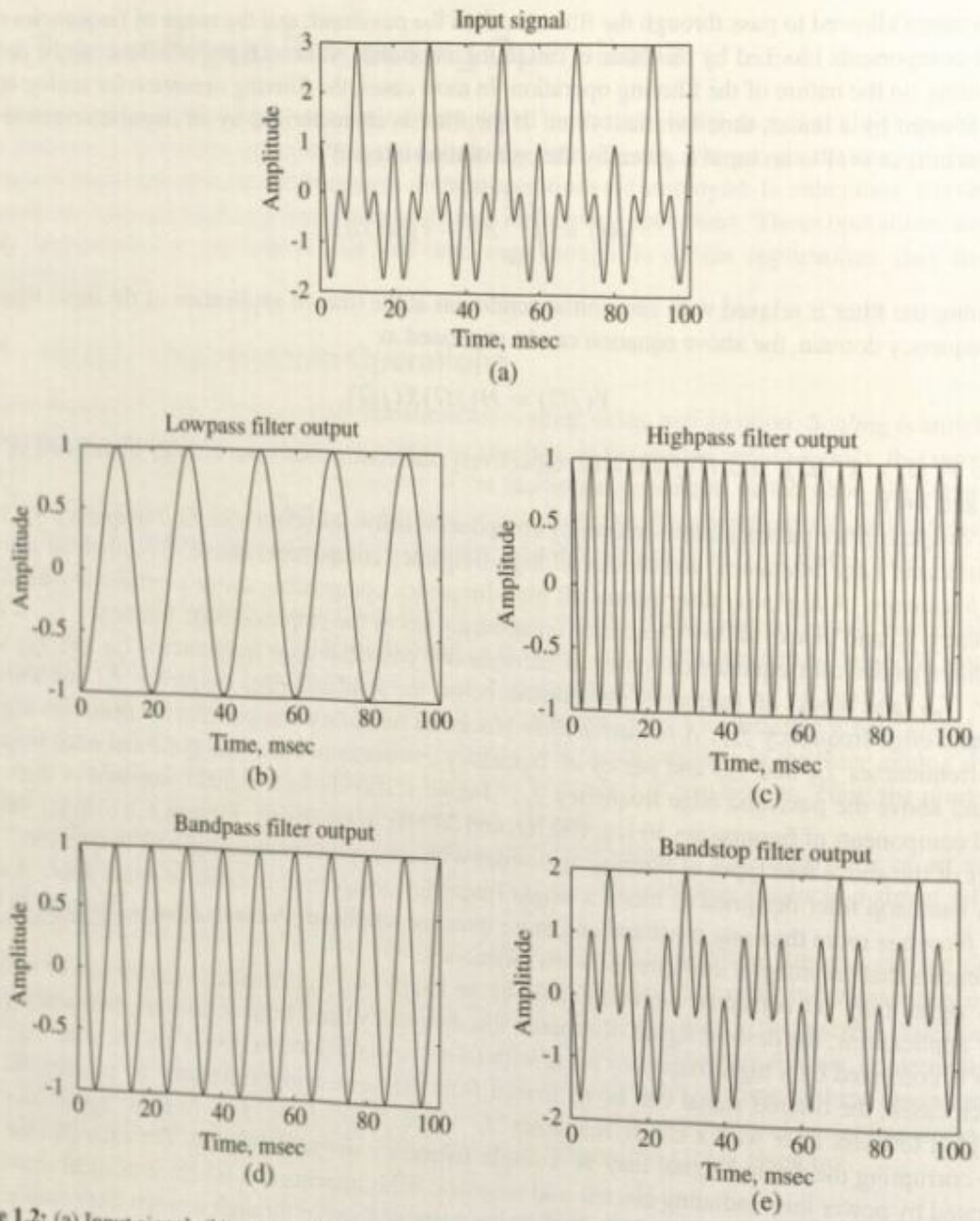


Figure 1.2: (a) Input signal, (b) output of a lowpass filter with a cutoff at 80 Hz, (c) output of a highpass filter with a cutoff at 150 Hz, (d) output of a bandpass filter with cutoffs at 80 Hz and 150 Hz, and (e) output of a bandstop filter with cutoffs at 80 Hz and 150 Hz.

desirable properties. One approach to generating a complex signal from a real signal is by employing a *Hilbert transformer* that is characterized by an impulse response $h_{HT}(t)$ given by [Fre94], [Opp83]

$$h_{HT}(t) = \frac{1}{\pi t}, \tag{1.4}$$

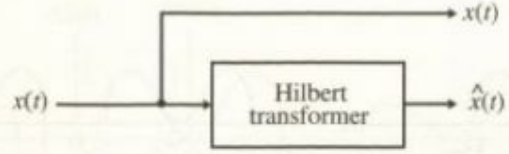


Figure 1.3: Generation of an analytic signal using a Hilbert transformer.

with a continuous-time Fourier transform $H_{HT}(j\Omega)$ given by

$$H_{HT}(j\Omega) = \begin{cases} -j, & \Omega > 0, \\ j, & \Omega < 0. \end{cases} \tag{1.5}$$

Let $x(t)$ denote a real analog signal with a continuous-time Fourier transform $X(j\Omega)$. The magnitude spectrum of a real signal exhibits even symmetry, while the phase spectrum exhibits odd symmetry. Thus, the spectrum $X(j\Omega)$ of a real signal $x(t)$ contains both positive and negative frequencies and can therefore be expressed as

$$X(j\Omega) = X_p(j\Omega) + X_n(j\Omega), \tag{1.6}$$

where $X_p(j\Omega)$ is the portion of $X(j\Omega)$ occupying the positive frequency range and $X_n(j\Omega)$ is the portion of $X(j\Omega)$ occupying the negative frequency range. If $x(t)$ is passed through a Hilbert transformer, its output $\hat{x}(t)$ has a spectrum $\hat{X}(j\Omega)$ given by

$$\hat{X}(j\Omega) = H_{HT}(j\Omega)X(j\Omega) = -jX_p(j\Omega) + jX_n(j\Omega). \tag{1.7}$$

It can be shown that $\hat{x}(t)$ is also a real signal. Consider the complex signal $y(t)$ formed by the sum of $x(t)$ and $\hat{x}(t)$:

$$y(t) = x(t) + j\hat{x}(t). \tag{1.8}$$

The signals $x(t)$ and $\hat{x}(t)$ are called, respectively, the *in-phase* and *quadrature components* of $y(t)$. The continuous-time Fourier transform of $y(t)$ is given by (Problem 3.9)

$$Y(j\Omega) = X(j\Omega) + j\hat{X}(j\Omega) = 2X_p(j\Omega). \tag{1.9}$$

Thus, the complex signal $y(t)$, called an *analytic signal*, has only positive-frequency components.

A block diagram representation of the scheme for the analytic signal generation from a real signal is sketched in Figure 1.3. One application of the Hilbert transformer is in the implementation of a single-sideband modulation as indicated in Figure 1.8 and discussed in Section 1.2.4.

1.2.4 Amplitude Modulation

For transmission of signals over long distances, a transmission medium such as cable, optical fiber, or the atmosphere is employed. Each such medium has a bandwidth that is more suitable for the efficient transmission of signals in the high-frequency range. As a result, for the transmission of a low-frequency signal over a channel, it is necessary to transform the signal to a high-frequency signal by means of a modulation operation. At the receiving end, the modulated high-frequency signal is demodulated, and the desired low-frequency signal is then extracted by further processing. There are four major types of modulation of analog signals: amplitude modulation, frequency modulation, phase modulation, and pulse

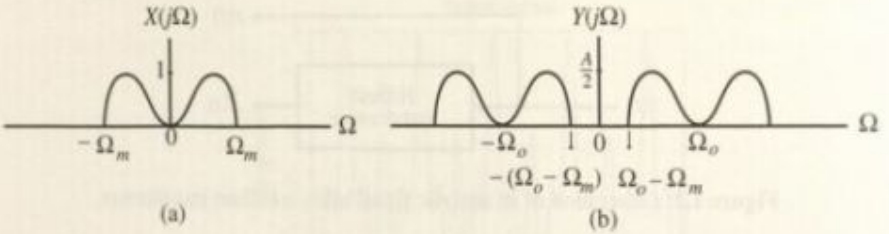


Figure 1.4: (a) Spectrum of the modulating signal $x(t)$ and (b) spectrum of the modulated signal $y(t)$. For convenience, both spectra are shown as real functions.

amplitude modulation. Of these schemes, amplitude modulation is conceptually simple and is discussed here [Fre94], [Opp83].

In the *amplitude modulation* scheme, the amplitude of a high-frequency sinusoidal signal $A\cos(\Omega_o t)$, called the *carrier signal*, is varied by the low-frequency band-limited signal $x(t)$, called the *modulating signal*, generating a high-frequency signal, called the *modulated signal*, $y(t)$ according to

$$y(t) = Ax(t)\cos(\Omega_o t). \tag{1.10}$$

Thus, amplitude modulation can be implemented by forming the product of the modulating signal with the carrier signal. It can be shown that the spectrum $Y(j\Omega)$ of $y(t)$ is given by

$$Y(j\Omega) = \tfrac{A}{2}X\left(j(\Omega - \Omega_o)\right) + \tfrac{A}{2}X\left(j(\Omega + \Omega_o)\right). \tag{1.11}$$

where $X(j\Omega)$ is the spectrum of the modulating signal $x(t)$. Figure 1.4 shows the spectra of the modulating signal and the modulated signal under the assumption that the carrier frequency Ω_o is greater than Ω_m, the highest frequency contained in $x(t)$. As seen from this figure, $y(t)$ is now a band-limited high-frequency signal with a bandwidth $2\Omega_m$ centered at Ω_o.

The portion of the amplitude-modulated signal between Ω_o and $\Omega_o + \Omega_m$ is called the *upper sideband*, whereas the portion between Ω_o and $\Omega_o - \Omega_m$ is called the *lower sideband*. Because of the generation of two sidebands and the absence of a carrier component in the modulated signal, the process is called *double-sideband suppressed carrier* (DSB-SC) *modulation*.

The *demodulation* of $y(t)$, assuming $\Omega_o > \Omega_m$, is carried out in two stages. First, the product of $y(t)$ with a sinusoidal signal of the same frequency as the carrier is formed. This results in

$$r(t) = y(t)\cos\Omega_o t = Ax(t)\cos^2\Omega_o t, \tag{1.12}$$

which can be rewritten as

$$r(t) = y(t)\cos\Omega_o t = \tfrac{A}{2}x(t) + \tfrac{A}{2}x(t)\cos(2\Omega_o t). \tag{1.13}$$

This result indicates that the product signal is composed of the original modulating signal scaled by a factor 1/2 and an amplitude-modulated signal with a carrier frequency $2\Omega_o$. The spectrum $R(j\Omega)$ of $r(t)$ is as indicated in Figure 1.5. The original modulating signal can now be recovered from $r(t)$ by passing it through a lowpass filter with a cutoff frequency Ω_c satisfying the relation $\Omega_m < \Omega_c < 2\Omega_o - \Omega_m$. The output of the filter is then a scaled replica of the modulating signal.

Figure 1.6 shows the block diagram representations of the amplitude modulation and demodulation schemes. The underlying assumption in the demodulation process outlined above is that a sinusoidal

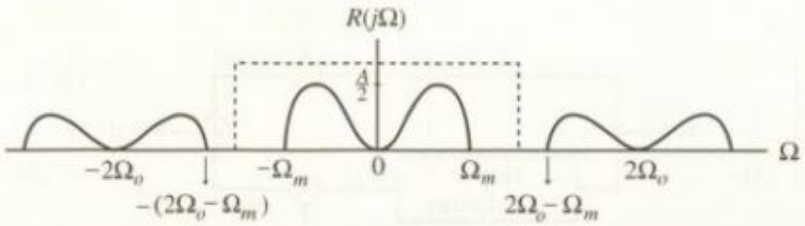


Figure 1.5: Spectrum of the product of the modulated signal and the carrier.

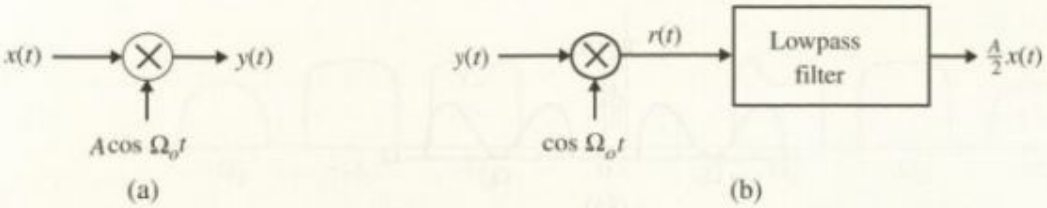


Figure 1.6: Schematic representations of the DSB-SC amplitude modulation and demodulation schemes: (a) modulator and (b) demodulator.

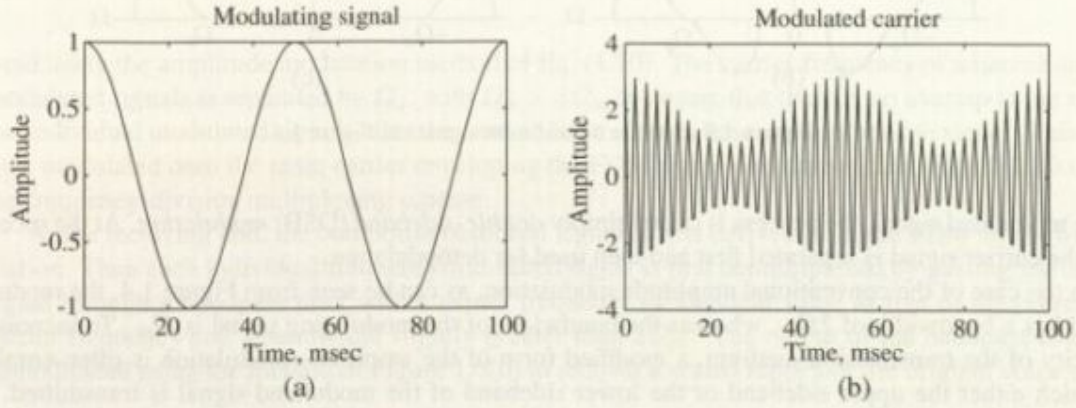


Figure 1.7: (a) A sinusoidal modulating signal of frequency 20 Hz and (b) modulated carrier with a carrier frequency of 400 Hz based on the DSB modulation.

signal identical to the carrier signal can be generated at the receiving end. In general, it is difficult to ensure that the demodulating sinusoidal signal has a frequency identical to that of the carrier all the time. To get around this problem, in the transmission of amplitude-modulated radio signals, the modulation process is modified so that the transmitted signal includes the carrier signal. This is achieved by redefining the amplitude modulation operation as follows:

$$y(t) = A[1 + mx(t)]\cos(\Omega_o t), \tag{1.14}$$

where m is a number chosen to ensure that $[1+mx(t)]$ is positive for all t. Figure 1.7 shows the waveforms of a modulating sinusoidal signal of frequency 20 Hz and the amplitude-modulated carrier obtained according to Eq. (1.14) for a carrier frequency of 400 Hz and $m = 0.5$. Note that the envelope of the modulated carrier is essentially the waveform of the modulating signal. As here the carrier is also present

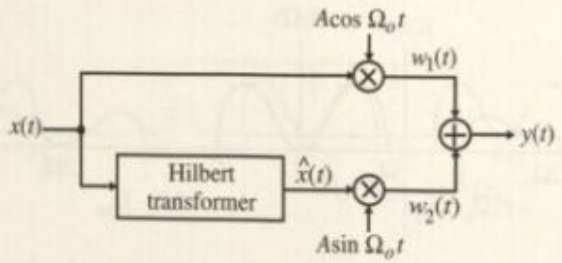


Figure 1.8: Single-sideband modulation scheme employing a Hilbert transformer.

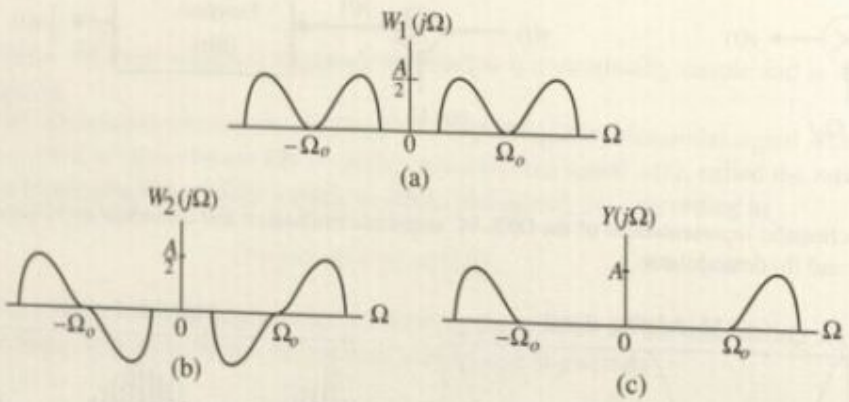


Figure 1.9: Spectra of pertinent signals in Figure 1.8.

in the modulated signal, the process is called simply *double-sideband* (DSB) *modulation*. At the receiving end, the carrier signal is separated first and then used for demodulation.

In the case of the conventional amplitude modulation, as can be seen from Figure 1.4, the modulated signal has a bandwidth of $2\Omega_m$, whereas the bandwidth of the modulating signal is Ω_m. To increase the capacity of the transmission medium, a modified form of the amplitude modulation is often employed in which either the upper sideband or the lower sideband of the modulated signal is transmitted. The corresponding procedure is called *single-sideband* (SSB) *modulation* to distinguish it from the double-sideband modulation scheme of Figure 1.6(a).

One way to implement the single-sideband amplitude modulation is indicated in Figure 1.8, where the Hilbert transformer used is defined by Eq. (1.4). The spectra of pertinent signals in Figure 1.8 are shown in Figure 1.9.

1.2.5 Multiplexing and Demultiplexing

For an efficient utilization of a wideband transmission channel, many narrow-bandwidth low-frequency signals are combined to form a composite wideband signal that is transmitted as a single signal. The process of combining these signals is called *multiplexing*, which is implemented to ensure that a replica of the original narrow-bandwidth low-frequency signals can be recovered at the receiving end. The recovery process is called *demultiplexing*.

One widely used method of combining different voice signals in a telephone communication system is the *frequency-division multiplexing* (FDM) scheme [Cou83], [Opp83]. Here, each voice signal, typically band-limited to a low-frequency band of width $2\Omega_m$, is frequency-translated into a higher frequency

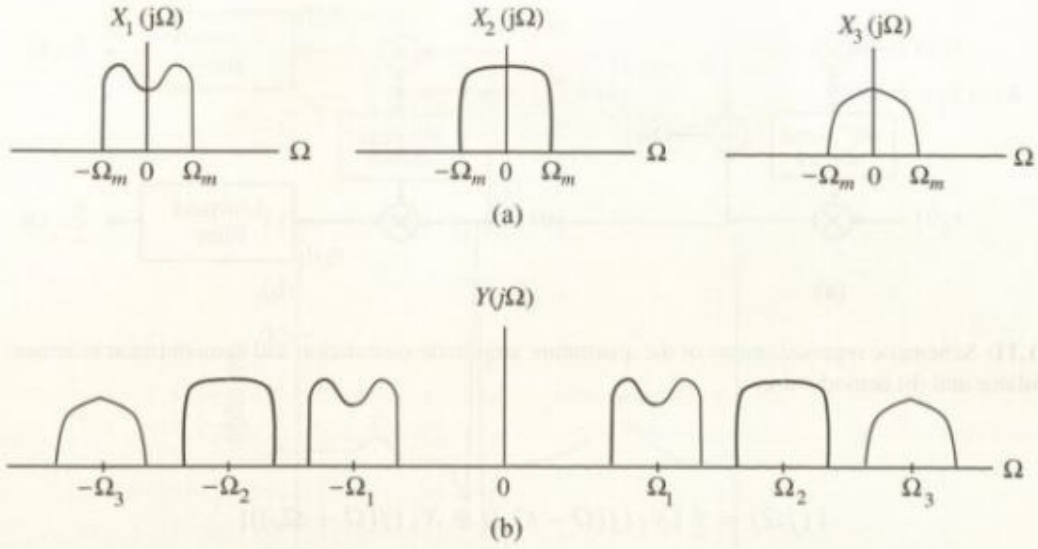


Figure 1.10: Illustration of the frequency-division multiplexing operation. (a) Spectra of three low-frequency signals and (b) spectra of the modulated composite signal.

band using the amplitude modulation method of Eq. (1.10). The carrier frequency of adjacent amplitude-modulated signals is separated by Ω_o, with $\Omega_o > 2\Omega_m$ to ensure that there is no overlap in the spectra of the individual modulated signals after they are added to form a baseband composite signal. This signal is then modulated onto the main carrier developing the FDM signal and transmitted. Figure 1.10 illustrates the frequency-division multiplexing scheme.

At the receiving end, the composite baseband signal is first derived from the FDM signal by demodulation. Then each individual frequency-translated signal is first demultiplexed by passing the composite signal through a bandpass filter with a center frequency of identical value as that of the corresponding carrier frequency and a bandwidth slightly greater than $2\Omega_m$. The output of the bandpass filter is then demodulated using the method of Figure 1.6(b) to recover a scaled replica of the original voice signal.

1.2.6 Quadrature Amplitude Modulation

We observed earlier that DSB amplitude modulation is half as efficient as SSB amplitude modulation with regard to utilization of the spectrum. The *quadrature amplitude modulation* (QAM) method uses DSB modulation to modulate two different signals so that they both occupy the same bandwidth; thus, QAM takes up only as much bandwidth as the SSB modulation method. To understand the basic idea behind the QAM approach, let $x_1(t)$ and $x_2(t)$ be two band-limited low-frequency signals with a bandwidth of Ω_m as indicated in Figure 1.4(a). The two modulating signals are individually modulated by the two carrier signals $A\cos(\Omega_o t)$ and $A\sin(\Omega_o t)$, respectively, and are summed, resulting in a composite signal $y(t)$ given by

$$y(t) = Ax_1(t)\cos(\Omega_o t) + Ax_2(t)\sin(\Omega_o t). \tag{1.15}$$

Note that the two carrier signals have the same carrier frequency Ω_o but have a phase difference of 90°. In general, the carrier $A\cos(\Omega_o t)$ is called the *in-phase component*, and the carrier $A\sin(\Omega_o t)$ is called the *quadrature component*. The spectrum $Y(j\Omega)$ of the composite signal $y(t)$ is now given by

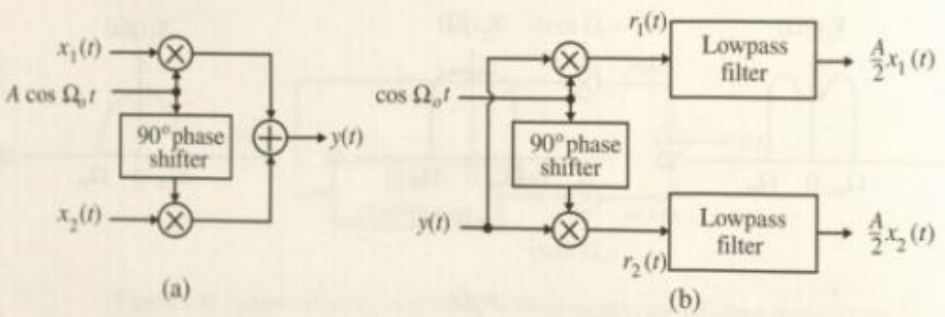


Figure 1.11: Schematic representations of the quadrature amplitude modulation and demodulation schemes: (a) modulator and (b) demodulator.

$$Y(j\Omega) = \tfrac{A}{2}\{X_1(j(\Omega-\Omega_o)) + X_1(j(\Omega+\Omega_o))\} + \tfrac{A}{2j}\{X_2(j(\Omega-\Omega_o)) - X_2(j(\Omega+\Omega_o))\} \tag{1.16}$$

and is seen to occupy the same bandwidth as the modulated signal obtained by a DSB modulation.

To recover the original modulating signals, the composite signal is multiplied by both the in-phase and the quadrature components of the carrier separately, resulting in two signals:

$$r_1(t) = y(t)\cos(\Omega_o t), \qquad r_2(t) = y(t)\sin(\Omega_o t). \tag{1.17}$$

Substituting $y(t)$ from Eq. (1.15) in Eq. (1.17), we obtain, after some algebra,

$$r_1(t) = \tfrac{A}{2}x_1(t) + \tfrac{A}{2}x_1(t)\cos(2\Omega_o t) + \tfrac{A}{2}x_2(t)\sin(2\Omega_o t),$$
$$r_2(t) = \tfrac{A}{2}x_2(t) + \tfrac{A}{2}x_1(t)\sin(2\Omega_o t) - \tfrac{A}{2}x_2(t)\cos(2\Omega_o t). \tag{1.18}$$

Lowpass filtering of $r_1(t)$ and $r_2(t)$ by filters with a cutoff at Ω_m yields the two modulating signals. Figure 1.11 shows the block diagram representations of the quadrature amplitude modulation and demodulation schemes.

As in the case of the DSB suppressed carrier modulation method, the QAM method also requires at the receiver an exact replica of the carrier signal employed in the transmitting end for accurate demodulation. It is therefore not employed in the direct transmission of analog signals, but finds applications in the transmission of discrete-time data sequences and in the transmission of analog signals converted into discrete-time sequences by sampling and analog-to-digital conversion.

1.2.7 Signal Generation

An equally important part of signal processing is synthetic signal generation. One of the simplest such signal generators is a device generating a sinusoidal signal, called an *oscillator*. Such a device is an integral part of the amplitude-modulation and demodulation system described in the previous two sections. It also has various other signal processing applications.

There are applications that require the generation of other types of periodic signals such as square waves and triangular waves. Certain types of random signals with a spectrum of constant amplitude for all frequencies, called *white noise*, often find applications in practice. One such application is in the generation of discrete-time synthetic speech signals.

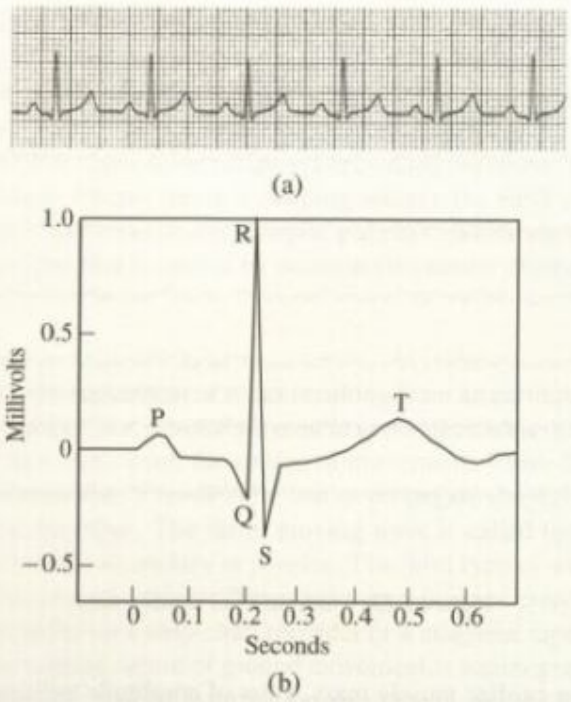


Figure 1.12: (a) A typical ECG trace and (b) one cycle of an ECG waveform.

1.3 Examples of Typical Signals[3]

To better understand the breadth of the signal processing task, we now examine a number of examples of some typical signals and their subsequent processing in typical applications.

Electrocardiography (ECG) Signal

The electrical activity of the heart is represented by the ECG signal [Sha81]. A typical ECG signal trace is shown in Figure 1.12(a). The ECG trace is essentially a periodic waveform. One cycle of the blood transfer process from the heart to the arteries is represented by one period of the ECG waveform as shown in Figure 1.12(b). This part of the waveform is generated by an electrical impulse originating at the sinoatrial node in the right atrium of the heart. The impulse causes contraction of the atria, which forces the blood in each atrium to squeeze into its corresponding ventricle. The resulting signal is called the P-wave. The atrioventricular node delays the excitation impulse until the blood transfer from the atria to the ventricles is completed, resulting in the P–R interval of the ECG waveform. The excitation impulse then causes contraction of the ventricles, which squeezes the blood into the arteries. This generates the QRS part of the ECG waveform. During this phase, the atria are relaxed and filled with blood. The T-wave of the waveform represents the relaxation of the ventricles. The complete process is repeated periodically, generating the ECG trace.

Each portion of the ECG waveform carries various types of information for the physician analyzing a patient's heart condition [Sha81]. For example, the amplitude and timing of the P and QRS portions

[3]This section has been adapted from *Handbook for Digital Signal Processing*, Sanjit K. Mitra and James F. Kaiser, Eds., ©1993, John Wiley & Sons. Adapted by permission of John Wiley & Sons.

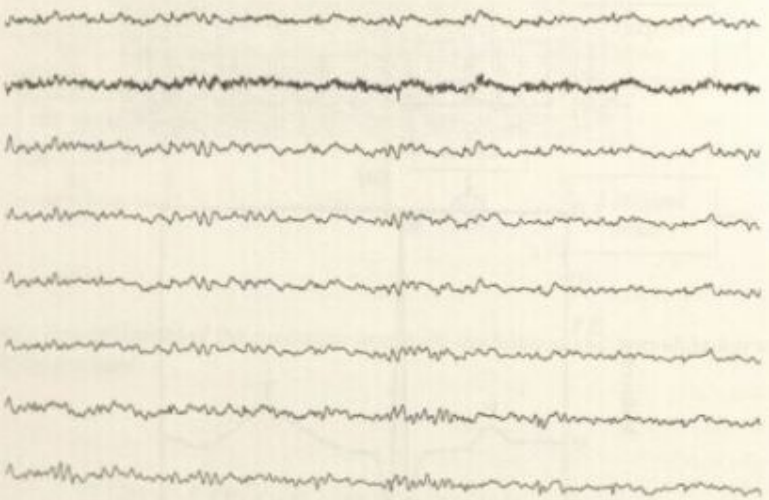


Figure 1.13: Multiple EEG signal traces.

indicate the condition of the cardiac muscle mass. Loss of amplitude indicates muscle damage, whereas increased amplitude indicates abnormal heart rates. Too long a delay in the atrioventricular node is indicated by a very long P–R interval. Likewise, blockage of some or all of the contraction impulses is reflected by intermittent synchronization between the P- and QRS-waves. Most of these abnormalities can be treated with various drugs, and the effectiveness of the drugs can again be monitored by observing the new ECG waveforms taken after the drug treatment.

In practice, there are various types of externally produced artifacts that appear in the ECG signal [Tom81]. Unless these interferences are removed, it is difficult for a physician to make a correct diagnosis. A common source of noise is the 60-Hz power lines whose radiated electric and magnetic fields are coupled to the ECG instrument through capacitive coupling and/or magnetic induction. Other sources of interference are the electromyographic signals that are the potentials developed by contracting muscles. These and other interferences can be removed with careful shielding and signal processing techniques.

Electroencephalogram (EEG) Signal

The summation of the electrical activity caused by the random firing of billions of individual neurons in the brain is represented by the EEG signal [Coh86], [Tom81]. In multiple EEG recordings, electrodes are placed at various positions on the scalp with two common electrodes placed on the earlobes, and potential differences between the various electrodes are recorded. A typical bandwidth of this type of EEG ranges from 0.5 to about 100 Hz, with the amplitudes ranging from 2 to 100 mV. An example of multiple EEG traces is shown in Figure 1.13.

Both frequency-domain and time-domain analyses of the EEG signal have been used for the diagnosis of epilepsy, sleep disorders, psychiatric malfunctions, and so forth. To this end, the EEG spectrum is subdivided into the following five bands: (1) the *delta* range, occupying the band from 0.5 to 4 Hz; (2) the *theta* range, occupying the band from 4 to 8 Hz; (3) the *alpha* range, occupying the band from 8 to 13 Hz; (4) the *beta* range, occupying the band from 13 to 22 Hz; and (5) the *gamma* range, occupying the band from 22 to 30 Hz.

The delta wave is normal in the EEG signals of children and sleeping adults. Since it is not common in alert adults, its presence indicates certain brain diseases. The theta wave is usually found in children even though it has been observed in alert adults. The alpha wave is common in all normal humans and is more pronounced in a relaxed and awake subject with closed eyes. Likewise, the beta activity is common in normal adults. The EEG exhibits rapid, low-voltage waves, called *rapid-eye-movement* (REM) waves, in a subject dreaming during sleep. Otherwise, in a sleeping subject, the EEG contains bursts of alpha-like waves, called *sleep spindles*. The EEG of an epileptic patient exhibits various types of abnormalities, depending on the type of epilepsy that is caused by uncontrolled neural discharges.

Seismic Signals

Seismic signals are caused by the movement of rocks resulting from an earthquake, a volcanic eruption, or an underground explosion [Bol93]. The ground movement generates elastic waves that propagate through the body of the earth in all directions from the source of movement. Three basic types of elastic waves are generated by the earth movement. Two of these waves propagate through the body of the earth, one moving faster with respect to the other. The faster moving wave is called the *primary* or *P-wave*, while the slower moving one is called the *secondary* or *S-wave*. The third type of wave is known as the *surface wave*, which moves along the ground surface. These seismic waves are converted into electrical signals by a seismograph and are recorded on a strip chart recorder or a magnetic tape.

Because of the three-dimensional nature of ground movement, a seismograph usually consists of three separate recording instruments that provide information about the movements in the two horizontal directions and one vertical direction and develops three records as indicated in Figure 1.14. Each such record is a one-dimensional signal. From the recorded signals, it is possible to determine the magnitude of the earthquake or nuclear explosion and the location of the source of the original earth movement.

Seismic signals also play an important role in the geophysical exploration for oil and gas [Rob80]. In this type of application, linear arrays of seismic sources, such as high-energy explosives, are placed at regular intervals on the ground surface. The explosions cause seismic waves to propagate through the subsurface geological structures and reflect back to the surface from interfaces between geological strata. The reflected waves are converted into electrical signals by a composite array of geophones laid out in certain patterns and displayed as a two-dimensional signal that is a function of time and space, called a *trace gather*, as indicated in Figure 1.15. Before these signals are analyzed, some preliminary time and amplitude corrections are made on the data to compensate for different physical phenomena. From the corrected data, the time differences between reflected seismic signals are used to map structural deformations, whereas the amplitude changes usually indicate the presence of hydrocarbons.

Speech Signals

The acoustic theory of speech production has led to a range of mathematical models for the representation of speech signals. A speech signal is created by exciting the vocal tract using either quasi-periodic puffs of air or by creating turbulent air flow around a constriction in the vocal tract or by a mixture of these two sound sources [Del93], [Rab78]. So-called *voiced sounds* are generated when air is forced through the tensed glottis, causing it to vibrate in an oscillatory manner and generating pseudo-periodic pulses of air that excite the vocal tract. Included in the class of voiced sounds are vowels such as /I/ (as in 'big') or /ae/ (as in 'bad'); voiced consonants such as /b/, /d/, /g/, /m/, /n/ and so on; and so-called *liquids* and *glides* such as /w/, /l/, /r/, and /y/.[4] are generated by forming a constriction at some point in the vocal tract,

[4]The sounds of a speech signal are usually represented pictorially using a "phonetic alphabet" by inserting the marker "/" on both sides of the letters representing the sound with uppercase letters representing voiced sounds that are stronger in amplitudes

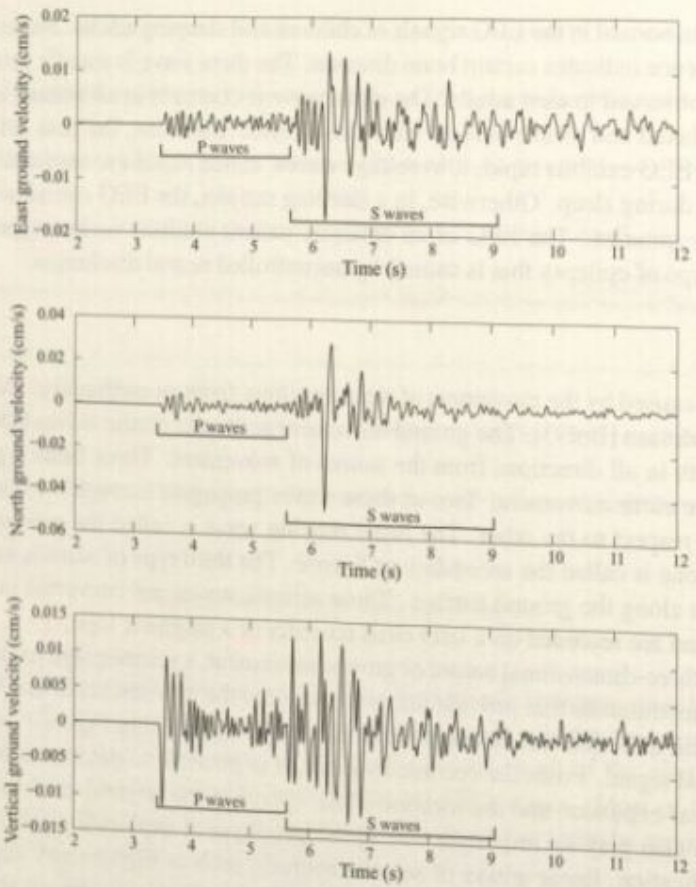


Figure 1.14: Chino Hills aftershock recorded at station Puddingstone Reservoir, Southern California Seismic Network. Southern CaliforniaEarthquake Data Center. 29 July 2008. <http://www.data.scec.org>. Approximate durations of the P-waves and S-waves have been added to the original seismograph.

which causes the air flow to become turbulent (noise-like) and to act as the excitation source for sounds such as /f/, /s/, /sh/ and so forth. Finally, there is a class of sounds that utilizes both sources of excitation and hence has characteristics of both voiced sounds and unvoiced sounds. Among this class of sounds are the *voiced fricatives* such as /v/, /z/, and /zh/.

There are sounds, called *plosive sounds*, whose characteristics change dynamically during their production, such as /p/, /t/, and /k/. For these sounds, the vocal tract has a total constriction at some point along the tract (thereby totally blocking the flow of air and equivalently the production of sound in the vocal tract). Pressure builds up behind the constriction causing a sudden release of pressure as the constriction is removed, leading to a sound, followed by a gradual release of the built-up air flow.

Speech Demo 1

Figure 1.16(a) depicts the speech waveform of a male utterance "I like digital signal processing ." The total duration of the speech waveform shown is 3 s. Magnified versions of /I/ in the word "like" and the /S/ segment in the word "processing" are sketched in Figures 1.16(b) and (c), respectively. The slowly varying low-frequency voiced waveform of /I/ and the high-frequency unvoiced fricative waveform of /S/ are evident from the magnified waveforms. The voiced waveform in Figure 1.16(b) is seen to be quasi-periodic and can be modeled by a sum of a finite number of sinusoids. The lowest frequency of oscillation in this representation is called the *fundamental frequency* or *pitch frequency*. The unvoiced waveform in

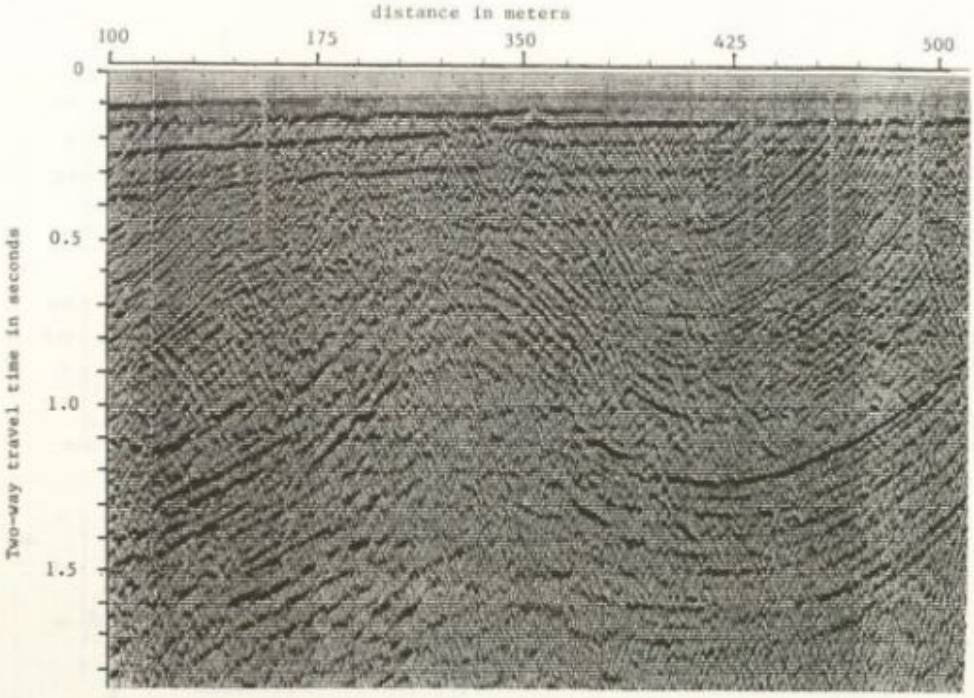


Figure 1.15: A typical seismic signal trace gather. (Courtesy of Institute for Crustal Research, University of California, Santa Barbara, CA.)

Figure 1.16(c) has no regular fine structure and is more noise-like.

One of the major applications of digital signal processing techniques is in the general area of speech processing. Problems in this area are usually divided into three groups: (1) speech analysis, (2) speech synthesis, and (3) speech analysis and synthesis [Opp78]. Digital speech analysis methods are used in automatic speech recognition, speaker verification, and speaker identification. Applications of digital speech synthesis techniques include reading machines for the automatic conversion of written text into speech and retrieval of data from computers in speech form by remote access through terminals or telephones. One example belonging to the third group is voice scrambling for secure transmission. Speech data compression for an efficient use of the transmission medium is another example of the use of speech analysis followed by synthesis. A typical speech signal after conversion into a digital form contains about 64,000 bits per second (bps). Depending on the desired quality of the synthesized speech, the original data can be compressed considerably; for example, down to about 1000 bps.

Musical Sound Signal

The electronic synthesizer is an example of the use of modern signal processing techniques [Ler83], [Moo77]. The natural sound generated by most musical instruments is generally produced by mechanical vibrations caused by activating some form of oscillator that then causes other parts of the instrument to vibrate. All these vibrations together in a single instrument generate the musical sound. In a violin, the primary oscillator is a stretched piece of string (cat gut). Its movement is caused by drawing a bow across it; this sets the wooden body of the violin vibrating, which in turn sets up vibrations of the air inside as well as outside the instrument. In a piano, the primary oscillator is a stretched steel wire that is set

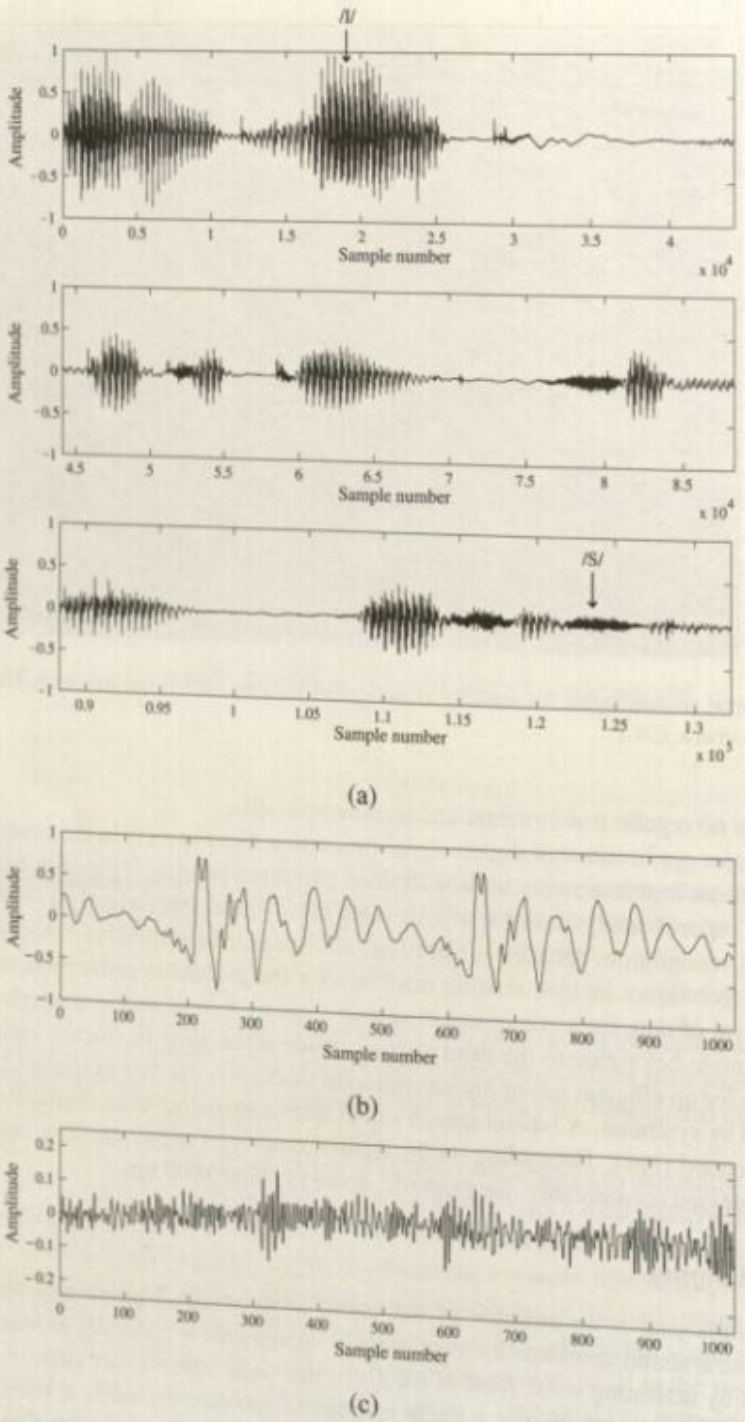


Figure 1.16: Speech waveform example: (a) sentence-length segment, (b) magnified version of the voiced segment (the letter /l/ in "like"), and (c) magnified version of the unvoiced segment (the letter /S/ in "processing").

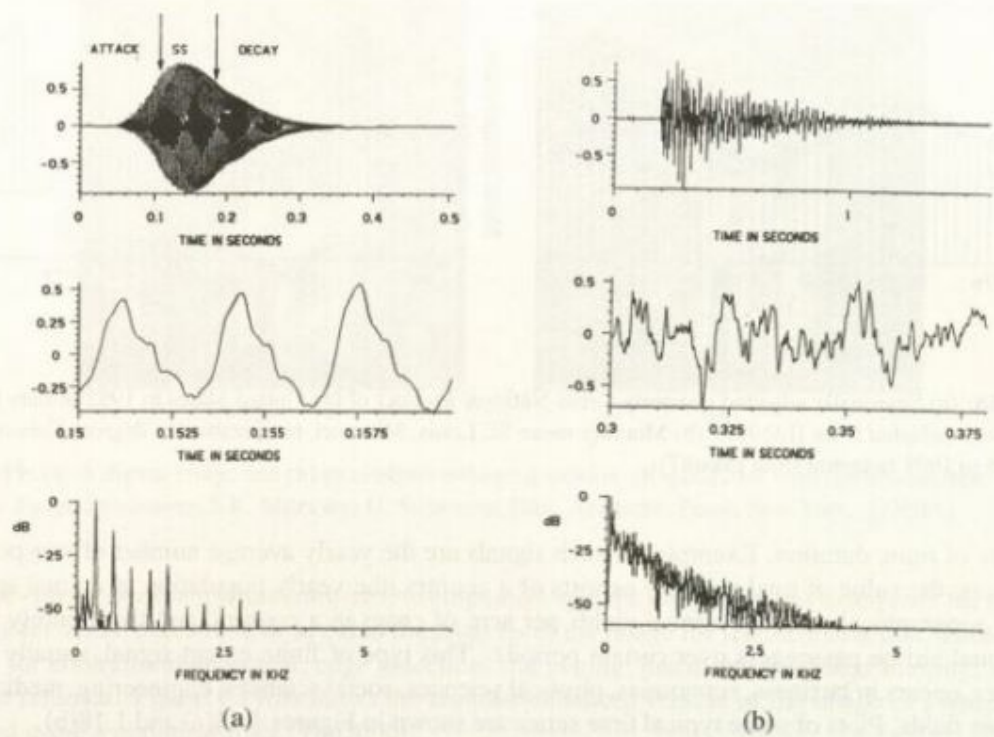


Figure 1.17: Waveforms of (a) the cello and (b) the bass drum. (Reproduced with permission from J.A. Moorer, Signal processing aspects of computer music: A survey, *Proceedings of the IEEE*, vol. 65, August 1977, pp. 1108–1137 ©1977 IEEE.)

into vibratory motion by the hitting of a hammer, which in turn causes vibrations in the wooden body (sounding board) of the piano. In wind or brass instruments, the vibration occurs in a column of air, and a mechanical change in the length of the air column by means of valves or keys regulates the rate of vibration.

The sound of orchestral instruments can be classified into two groups: *quasi-periodic* and *aperiodic*. Quasi-periodic sounds can be described by a sum of a finite number of sinusoids with independently varying amplitudes and frequencies. Figure 1.17(a) and (b) show the sound waveforms of two different instruments, the cello and the bass drum, respectively. In each figure, the top waveform is the plot of an entire isolated note, whereas the bottom plot shows an expanded version of a portion of the note: 10 ms for the cello and 80 ms for the bass drum. The waveform of the note from a cello is seen to be quasi-periodic. On the other hand, the bass drum waveform is clearly aperiodic. The tone of an orchestral instrument is commonly divided into three segments called the *attack* part, the *steady-state* part, and the *decay* part. Figure 1.17 illustrates this division for the two tones. Note that the bass drum tone of Figure 1.17(b) shows no steady-state part. A reasonable approximation of many tones is obtained by splicing together these parts. However, high-fidelity reproduction requires a more complex model.

Time Series

The signals described thus far are continuous functions with time as the independent variable. In many cases, the signals of interest are naturally discrete functions of the independent variables. Often such

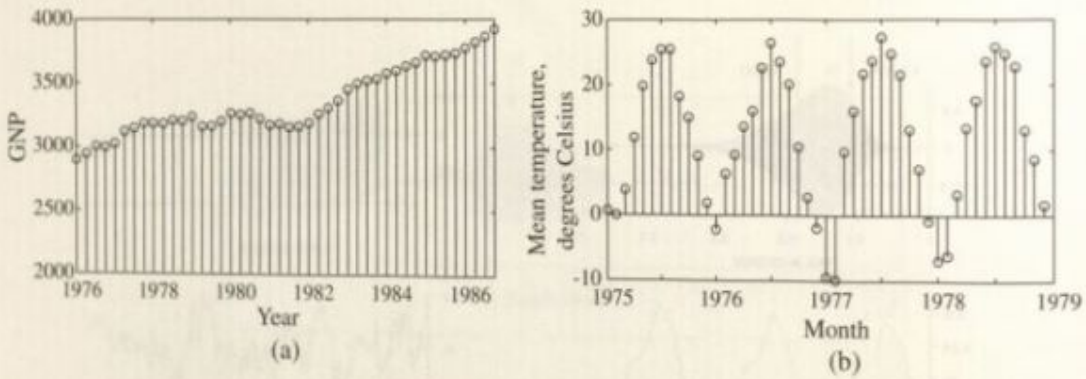


Figure 1.18: (a) Seasonally adjusted quarterly Gross National Product of the United States in 1982 dollars from 1976 to 1986 (adapted from [Lüt91]). (b) Monthly mean St. Louis, Missouri, temperature in degrees Celsius for the years 1975 to 1978 (adapted from [Mar87]).

signals are of finite duration. Examples of such signals are the yearly average number of sunspots, daily stock prices, the value of total monthly exports of a country, the yearly population of animal species in a certain geographical area, the annual yields per acre of crops in a country, and the monthly totals of international airline passengers over certain periods. This type of finite extent signal, usually called a *time series*, occurs in business, economics, physical sciences, social sciences, engineering, medicine, and many other fields. Plots of some typical time series are shown in Figures 1.18(a) and 1.18(b).

There are many reasons for analyzing a particular time series [Box70]. In some applications, there may be a need to develop a model to determine the nature of the dependence of the data on the independent variable and use it to forecast the future behavior of the series. As an example, in business planning, reasonably accurate sales forecasts are necessary. Some types of series possess seasonal or periodic components, and it is important to extract these components. The study of sunspot numbers is important for predicting climate variations. Invariably, the time series data are noisy, and their representations require models based on their statistical properties.

Images

As indicated earlier, an image is a two-dimensional signal whose intensity at any point is a function of two spatial variables. Common examples are photographs, still video images, radar and sonar images, and chest and dental X-rays. An image sequence, such as that seen in a television, is essentially a three-dimensional signal for which the image intensity at any point is a function of three variables: two spatial variables and time. Figure 1.19(a) shows the photograph of a digital image.

The basic problems in image processing are image signal representation and modeling, enhancement, restoration, reconstruction from projections, analysis, and coding [Jai89].

Each picture element in a specific image represents a certain physical quantity; a characterization of the element is called the *image representation*. For example, a photograph represents the luminances of various objects as seen by the camera. An infrared image taken by a satellite or an airplane represents the temperature profile of a geographical area. Depending on the type of image and its applications, various types of image models are usually defined. Such models are also based on perception and on local or global characteristics. The nature and performance of the image processing algorithms depend on the image model being used.

(a) (b)

Figure 1.19: (a) A digital image and (b) its contrast-enhanced version. (Reproduced with permission from *Nonlinear Image Processing*, S.K. Mitra and G. Sicuranza, Eds., Academic Press, New York, ©2000.)

Image enhancement algorithms are used to emphasize specific image features to improve the quality of the image for visual perception or to aid in the analysis of the image for feature extraction. These include methods for contrast enhancement, edge detection, sharpening, linear and nonlinear filtering, zooming, and noise removal. Figure 1.19(b) shows the contrast-enhanced version of the image of Figure 1.19(a), developed using a nonlinear filter [Thu2000].

The algorithms used for elimination or reduction of degradations in an image, such as blurring and geometric distortion caused by the imaging system and/or its surroundings, are known as *image restoration*. *Image reconstruction* from projections involves the development of a two-dimensional image slice of a three-dimensional object from a number of planar projections obtained from various angles. By creating a number of contiguous slices, a three-dimensional image giving an inside view of the object is developed.

Image analysis methods are employed to develop a quantitative description and classification of one or more desired objects in an image.

For digital processing, an image needs to be sampled and quantized using an analog-to-digital converter. A reasonable size digital image in its original form takes a considerable amount of memory space for storage. For example, an image of size 512×512 samples with 8-bit resolution per sample contains over 2 million bits. *Image coding* methods are used to reduce the total number of bits in an image without any degradation in visual perception quality as in speech coding; for example, down to about 1 bit per sample on the average.

1.4 Typical Signal Processing Applications[5]

There are numerous applications of signal processing that we often encounter in our daily lives without being aware of them. Originally the signal processing algorithms used in these applications were carried out in the continuous-time domain. However, they are now being increasingly implemented using discrete-signal processing algorithms. Due to space limitations, it is not possible to discuss all of these applications. However, an overview of selected applications is presented.

[5]This section has been adapted from *Handbook for Digital Signal Processing*, Sanjit K. Mitra and James F. Kaiser, Eds., ©1993, John Wiley & Sons. Adapted by permission of John Wiley & Sons.

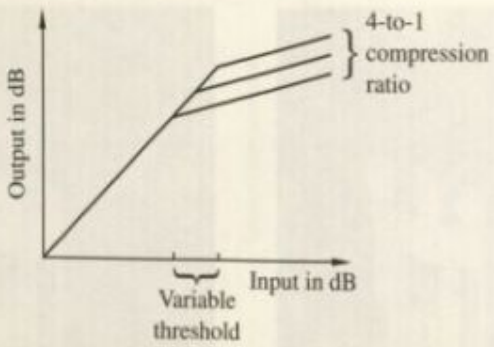


Figure 1.20: Transfer characteristic of a typical compressor.

1.4.1 Sound Recording Applications

The recording of most musical programs nowadays is usually made in an acoustically inert studio. The sound from each instrument is picked up by its own microphone closely placed to the instrument and is recorded on a single track in a multitrack tape recorder containing as many as 48 tracks. The signals from individual tracks in the master recording are then edited and combined by the sound engineer in a *mix-down* system to develop a two-track stereo recording. There are a number of reasons for following this approach. First, the closeness of each individual microphone to its assigned instrument provides a high degree of separation between the instruments and minimizes the background noise in the recording. Second, the sound part of one instrument can be rerecorded later if necessary. Third, during the mix-down process, the sound engineer can manipulate individual signals by using a variety of signal processing devices to alter the musical balances between the sounds generated by the instruments, can change the timbre, and can add natural room acoustics effects and other special effects [Ble78], [Ear76].

Various types of signal processing techniques are utilized in the mix-down phase. Some are used to modify the spectral characteristics of the sound signal and to add special effects, whereas others are used to improve the quality of the transmission medium. The signal processing circuits most commonly used are (1) compressors and limiters, (2) expanders and noise gates, (3) equalizers and filters, (4) noise reduction systems, (5) delay and reverberation systems, and (6) circuits for special effects [Ble78], [Ear76], [Hub89], [Wor89]. These operations are usually performed on the original analog audio signals and are implemented using analog circuit components. However, there is a growing trend toward all digital implementation and its use in the processing of the digitized versions of the analog audio signals [Ble78].

Compressors and limiters. These devices are used for the compression of the dynamic range of an audio signal. The compressor can be considered as an amplifier with two gain levels: the gain is unity for input signal levels below a certain threshold and less than unity for signals with levels above the threshold. The threshold level is adjustable over a wide range of the input signal. Figure 1.20 shows the transfer characteristic of a typical compressor.

The parameters characterizing a compressor are its compression ratio, threshold level, attack time, and release time, which are illustrated in Figure 1.21.

When the input signal level suddenly rises above a prescribed threshold, the time taken by the compressor to adjust its normal unity gain to the lower value is called the *attack time*. Because of this effect, the output signal exhibits a slight degree of overshoot before the desired output level is reached. A zero attack time is desirable to protect the system from sudden high-level transients. However, in this case,

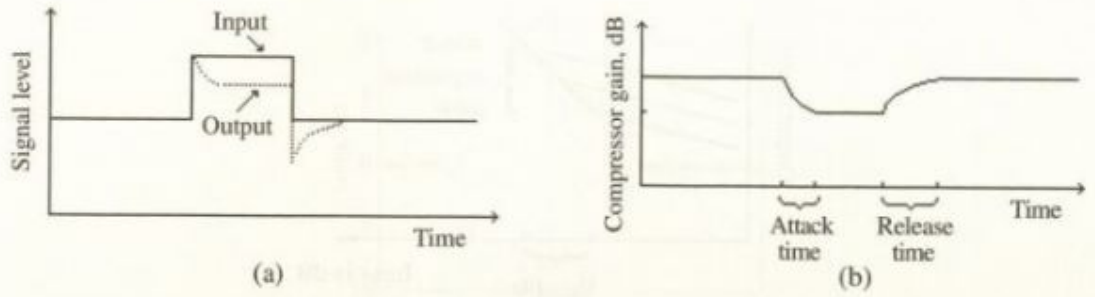


Figure 1.21: Parameters characterizing a typical compressor.

the impact of sharp musical attacks is eliminated, resulting in a dull, "lifeless" sound [Wor89]. A longer attack time causes the output to sound more percussive than normal.

Similarly, the time taken by the compressor to reach its normal unity gain value when the input level suddenly drops below the threshold is called the *release time* or *recovery time*. If the input signal fluctuates rapidly around the threshold in a small region, the compressor gain also fluctuates up and down. In such a situation, the rise and fall of background noise results in an audible effect called *breathing* or *pumping*, which can be minimized with a longer release time for the compressor gain.

There are various applications of the compressor unit in musical recording [Ear76]. For example, it can be used to eliminate variations in the peaks of an electric bass output signal by clamping them to a constant level, thus providing an even and solid bass line. To maintain the original character of the instrument, it is necessary to use a compressor with a long recovery time compared to the natural decay rate of the electric bass. The device is also useful to compensate for the wide variations in the signal level produced by a singer who moves frequently, changing the distance from the microphone.

A compressor with a compression ratio of 10-to-1 or greater is called a *limiter* because its output levels are essentially clamped to the threshold level. The limiter is used to prevent overloading of amplifiers and other devices caused by signal peaks exceeding certain levels.

Expanders and noise gates. The expander's function is opposite that of the compressor. It is also an amplifier with two gain levels: the gain is unity for input signal levels above a certain threshold and less than unity for signals with levels below the threshold. The threshold level is again adjustable over a wide range of the input signal. Figure 1.22 shows the transfer characteristic of a typical expander. The *expander* is used to expand the dynamic range of an audio signal by boosting the high-level signals and attenuating the low-level signals. The device can also be used to reduce noise below a threshold level.

The expander is characterized by its expansion ratio, threshold level, attack time, and release time. Here, the time taken by the device to reach the normal unity gain for a sudden change in the input signal to a level above the threshold is defined as the *attack time*. Likewise, the time required by the device to lower the gain from its normal value of one for a sudden decrease in the input signal level is called the *release time*. The noise gate is a special type of expander that heavily attenuates signals with levels below the threshold. It is used, for example, to totally cut off a microphone during a musical pause so as not to pass the noise being picked up by the microphone.

Equalizers and filters. Various types of filters are used to modify the frequency response of a recording or the monitoring channel. One such filter, called the *shelving filter*, provides boost (rise) or cut (drop) in the frequency response at either the low or the high end of the audio frequency range while not affecting the frequency response in the remaining range of the audio spectrum, as shown in Figure 1.23. *Peaking*

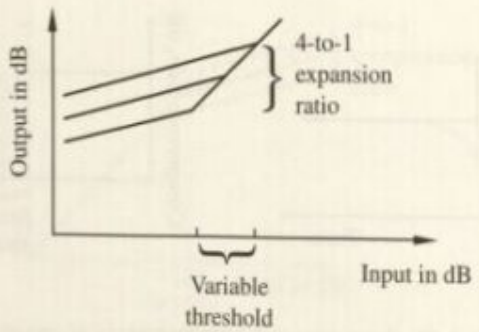


Figure 1.22: Transfer characteristic of a typical expander.

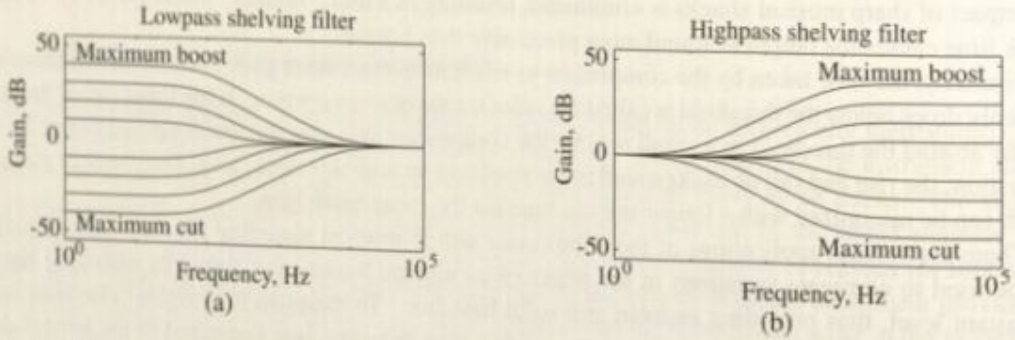


Figure 1.23: Frequency responses of (a) low-frequency shelving filter and (b) high-frequency shelving filter.

filters are used for midband equalization and are designed to have either a bandpass response to provide a boost or a bandstop response to provide a cut, as indicated in Figure 1.24.

The parameters characterizing a low-frequency shelving filter are the two frequencies f_{1L} and f_{2L}, where the magnitude response begins tapering up or down from a constant level and the low-frequency gain levels in dB. Likewise, the parameters characterizing a high-frequency shelving filter are the two frequencies f_{1H} and f_{2H}, where the magnitude response begins tapering up or down from a constant level and the high-frequency gain levels in dB. In the case of a peaking filter, the parameters of interest are the center frequency f_0, the 3-dB bandwidth Δf of the bell-shaped curve, and the gain level at the center frequency. Most often, the quality factor $Q = f_0/\Delta f$ is used to characterize the shape of the frequency response instead of the bandwidth Δf.

A typical equalizer consists of a cascade of a low-frequency shelving filter, a high-frequency shelving filter, and three or more peaking filters with adjustable parameters to provide adjustment of the overall equalizer frequency response over a broad range of frequencies in the audio spectrum. In a *parametric equalizer*, each individual parameter of its constituent filter blocks can be varied independently without affecting the parameters of the other filters in the equalizer.

The *graphic equalizer* consists of a cascade of peaking filters with fixed center frequencies but adjustable gain levels that are controlled by vertical slides in the front panel. The physical position of the slides reasonably approximates the overall equalizer magnitude response, as shown schematically in Figure 1.25.

Other types of filters that also find applications in the musical recording and transfer processes are the *lowpass*, *highpass*, and *notch filters*. Their corresponding frequency responses are indicated in Fig-

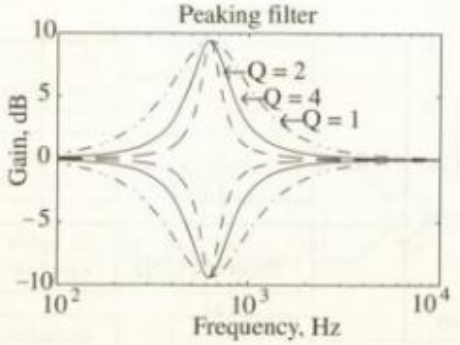


Figure 1.24: Peaking filter frequency response.

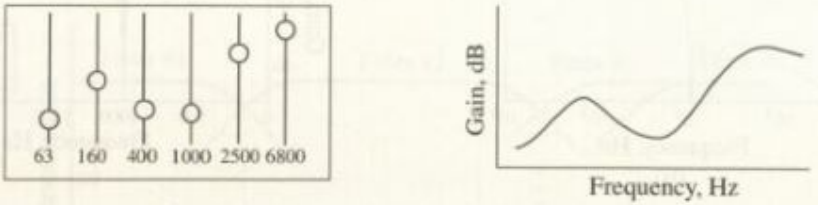


Figure 1.25: Graphic equalizer: (a) control panel settings and (b) corresponding frequency response. (Adapted from [Ear76].)

ure 1.26. The notch filter is designed to attenuate a particular frequency component and has a narrow notch width so as not to affect the rest of the musical program.

Two major applications of equalizers and filters in recording are to correct certain types of problems that may have occurred during the recording or the transfer process and to alter the harmonic or timbral contents of a recorded sound purely for musical or creative purposes [Ear76]. For example, a direct transfer of a musical recording from old 78 rpm disks to a wideband playback system will be highly noisy due to the limited bandwidth of the old disks. To reduce this noise, a bandpass filter with a passband matching the bandwidth of the old records is utilized. Often, older recordings are made more pleasing by adding a broad high-frequency peak in the 5- to 10-kHz range and by shelving out some of the lower frequencies. The notch filter is particularly useful in removing 60-Hz power supply hum.

In creating a program by mixing down a multichannel recording, the recording engineer usually employs equalization of individual tracks for creative reasons [Ear76]. For example, a "fullness" effect can be added to weak instruments such as the acoustical guitar by boosting frequency components in the range 100 to 300 Hz. Similarly, by boosting the 2- to 4-kHz range, the transients caused by the fingers against the string of an acoustical guitar can be made more pronounced. A high-frequency shelving boost above the 1- to 2-kHz range increases the "crispness" in percussion instruments such as the bongo or snare drums.

Noise reduction system. The overall dynamic range of human hearing is over 120 dB. However, most recording and transmission mediums have a much smaller dynamic range. The music to be recorded must be above the sound background or noise. If the background noise is around 30 dB, the dynamic range available for the music is only 90 dB, requiring dynamic range compression for noise reduction.

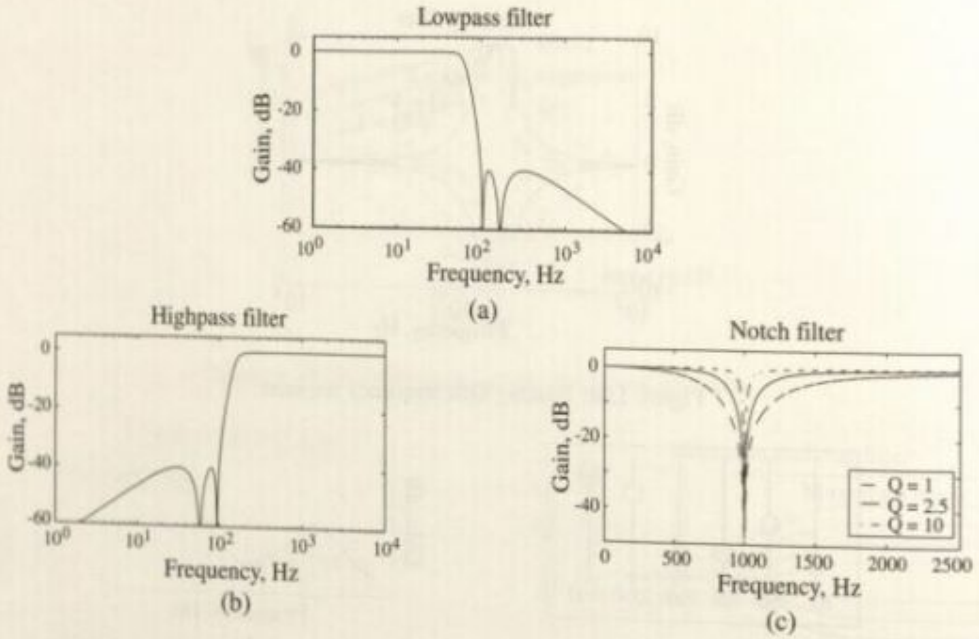


Figure 1.26: Frequency responses of other types of filters: (a) lowpass filter, (b) highpass filter, and (c) notch filter.

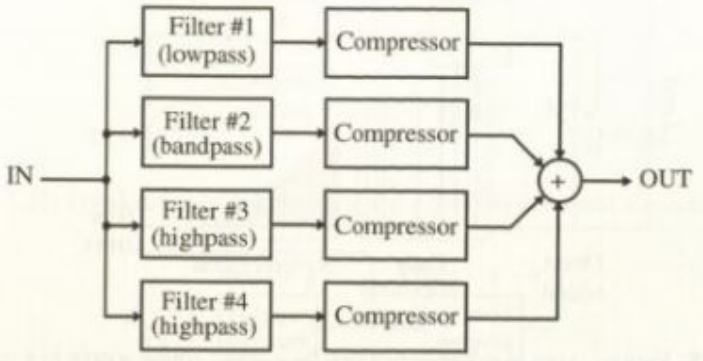
A noise reduction system consists of two parts. The first part provides the compression during the recording mode, while the second part provides the complementary expansion during the playback mode. To this end, the most popular methods in musical recording are the Dolby noise reduction schemes, of which there are several types [Ear76], [Hub89], [Wor89].

In the Dolby A-type method used in professional recording, for the recording mode, the audio signal is split into four frequency bands by a bank of four filters; separate compression is provided in each band and the outputs of the compressors are combined, as indicated in Figure 1.27. Moreover, the compression in each band is restricted to a 20-dB input range from −40 to −20 dB. Below the lower threshold (−40 dB), very low level signals are boosted by 10 dB, and above the upper threshold (−20 dB), the system has unity gain, passing the high-level signals unaffected. The transfer characteristic for the record mode is thus as shown in Figure 1.28.

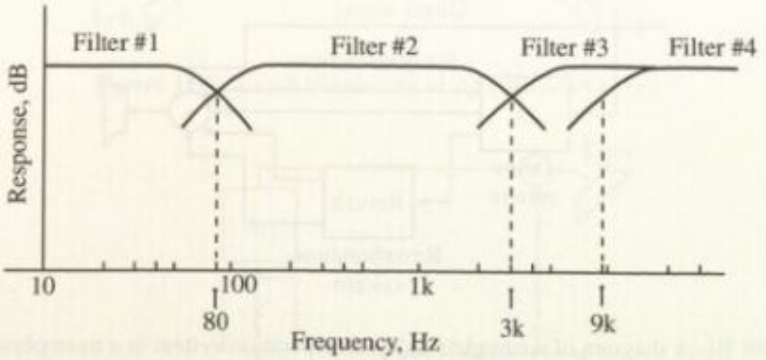
In the playback mode, the scheme is essentially the same as that in the recording mode, except here the compressors are replaced by expanders with complementary transfer characteristics, as indicated in Figure 1.28. Here, the expansion is limited to a 10-dB input range from −30 to −20 dB. Above the upper threshold (−20 dB), very high level signals are cut by 10 dB, while below the lower threshold (−30 dB), the system has unity gain passing the low-level signals unaffected.

Note that for each band, a 2-to-1 compression is followed by a 1-to-2 complementary expansion such that the dynamic range of the signal at the input of the compressor is exactly equal to that at the expander output. This type of overall signal processing operation is often called *companding*. Moreover, the companding operation in one band has no effect on a signal in another band and may often be masked by other bands with no companding.

Delay and reverberation systems. Music generated in an inert studio does not sound natural compared to the music performed inside a room, such as a concert hall. In the latter case, the sound waves propagate in all directions and reach the listener from various directions and at various times, depending



(a)



(b)

Figure 1.27: The Dolby A-type noise reduction scheme for the recording mode. (a) Block diagram and (b) frequency responses of the four filters with cutoff frequencies as shown.

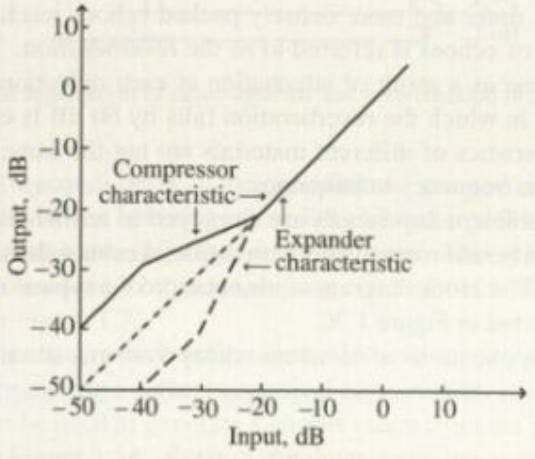


Figure 1.28: Compressor and expander transfer characteristic for the Dolby A-type noise reduction scheme.

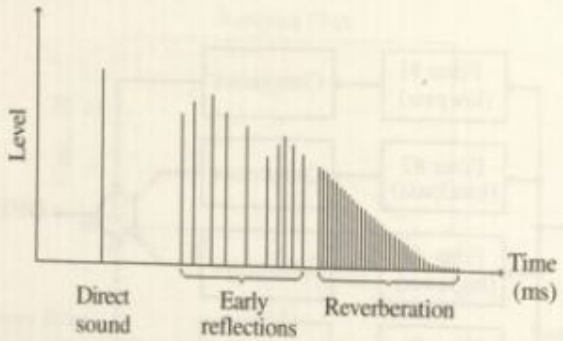


Figure 1.29: Various types of echoes generated by a single sound source in a room.

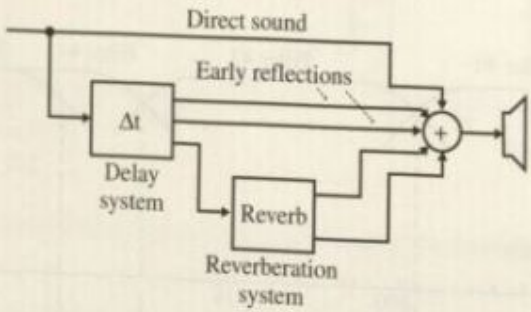


Figure 1.30: Block diagram of a complete delay-reverberation system in a monophonic system.

on the distance travelled by the sound waves from the source to the listener. The sound wave coming directly to the listener, called the *direct sound*, reaches first and determines the listener's perception of the location, size, and nature of the sound source. This is followed by a few closely spaced echoes, called *early reflections*, generated by reflections of sound waves from all sides of the room and reaching the listener at irregular times. These echoes provide the listener's subconscious cues as to the size of the room. After these early reflections, more and more densely packed echoes reach the listener due to multiple reflections. The latter group of echoes is referred to as the *reverberation*. The amplitude of the echoes decays exponentially with time as a result of attenuation at each reflection. Figure 1.29 illustrates this concept. The period of time in which the reverberation falls by 60 dB is called the *reverberation time*. Since the absorption characteristics of different materials are not the same at different frequencies, the reverberation time varies from frequency to frequency.

Delay systems with adjustable delay factors are employed to artificially create the early reflections. Electronically generated reverberation combined with artificial echo reflections are usually added to the recordings made in a studio. The block diagram representation of a typical delay-reverberation system in a monophonic system is depicted in Figure 1.30.

There are various other applications of electronic delay systems, some of which are described next [Ear76].

Special effects. By feeding in the same sound signal through an adjustable delay and gain control, as indicated in Figure 1.31, it is possible to vary the localization of the sound source from the left speaker to the right for a listener located on the plane of symmetry. For example, in Figure 1.31, a 0-dB loss in

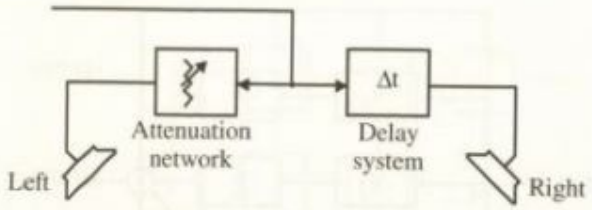


Figure 1.31: Localization of sound source using delay systems and attenuation network.

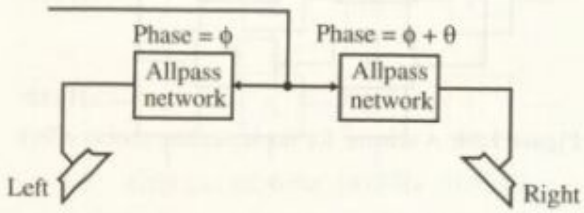


Figure 1.32: Sound broadening using allpass networks.

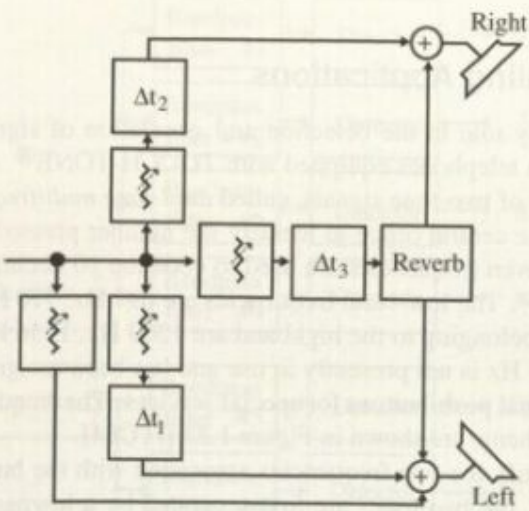


Figure 1.33: A possible application of delay systems and reverberation in a stereophonic system.

the left channel and a few milliseconds delay in the right channel give the impression of a localization of the sound source at the left. However, lowering of the left-channel signal level by a few-dB loss results in a phantom image of the sound source moving toward the center. This scheme can be further extended to provide a degree of *sound broadening* by phase shifting one channel with respect to the other through allpass networks[6] as shown in Figure 1.32.

Another application of the delay-reverberation system is in the processing of a single track into a pseudo-stereo format while simulating a natural acoustical environment, as illustrated in Figure 1.33.

The delay system can also be used to generate a chorus effect from the sound of a soloist. The basic scheme used is illustrated in Figure 1.34. Each of the delay units has a variable delay controlled by a low-frequency, pseudo-random noise source to provide a random pitch variation [Ble78].

[6] An allpass network is characterized by a magnitude spectrum that is equal to one for all frequencies.

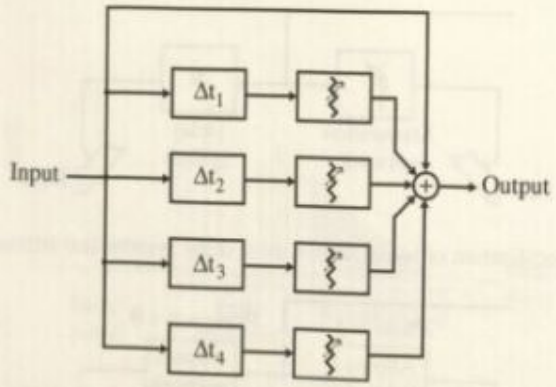


Figure 1.34: A scheme for implementing chorus effect.

It should be pointed out here that additional signal processing is employed to make the stereo submaster developed by the sound engineer more suitable for the record-cutting lathe or the cassette tape duplicator.

1.4.2 Telephone Dialing Applications

Signal processing plays a key role in the detection and generation of signaling tones for push-button telephone dialing [Dar76]. In telephones equipped with TOUCH-TONE® dialing, the pressing of each button generates a unique set of two-tone signals, called *dual-tone multifrequency* (DTMF) *signals*, that are processed at the telephone central office to identify the number pressed by determining the two associated tone frequencies. Seven frequencies are used to code the 10 decimal digits and the two special buttons marked " * " and " # ". The low-band frequencies are 697 Hz, 770 Hz, 852 Hz, and 941 Hz. The remaining three frequencies belonging to the highband are 1209 Hz, 1336 Hz, and 1477 Hz. The fourth high-band frequency of 1633 Hz is not presently in use and has been assigned for future applications to permit the use of four additional push-buttons for special services. The frequency assignments used in the TOUCH-TONE® dialing scheme are shown in Figure 1.35 [ITU84].

The scheme used to identify the two frequencies associated with the button that has been pressed is shown in Figure 1.36. Here, the two tones are first separated by a lowpass and a highpass filter. The passband cutoff frequency of the lowpass filter is slightly above 1000 Hz, whereas that of the highpass filter is slightly below 1200 Hz. The output of each filter is next converted into a square wave by a limiter and then processed by a bank of bandpass filters with narrow passbands. The four bandpass filters in the low-frequency channel have center frequencies at 697 Hz, 770 Hz, 852 Hz, and 941 Hz. The four bandpass filters in the high-frequency channel have center frequencies at 1209 Hz, 1336 Hz, 1477 Hz, and 1633 Hz. The detector following each bandpass filter develops the necessary dc switching signal if its input voltage is above a certain threshold.

All the signal processing functions described above are usually implemented in practice in the analog domain. However, increasingly, these functions are being implemented using digital techniques.[7]

[7] See Section 14.1.

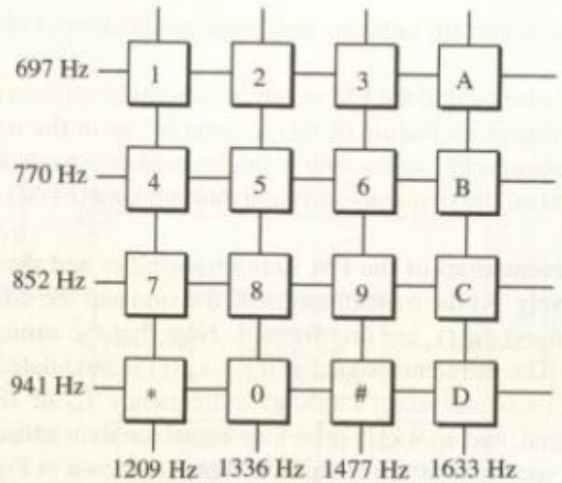


Figure 1.35: The tone frequency assignments for TOUCH-TONE® dialing.

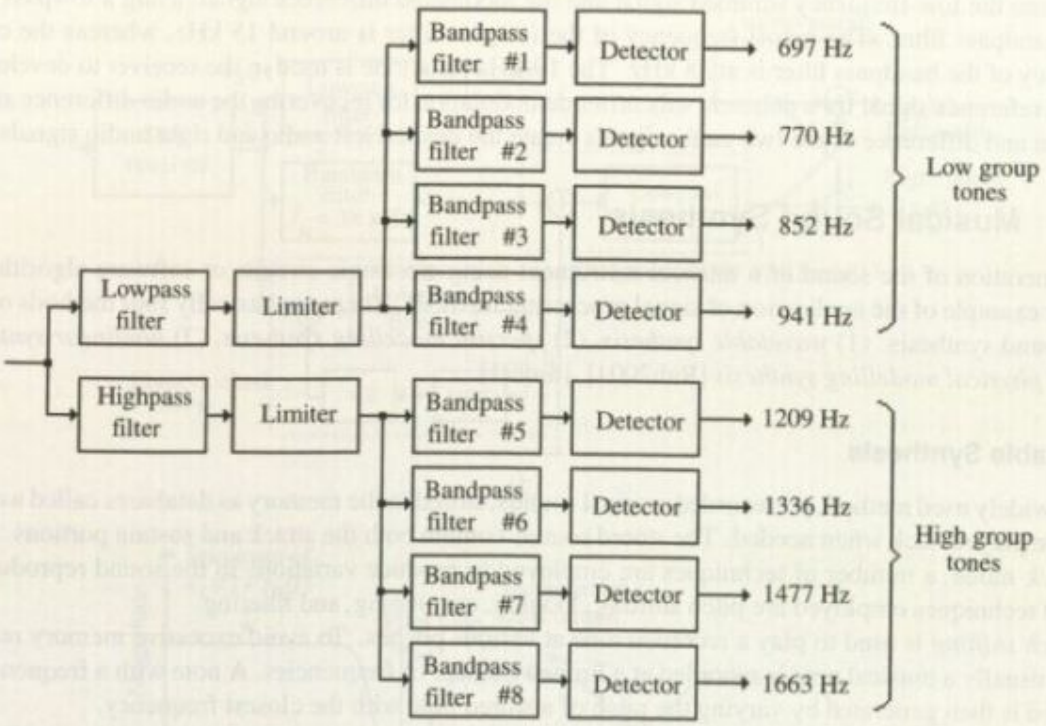


Figure 1.36: The tone detection scheme for TOUCH-TONE® dialing.

1.4.3 FM Stereo Applications

For wireless transmission of a signal occupying a low-frequency range, such as an audio signal, it is necessary to transform the signal to a high-frequency range by modulating it onto a high-frequency carrier. At the receiver, the modulated signal is demodulated to recover the low-frequency signal. The signal processing operations used for wireless transmission are modulation, demodulation, and filtering. Two

commonly used modulation schemes for radio are amplitude modulation (AM) and frequency modulation (FM).

We next review the basic idea behind the FM stereo broadcasting and reception scheme as used in the United States [Cou83]. An important feature of this scheme is that at the receiving end, the signal can be heard over a standard monaural FM radio with a single speaker or over a stereo FM radio with two speakers. The system is based on the frequency-division multiplexing (FDM) method described earlier in Section 1.2.5.

The block diagram representations of the FM stereo transmitter and the receiver are shown in Figure 1.37(a) and (b), respectively. At the transmitting end, the sum and the difference of the left and right channel audio signals, $s_L(t)$ and $s_R(t)$, are first formed. Note that the summed signal $s_L(t) + s_R(t)$ is used in monaural FM radio. The difference signal $s_L(t) - s_R(t)$ is modulated using the double-sideband suppressed carrier (DSB-SC) scheme[8] using a subcarrier frequency f_{sc} of 38 kHz. The summed signal, the modulated difference signal, and a 19-kHz pilot tone signal are then added, developing the composite baseband signal $s_B(t)$. The spectrum of the composite signal is shown in Figure 1.37(c). The baseband signal is next modulated onto the main carrier frequency f_c using the frequency modulation method. At the receiving end, the FM signal is demodulated to derive the baseband signal $s_B(t)$, which is then separated into the low-frequency summed signal and the modulated difference signal, using a lowpass filter and a bandpass filter. The cutoff frequency of the lowpass filter is around 15 kHz, whereas the center frequency of the bandpass filter is at 38 kHz. The 19-kHz pilot tone is used in the receiver to develop the 38-kHz reference signal for a coherent subcarrier demodulation for recovering the audio difference signal. The sum and difference of the two audio signals create the desired left audio and right audio signals.

1.4.4 Musical Sound Synthesis

The generation of the sound of a musical instrument using electronic circuits or software algorithms is another example of the application of signal processing methods. There are basically four methods of musical sound synthesis: (1) *wavetable synthesis*, (2) *spectral modelling synthesis*, (3) *nonlinear synthesis*, and (4) *physical modelling synthesis* [Rab2001], [Smi91].

Music Demo 1

Wavetable Synthesis

In this widely used method, prerecorded musical sounds, stored in the memory as databases called *wavetables*, are played back when needed. The stored sounds contain both the attack and sustain portions. In the playback mode, a number of techniques are employed to produce variations in the sound reproduction. Typical techniques employed are pitch shifting, looping, enveloping, and filtering.

Pitch shifting is used to play a recorded tone at various pitches. To avoid excessive memory requirements, usually a musical note is recorded at a limited number of frequencies. A note with a frequency not recorded is then generated by varying the pitch of a stored note with the closest frequency.

Looping is used to extend the playback time of a recorded note. It is implemented by reading out repeatedly the stored data.

As the attack, decay, sustain, and release portions of the envelope of a recorded note are often lost due to looping, they can be regenerated or changed by *enveloping*. It is implemented by changing the amplitude of the recorded note with a time-varying gain function.

Filtering is used to modify the spectral properties of a tone. It provides an improved time-dependent control of the recorded sound and is often used to correct the amplitude change caused by enveloping.

[8] See Section 1.2.4.

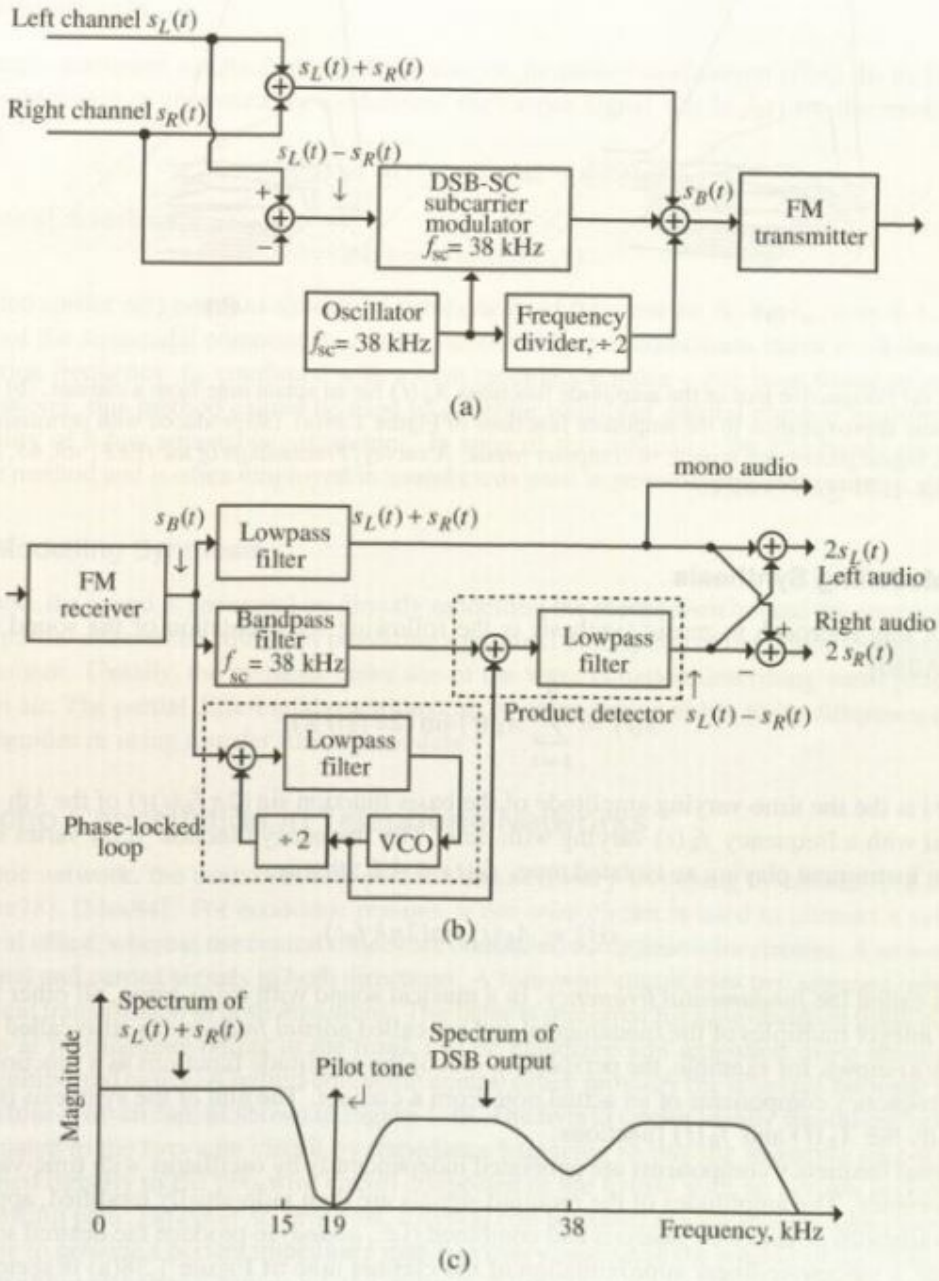
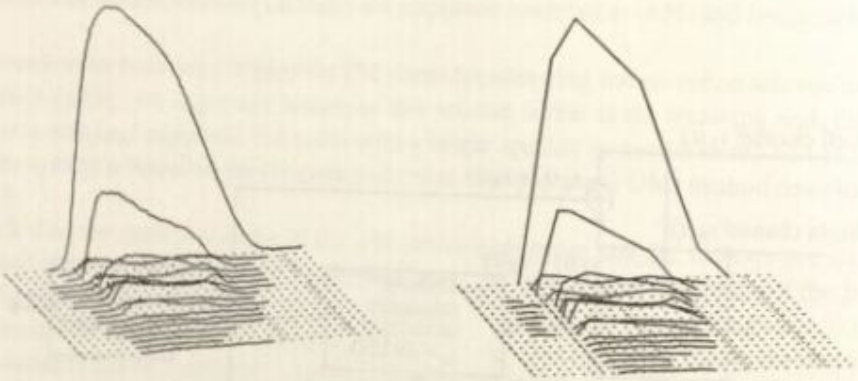


Figure 1.37: The FM stereo system: (a) transmitter, (b) receiver, and (c) spectrum of the composite baseband signal $s_B(t)$.



(a) (b)

Figure 1.38: (a) Perspective plot of the amplitude functions $A_k(t)$ for an actual note from a clarinet. (b) A piecewise-linear approximation to the amplitude functions of Figure 1.38(a). (Reproduced with permission from J. A. Moorer, Signal processing aspects of computer music: A survey, *Proceedings of the IEEE*, vol. 65, August 1977, pp. 1108–1137 ©1977 IEEE.)

Music Demo 2

Spectral Modelling Synthesis

The basis of this approach to music synthesis is the following representation of the sound signal $s(t)$ [Moo77], [All80]:

$$s(t) = \sum_{k=1}^{N} A_k(t) \sin(2\pi f_k(t)t), \tag{1.19}$$

where $A_k(t)$ is the the time-varying amplitude of the basis function $\sin(2\pi f_k(t)t)$ of the kth component of the signal with a frequency $f_k(t)$ varying with time. The frequency function $f_k(t)$ varies slowly with time. For an instrument playing an isolated tone, $f_k(t) = kf_o$; that is,

$$s(t) = A_k(t) \sin(2\pi k f_o t), \tag{1.20}$$

where f_o is called the *fundamental frequency*. In a musical sound with many tones, all other frequencies are usually integer multiples of the fundamental and are called *partial frequencies*, also called *harmonics*. Figure 1.38(a) shows, for example, the perspective plot of the amplitude functions as a function of time of 17 partial frequency components of an actual note from a clarinet. The aim of the synthesis is to produce electronically the $A_k(t)$ and $f_k(t)$ functions.

The partial frequency components are generated independently by oscillators with time-varying oscillation frequencies. The amplitudes of the required signals are then individually modified, approximating the actual variations obtained by analysis and combined (i.e., *added*) to produce the desired sound signal. For example, a piecewise-linear approximation of the clarinet note of Figure 1.38(a) is sketched in Figure 1.38(b) and can be used to generate a reasonable replica of the note. Usually, some alterations to the amplitude and frequency functions may be needed before the music that is generated sounds as close as possible to that of the original instrument.

A recent variation to the additive synthesis is the *granular synthesis*. In this method, the basis functions of Eq. (1.19) are concentrated in frequency and time and are called *grains* or *atoms*. These basis functions

are generated in one of several ways, such as windowed segments of sine waves, from wavetables, with wavelet expansions.[9]

Nonlinear Synthesis

Music Demo 3

A fairly simple nonlinear synthesis method is based on *frequency modulation* (FM). In its barest form, the sound signal $s(t)$ is generated by modulating the carrier signal $\sin(2\pi f_0 t)$ by the modulator $\phi(t)$, resulting in

$$s(t) = A(t)\sin(2\pi f_0 t + \phi(t)). \tag{1.21}$$

For a sinusoidal modulator

$$\phi(t) = \kappa \sin(2\pi f_m t), \tag{1.22}$$

the modulated carrier $s(t)$ contains sinusoidal components of frequencies $f_0 + nf_m, n = 0, 1, 2, \ldots$. The amplitudes of the sinusoidal components can be varied using the modulation index κ. A small value of the modulation frequency f_m combined with a high modulation index κ has been found to produce rich sounds. However, this method cannot be used to generate sounds of natural musical instruments due to the availability of a few adjustable parameters. In spite of this difficulty, the FM-based approach is an inexpensive method and is often employed in sound cards used in personal computers and in synthesizers.

Physical Modelling Synthesis

Music Demo 4

In this method, the sound is generated by directly modelling the mechanism behind the sound production. It results in partial differential equations providing a physical description of the major vibrating structures of the instrument. Usually, the methods make use of the wave equation describing wave propagation in solids and in air. The partial differential equations can be solved using either finite difference methods or digital waveguides or using transfer function models.

1.4.5 Echo Cancellation in Telephone Networks

In a telephone network, the central offices perform the necessary switching to connect two subscribers [Dut80], [Fre78], [Mes84]. For economic reasons, a *two-wire* circuit is used to connect a subscriber to his/her central office, whereas the central offices are connected using *four-wire* circuits. A two-wire circuit is bidirectional and carries signals in both directions. A four-wire circuit uses two separate unidirectional paths for signal transmission in both directions. The latter is preferred for long-distance trunk connections since signals at intermediate points in the trunk can be equalized and amplified using repeaters and, if necessary, multiplexed easily. A hybrid coil in the central office provides the interface between a two-wire circuit and a four-wire circuit, as shown in Figure 1.39. The hybrid circuit ideally should provide a perfect impedance match to the two-wire circuit by impedance balancing so that the incoming four-wire receive signal is passed directly to the two-wire circuit connected to the hybrid with no portion appearing in the four-wire transmit path. However, to save cost, a hybrid coil is shared among several subscribers. Thus, it is not possible to provide a perfect impedance match in every case since the lengths of the subscriber lines vary. The resulting imbalance causes a large portion of the incoming received signal from the distance talker to appear in the transmit path, and it is returned to the talker as an echo. Figure 1.40 illustrates the normal transmission between a talker and a listener, as well as two possible major echo paths.

[9] See Section 14.6.1.

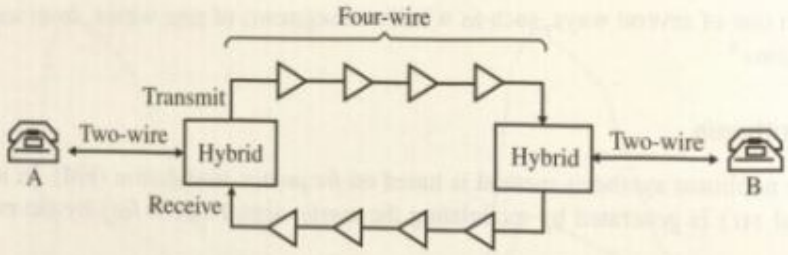


Figure 1.39: Basic 2/4-wire interconnection scheme.

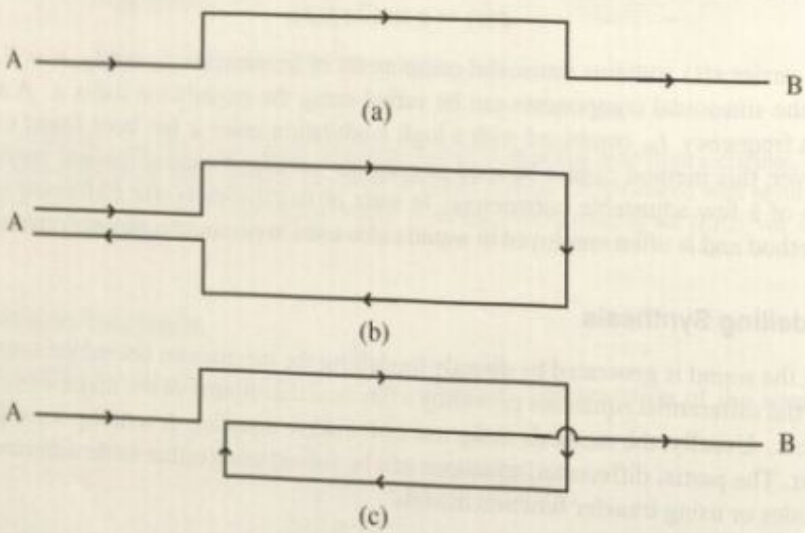


Figure 1.40: Various signal paths in a telephone network. (a) Transmission path from talker A to listener B, (b) echo path for talker A, and (c) echo path for listener B.

The effect of the echo can be annoying to the talker, depending on the amplitude and delay of the echo, that is, on the length of the trunk circuit. The effect of the echo is worst for telephone networks involving geostationary satellite circuits, where the echo delay is about 540 ms.

Several methods are followed to reduce the effect of the echo. In trunk circuits up to 3000 km in length, adequate reduction of the echo is achieved by introducing additional signal loss in both directions of the four-wire circuit. In this scheme, an improvement in the signal-to-echo ratio is realized since the echo undergoes loss in both directions, while the signals are attenuated only once.

For distances greater than 3000 km, echoes are controlled by means of an echo suppressor inserted in the trunk circuit, as indicated in Figure 1.41(a). The device is essentially a voice-activated switch implementing two functions. It first detects the direction of the conversation and then blocks the opposite path in the four-wire circuit. Even though it introduces distortion when both subscribers are talking by clipping parts of the speech signal, the echo suppressor has provided a reasonably acceptable solution for terrestrial transmission.

For telephone conversation involving satellite circuits, an elegant solution is based on the use of an echo canceler. The circuit generates a replica of the echo using the signal in the receive path and subtracts it from the signal in the transmit path, as indicated in Figure 1.41(b). Basically, it is an adaptive filter

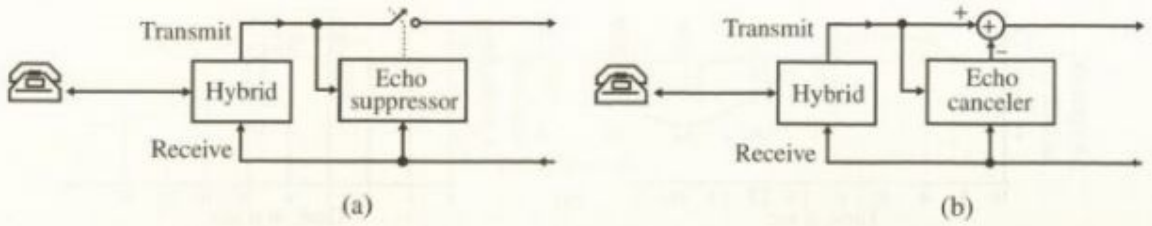


Figure 1.41: (a) Echo suppression scheme and (b) echo cancellation scheme.

Figure 1.42: Scheme for the digital processing of an analog signal.

structure whose parameters are adjusted using certain adaptation algorithms until the residual signal is satisfactorily minimized.[10] Typically, an echo reduction of about 40 dB is considered satisfactory in practice. To eliminate the problem generated when both subscribers are talking, the adaptation algorithm is disabled when the signal in the transmit path contains both the echo and the signal generated by the speaker closer to the hybrid coil.

1.5 Why Digital Signal Processing?[11]

In some sense, the origin of digital signal processing techniques can be traced back to the seventeenth century when finite difference methods, numerical integration methods, and numerical interpolation methods were developed to solve physical problems involving continuous variables and functions. The more recent interest in digital signal processing arose in the 1950s with the availability of large digital computers. Initial applications were primarily concerned with the simulation of analog signal processing methods. Around the beginning of the 1960s, researchers began to consider digital signal processing as a separate field by itself. Since then, there have been significant and myriad developments and breakthroughs in both theory and applications of digital signal processing.

Digital processing of an analog signal consists basically of three steps: conversion of the analog signal into a digital form; processing of the digital version; and finally, conversion of the processed digital signal back into an analog form. Figure 1.42 shows the overall scheme in a block diagram form.

Since the amplitude of the analog input signal varies with time, a *sample-and-hold* (S/H) circuit is used first to sample the analog input at periodic intervals and hold the sampled value constant at the input of the *analog-to-digital* (A/D) *converter* to permit accurate digital conversion. The input to the A/D converter is a staircase-type analog signal if the S/H circuit holds the sampled value until the next sampling instant. The output of the A/D converter is a binary data stream that is next processed by the digital processor implementing the desired signal processing algorithm. The output of the digital processor, another binary data stream, is then converted into a staircase-type analog signal by the *digital-to-analog* (D/A) *converter*. The lowpass filter at the output of the D/A converter then removes all undesired high-

[10] For a review of adaptive filtering methods, see [Cio93].

[11] This section has been adapted from *Handbook for Digital Signal Processing*, Sanjit K. Mitra and James F. Kaiser, Eds., ©1993, John Wiley & Sons. Adapted by permission of John Wiley & Sons.

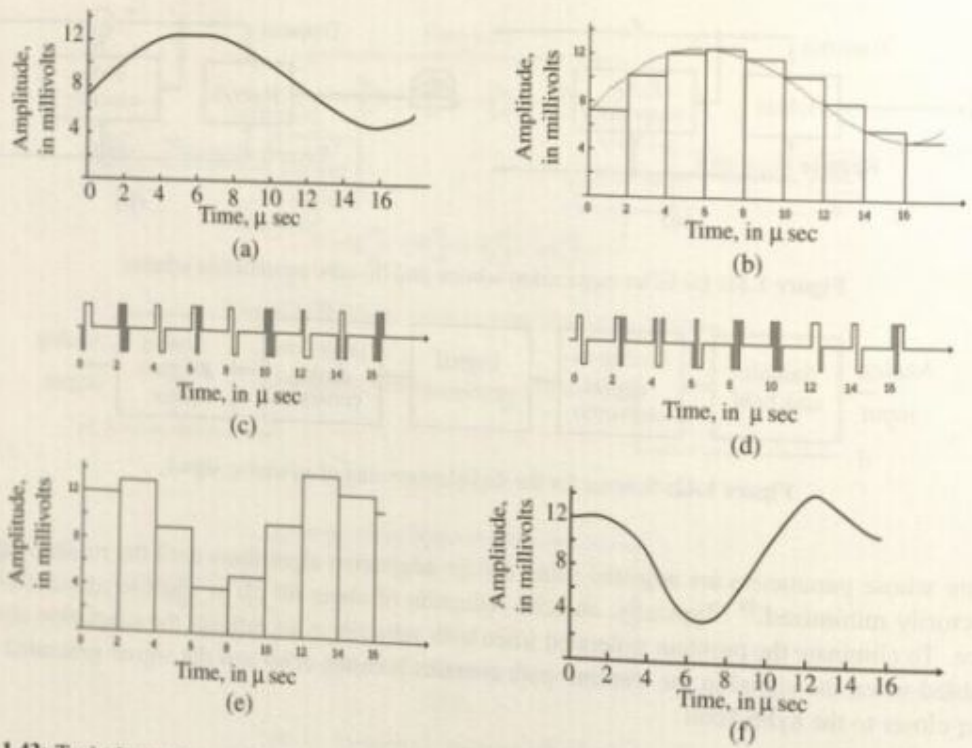


Figure 1.43: Typical waveforms of signals appearing at various stages in Figure 1.42. (a) Analog input signal, (b) output of the S/H circuit, (c) A/D converter output, (d) output of the digital processor, (e) D/A converter output, and (f) analog output signal. In (c) and (d), the digital HIGH and LOW levels are shown as positive and negative pulses for clarity.

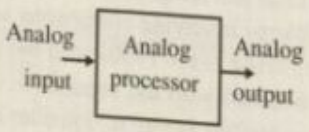


Figure 1.44: Analog processing of analog signals.

frequency components and delivers at its output the desired processed analog signal. Figure 1.43 illustrates the waveforms of the pertinent signals at various stages in the above process, where for clarity the two levels of the binary signals are shown as a positive and a negative pulse, respectively.

In contrast to the above, a direct analog processing of an analog signal is conceptually much simpler since it involves only a single processor, as illustrated in Figure 1.44. It is therefore natural to ask what the advantages are of digital processing of an analog signal.

There are of course many advantages in choosing digital signal processing. The most important ones are discussed next [Bel2000], [Pro96].

Unlike analog circuits, the operation of digital circuits does not depend on precise values of the digital signals. As a result, a digital circuit is less sensitive to tolerances of component values and is fairly independent of temperature, aging, and most other external parameters. A digital circuit can be reproduced easily in volume quantities and does not require any adjustments either during construction or later while in use. Moreover, it is amenable to full integration, and with the recent advances in *very large scale*

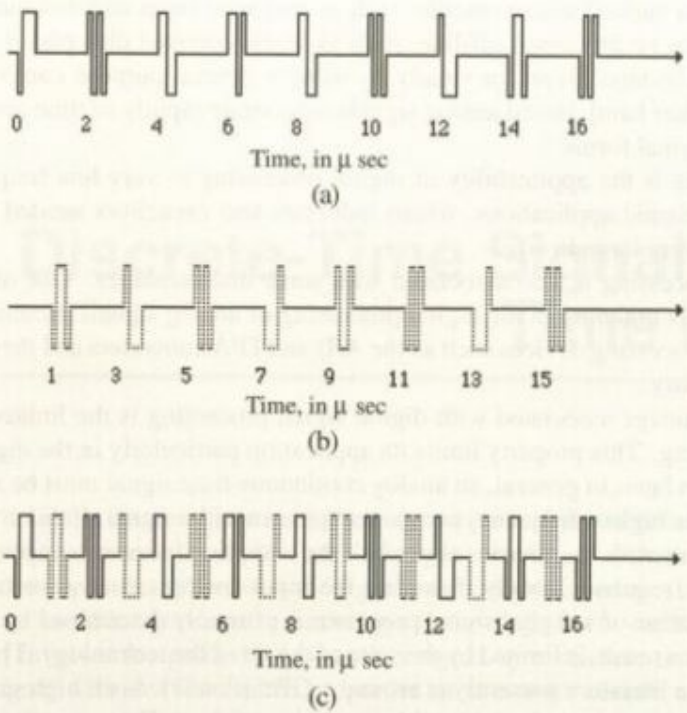


Figure 1.45: Illustration of the time-sharing concept. The signal shown in (c) has been obtained by time-multiplexing the signals shown in (a) and (b).

integrated (VLSI) circuits, it has been possible to integrate highly sophisticated and complex digital signal processing systems on a single chip.

In a digital processor, the signals and the coefficients describing the processing operation are represented as binary words. Thus, any desirable accuracy can be achieved by simply increasing the wordlength, subject to cost limitation. Moreover, the dynamic ranges for signals and coefficients can be increased still further by using floating-point arithmetic if necessary.

Digital processing allows the sharing of a given processor among a number of signals by timesharing, thus reducing the cost of processing per signal. Figure 1.45 illustrates the concept of timesharing, where two digital signals are combined into one by time-division multiplexing. The multiplexed signal can then be fed into a single processor. By switching the processor coefficients prior to the arrival of each signal at the input of the processor, the processor can be made to look like two different systems. Finally, by demultiplexing the output of the processor, the processed signals can be separated.

Digital implementation permits easy adjustment of processor characteristics during processing, such as that needed in implementing adaptive filters. Such adjustments can be simply carried out by periodically changing the coefficients of the algorithm representing the processor characteristics. Another application of the changing of coefficients is in the realization of systems with programmable characteristics, such as frequency selective filters with adjustable cutoff frequencies. Filter banks with guaranteed complementary frequency response characteristics are easily implemented in digital form.

Digital implementation allows the realization of certain characteristics not possible with analog implementation, such as exact linear phase and multirate processing. Digital circuits can be cascaded without any loading problems, unlike analog circuits. Digital signals can be stored almost indefinitely without any

loss of information on various storage media, such as magnetic tapes and disks and optical disks. Such stored signals can later be processed off-line, such as in the compact disk player, the digital video disk player, the digital audio tape player, or simply by using a general purpose computer as in seismic data processing. On the other hand, stored analog signals deteriorate rapidly as time progresses and cannot be recovered in their original forms.

Another advantage is the applicability of digital processing to very low frequency signals, such as those occurring in seismic applications, where inductors and capacitors needed for analog processing would be physically very large in size.

Digital signal processing is also associated with some disadvantages. One obvious disadvantage is the increased system complexity in the digital processing of analog signals because of the need for additional pre- and postprocessing devices such as the A/D and D/A converters and their associated filters and complex digital circuitry.

A second disadvantage associated with digital signal processing is the limited range of frequencies available for processing. This property limits its application particularly in the digital processing of analog signals. As shown later, in general, an analog continuous-time signal must be sampled at a frequency that is at least twice the highest frequency component present in the signal. If this condition is not satisfied, then signal components with frequencies above half the sampling frequency appear as signal components below this particular frequency, totally distorting the input analog signal waveform. The available frequency range of operation of a digital signal processor is primarily determined by the S/H circuit and the A/D converter and, as a result, is limited by the state of the art of the technology. The highest sampling frequency reported in the literature presently is around 1 GHz [Pou87]. Such high sampling frequencies are not usually used in practice since the achievable resolution of the A/D converter, given by the wordlength of the digital equivalent of the analog sample, decreases with an increase in the speed of the converter. For example, the reported resolution of an A/D converter operating at 1 GHz is 6 bits [Pou87]. On the other hand, in most applications, the required resolution of an A/D converter is from 12 bits to around 16 bits. Consequently, a sampling frequency of at most 10 MHz is presently a practical upper limit. This upper limit, however, is getting larger and larger with advances in technology.

The third disadvantage stems from the fact that digital systems are constructed using active devices that consume electrical power. For example, the WE DSP32C Digital Signal Processor chip contains over 405,000 transistors and dissipates around 1 watt. On the other hand, a variety of analog processing algorithms can be implemented using passive circuits employing inductors, capacitors, and resistors that do not need power. Moreover, active devices are less reliable than passive components.

Another disadvantage of digital signal processing is due to the effects resulting from the algorithms implemented with finite precision arithmetic in hardware or software. By properly designing the algorithm and its implementation, the effects of the finite wordlength can be minimized in most cases.

However, the advantages far outweigh the disadvantages in various applications, and with the continuing decrease in the cost of digital processor hardware, applications of digital signal processing are increasing rapidly.

Chapter 2

Discrete-Time Signals in the Time Domain

As indicated in Figure 1.1(c), a discrete-time signal in its most basic form is defined at equally spaced discrete values of time, the independent variable, with the signal amplitude at these discrete times being a continuous variable. Consequently, a discrete-time signal can be represented as a sequence of numbers, with the independent time variable represented as an integer in the range from $-\infty$ to $+\infty$. Discrete-time signal processing, then, involves the processing of such a signal by a discrete-time system to develop another discrete-time signal with more desirable properties or to extract certain information about the original signal.

In many applications involving continuous-time signals, it is becoming increasingly more attractive to process the continuous-time signal by discrete-time signal processing methods. To this end, the continuous-time signal is first converted into an "equivalent" discrete-time signal by periodic sampling; the resulting signal then is processed by a discrete-time system to generate another discrete-time signal, and the latter is converted into an equivalent continuous-time signal, if necessary. As we shall show later in Section 3.8.1, under certain (ideal) conditions, the conversion of a continuous-time signal can be carried out such that the discrete-time equivalent has all the information contained in the original continuous-time signal and, if necessary, can be converted back into the original continuous-time signal without any distortion.

Thus, to understand the theory of digital signal processing and the design of discrete-time systems, we need to know how to mathematically represent discrete-time signals and systems. Such representations can be either in the time domain or in the frequency domain. In this chapter and the following chapter we restrict our attention to discrete-time signals.

In this chapter, we consider the mathematical representations of discrete-time signals in the time domain and several concepts associated with such representations. We first consider the representation in the time domain. We then describe several elementary operations on discrete-time signals and the devices used to implement these operations. These operations form the basic building blocks in developing discrete-time systems for the processing of discrete-time signals. We next discuss various classifications of the discrete-time signals that often can be used to simplify the signal processing algorithms. A number of basic operations that generate other sequences from one or more sequences are described next. We then describe several basic discrete-time signals or sequences that play important roles in the time-domain characterization of arbitrary discrete-time signals and discrete-time systems. Next we discuss the relation between a continuous-time signal and its discrete-time version, obtained by sampling the former at uniform intervals. In addition, we point out the condition under which the discrete-time signal obtained by uniform sampling can uniquely represent the parent continuous-time signal. One type of similarity

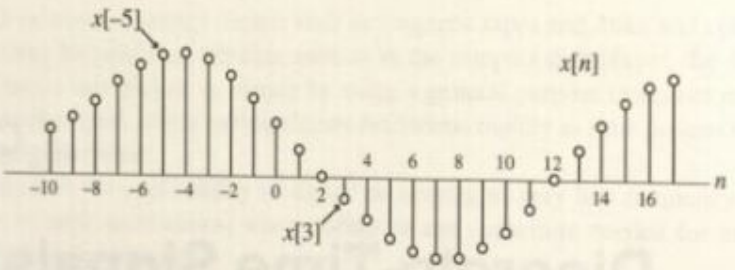


Figure 2.1: Graphical representation of a discrete-time sequence $\{x[n]\}$.

measure between a pair of discrete-time signals is given by the cross-correlation sequence which then is introduced and some of its properties are investigated. Finally the chapter concludes with a qualitative discussion of discrete-time random signals.

Throughout this chapter and successive chapters, we make extensive use of MATLAB to illustrate, through computer simulations, the various concepts introduced.

2.1 Time-Domain Representation

As indicated earlier, in digital signal processing, signals are represented as sequences of numbers called *samples*. A sample value of a typical discrete-time signal or sequence is a function of the independent integer-valued variable defined in the range $-\infty$ and ∞. For example, $x[n]$ is the amplitude of a discrete-time signal $\{x[n]\}$ at the time instant n where $-\infty < n < \infty$. It should be noted that $x[n]$ is defined only for integer values of n and is undefined for noninteger values of n. If a discrete-time signal is written as a sequence of numbers inside braces, the location of the sample value associated with the time index $n = 0$ is indicated by an arrow $\uparrow$ under it. The sample values to its right are for positive values of n, and the sample values to its left are for negative values of n. An example of a discrete-time signal with real-valued samples is given by

$$\{x[n]\} = \{\ldots, 0.95, -0.2, \underset{\uparrow}{2.17}, 1.1, 0.2, -3.67, 2.9, -0.8, 4.1, \ldots\}. \tag{2.1}$$

For the above signal, $x[-1] = -0.2$, $x[0] = 2.17$, $x[1] = 1.1$, and so on. The graphical representation of a sequence $\{x[n]\}$ with real-valued samples is illustrated in Figure 2.1.

In some applications a discrete-time sequence $\{x[n]\}$ is generated by periodically sampling a continuous-time signal $x_a(t)$ at uniform time intervals:

$$x[n] = x_a(t)|_{t=nT} = x_a(nT), \qquad n = \ldots, -2, -1, 0, 1, 2, \ldots \tag{2.2}$$

as illustrated in Figure 2.2. The spacing T between two consecutive samples in Eq. (2.2) is called the *sampling interval* or *sampling period*. The reciprocal of the sampling interval T, denoted as F_T, is called the *sampling frequency*

$$F_T = \frac{1}{T}. \tag{2.3}$$

The unit of sampling frequency here is cycles per second, or Hertz (Hz), if the sampling period has physical dimensions of time. In the case of an image, a spatial sampling is used in both horizontal and vertical directions. In this case, the horizontal and vertical sampling frequencies are in cycles per unit of distance in the two directions defining the two sampling periods.

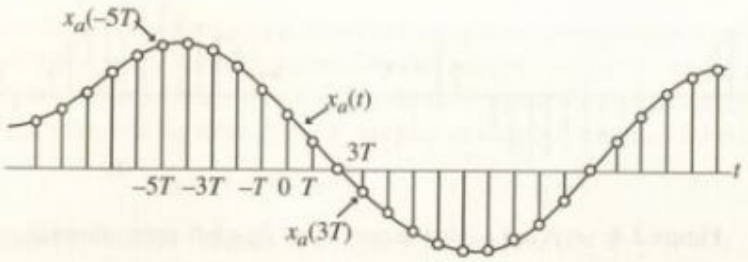


Figure 2.2: Sequence generated by sampling a continuous-time signal $x_a(t)$.

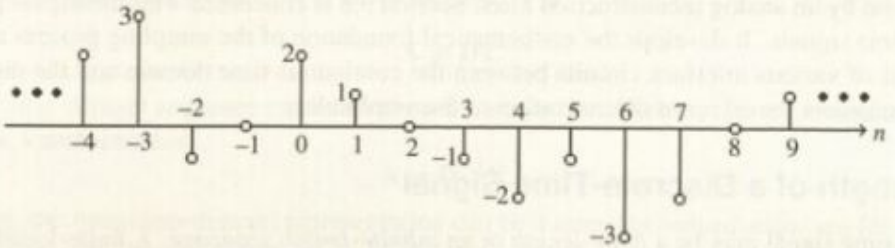


Figure 2.3: A digital signal.

It should be noted that, whether or not a sequence $\{x[n]\}$ has been obtained by sampling, the quantity $x[n]$ is called the *nth sample* of the sequence. For a sequence $\{x[n]\}$, the nth sample value $x[n]$ can, in general, take any real or complex value. If $x[n]$ is real for all values of n, then $\{x[n]\}$ is a *real sequence*. On the other hand, if the nth sample value is complex for one or more values of n, then it is a *complex sequence*. By separating the real and imaginary parts of $x[n]$, we can write a complex sequence $\{x[n]\}$ as

$$\{x[n]\} = \{x_{\text{re}}[n]\} + j\,\{x_{\text{im}}[n]\}, \tag{2.4}$$

where $x_{\text{re}}[n]$ and $x_{\text{im}}[n]$ are the real and the imaginary parts of $x[n]$, respectively, and are real sequences. For a real sequence, $x_{\text{im}}[n] = 0$, and for a purely imaginary sequence, $x_{\text{re}}[n] = 0$, for all values of n. The complex conjugate sequence of $\{x[n]\}$ is usually denoted by $\{x^*[n]\}$ and written as $\{x^*[n]\} = \{x_{\text{re}}[n]\} - j\,\{x_{\text{im}}[n]\}$.[1] Often the braces are ignored to denote a sequence if there is no ambiguity.

As defined in the previous chapter, there are basically two types of discrete-time signals: sampled-data signals, in which the samples are continuous-valued; and digital signals, in which the samples are discrete-valued. The pertinent signals in a practical digital signal processing system are digital signals obtained by quantizing the sample values either by *rounding* or by *truncation*. For example, the digital signal $\{\hat{x}[n]\}$ obtained by rounding the sample values of the discrete-time sequence $x[n]$ of Eq. (2.1) to the nearest integer values is given by

$$\{\hat{x}[n]\} = \{\ldots,\ 1,\ 0,\ \underset{\uparrow}{2},\ 1,\ 0,\ -4,\ 3,\ -1,\ 4,\ \ldots\}.$$

Figure 2.3 shows a digital signal with amplitudes taking discrete integer values in the range from -3 to 3.

For digital processing of a continuous-time signal, it is first converted into an equivalent digital signal by means of a sample-and-hold circuit, followed by an analog-to-digital converter. The processed

[1] The complex conjugation operation is denoted by the superscript symbol *.

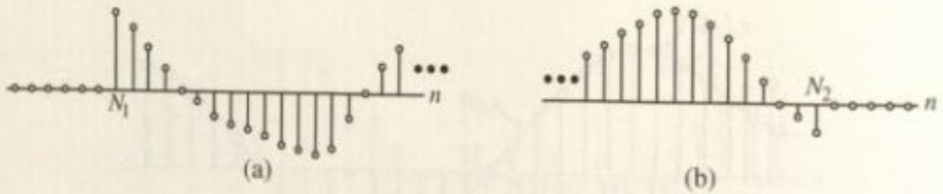


Figure 2.4: (a) A right-sided sequence and (b) a left-sided sequence.

digital signal is then converted back into an equivalent continuous-time signal by a digital-to-analog converter, followed by an analog reconstruction filter. Section 3.8 is concerned with the digital processing of continuous-time signals. It develops the mathematical foundation of the sampling process and describes the operations of various interface circuits between the continuous-time domain and the digital domain. Chapter 12 considers the effect of discretization of the amplitudes.

2.1.1 Length of a Discrete-Time Signal

The discrete-time signal may be a *finite-length* or an *infinite-length sequence*. A finite-length (also called *finite-duration* or *finite-extent*) sequence is defined only for a finite time interval:

$$N_1 \leq n \leq N_2, \tag{2.5}$$

where $-\infty < N_1$ and $N_2 < \infty$ with $N_2 \geq N_1$. The *length or duration* N of the above finite-length sequence is

$$N = N_2 - N_1 + 1. \tag{2.6}$$

A length-N discrete-time sequence consists of N samples and is often referred to as an N-point sequence. A finite-length sequence also can be considered as an infinite-length sequence by assigning zero values to samples whose arguments are outside the above range. The process of lengthening a sequence by adding zero-valued samples is called *appending with zeros* or *zero-padding*. In some applications, a finite-length sequence is also made a longer sequence by appending it with non-zero samples.

There are three types of infinite-length sequences. A *right-sided sequence* $x[n]$ has zero-valued samples for $n < N_1$; that is,

$$x[n] = 0 \qquad \text{for } n < N_1, \tag{2.7}$$

where N_1 is a finite integer that can be positive or negative. If $N_1 \geq 0$, a right-sided sequence usually is called a *causal sequence*.[2] Likewise, a *left-sided sequence* $x[n]$ has zero-valued samples for $n > N_2$; that is,

$$x[n] = 0 \qquad \text{for } n > N_2, \tag{2.8}$$

where N_2 is a finite integer that can be positive or negative. If $N_2 \leq 0$, a left-sided sequence usually is called an *anticausal sequence*. A general *two-sided sequence* is defined for both positive and negative values of n. Figure 2.4 illustrates the above two types of one-sided sequences.

For simplicity, in finite-length sequences defined for positive values of the time index n beginning at $n = 0$, the first sample in the sequence always will be assumed to be the one associated with the time index $n = 0$ without the arrow being explicitly shown under it.

We have so far considered the sequence of numbers representing a discrete-time signal in the time domain. As we shall show in Chapter 5, a finite-length discrete-time signal can also be represented as a

[2]The term *causal sequence* has its origin in the causality condition of a discrete-time system as discussed in Section 4.4.4.

finite-length sequence in a transform domain where the sample of the sequence is a function of a discrete variable in the transform domain, usually denoted by the integer variable k. It is a common practice to denote the transform-domain representation of a discrete-time signal $x[n]$ with capital letters; for example, $X[k]$. If we denote the vector of the time-domain samples of length N defined for $0 \leq n \leq N-1$, as[3]

$$\mathbf{x} = [\,x[0] \quad x[1] \quad \cdots \quad x[N-1]\,]^t\,,$$

the vector of the transform-domain samples of length N defined for $0 \leq k \leq N-1$,

$$\mathbf{X} = [\,X[0] \quad X[1] \quad \cdots \quad X[N-1]\,]^t\,,$$

is obtained by multiplying $\mathbf{x}$ by an $N \times N$ invertible matrix $\mathbf{D}$:

$$\mathbf{X} = \mathbf{D}\mathbf{x}.$$

The original time-domain sequence can be recovered from its transform-domain representation by applying an inverse transformation

$$\mathbf{x} = \mathbf{D}^{-1}\mathbf{X}.$$

In some cases, the transform-domain representation can be a complex-valued sequence for a real-valued time-domain sequence. The transform-domain sequence often is processed using some of the operations used in the processing of discrete-time signals in the time domain. Some elementary operations on sequences are discussed in Section 2.2.1.

2.1.2 Strength of a Discrete-Time Signal

The strength of a discrete-time signal usually is given by its norm. In general, the $\mathcal{L}_p$-*norm* of a sequence $\{x[n]\}$ is defined by

$$\| x \|_p = \left(\sum_{n=-\infty}^{\infty} |x[n]|^p \right)^{1/p}, \tag{2.9}$$

where p is a positive integer. In practice, the value of p used is typically 1 or 2 or ∞. The $\mathcal{L}_\infty$-norm is given by the *peak absolute value* of $\{x[n]\}$, that is,

$$\| x \|_\infty = |x|_{\max}. \tag{2.10}$$

It follows from Eq. (2.9) that for a length-N sequence, $\| x \|_2 / \sqrt{N}$ is the root-mean-squared (rms) value of $\{x[n]\}$, and $\| x \|_1 / N$, is the mean absolute value of $\{x[n]\}$. It can be shown that (Problem 2.2)

$$\| x \|_2 \leq \| x \|_1 .$$

One application of the norm is in estimating the error in the approximation of a discrete-time signal by another discrete-time signal. Let the length-N sequence $y[n], 0 \leq n \leq N-1$, represent an approximation of the length-N sequence $x[n], 0 \leq n \leq N-1$ in some sense. For example, $x[n]$ may be a discrete-time signal corrupted by an additive noise and $y[n]$ is the signal obtained after the removal of the noise. There are several definitions of the error estimate. One estimate of the quality of the approximation is given by the *mean-squared error* (MSE)

$$\text{MSE} = \frac{1}{N} \sum_{i=0}^{N-1} (|y[n] - x[n]|)^2 = \frac{1}{N} (\| y[n] - x[n] \|_2)^2 . \tag{2.11}$$

[3]The superscript "t" denotes matrix transposition.

which is the square of the $\mathcal{L}_2$-norm of the difference signal divided by the length of the signals. Another estimate of the approximation is given by the *relative error*, which is the ratio of the $\mathcal{L}_2$-norm of the difference signal and $\mathcal{L}_2$-norm of the original signal:

$$E_{rel} = \left(\frac{\sum_{n=0}^{N-1} |y[n] - x[n]|^2}{\sum_{n=0}^{N-1} |x[n]|^2} \right)^{1/2} = \frac{\| y[n] - x[n] \|_2}{\| x[n] \|_2}. \tag{2.12}$$

The norm of a finite-length sequence `x` can be computed using the M-file `norm` in MATLAB. The *$\mathcal{L}_2$-norm* of `x` is obtained using either `norm(x)` or `norm(x,2)`. The *$\mathcal{L}_1$-norm* of `x` can be determined using `norm(x,1)`. Finally, *$\mathcal{L}_\infty$-norm* of `x` is evaluated using `norm(x,inf)`.

2.2 Operations on Sequences

Digital signal processing is concerned with the processing of one or more discrete-time signals to generate another signal or signals with more desirable properties. For example, a signal of interest may be corrupted by an additive noise during its transmission through a medium, and it is necessary to operate on the noisy signal to obtain a reasonable replica of the original uncorrupted signal. In some applications, several different discrete-time signals are combined at the transmitting end for efficient transmission and at the receiving end they are separated by operating on the combined signal. In most cases, the operation defining a particular discrete-time system is composed of some elementary operations that we describe next.

2.2.1 Elementary Operations

Let $x[n]$ and $y[n]$ be two known sequences. By forming the *product* of the sample values of these two sequences at each instant, we form a new sequence $w_1[n]$:

$$w_1[n] = x[n] \cdot y[n]. \tag{2.13}$$

In some applications, the product operation is also known as *modulation*. The device implementing the modulation operation is called a *modulator*, and its schematic representation is shown in Figure 2.5(a).

An application of the product operation is illustrated in Example 2.14. Another application of the product operation is in forming a finite-length sequence from an infinite-length sequence by multiplying the latter with a finite-length sequence called a *window sequence*.

The second basic operation is the scalar *multiplication*, whereby a new sequence is generated by multiplying each sample of a sequence $x[n]$ by a scalar A:

$$w_2[n] = Ax[n]. \tag{2.14}$$

The device implementing the multiplication operation is called a *multiplier*, and its schematic representation is shown in Figure 2.5(b).

The third basic operation is the *addition* by which a new sequence $w_3[n]$ is obtained by adding the sample values of two sequences $x[n]$ and $y[n]$:

$$w_3[n] = x[n] + y[n]. \tag{2.15}$$

The device implementing the addition operation is called an *adder*, and its schematic representation is shown in Figure 2.5(c). By inverting the signs of all samples of the sequence $y[n]$, an adder can also be used to implement the *subtraction* operation.

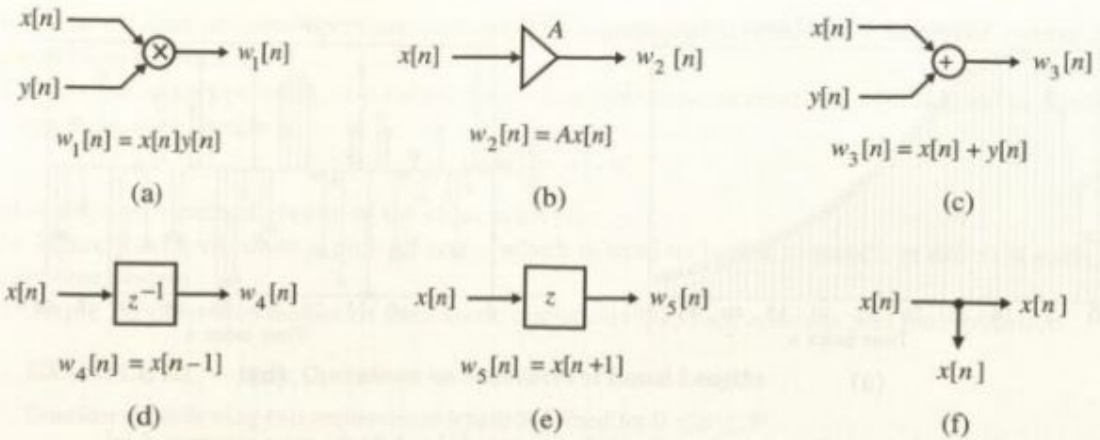


Figure 2.5: Schematic representations of basic operations on sequences: (a) modulator, (b) multiplier, (c) adder, (d) unit delay, (e) unit advance, and (f) pick-off node.

A very simple application of the addition operation is in improving the quality of measured data that has been corrupted by an additive random noise. In many cases, the actual uncorrupted data vector $\mathbf{s}$ remains essentially the same from one measurement to the next, while the additive noise vector is random and not reproducible. Let $\mathbf{d}_i$ denote the noise vector corrupting the i-th measurement of the uncorrupted data vector $\mathbf{s}$:

$$\mathbf{x}_i = \mathbf{s} + \mathbf{d}_i.$$

The average data vector, called the *ensemble average*, obtained after K measurements is then given by

$$\mathbf{x}_{\text{ave}} = \frac{1}{K}\sum_{i=1}^{K}(\mathbf{x}_i) = \frac{1}{K}\sum_{i=1}^{K}(\mathbf{s} + \mathbf{d}_i) = \mathbf{s} + \frac{1}{K}\left(\sum_{i=1}^{K}\mathbf{d}_i\right).$$

For a very large value of K, $\mathbf{x}_{\text{ave}}$ is usually a reasonable replica of the desired data vector $\mathbf{s}$, as the samples of the average of the summed noise vector $\frac{1}{K}(\sum_{i=1}^{K}\mathbf{d}_i)$ become very small due to the randomness of the noise. Example 2.1 illustrates ensemble averaging.

EXAMPLE 2.1 Illustration of Ensemble Averaging

We assume for simplicity that the original uncorrupted data is given by [Sch75]

$$s[n] = 2[n(0.9)^n]. \tag{2.16}$$

The MATLAB program to generate the above data $s[n]$, the noise $d[n]$, and the ensemble average are given in Program 2_1. The two sequences generated using the above program are shown in Figure 2.6. One sample of the noise-corrupted data $s[n] + d_i[n]$ and the ensemble average obtained after 50 measurements is shown in Figure 2.7. It can be seen that the ensemble average shown in Figure 2.7(b) is nearly the same as the original uncorrupted data shown in Figure 2.6(a).

Program 2_1.m

An application of ensemble averaging is in the power spectrum estimation of a random signal, discussed in the CD accompanying this book.

The *time-shifting* operation illustrated below in Eq. (2.17) shows the relation between $x[n]$ and its time-shifted version $w_4[n]$:

$$w_4[n] = x[n - N], \tag{2.17}$$

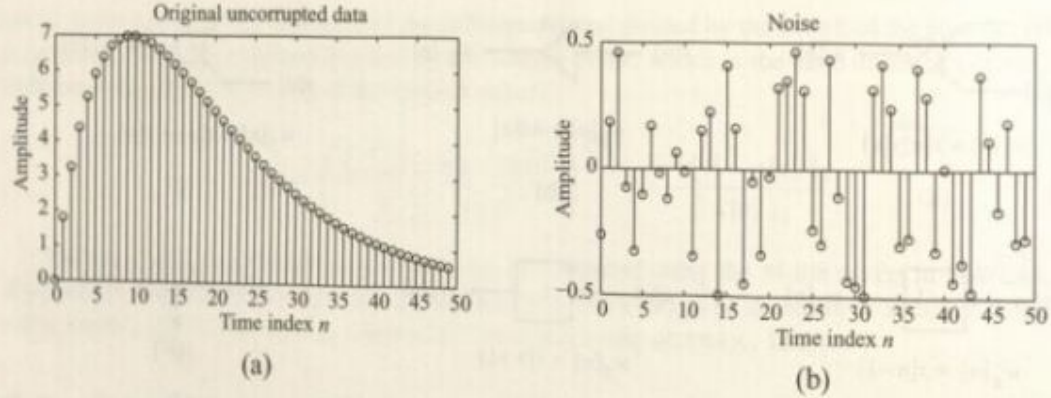


Figure 2.6: (a) The original uncorrupted sequence $s[n]$ and (b) the noise sequence $d_i[n]$.

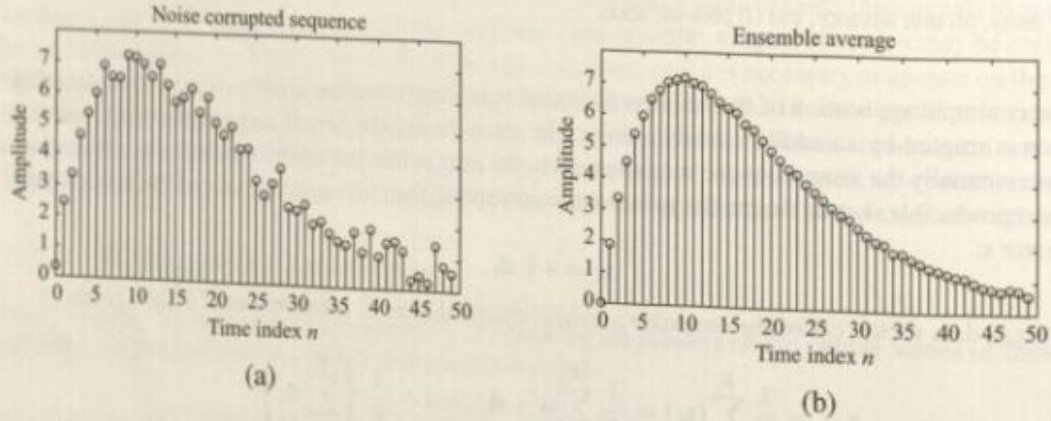


Figure 2.7: (a) A sample of the corrupted sequence and (b) the ensemble average after 50 measurements.

where N is an integer. If $N > 0$, it is a *delaying* operation, and if $N < 0$, it is an *advancing* operation. For $N = 1$, we have the input–output relation

$$w_4[n] = x[n-1],$$

and the device implementing the delay operation by one sample period is called a *unit delay*. In terms of z-transform, introduced later in Chapter 6, the above relation can be rewritten as

$$W_4(z) = z^{-1}X(z),$$

where $W_4(z)$ and $X(z)$ are, respectively, the z-transforms of the output sequence $w_4[n]$ and the input sequence $x[n]$. It is a usual practice to represent schematically the unit delay operation using the symbol z^{-1}, as shown in Figure 2.5(d).

The opposite of the unit delay operation is the *unit advance* operation, defined by

$$w_5[n] = x[n+1],$$

which in terms of the z-transform is given by

$$W_5(z) = z\,X(z).$$

As a result, the unit advance operation commonly is represented schematically using the symbol z, as shown in Figure 2.5(e).

The *time-reversal* operation, also called the *folding operation*, is another useful scheme to develop a new sequence. An example is

$$w_6[n] = x[-n], \tag{2.18}$$

which is the time-reversed version of the sequence $x[n]$.

In Figure 2.5(f), we show a *pick-off node*, which is used to feed a sequence to different parts of a discrete-time system.

Example 2.2 illustrates the use of three basic operations: product, addition, and multiplication.

EXAMPLE 2.2 Basic Operations on Sequences of Equal Lengths

Consider the following two sequences of length 5 defined for $0 \leq n \leq 4$:

$$c[n] = \{3.2, \quad 41, \quad 36, \quad -9.5, \quad 0\},$$
$$d[n] = \{1.7, \quad -0.5, \quad 0, \quad 0.8, \quad 1\}.$$

Several new sequences of length 5 generated from the above sequences are given by

$$w_1[n] = c[n] \cdot d[n] = \{5.44, \quad -20.5, \quad 0, \quad -7.6, \quad 0\},$$
$$w_2[n] = c[n] + d[n] = \{4.9, \quad 40.5, \quad 36, \quad -8.7, \quad 1\},$$
$$w_3[n] = \tfrac{7}{2}c[n] = \{11.2, \quad 143.5, \quad 126, \quad -33.25, \quad 0\}.$$

As indicated by Example 2.2, operations on two or more sequences to generate a new sequence can be carried out if all sequences are of the same length and defined for the same range of the time index n. However, if the sequences are not of equal length, all sequences can be made to have the same range of the time index n by appending zero-valued samples to the sequence(s) of smaller lengths. This process is illustrated in Example 2.3.

EXAMPLE 2.3 Basic Operations on Sequences of Unequal Lengths

Consider a sequence $\{g[n]\}$ of length 3 defined for $0 \leq n \leq 2$ given by

$$\{g[n]\} = \{-21, \quad 1.5, \quad 3\}.$$

It is clear that we cannot develop another sequence by operating on this sequence and any one of the length-5 sequences of Example 2.2. However, it is possible to treat $\{g[n]\}$ as a sequence of length 5 and defined for $0 \leq n \leq 4$ by appending it with two zero-valued samples:

$$\{g_e[n]\} = \{-21, \quad 1.5, \quad 3, \quad 0, \quad 0\}.$$

Examples of new sequences generated from $\{g_e[n]\}$ and $c[n]$ of the previous example are indicated below:

$$\{w_4[n]\} = \{c[n] \cdot g_e[n]\} = \{-67.2, \quad 61.5, \quad 108, \quad 0, \quad 0\},$$
$$\{w_5[n]\} = \{c[n] + g_e[n]\} = \{-17.8, \quad 42.5, \quad 39, \quad -9.5, \quad 0\}.$$

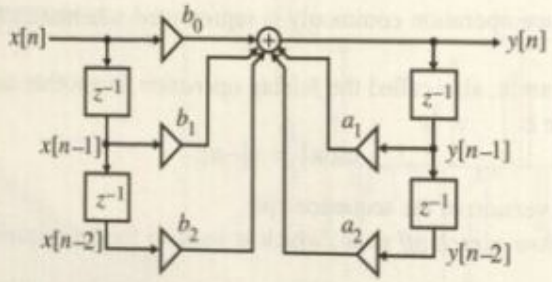


Figure 2.8: Schematic representation of Eq. (2.19).

2.2.2 Combination of Elementary Operations

In most applications, combinations of the above elementary operations are used. An illustration of such combinations is given in Example 2.4.

EXAMPLE 2.4 Illustration of Combination of Basic Operations on Sequences

In some applications, a sequence is generated by a weighted combination of another sequence and its past values, along with a weighted combination of past values of the sequence being generated, as illustrated in Eq. (2.19):

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + a_1y[n-1] + a_2y[n-2]. \tag{2.19}$$

As the above equation points out, the sequence $y[n]$ is generated by multiplying the present value of the sequence $x[n]$ and its two past values, $x[n-1]$ and $x[n-2]$, with constants b_0, b_1, and b_2, respectively, and multiplying two past values of the sequence $y[n]$, $y[n-1]$ and $y[n-2]$, with constants a_1 and a_2, respectively, and finally adding all weighted samples. Thus, the complicated operation of Eq. (2.19) makes use of three elementary operations: addition, multiplication, and delaying. A schematic representation of the operation given in Eq. (2.19) using the schematic representations of the adder, multiplier, and unit delay is shown Figure 2.8.

A more complicated combination of two sequences to generate another sequence is the convolution sum discussed next.

2.2.3 Convolution Sum

Let $x[n]$ and $h[n]$ denote two sequences. The sequence $y[n]$ generated by the *convolution sum* of these two sequences is given by

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k], \tag{2.20a}$$

which alternately can be written as

$$y[n] = \sum_{k=-\infty}^{\infty} x[n-k]h[k] \tag{2.20b}$$

by a simple change of variables, provided the convolution sum given in Eq. (2.20a) is convergent. As can be seen from the above two equations, the generation of the sequence $y[n]$ by the convolution sum

involves four basic operations: time-reversal, multiplication, addition, and delay. We shall show later in Section 4.4.1 that a certain class of discrete-time system is characterized completely in the time domain by the convolution sum.

For brevity, the convolution sum expression will be compactly written as

$$y[n] = x[n] \circledast h[n], \tag{2.21}$$

where the notation $\circledast$ denotes the convolution sum.[4] Example 2.5 illustrates the computation of the convolution sum.

EXAMPLE 2.5 Illustration of Convolution Sum Computation

We systematically develop the sequence $y[n]$ generated by the convolution of the two finite-length sequences $x[n]$ and $h[n]$ shown in Figure 2.9(a). As can be seen from the plot of the shifted time-reversed version $\{h[n-k]\}$ for $n < 0$ sketched in Figure 2.9(b) for $n = -3$, for any value of the sample index k, the kth sample of either $\{x[k]\}$ or $\{h[n-k]\}$ is zero. As a result, the product of the kth samples is always zero for any k, and consequently, the convolution sum of Eq. (2.20a) leads to

$$y[n] = 0 \qquad \text{for } n < 0.$$

Consider now the calculation of $y[0]$. We form $\{h[-k]\}$ as shown in Figure 2.9(c). The product sequence $\{x[k]h[-k]\}$ is plotted in Figure 2.9(d), which has a single nonzero sample for $k = 0$, $x[0]h[0]$. Thus,

$$y[0] = x[0]h[0] = -2.$$

For the computation of $y[1]$, we shift $\{h[-k]\}$ to the right by one sample period to form $\{h[1-k]\}$, as sketched in Figure 2.9(e). The product sequence $\{x[k]h[1-k]\}$ shown in Figure 2.9(f) has one nonzero sample for $k = 0$, $x[0]h[1]$. As a result,

$$y[1] = x[0]h[1] + x[1]h[0] = -4 + 0 = -4.$$

To calculate $y[2]$, we form $\{h[2-k]\}$, as indicated in Figure 2.9(g). The resulting product sequence $\{x[k]h[2-k]\}$ is plotted in Figure 2.9(h), which leads to

$$y[2] = x[0]h[2] + x[1]h[1] + x[2]h[0] = 0 + 0 + 1 = 1.$$

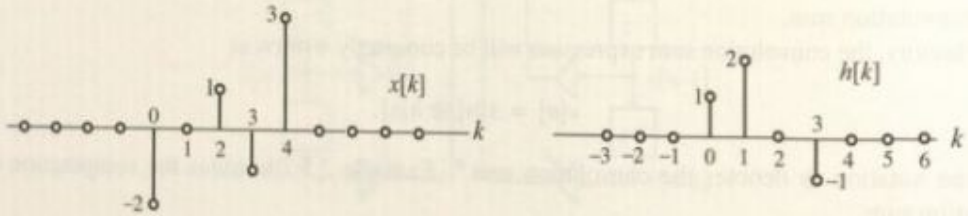
This process is continued, resulting in

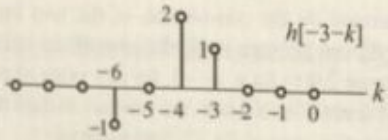
$$\begin{aligned}
y[3] &= x[0]h[3] + x[1]h[2] + x[2]h[1] + x[3]h[0] = 2 + 0 + 2 - 1 = 3,\\
y[4] &= x[1]h[3] + x[2]h[2] + x[3]h[1] + x[4]h[0] = 0 + 0 - 2 + 3 = 1,\\
y[5] &= x[2]h[3] + x[3]h[2] + x[4]h[1] = -1 + 0 + 6 = 5,\\
y[6] &= x[3]h[3] + x[4]h[2] = 1 + 0 = 1,\\
y[7] &= x[4]h[3] = -3.
\end{aligned}$$

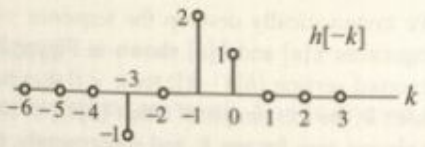
From the plot of $\{h[n-k]\}$ for $n > 7$ as shown in Figure 2.9(i) and the plot of $\{x[k]\}$ in Figure 2.9(a), it can be seen that there is no overlap between these two sequences. Hence,

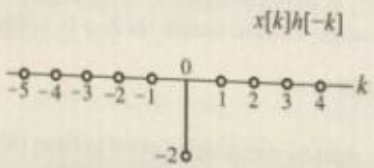
$$y[n] = 0 \qquad \text{for } n > 7.$$

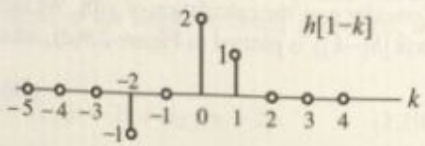
The sequence $\{y[n]\}$ as obtained above is sketched in Figure 2.10.

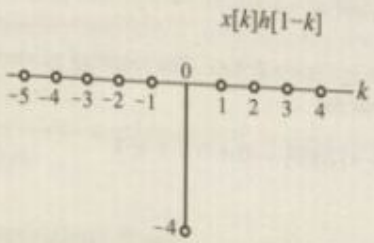


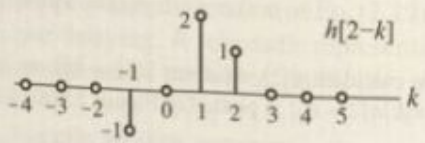


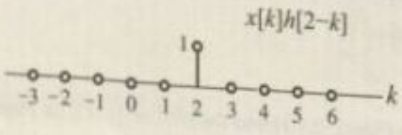












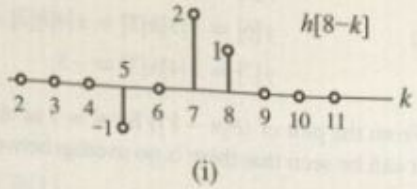


Figure 2.9: Illustration of the convolution process.

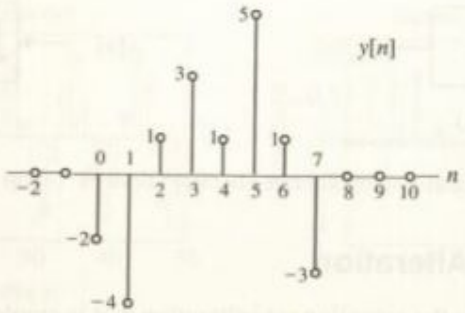


Figure 2.10: Sequence generated by the convolution.

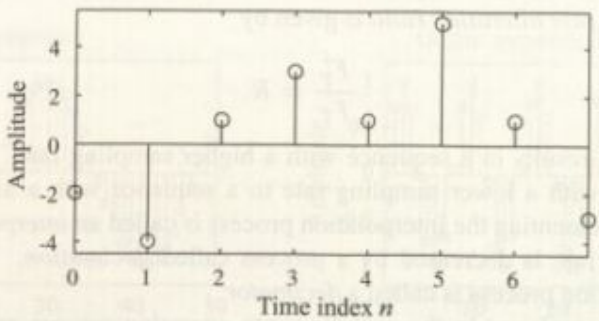


Figure 2.11: Sequence generated by convolution using MATLAB.

It should be noted that the sum of the indices of each sample product inside the summation of either Eq. (2.20a) or Eq. (2.20b) is equal to the index of the sample being generated by the convolution sum operation. For example, the sum in the computation of $y[3]$ in the above example involves the products $x[0]h[3]$, $x[1]h[2]$, $x[2]h[1]$, and $x[3]h[0]$. The sum of the indices in each of these four products is equal to 3, which is the index of the sample $y[3]$.

As can be seen from Example 2.5, the convolution of two finite-length sequences results in a finite-length sequence. In this example, the convolution of a sequence $\{x[n]\}$ of length 5 with a sequence $\{h[n]\}$ of length 4 resulted in a sequence $\{y[n]\}$ of length 8. In general, if the lengths of the two sequences being convolved are M and N, then the resulting sequence after convolution is of length $M + N - 1$ (Problem 2.5).

In MATLAB, the M-file `conv` implements the convolution of two finite-length sequences. Its application is illustrated in Example 2.6.

EXAMPLE 2.6 Convolution Computation Using MATLAB

Program 2_2 given in the accompanying CD can be used to compute the convolution of two finite-length sequences. The plot of the sequence generated by convolving the two sequences of Example 2.5 using Program 2_2 is indicated in Figure 2.11. As expected, the result is identical to that derived in Example 2.5.

Program 2_2.m

[4] In the literature, the symbol for the convolution sum is $*$ without the circle. However, as the superscript $*$ is always used for denoting the complex conjugation operation, in this text we have adopted the symbol $\circledast$ to denote the convolution sum operation.

Figure 2.12: Representation of basic sampling rate alteration devices: (a) up-sampler and (b) down-sampler.

2.2.4 Sampling Rate Alteration

Another quite useful operation is the sampling rate alteration that is employed to generate a new sequence with a sampling rate higher or lower than that of a given sequence. Thus, if $x[n]$ is a sequence with a sampling rate of F_T Hz and it is used to generate another sequence $y[n]$ with a desired sampling rate of F_T' Hz, then the *sampling rate alteration ratio* is given by

$$\frac{F_T'}{F_T} = R. \tag{2.22}$$

The operation for $R > 1$ results in a sequence with a higher sampling rate. This operation is used in interpolating a sequence with a lower sampling rate to a sequence with a higher sampling rate. The discrete-time system implementing the interpolation process is called an *interpolator*. On the other hand, if $R < 1$, the sampling rate is decreased by a process called *decimation*. The discrete-time system implementing the decimation process is called a *decimator*.

The basic operations employed in the sampling rate alteration process are called *up-sampling* and *down-sampling*. These operations play important roles in multirate discrete-time systems and are considered in Chapters 13 and 14.

In up-sampling by an integer factor $L > 1$, $L - 1$ equidistant zero-valued samples are inserted by the up-sampler between each two consecutive samples of the input sequence $x[n]$ to develop an output sequence $x_u[n]$ according to the relation

$$x_u[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \ldots, \\ 0, & \text{otherwise.} \end{cases} \tag{2.23}$$

The sampling rate of $x_u[n]$ is L times larger than that of the original sequence $x[n]$.

The block-diagram representation of the up-sampler, also called a *sampling rate expander*, is shown in Figure 2.12(a). Figure 2.13 illustrates the up-sampling operation for an up-sampling factor of $L = 3$. The interpolator consists of an up-sampler followed by a discrete-time system that replaces the inserted zero-valued samples of $x_u[n]$ with more appropriate values given by a linear combination of samples of $x[n]$. A simple example of an interpolator is given in Section 4.1.

Conversely, the down-sampling operation by an integer factor $M > 1$ on a sequence $x[n]$ consists of keeping every Mth sample of $x[n]$ and removing $M - 1$ between samples, generating an output sequence $y[n]$ according to the relation

$$y[n] = x[nM]. \tag{2.24}$$

This results in a sequence $y[n]$ whose sampling rate is $(1/M)$-th that of $x[n]$. Basically, all input samples with indices equal to an integer multiple of M are retained at the output, and all others are discarded.

The schematic representation of the *down-sampler* or *sampling rate compressor* is shown in Figure 2.12(b). Figure 2.14 illustrates the down-sampling operation for a down-sampling factor of $M = 3$. The decimator consists of a discrete-time system followed by a down-sampler. The discrete-time system preceding the down-sampler ensures that the input signal $x[n]$ is appropriately band-limited to prevent aliasing that is caused by the down-sampling operation.

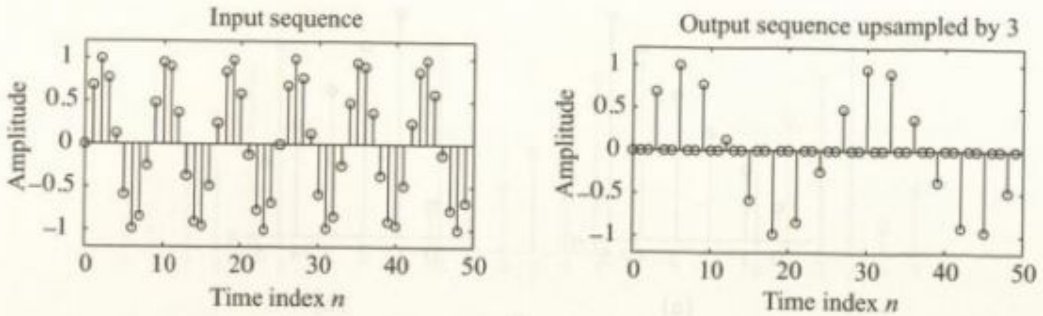


Figure 2.13: Illustration of the up-sampling process.

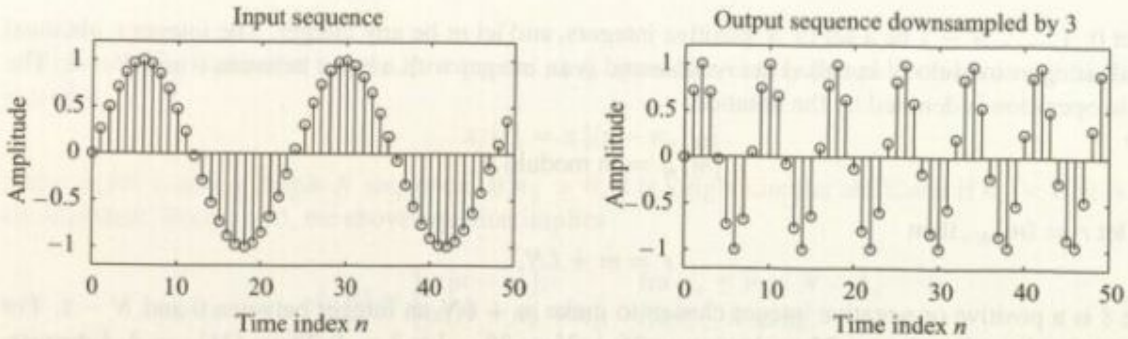


Figure 2.14: Illustration of the down-sampling process.

2.3 Operations on Finite-Length Sequences

Some of the operations on sequences described in the previous section are not applicable to finite-length sequences. Let $x[n]$ be such a sequence of length N defined for $0 \leq n \leq N-1$, which has sample values equal to zero for $n < 0$ and $n \geq N$. A time-reversal operation on $x[n]$ results in a length-N sequence $x[-n]$ defined for $-(N-1) \leq n \leq 0$. Likewise, a linear time-shift of the sequence by an integer valued M results in a length-N sequence $x[n+M]$, which no longer is defined for $0 \leq n \leq N-1$. We thus need to define another type of shifting that will keep the shifted sequence in the range $0 \leq n \leq N-1$. Similarly, a convolution sum of a length-N sequence $x[n]$ defined for $0 \leq n \leq N-1$ with another length-N sequence $h[n]$ defined for $0 \leq n \leq N-1$ will generate a sequence of length $2N-1$ defined for $0 \leq n \leq 2N-2$. Consequently, another type of convolution needs to be defined that will ensure the convolved sequence is in the range $0 \leq n \leq N-1$. In this section, we consider the time-reversal and time-shifting operations for finite-length sequences. The convolution sum–like operation for finite-length sequences is described in Section 5.4.

2.3.1 Circular Time-Reversal Operation

The time-reversal operation on a finite-length sequence that develops a sequence also defined for the same range of values of the time index n as that of the original sequence is obtained by using the *modulo operation*.

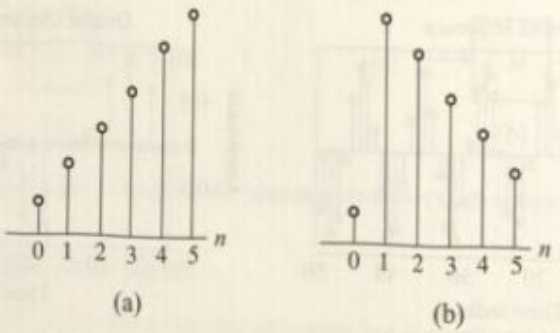


Figure 2.15: Illustration of a circular time-reversal of a finite-length sequence. (a) $x[n]$, (b) $y[n] = x[\langle -n \rangle_6]$.

Let $0, 1, \ldots, N-1$ be a set of N positive integers, and let m be any integer. The integer r obtained by evaluating m modulo N is called the *residue* and is an integer with a value between 0 and $N-1$. The modulo operation is denoted by the notation

$$\langle m \rangle_N = m \text{ modulo } N.$$

If we let $r = \langle m \rangle_N$, then

$$r = m + \ell N,$$

where ℓ is a positive or negative integer chosen to make $m + \ell N$ an integer between 0 and $N-1$. For example, for $N = 7$ and $m = 25$ we have $r = 25 + 7\ell = 25 - 7 \times 3 = 4$. Thus, $\langle 25 \rangle_7 = 4$. Likewise, for $N = 7$ and $m = -15$, we get $r = -15 + 7\ell = -15 + 7 \times 3 = 6$. Hence $\langle -15 \rangle_7 = 6$. The modulo operation can be carried out in MATLAB using the M-file mod.

Thus, the time-reversed version $\{y[n]\}$ of the length-N sequence $\{x[n]\}$ defined for $0 \leq n \leq N-1$ is given by $\{y[n]\} = \{x[\langle -n \rangle_N]\}$. We illustrate the circular time-reversal operation in Example 2.7.

EXAMPLE 2.7 Illustration of Circular Time-Reversal Operation

Consider the length-5 sequence $\{x[n]\} = \{x[0], x[1], x[2], x[3], x[4]\}$. Its circular time-reversed version is given by $\{y[n]\} = \{x[\langle -n \rangle_5]\}$. The 5 samples of $\{y[n]\}$ are thus as given below:

$$\begin{aligned} y[0] &= x[\langle 0 \rangle_5] = x[0], \\ y[1] &= x[\langle -1 \rangle_5] = x[4], \\ y[2] &= x[\langle -2 \rangle_5] = x[3], \\ y[3] &= x[\langle -3 \rangle_5] = x[2], \\ y[4] &= x[\langle -4 \rangle_5] = x[1]. \end{aligned}$$

Hence, $\{y[n]\} = \{x[0], x[4], x[3], x[2], x[1]\}$.

The concept of a circular time-reversal of a finite-length sequence is illustrated in Figure 2.15.

2.3.2 Circular Shift of a Sequence

The time-shifting operation for a finite-length sequence that results in another sequence of the same length and defined for the same range of values of n is referred to as the *circular time-shifting* operation. Such a shifting operation is achieved by using the modulo operation.

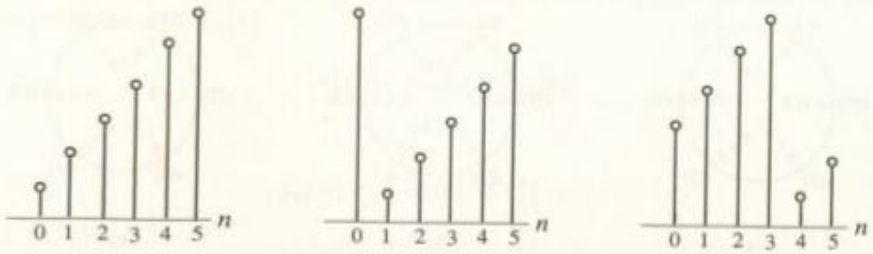


Figure 2.16: Illustration of a circular shift of a finite-length sequence. (a) $x[n]$, (b) $x[\langle n-1\rangle_6] = x[\langle n+5\rangle_6]$, and (c) $x[\langle n-4\rangle_6] = x[\langle n+2\rangle_6]$.

The circular shift of a length-N sequence $x[n]$ by an amount n_o sample periods is defined by the equation

$$x_c[n] = x\left[\langle n - n_o\rangle_N\right], \tag{2.25}$$

where $x_c[n]$ is also a length-N sequence. If $n_o > 0$, it is a right circular shift, and if $n_o < 0$, it is a left circular shift. For $n_o > 0$, the above equation implies

$$x_c[n] = \begin{cases} x[n-n_o], & \text{for } n_o \le n \le N-1, \\ x[N-n_o+n], & \text{for } 0 \le n < n_o. \end{cases} \tag{2.26}$$

The concept of a circular shift of a finite-length sequence is illustrated in Figure 2.16. Figure 2.16(a) shows a length-6 sequence $x[n]$. Figure 2.16(b) shows its circularly shifted version shifted to the right by 1 sample period or, equivalently, shifted to the left by 5 sample periods. Likewise, Figure 2.16(c) depicts its circularly shifted version shifted to the right by 4 sample periods or, equivalently, shifted to the left by 2 sample periods.

As can be seen from Figures 2.16(b) and (c), a right circular shift by n_o is equivalent to a left circular shift by $N - n_o$ sample periods. It should be noted that a circular shift by an integer number n_o greater than N is equivalent to a circular shift by $\langle n_o\rangle_N$. For finite-length sequences defined for a specific time interval, the *circular time-reversal* operation, a time-reversal–like operation, is defined using the modulo arithmetic. Specifically, for a length-N sequence $x[n]$, $0 \le n < N-1$, the circularly time-reversed sequence is also of length N and given by

$$x[\langle -n\rangle_N] = x[\langle N-n\rangle_N].$$

Since $\langle -n\rangle_N = N - n$ for $1 \le n < N-1$, the circular time-reversal operation can be implemented using

$$x[\langle -n\rangle_N] = \begin{cases} x[N-n], & \text{for } 1 \le n \le N-1, \\ x[n], & \text{for } n = 0. \end{cases}$$

If we view the length-N sequence displayed on a circle at N equally spaced points, then the circular shift operation can be considered as a clockwise or anticlockwise rotation of the sequence by n_o sample spacings on the circle. This is illustrated in Figure 2.17.

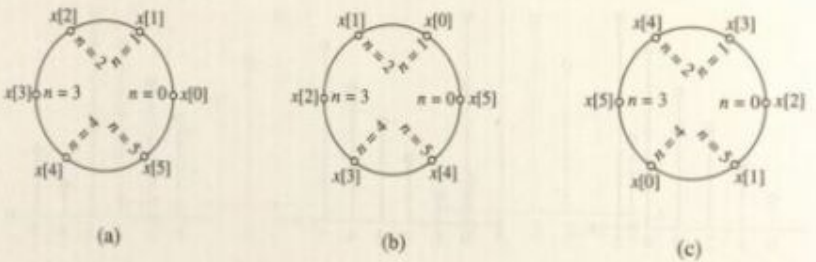


Figure 2.17: Alternate illustration of a circular shift of a finite-length sequence. (a) $x[n]$, (b) $x[\langle n-1\rangle_6] = x[\langle n+5\rangle_6]$, and (c) $x[\langle n-4\rangle_6] = x[\langle n+2\rangle_6]$.

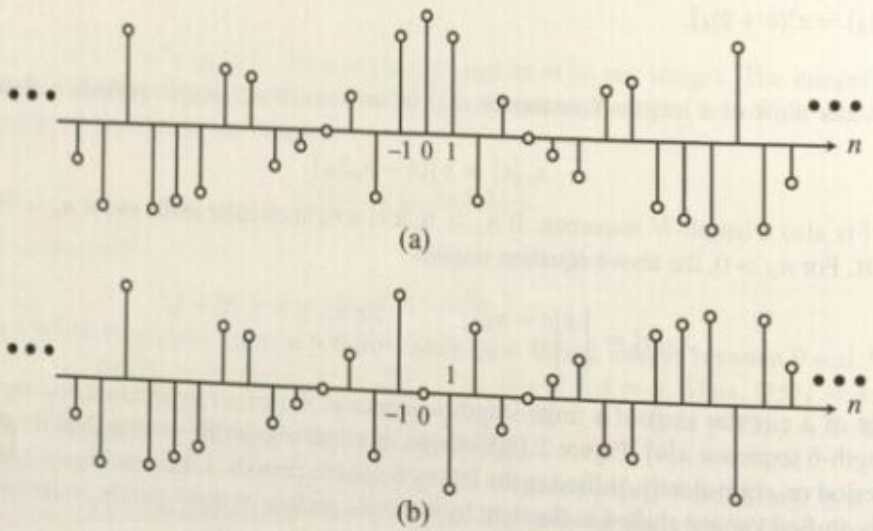


Figure 2.18: (a) An even sequence and (b) an odd sequence.

2.3.3 Classification of Sequences

A discrete-time signal can be classified into several types based on its specific characteristics. One classification discussed earlier is in terms of the number of samples defining the sequence. Another classification is with respect to the symmetry exhibited by the samples with respect to the time index $n = 0$. A discrete-time signal can also be classified in terms of its other properties such as periodicity, summability, energy, and power. These properties can often be exploited in simplifying some signal processing algorithms.

Classification Based on Symmetry

A sequence $x[n]$ is called a *conjugate-symmetric sequence* if $x[n] = x^*[-n]$. A real conjugate-symmetric sequence is called an *even sequence*. A sequence $x[n]$ is called a *conjugate-antisymmetric sequence* if $x[n] = -x^*[-n]$. A real conjugate-antisymmetric sequence is called an *odd sequence*. For a conjugate-antisymmetric sequence $x[n]$, the sample value at $n = 0$ must be purely imaginary. Consequently, for an odd sequence $x[0] = 0$. Examples of even and odd sequences are shown in Figure 2.18.

Any complex sequence $x[n]$ can be expressed as a sum of its conjugate-symmetric part $x_{cs}[n]$ and its conjugate-antisymmetric part $x_{ca}[n]$:

$$x[n] = x_{cs}[n] + x_{ca}[n], \tag{2.27}$$

where

$$x_{cs}[n] = \tfrac{1}{2}\left(x[n] + x^*[-n]\right), \tag{2.28a}$$

$$x_{ca}[n] = \tfrac{1}{2}\left(x[n] - x^*[-n]\right). \tag{2.28b}$$

As indicated by Eqs. (2.28a) and (2.28b), the computation of the conjugate-symmetric and conjugate-antisymmetric parts of a sequence involves conjugation, time-reversal, addition, and multiplication operations. Because of the time-reversal operation, the decomposition of a finite-length sequence into a sum of a conjugate-symmetric sequence and a conjugate-antisymmetric sequence is possible if the parent sequence is of odd length defined for a symmetric interval, $-M \leq 0 \leq M$.

EXAMPLE 2.8 Generation of Symmetric Parts of a Complex Sequence

Consider the finite-length sequence of length 7 defined for $-3 \leq n \leq 3$:

$$\{g[n]\} = \{0,\ 1 + j4,\ -2 + j3,\ \underset{\uparrow}{4 - j2},\ -5 - j6,\ -j2,\ 3\}.$$

To determine its conjugate-symmetric part $g_{cs}[n]$ and its conjugate-antisymmetric part $g_{ca}[n]$, we form

$$\{g^*[n]\} = \{0,\ 1 - j4,\ -2 - j3,\ \underset{\uparrow}{4 + j2},\ -5 + j6,\ j2,\ 3\},$$

whose time-reversed version is then given by

$$\{g^*[-n]\} = \{3,\ j2,\ -5 + j6,\ \underset{\uparrow}{4 + j2},\ -2 - j3,\ 1 - j4,\ 0\}.$$

Using Eq. (2.28a), we thus arrive at

$$\{g_{cs}[n]\} = \{1.5,\ 0.5 + j3,\ -3.5 + j4.5,\ \underset{\uparrow}{4},\ -3.5 - j4.5,\ 0.5 - j3,\ 1.5\}.$$

Likewise, using Eq. (2.28b), we get

$$\{g_{ca}[n]\} = \{-1.5,\ 0.5 + j,\ 1.5 - j1.5,\ \underset{\uparrow}{-j2},\ -1.5 - j1.5,\ -0.5 + j,\ 1.5\}.$$

It easily can be verified that $g_{cs}[n] = g_{cs}^*[-n]$ and $g_{ca}[n] = -g_{ca}^*[-n]$.

In a similar manner, any real sequence $x[n]$ can be expressed as a sum of its even part $x_{ev}[n]$ and its odd part $x_{od}[n]$:

$$x[n] = x_{ev}[n] + x_{od}[n], \tag{2.29}$$

where

$$x_{ev}[n] = \tfrac{1}{2}\left(x[n] + x[-n]\right), \tag{2.30a}$$

$$x_{od}[n] = \tfrac{1}{2}\left(x[n] - x[-n]\right). \tag{2.30b}$$

The symmetry properties of sequences often simplify their respective frequency-domain representations and can be exploited in signal analysis. Implications of the symmetry conditions are considered in the frequency-domain descriptions of sequences discussed in Section 3.2.3.

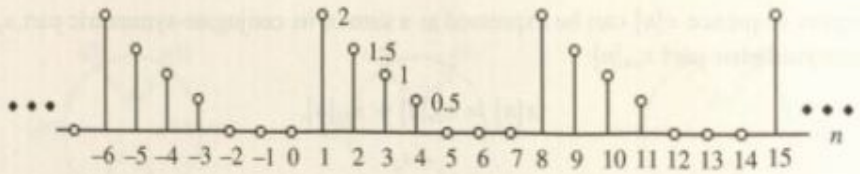


Figure 2.19: An example of a periodic sequence.

Periodic and Aperiodic Signals

A sequence $\tilde{x}[n]$ satisfying

$$\tilde{x}[n] = \tilde{x}[n + kN] \qquad \text{for all } n \tag{2.31}$$

is called a *periodic sequence* with a *period* N, where N is a positive integer and k is any integer. An example of a periodic sequence that has a period $N = 7$ samples is shown in Figure 2.19. A sequence is called an *aperiodic sequence* if it is not periodic. To distinguish a periodic sequence from an aperiodic sequence, we shall denote the former with a "~" on top. The *fundamental period* N_f of a periodic signal is the smallest value of N for which Eq. (2.31) holds.

The sum of two or more periodic sequences is also a periodic sequence. If $\tilde{x}_a[n]$ and $\tilde{x}_b[n]$ are two periodic sequences with fundamental periods N_a and N_b, respectively, then the sequence $\tilde{y}[n] = \tilde{x}_a[n] + \tilde{x}_b[n]$ is a periodic sequence with a fundamental period N given by the *least common multiple* (LCM) of N_a and N_b, respectively; that is,

$$N = \text{LCM}(N_a, N_b). \tag{2.32}$$

$\text{LCM}(N_a, N_b)$ can be computed by dividing the product of N_a and N_b by the *greatest common divisor* (GCD) of N_a and N_b, respectively:

$$\text{LCM}(N_a, N_b) = \frac{N_a N_b}{\text{GCD}(N_a, N_b)}. \tag{2.33}$$

Likewise, the product of two or more periodic sequences is again a periodic sequence. Here, if $\tilde{x}_a[n]$ and $\tilde{x}_b[n]$ are two periodic sequences with fundamental periods N_a and N_b, respectively, then the sequence $\tilde{w}[n] = \tilde{x}_a[n]\tilde{x}_b[n]$ is a periodic sequence with a period given by Eq. (2.32). It should be noted that in some cases, the fundamental period can be smaller than that given by Eq. (2.32). The LCM and GCD of two positive integers can be computed using the M-files `lcm` and `gcd` in MATLAB.

Energy and Power Signals

The total *energy* of a sequence $x[n]$ is defined by

$$\mathcal{E}_x = \sum_{n=-\infty}^{\infty} |x[n]|^2. \tag{2.34}$$

An infinite-length sequence with finite sample values may or may not have finite energy, as illustrated in Example 2.9.

EXAMPLE 2.9 Examples of Finite Energy and Infinite Energy Signals

The infinite-length sequence $x_1[n]$ defined by

$$x_1[n] = \begin{cases} \frac{1}{n}, & n \geq 1, \\ 0, & n \leq 0, \end{cases} \tag{2.35}$$

has an energy equal to

$$\mathcal{E}_{x_1} = \sum_{n=1}^{\infty} \left(\frac{1}{n}\right)^2,$$

which converges to $\pi^2/6$, indicating that $x_1[n]$ has finite energy. However, the infinite-length sequence $x_2[n]$ defined by

$$x_2[n] = \begin{cases} \frac{1}{\sqrt{n}}, & n \geq 1, \\ 0, & n \leq 0, \end{cases} \tag{2.36}$$

has an energy equal to

$$\mathcal{E}_{x_2} = \sum_{n=1}^{\infty} \frac{1}{n},$$

which does not converge. As a result, the sequence $x_2[n]$ has infinite energy.

The *average power* of an aperiodic sequence $x[n]$ is defined by

$$\mathcal{P}_x = \lim_{K\to\infty} \frac{1}{2K+1} \sum_{n=-K}^{K} |x[n]|^2. \tag{2.37}$$

The average power of a sequence can be related to its energy by defining its energy over a finite interval $-K \leq n \leq K$ as

$$\mathcal{E}_{x,K} = \sum_{n=-K}^{K} |x[n]|^2. \tag{2.38}$$

Then,

$$\mathcal{P}_x = \lim_{K\to\infty} \frac{1}{2K+1} \mathcal{E}_{x,K}. \tag{2.39}$$

It can be seen from the above equation that if the energy of a signal $\mathcal{E}_x$ is finite, then its average power $\mathcal{P}_x$ is zero.

The average power of an infinite-length sequence may be finite or infinite. An infinite energy signal with finite average power is called a *power signal*. Likewise, a finite energy signal with zero average power is called an *energy signal*. An example of a power signal is a periodic sequence that has a finite average power but infinite energy. An example of an energy signal is a finite-length sequence that has finite energy but zero average power. The average power of a periodic sequence $\tilde{x}[n]$ with a period N is given by

$$\mathcal{P}_x = \frac{1}{N} \sum_{n=0}^{N-1} |\tilde{x}[n]|^2. \tag{2.40}$$

EXAMPLE 2.10 Example of a Power Signal

Consider the causal sequence defined by

$$x[n] = \begin{cases} 3(-1)^n, & n \geq 0, \\ 0, & n < 0. \end{cases}$$

It follows from Eq. (2.34) that $x[n]$ has infinite energy. On the other hand, from Eq. (2.37), its average power is given by

$$\mathcal{P}_x = \lim_{K\to\infty} \frac{1}{2K+1} \left(9 \sum_{n=0}^{K} 1\right) = \lim_{K\to\infty} \frac{9(K+1)}{2K+1} = 4.5,$$

which is finite.

Other Types of Classification

A sequence $x[n]$ is said to be *bounded* if each of its samples is of magnitude less than or equal to a finite positive number B_x; that is,

$$|x[n]| \leq B_x < \infty. \tag{2.41}$$

The periodic sequence of Figure 2.19 is a bounded sequence with a bound $B_x = 2$.

A sequence $x[n]$ is said to be *absolutely summable* if

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty. \tag{2.42}$$

A sequence is said to be *square-summable* if

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty. \tag{2.43}$$

A square-summable sequence, therefore, has finite energy and is an energy signal if it also has zero power. An example of a sequence that is square-summable but not absolutely summable is

$$x_a[n] = \begin{cases} \frac{\sin \omega_c n}{\pi n}, & n < 0, n > 0, \\ \frac{\omega_c}{\pi}, & n = 0. \end{cases}$$

Examples of sequences that are neither absolutely summable nor square-summable are

$$x_b[n] = \sin \omega_c n, \qquad -\infty < n < \infty,$$
$$x_c[n] = K, \qquad -\infty < n < \infty,$$

where K is a constant.

The sequence $\tilde{y}[n]$ obtained by adding an absolutely summable sequence $x[n]$ with its replicas shifted by integer multiples of N,

$$\tilde{y}[n] = \sum_{k=-\infty}^{\infty} x[n + kN], \tag{2.44}$$

where N is a positive integer, is a periodic sequence with a period N (Problem 2.25). The periodic sequence $\tilde{y}[n]$ is called an *N-periodic extension* of $x[n]$.

2.4 Typical Sequences and Sequence Representation

We now consider several special sequences that play important roles in the analysis and design of discrete-time systems. For example, an arbitrary sequence can be expressed in terms of some of these basic sequences. Another fundamental application that is the key behind discrete-time signal processing is the representation of a class of discrete-time systems in terms of the response of the system to certain basic sequences. This representation permits the computation of the response of the discrete-time system to arbitrary discrete-time signals if the latter can be expressed in terms of these basic sequences.

2.4.1 Some Basic Sequences

The most common basic sequences are the unit sample sequence, the unit step sequence, the sinusoidal sequence, and the exponential sequence. These sequences are defined next.

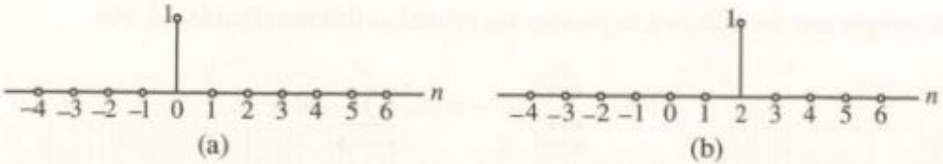


Figure 2.20: (a) The unit sample sequence $\{\delta[n]\}$ and (b) the shifted unit sample sequence $\{\delta[n-2]\}$.

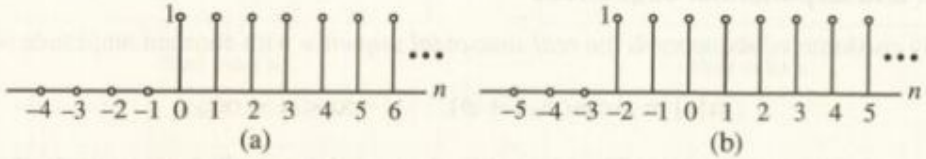


Figure 2.21: (a) The unit step sequence $\{\mu[n]\}$ and the shifted unit step sequence $\{\mu[n+2]\}$.

Unit Sample Sequence

The simplest and one of the most useful sequences is the *unit sample sequence*, often called the *discrete-time impulse* or the *unit impulse*, as shown in Figure 2.20(a). It is denoted by $\delta[n]$ and defined by

$$\delta[n] = \begin{cases} 1, & n = 0, \\ 0, & n \neq 0. \end{cases} \tag{2.45}$$

The unit sample sequence shifted by k samples is thus given by

$$\delta[n-k] = \begin{cases} 1, & n = k, \\ 0, & n \neq k. \end{cases}$$

Figure 2.20(b) shows $\delta[n-2]$. We shall show in Section 2.4.3 that any arbitrary sequence can be represented as a sum of weighted time-shifted unit sample sequences. In Section 3.4, we demonstrate that a certain class of discrete-time systems is completely characterized in the time domain by its output response to a unit impulse input. Furthermore, knowing this particular response of the system, we can compute its response to any arbitrary input sequence.

Unit Step Sequence

A second basic sequence is the *unit step sequence*, shown in Figure 2.21(a). It is denoted by $\mu[n]$ and is defined by

$$\mu[n] = \begin{cases} 1, & n \geq 0, \\ 0, & n < 0. \end{cases} \tag{2.46}$$

The unit step sequence shifted by k samples is thus given by

$$\mu[n-k] = \begin{cases} 1, & n \geq k, \\ 0, & n < k. \end{cases}$$

Figure 2.21(b) shows $\mu[n+2]$.

The unit sample and the unit step sequences are related as follows (Problem 2.26):

$$\mu[n] = \sum_{m=0}^{\infty} \delta[n-m] = \sum_{k=-\infty}^{n} \delta[k], \tag{2.47a}$$

$$\delta[n] = \mu[n] - \mu[n-1]. \tag{2.47b}$$

Sinusoidal and Exponential Sequences

A commonly encountered sequence is the *real sinusoidal sequence* with constant amplitude of the form

$$x[n] = A\cos(\omega_o n + \phi), \qquad -\infty < n < \infty, \tag{2.48}$$

where A, ω_o, and ϕ are real numbers. The parameters A, ω_o, and ϕ are called, respectively, the *amplitude*, the *normalized angular frequency*, and the *phase* of the sinusoidal sequence $x[n]$.

Figure 2.22 shows different types of real sinusoidal sequences. The real sinusoidal sequence of Eq. (2.48) can be written alternatively as

$$x[n] = x_i[n] + x_q[n], \tag{2.49}$$

where $x_i[n]$ and $x_q[n]$ are, respectively, the *in-phase* and the *quadrature components* of $x[n]$, and are given by

$$x_i[n] = A\cos\phi\cos(\omega_o n), \qquad x_q[n] = -A\sin\phi\sin(\omega_o n). \tag{2.50}$$

In the above equations, $A\cos\phi$ is the amplitude of the sinusoidal in-phase component $x_i[n]$ and $A\sin\phi$ is the amplitude of the sinusoidal quadrature component $x_q[n]$.

Another set of basic sequences is formed by taking the nth sample value to be the nth power of a real or complex constant. Such sequences are termed *exponential sequences*, and their most general form is given by

$$x[n] = A\alpha^n, \qquad -\infty < n < \infty, \tag{2.51}$$

where A and α are real or complex numbers. By expressing

$$\alpha = e^{(\sigma_o + j\omega_o)}, \qquad A = |A|e^{j\phi},$$

we can rewrite Eq. (2.51) as

$$x[n] = Ae^{(\sigma_o + j\omega_o)n} = |A|e^{\sigma_o n}e^{j(\omega_o n + \phi)} \tag{2.52a}$$

$$= |A|e^{\sigma_o n}\cos(\omega_o n + \phi) + j\,|A|e^{\sigma_o n}\sin(\omega_o n + \phi), \tag{2.52b}$$

to arrive at an alternative general form of a *complex exponential sequence* where σ_o, ϕ, and ω_o are now real numbers. If we write $x[n] = x_{\text{re}}[n] + jx_{\text{im}}[n]$, then from Eq. (2.52b),

$$x_{\text{re}}[n] = |A|e^{\sigma_o n}\cos(\omega_o n + \phi),$$

$$x_{\text{im}}[n] = |A|e^{\sigma_o n}\sin(\omega_o n + \phi),$$

where $|A|e^{\sigma_o n}$ is the amplitude of the sinusoidal signal $x_{\text{re}}[n]$ and $|A|e^{\sigma_o n}$ is the amplitude of the sinusoidal signal $x_{\text{im}}[n]$. Thus, the real and imaginary parts of a complex exponential sequence are real sinusoidal sequences with constant ($\sigma_o = 0$), growing ($\sigma_o > 0$), or decaying ($\sigma_o < 0$) amplitudes for

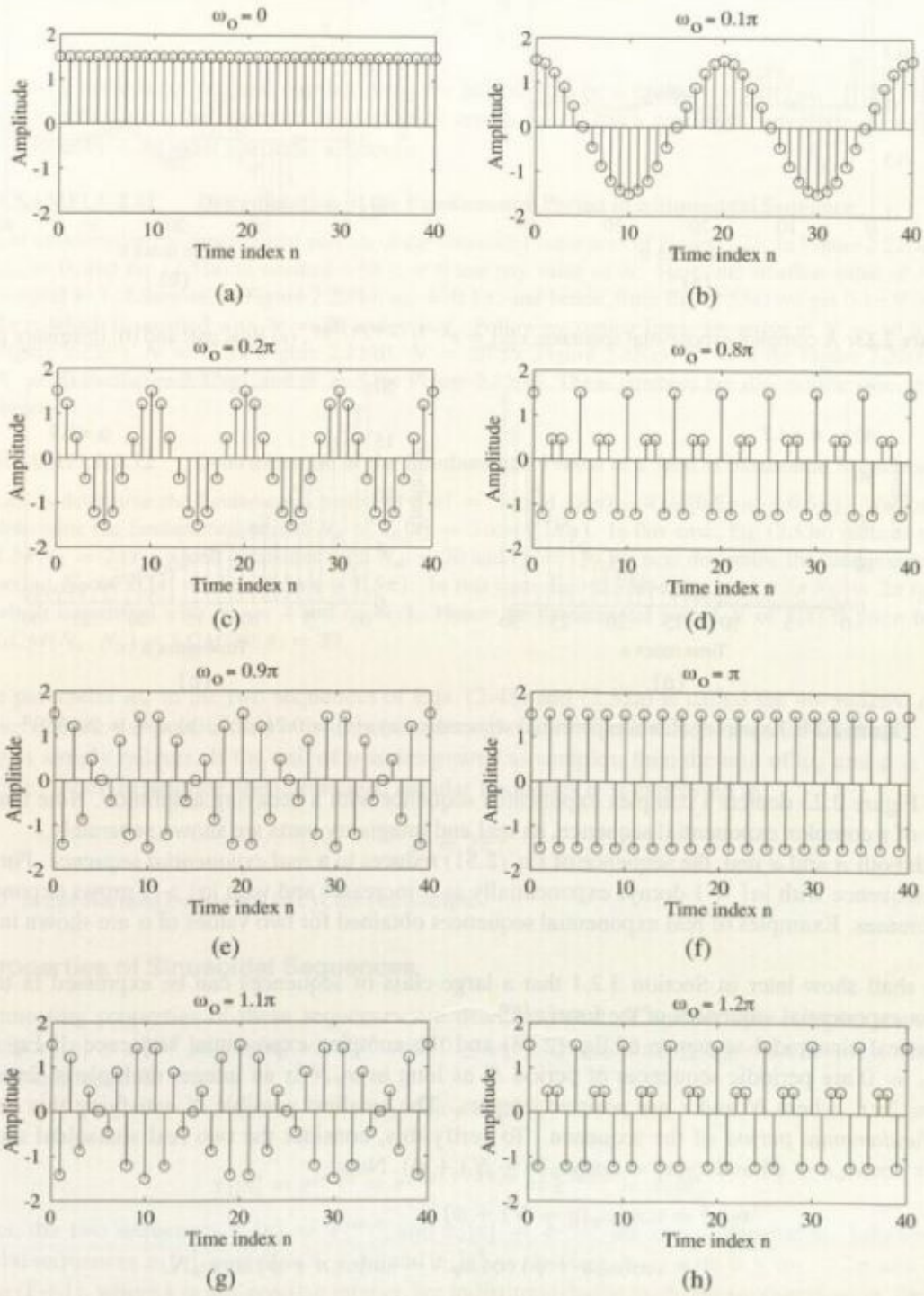


Figure 2.22: A family of sinusoidal sequences given by $x[n] = 1.5\cos\omega_o n$: (a) $\omega_o = 0$, (b) $\omega_o = 0.1\pi$, (c) $\omega_o = 0.2\pi$, (d) $\omega_o = 0.8\pi$, (e) $\omega_o = 0.9\pi$, (f) $\omega_o = \pi$, (g) $\omega_o = 1.1\pi$, and (h) $\omega_o = 1.2\pi$.

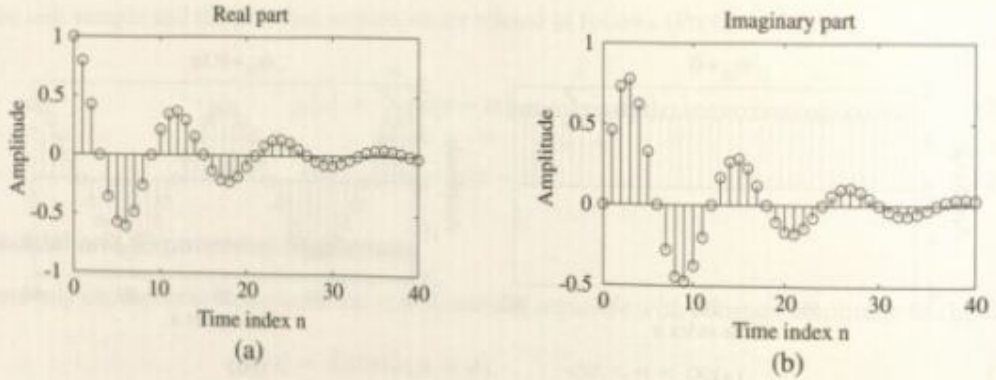


Figure 2.23: A complex exponential sequence $x[n] = e^{(-1/12+j\pi/6)n}$. (a) Real part and (b) imaginary part.

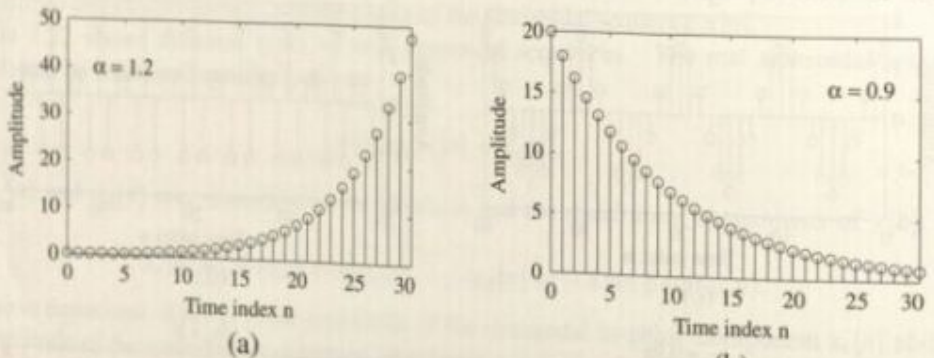


Figure 2.24: Examples of real exponential sequences: (a) $x[n] = 0.2(1.2)^n$, (b) $x[n] = 20(0.9)^n$.

$n > 0$. Figure 2.23 depicts a complex exponential sequence with a decaying amplitude. Note that in the display of a complex exponential sequence, its real and imaginary parts are shown separately.

With both A and α real, the sequence of Eq. (2.51) reduces to a *real exponential sequence*. For $n \geq 0$, such a sequence with $|\alpha| < 1$ decays exponentially as n increases and with $|\alpha| > 1$ grows exponentially as n increases. Examples of real exponential sequences obtained for two values of α are shown in Figure 2.24.

We shall show later in Section 3.2.1 that a large class of sequences can be expressed in terms of complex exponential sequences of the form $e^{j\omega n}$.

The real sinusoidal sequence of Eq. (2.48) and the complex exponential sequence of Eq. (2.52a) with $\sigma_o = 0$ are periodic sequences of period N as long as $\omega_o N$ is an integer multiple of 2π; that is, $\omega_o N = 2\pi r$, where N and r are positive integers. The smallest possible N satisfying this condition is the *fundamental period* of the sequence. To verify this, consider the two real sinusoidal sequences $x_1[n] = \cos(\omega_o n + \phi)$ and $x_2[n] = \cos(\omega_o(n + N) + \phi)$. Now

$$x_2[n] = \cos(\omega_o(n + N) + \phi)$$

$$= \cos(\omega_o n + \phi)\cos\omega_o N - \sin(\omega_o n + \phi)\sin\omega_o N,$$

which will be equal to $\cos(\omega_o n + \phi) = x_1[n]$ only if $\sin\omega_o N = 0$ and $\cos\omega_o N = 1$. These two conditions are satisfied if and only if $\omega_o N$ is an integer multiple of 2π; that is,

$$\omega_o N = 2\pi r \tag{2.53a}$$

or

$$\frac{2\pi}{\omega_o} = \frac{N}{r}. \tag{2.53b}$$

If $2\pi/\omega_o$ is a noninteger rational number, then the period will be a multiple of $2\pi/\omega_o$. If $2\pi/\omega_o$ is not a rational number, then the sequence is aperiodic even though it has a sinusoidal envelope. For example, $x[n] = \cos(\sqrt{3}n + \phi)$ is an aperiodic sequence.

EXAMPLE 2.11 Determination of the Fundamental Period of a Sinusoidal Sequence

Let us determine the fundamental periods of the sinusoidal sequences of Figure 2.22. In Figure 2.22(a), $\omega_o = 0$, and Eq. (2.53a) is satisfied with $r = 0$ and any value of N. Here, the smallest value of N is equal to 1. Likewise, in Figure 2.22(b), $\omega_o = 0.1\pi$, and hence, from Eq. (2.53a) we get $0.1\pi N = 2\pi r$, which is satisfied with $N = 20$ and $r = 1$. Following similar lines, we arrive at $N = 10$ for Figure 2.22(c), $N = 5$ for Figure 2.22(d), $N = 20$ for Figure 2.22(e), $N = 2$ for Figure 2.22(f), $N = 20$ for Figure 2.22(g), and $N = 5$ for Figure 2.22(h). These numbers are also evident from the figures.

EXAMPLE 2.12 Determination of the Fundamental Period of a Sum of Sinusoidal Sequences

Let us determine the fundamental period of $\tilde{x}[n] = 3\cos(1.3\pi n) - 4\sin(0.5\pi n + 0.5\pi)$. We first determine the fundamental period N_a of $\tilde{x}_a[n] = 3\cos(1.3\pi n)$. In this case, Eq. (2.53a) reduces to $1.3\pi N_a = 2\pi r_a$, which is satisfied with $N_a = 20$ and $r_a = 13$. We next determine the fundamental period N_b of $\tilde{x}_b[n] = 4\sin(0.5\pi n + 0.5\pi)$. In this case, Eq. (2.53a) reduces to $0.5\pi N_b = 2\pi r_b$, which is satisfied with $N_b = 4$ and $r_b = 1$. Hence the fundamental period N of $\tilde{x}[n]$ is given by $LCM(N_a, N_b) = LCM(20, 4) = 20$.

The parameter ω_o in the two sequences of Eqs. (2.48) and (2.52a) is called the *normalized angular frequency*. Since the time instant n is dimensionless, the unit of normalized angular frequency ω_o and *phase* ϕ is simply radians. If the unit of n is designated as samples, then the unit of ω_o and ϕ is radians per sample. Often in practice, the normalized angular frequency ω is expressed as

$$\omega = 2\pi f, \tag{2.54}$$

where f is *normalized frequency* in cycles per sample.

Two Properties of Sinusoidal Sequences

Two interesting properties of these sequences are discussed next. Consider two complex exponential sequences $x_1[n] = e^{j\omega_1 n}$ and $x_2[n] = e^{j\omega_2 n}$ with $0 \le \omega_1 < 2\pi$ and $2\pi k \le \omega_2 < 2\pi(k+1)$, where k is any positive integer. If

$$\omega_2 = \omega_1 + 2\pi k, \tag{2.55}$$

then

$$x_2[n] = e^{j\omega_2 n} = e^{j(\omega_1 + 2\pi k)n} = e^{j\omega_1 n} = x_1[n].$$

Thus, the two sequences $x_1[n] = e^{j\omega_1 n}$ and $x_2[n] = e^{j\omega_2 n}$ are indistinguishable. Likewise, two sinusoidal sequences $x_1[n] = \cos(\omega_1 n + \phi)$ and $x_2[n] = \cos(\omega_2 n + \phi)$ with $0 \le \omega_1 < 2\pi$ and $2\pi k \le \omega_2 < 2\pi(k+1)$, where k is any positive integer, are indistinguishable from one another if $\omega_2 = 2\pi k + \omega_1$. In other words, any exponential or sinusoidal sequence of a normalized angular frequency ω_2 with a value outside the frequency range $0 \le \omega < 2\pi$ is equivalent to an exponential or sinusoidal sequence of a normalized angular frequency of value ω_2 modulo 2π.

The second interesting feature of the discrete-time sinusoidal signals follows from the property of the sine and cosine functions. To illustrate this feature, consider two sinusoidal sequences $x_1[n] = \cos(\omega_1 n)$ and $x_2[n] = \cos(\omega_2 n)$ with $0 \leq \omega_1 < \pi$ and $\pi \leq \omega_2 < 2\pi$. Let $\omega_2 = 2\pi - \omega_1$. Then it follows that $x_2[n] = \cos(2\pi n - \omega_1 n) = \cos(\omega_1 n) = x_1[n]$. Thus, the two sinusoidal sequences $x_1[n]$ and $x_2[n]$ are indistinguishable from one another. Hence, a sinusoidal sequence with a normalized angular frequency ω_2 in the range $\pi \leq \omega_2 < 2\pi$ assumes the identity of a sinusoidal sequence with a normalized angular frequency $\omega_1 = 2\pi - \omega_2$ in the range $0 \leq \omega_1 < \pi$. The frequency π is thus called the *folding frequency*. An implication of this property of the sinusoidal sequences can be seen in Figure 2.22. The frequency of oscillation of the discrete-time sinusoidal sequence $x[n] = A\cos(\omega_o n)$ increases as ω_o increases from 0 to π, and then the frequency of oscillation decreases as ω_o increases from π to 2π. As a result of the first property, a normalized angular frequency ω_o in the neighborhood of $\omega = 2\pi k$ is indistinguishable from a normalized angular frequency $\omega_o - 2\pi k$ in the neighborhood of $\omega = 0$, and a normalized angular frequency ω_o in the neighborhood of $\omega = \pi(2k + 1)$ is indistinguishable from a normalized angular frequency $\omega_o - \pi(2k+1)$ in the neighborhood of $\omega = \pi$ for any integer value of k. Therefore, frequencies in the neighborhood of $\omega_o = 2\pi k$ are usually called *low frequencies*, and frequencies in the neighborhood of $\omega_o = \pi(2k + 1)$ are called *high frequencies*. For example, $v_1[n] = \cos(0.1\pi n) = \cos(1.9\pi n)$, shown in Figure 2.22(b), is a low-frequency signal, whereas, $v_2[n] = \cos(0.8\pi n) = \cos(1.2\pi n)$, shown in Figure 2.22(d) and (h), is a high-frequency signal.

Rectangular Window Sequence

The *rectangular window sequence*, also called a *box-car sequence*, is a sequence $w_R[n]$ with unity sample values in a finite range $N_1 \leq n \leq N_2$ and zero-valued samples outside this range; that is

$$w_R[n] = \begin{cases} 0, & n < N_1, \\ 1, & N_1 \leq n \leq N_2, \\ 0, & n > N_2. \end{cases} \tag{2.56}$$

An application of the above sequence is in extracting all samples in the range $N_1 \leq n \leq N_2$ of an infinite-length or very long length sequence $x[n]$ and setting all other samples outside the above range to zero values by multiplying $x[n]$ with $w_R[n]$. Thus,

$$x[n] \cdot w_R[n] = \begin{cases} 0, & n < N_1, \\ x[n], & N_1 \leq n \leq N_2, \\ 0, & n > N_2. \end{cases} \tag{2.57}$$

The operation described in the above equations is called *windowing* and plays an important role in the design of certain types of digital filters (Section 10.2). It is also used in the short-term analysis of non-stationary signals.

Generation of Sequences with Complex Waveforms

The elementary operations described in Section 2.2.1 can be combined to generate sequences with more complex waveforms. We illustrate this application in Example 2.13.

EXAMPLE 2.13 Generation of a Square Wave Sequence

Consider the three sinusoidal sequences $x_1[n]$, $x_2[n]$, and $x_3[n]$ with normalized angular frequencies 0.05, 0.15, and 0.25, respectively, as given below:

$$x_1[n] = \sin(0.05\pi n)\mu[n],$$

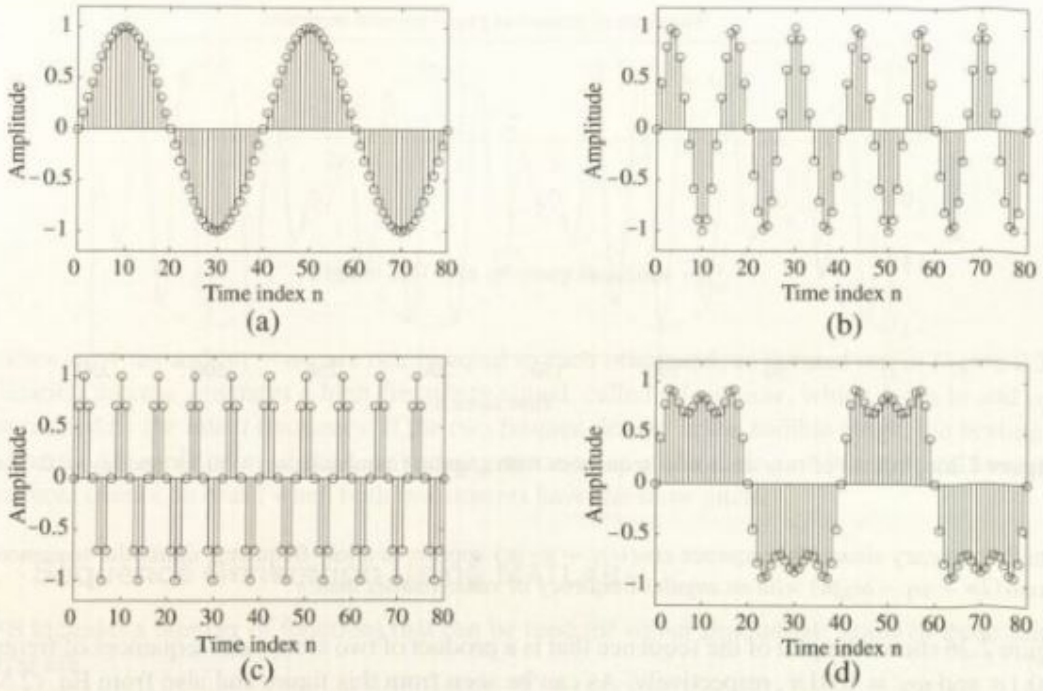


Figure 2.25: Sequences of Example 2.13: (a) $x_1[n]$, (b) $x_2[n]$, (c) $x_3[n]$, and (d) $y[n]$.

$$x_2[n] = \sin(0.15\pi n)\mu[n],$$

$$x_3[n] = \sin(0.25\pi n)\mu[n].$$

A plot of these three sinusoidal sequences is given in Figure 2.25(a)–(c), respectively. A new sequence $y[n]$ is formed by combining the above three sequences as follows:

$$y[n] = x_1[n] + \tfrac{1}{3}x_2[n] + \tfrac{1}{5}x_3[n],$$

whose plot is indicated in Figure 2.25(d). It can be seen that this new sequence has almost a square waveform.

An application of the modulation operation discussed earlier in Section 2.2.1 is to transform a sequence with low-frequency sinusoidal components to a sequence with high-frequency components by modulating the former with a sinusoidal sequence of very high frequency, as illustrated in Example 2.14.

EXAMPLE 2.14 Illustration of the Modulation Operation

Let $x_1[n] = \cos(\omega_1 n)$ and $x_2[n] = \cos(\omega_2 n)$ with $\pi > \omega_2 >> \omega_1 > 0$. The product sequence $y[n] = x_1[n]x_2[n]$ is then given by

$$y[n] = \cos(\omega_1 n) \cdot \cos(\omega_2 n). \tag{2.58}$$

Using a trigonometric identity, we arrive at

$$y[n] = \tfrac{1}{2}\cos((\omega_1 + \omega_2)n) + \tfrac{1}{2}\cos((\omega_2 - \omega_1)n). \tag{2.59}$$

The new sequence $y[n]$ is thus composed of two sinusoidal sequences of frequencies $\omega_1 + \omega_2$ and $\omega_2 - \omega_1$. It should be noted that, because of the property given by Eq. (2.55), if $\omega_1 + \omega_2 > \pi$, the

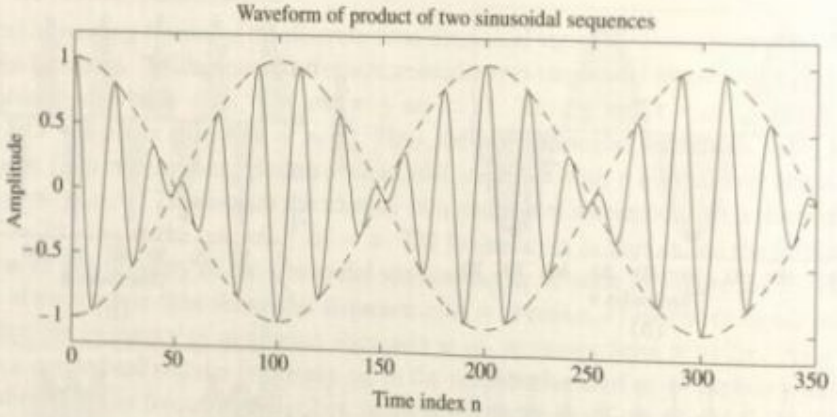


Figure 2.26: Product of two sinusoidal sequences with angular frequencies $\omega_1 = 0.1\pi$ and $\omega_2 = 0.01\pi$.

high-frequency sinusoidal sequence $\cos((\omega_1 + \omega_2)n)$ appears as a low-frequency sinusoidal sequence $\cos((2\pi - \omega_1 - \omega_2)n)$ with an angular frequency of value smaller than π.[5]

Figure 2.26 shows the plot of the sequence that is a product of two sinusoidal sequences of frequencies $\omega_1 = 0.1\pi$ and $\omega_2 = 0.01\pi$, respectively. As can be seen from this figure and also from Eq. (2.58), the product signal is a high-frequency sinusoidal sequence with an amplitude sinusoidally varying with time at a lower frequency.

Fundamental and Harmonic Components

It should be noted that the angular frequency of the sequence $x_2[n]$ in Example 2.13 is 3 times that of $x_1[n]$, and the angular frequency of the sequence $x_3[n]$ is 5 times that of $x_1[n]$. The sinusoidal sequences whose angular frequencies are integer multiples of a sinusoidal sequence of lower angular frequency are called the *harmonics*, whereas the sinusoidal sequence with the lower frequency is called the *fundamental component*, with its frequency being called the *fundamental frequency*. The harmonic component with a frequency that is k times that of the fundamental frequency is called the *k-th harmonic*. In Example 2.13, the sequence $x_1[n]$ is the fundamental component with a fundamental angular frequency of 0.05 radian per sample. The sequences $x_2[n]$ and $x_3[n]$, are respectively, the third and fifth harmonics.

Any periodic sequence can be expressed in the form of linearly weighted combinations of a fundamental and a series of harmonic components called the *Fourier series* expansion. The weights associated with each component in the expansion are called the Fourier series coefficients. In Example 2.13, the Fourier series coefficients of $x_1[n]$, $x_2[n]$, and $x_3[n]$, are respectively, 1, 1/3, and 1/5. The development of the Fourier series expansion of a periodic sequence is considered in Problem 5.3.

Beat Notes

As shown in Example 2.14, the multiplication of two sinusoidal signals $x_1[n]$ and $x_2[n]$ with normalized angular frequencies ω_1 and ω_2, respectively, results in a sum of two sinusoidal signals whose normalized angular frequencies are $\omega_1 + \omega_2$ and $\omega_1 - \omega_2$, respectively; that is, the sum and differences of the normalized angular frequencies of the original sinusoidal signals. If $\omega_2 \ll \omega_1$, then the two normalized angular

[5]The appearance of a high-frequency signal $\cos((\omega_1 + \omega_2)n)$ as a low-frequency signal $\cos((2\pi - \omega_1 - \omega_2)n)$ is called aliasing (see Section 2.5).

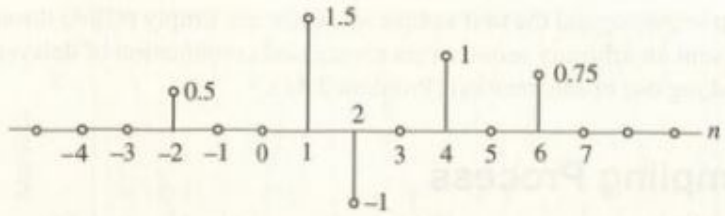


Figure 2.27: An arbitrary sequence $x[n]$.

frequencies $\omega_1 + \omega_2$ and $\omega_1 - \omega_2$ are nearly equal to each other, and, as pointed out in Figure 2.26, the multiplication process generates a high-frequency signal, called a *beat note*, which fades in and out at a rate determined by the lower frequency. If the two frequencies are in the audible range, the beating effect can be heard. An application of the beat signal generation is in the tuning of two musical instruments. The beat note cannot be heard when both instruments have the same pitch.

2.4.2 Sequence Generation Using MATLAB

MATLAB includes a number of functions that can be used for signal generation. Some of these functions of interest are

```
exp,   sin,   cos,   square,   sawtooth
```

For example, the code fragments to generate a length-N complex exponential sequence with an exponent $a + jb$ and of the form shown in Figure 2.23 is given by

```
n = 1:N;
x = K*exp((a + b*i)*n);
```

Program 2_3.m
Program 2_4.m

The complete code is given in Program 2_3. Likewise, the code fragments to generate a length-$(N + 1)$ real exponential sequence with an exponent a and of the form shown in Figure 2.24 is given by

```
n = 0:N;
x = K*a.^n;
```

The complete code is given in Program 2_4. Another type of sequence generation using MATLAB can be found earlier in Example 2.1.

2.4.3 Representation of an Arbitrary Sequence

An arbitrary sequence can be represented in the time domain as a weighted sum of the delayed and advanced versions of a basic sequence. A commonly used basic sequence in the representation of an arbitrary sequence is the unit sample sequence. For example, the sequence $x[n]$ of Figure 2.27 can be expressed as

$$x[n] = 0.5\,\delta[n+2] + 1.5\,\delta[n-1] - \delta[n-2] + \delta[n-4] + 0.75\,\delta[n-6]. \tag{2.60}$$

An implication of this type of representation is considered in Section 4.4.1, where we develop the general expression for calculating the output sequence of certain types of discrete-time systems for an arbitrary input sequence.

Since the unit step sequence and the unit sample sequence are simply related through Eq. (2.47b), it is also possible to represent an arbitrary sequence as a weighted combination of delayed and advanced unit step sequences by making use of this relation (Problem 2.44).

2.5 The Sampling Process

We indicated earlier that often the discrete-time sequence is developed by uniformly sampling a continuous-time signal $x_a(t)$, as illustrated in Figure 2.2. The relation between the two signals is given by Eq. (2.2), where the time variable t of the continuous-time signal is related to the time variable n of the discrete-time signal only at discrete-time instants t_n given by

$$t_n = nT = \frac{n}{F_T} = \frac{2\pi n}{\Omega_T}, \tag{2.61}$$

with $F_T = 1/T$ denoting the sampling frequency and $\Omega_T = 2\pi F_T$ denoting the sampling angular frequency. For example, if the continuous-time signal is

$$x_a(t) = A\cos(2\pi f_o t + \phi) = A\cos(\Omega_o t + \phi), \tag{2.62}$$

the corresponding discrete-time signal is given by

$$\begin{aligned} x[n] &= A\cos(\Omega_o nT + \phi) \\ &= A\cos\left(\frac{2\pi\Omega_o}{\Omega_T}n + \phi\right) = A\cos(\omega_o n + \phi), \end{aligned} \tag{2.63}$$

where

$$\omega_o = \frac{2\pi\Omega_o}{\Omega_T} = \Omega_o T \tag{2.64}$$

is the normalized angular frequency of the discrete-time signal $x[n]$. The unit of the normalized angular frequency ω_o is radians per sample, while the unit of the angular frequency Ω_o of the continuous-time signal is radians per second and the unit of the frequency f_o is Hertz if the sampling period T has the physical dimension of time.

EXAMPLE 2.15 Ambiguity in the Discrete-Time Representation of Continuous-Time Signals

Consider the three sequences generated by uniformly sampling the three cosine functions of frequencies 3 Hz, 7 Hz, and 13 Hz, respectively: $g_1(t) = \cos(6\pi t/\text{sec})$, $g_2(t) = \cos(14\pi t/\text{sec})$, and $g_3(t) = \cos(26\pi t/\text{sec})$ with a sampling rate of 10 Hz; that is, with $T = 0.1$ sec. The derived sequences are therefore

$$g_1[n] = \cos(0.6\pi n), \qquad g_2[n] = \cos(1.4\pi n), \qquad g_3[n] = \cos(2.6\pi n).$$

Plots of these sequences (shown with circles) and their parent time functions are given in Figure 2.28. Note from these plots that each sequence has exactly the same sample value for any given n. This also can be verified by observing that

$$g_2[n] = \cos(1.4\pi n) = \cos((2\pi - 0.6\pi)n) = \cos(0.6\pi n)$$

and

$$g_3[n] = \cos(2.6\pi n) = \cos((2\pi + 0.6\pi)n) = \cos(0.6\pi n).$$

As a result, all three sequences above are identical, and it is difficult to associate a unique continuous-time function with any one of these sequences. In fact, all cosine waveforms of frequencies given by $(10k \pm 3)$ Hz, with k being any nonnegative integer, lead to the sequence $g_1[n] = \cos(0.6\pi n)$ when sampled at a 10-Hz rate.

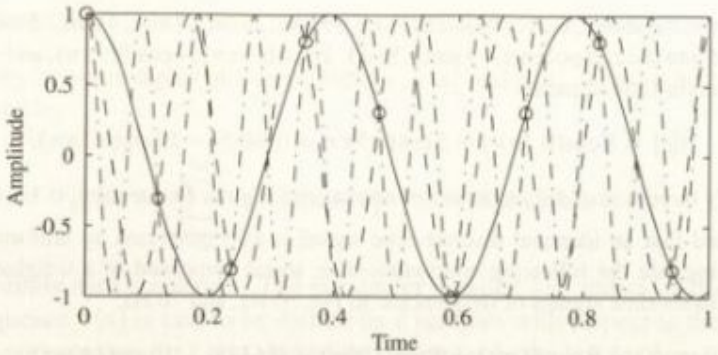


Figure 2.28: Ambiguity in the discrete-time representation of continuous-time signals. $g_1(t)$ is shown with the solid line, $g_2(t)$ is shown with the dashed line, $g_3(t)$ is shown with the dashed-dot line, and the sequence obtained by sampling is shown with circles.

In the general case, the family of continuous-time sinusoids

$$x_{a,k}(t) = A\cos(\pm(\Omega_o t + \phi) + k\Omega_T t), \qquad k = 0, \pm 1, \pm 2, \ldots \tag{2.65}$$

leads to identical sampled signals:

$$\begin{aligned} x_{a,k}(nT) &= A\cos((\Omega_o + k\Omega_T)nT + \phi) = A\cos\left(\frac{2\pi(\Omega_o + k\Omega_T)n}{\Omega_T} + \phi\right) \\ &= A\cos\left(\frac{2\pi\Omega_o n}{\Omega_T} + \phi\right) = A\cos(\omega_o n + \phi) = x[n]. \end{aligned} \tag{2.66}$$

The above phenomenon of a continuous-time sinusoidal signal of higher frequency acquiring the identity of a sinusoidal sequence of lower frequency after sampling is called *aliasing*. Since there are an infinite number of continuous-time functions that can lead to a given sequence when sampled periodically, additional conditions need to be imposed so that the sequence $\{x[n]\} = \{x_a(nT)\}$ can uniquely represent the parent continuous-time function $x_a(t)$. In which case, $x_a(t)$ can be fully recovered from a knowledge of $\{x[n]\}$.

EXAMPLE 2.16 Illustration of Aliasing

Determine the discrete-time signal $v[n]$ obtained by uniformly sampling a continuous-time signal $v_a(t)$ composed of a weighted sum of five sinusoidal signals of frequencies 30 Hz, 150 Hz, 170 Hz, 250 Hz, and 330 Hz, at a sampling rate of 200 Hz, as given below:

$$v_a(t) = 6\cos(60\pi t/\text{sec}) + 3\sin(300\pi t/\text{sec}) + 2\cos(340\pi t/\text{sec}) + 4\cos(500\pi t/\text{sec}) + 10\sin(660\pi t/\text{sec}).$$

The sampling period $T = 1/200 = 0.005$ sec. Hence, the generated discrete-time signal $v[n]$ is given by

$$\begin{aligned} v[n] &= 6\cos(0.3\pi n) + 3\sin(1.5\pi n) + 2\cos(1.7\pi n) + 4\cos(2.5\pi n) \\ &\quad + 10\sin(3.3\pi n) \\ &= 6\cos(0.3\pi n) + 3\sin((2\pi - 0.5\pi)n) + 2\cos((2\pi - 0.3\pi)n) \\ &\quad + 4\cos((2\pi + 0.5\pi)n) + 10\sin((4\pi - 0.7\pi)n) \\ &= 6\cos(0.3\pi n) - 3\sin(0.5\pi n) + 2\cos(0.3\pi n) + 4\cos(0.5\pi n) \\ &\quad - 10\sin(0.7\pi n). \end{aligned}$$


Aliasing Demo

As can be seen from the above, the components $3\sin(1.5\pi n)$, $2\cos(1.7\pi n)$, $4\cos(2.5\pi n)$, and $10\sin(3.3\pi n)$ have been aliased into the components $-3\sin(0.5\pi n)$, $2\cos(0.3\pi n)$, $4\cos(0.5\pi n)$, and $-10\sin(0.7\pi n)$, resulting in a discrete-time sequence

$$v[n] = 8\cos(0.3\pi n) + 5\cos(0.5\pi n + 0.6435) - 10\sin(0.7\pi n),$$

composed of only three sinusoidal sequences of normalized angular frequencies: 0.3π, 0.5π, and 0.7π.

It should be noted that an identical discrete-time signal is also generated by uniformly sampling at a 200-Hz sampling rate the following continuous-time signal composed of a weighted sum of three sinusoidal continuous-time signals of frequencies 30 Hz, 50 Hz, and 70 Hz:

$$w_a(t) = 8\cos(60\pi t) + 5\cos(100\pi t + 0.6435) - 10\sin(140\pi t).$$

Another example of a continuous-time signal generating the same discrete-time sequence is

$$u_a(t) = 2\cos(60\pi t/\text{sec}) + 4\cos(100\pi t/\text{sec}) + 10\sin(260\pi t/\text{sec}) + 6\cos(460\pi t/\text{sec}) + 3\sin(700\pi t/\text{sec}),$$

which is composed of a weighted sum of five sinusoidal continuous-time signals of frequencies 30 Hz, 50 Hz, 130 Hz, 230 Hz, and 350 Hz, respectively.

It follows from Eq. (2.64) that if $\Omega_T > 2\Omega_o$, then the corresponding normalized digital frequency ω_o of the discrete-time signal $x[n]$ obtained by sampling the parent continuous-time signal $x_a(t)$ will be in the range $-\pi < \omega < \pi$, implying no aliasing. On the other hand, if $\Omega_T < 2\Omega_o$, the normalized digital frequency will fold into a lower digital frequency ω_o in the range $-\pi < \omega < \pi$ because of aliasing. The value of ω_o is given by $2\pi\Omega_o/\Omega_T$ modulo 2π. Hence, to prevent aliasing, the sampling frequency Ω_T should be greater than 2 times the frequency Ω_o of the sinusoidal signal being sampled. Generalizing the above result, we observe that if we have an arbitrary continuous-time signal $x_a(t)$ that can be represented as a weighted sum of a number of sinusoidal signals, then $x_a(t)$ can also be represented uniquely by its sampled version $\{x[n]\}$ if the sampling frequency Ω_T is chosen to be greater than 2 times the highest frequency contained in $x_a(t)$. The condition to be satisfied by the sampling frequency to prevent aliasing is called the *sampling theorem*, which is formally derived later in Section 3.8.1.

The discrete-time signal obtained by sampling a continuous-time signal $x_a(t)$ may be represented as a sequence $\{x_a[nT]\}$. However, we shall use the more common notation $\{x[n]\}$ for simplicity. It should be noted that when dealing with the sampled version of a continuous-time function, it is essential to know the numerical value of the sampling period T.

2.6 Correlation of Signals

There are applications where it is necessary to compare one reference signal with one or more signals to determine the similarity between the pair and to determine additional information based on the similarity. A measure of the similarity between signals is given by the correlation sequence. For example, in digital communications, a set of data symbols are represented by a set of unique discrete-time sequences. If one of these sequences is transmitted, the receiver has to determine which particular sequence has been received by comparing the received signal with every member of possible sequences from the set. Similarly, in radar and sonar applications, the received signal reflected from the target is the delayed version of the transmitted signal, and by measuring the delay, one can determine the location of the target. The detection problem gets more complicated in practice, as often the received signal is corrupted by additive random noise. In this section, we define the correlation sequence and study its properties.

2.6.1 Definitions

A measure of similarity between a pair of energy signals, $x[n]$ and $y[n]$, is given by the *cross-correlation sequence* $r_{xy}[\ell]$ defined by

$$r_{xy}[\ell] = \sum_{n=-\infty}^{\infty} x[n]y[n-\ell], \quad \ell = 0, \pm 1, \pm 2, \ldots. \tag{2.67}$$

provided the above infinite sum converges. The parameter ℓ called *lag*, indicates the time-shift between the pair. The time sequence $y[n]$ is said to be shifted by ℓ samples with respect to the reference sequence $x[n]$ to the right for positive values of ℓ and shifted by ℓ samples to the left for negative values of ℓ.

The ordering of the subscripts xy in Eq. (2.67) specifies that $x[n]$ is the reference sequence that remains fixed in time, whereas the sequence $y[n]$ is being shifted with respect to $x[n]$. If we wish to make $y[n]$ the reference sequence and shift the sequence $x[n]$ with respect to $y[n]$, then the corresponding cross-correlation sequence is given by

$$\begin{aligned} r_{yx}[\ell] &= \sum_{n=-\infty}^{\infty} y[n]x[n-\ell] \\ &= \sum_{m=-\infty}^{\infty} y[m+\ell]x[m] = r_{xy}[-\ell]. \end{aligned} \tag{2.68}$$

Thus, $r_{yx}[\ell]$ is obtained by time-reversing the sequence $r_{xy}[\ell]$.

The *autocorrelation* sequence of $x[n]$ is given by

$$r_{xx}[\ell] = \sum_{n=-\infty}^{\infty} x[n]x[n-\ell], \tag{2.69}$$

obtained by setting $y[n] = x[n]$ in Eq. (2.67). Note from Eq. (2.69) that $r_{xx}[0] = \sum_{n=-\infty}^{\infty} x^2[n] = \mathcal{E}_x$, the energy of the signal $x[n]$. From Eq. (2.68), it follows that $r_{xx}[\ell] = r_{xx}[-\ell]$, implying that $r_{xx}[\ell]$ is an even sequence for a real sequence $x[n]$.

An examination of Eq. (2.67) reveals that the expression for the cross-correlation looks quite similar to that of the convolution given by Eq. (2.20a). This similarity is much clearer if we rewrite Eq. (2.67) as

$$r_{xy}[\ell] = \sum_{n=-\infty}^{\infty} x[n]y[-(\ell-n)] = x[\ell] \circledast y[-\ell]. \tag{2.70}$$

It thus follows that the cross-correlation of the sequence $x[n]$ with the reference sequence $y[n]$ is simply given by the convolution sum of $x[n]$ with $y[-n]$, the time-reversed version of $y[n]$. Likewise, the autocorrelation of $x[n]$ is given by the convolution sum of itself with its time-reversed version.

2.6.2 Properties of Autocorrelation and Cross-Correlation Sequences

We next derive some basic properties of the autocorrelation and cross-correlation sequences [Pro96]. Consider two finite-energy sequences $x[n]$ and $y[n]$. Now, the energy of the combined sequence $ax[n] +$

$y[n-\ell]$ is also finite and nonnegative. That is,

$$\sum_{n=-\infty}^{\infty}(ax[n]+y[n-\ell])^2 = a^2\sum_{n=-\infty}^{\infty}x^2[n] + 2a\sum_{n=-\infty}^{\infty}x[n]y[n-\ell] + \sum_{n=-\infty}^{\infty}y^2[n-\ell]$$

$$= a^2 r_{xx}[0] + 2a r_{xy}[\ell] + r_{yy}[0] \geq 0, \tag{2.71}$$

where $r_{xx}[0] = \mathcal{E}_x > 0$ and $r_{yy}[0] = \mathcal{E}_y > 0$ are energies of the sequences $x[n]$ and $y[n]$, respectively. We can rewrite Eq. (2.71) as

$$\begin{bmatrix} a & 1 \end{bmatrix}\begin{bmatrix} r_{xx}[0] & r_{xy}[\ell] \\ r_{xy}[\ell] & r_{yy}[0] \end{bmatrix}\begin{bmatrix} a \\ 1 \end{bmatrix} \geq 0$$

for any finite value of a. Or, in other words, the matrix

$$\begin{bmatrix} r_{xx}[0] & r_{xy}[\ell] \\ r_{xy}[\ell] & r_{yy}[0] \end{bmatrix}$$

is positive semi-definite. This implies

$$r_{xx}[0]r_{yy}[0] - r_{xy}^2[\ell] \geq 0$$

or equivalently,

$$|r_{xy}[\ell]| \leq \sqrt{r_{xx}[0]r_{yy}[0]} = \sqrt{\mathcal{E}_x\mathcal{E}_y}. \tag{2.72}$$

The above inequality provides an upper bound for the cross-correlation sequence samples. If we set $y[n] = x[n]$, the above reduces to

$$|r_{xx}[\ell]| \leq r_{xx}[0] = \mathcal{E}_x. \tag{2.73}$$

This is a significant result as it states that at zero lag ($\ell = 0$), the sample value of the autocorrelation sequence has its maximum value.

To derive an additional property of the cross-correlation sequence, consider the case

$$y[n] = \pm\, b\, x[n-N],$$

where N is an integer and $b > 0$ is an arbitrary number. In this case $\mathcal{E}_y = b^2\mathcal{E}_x$, and therefore,

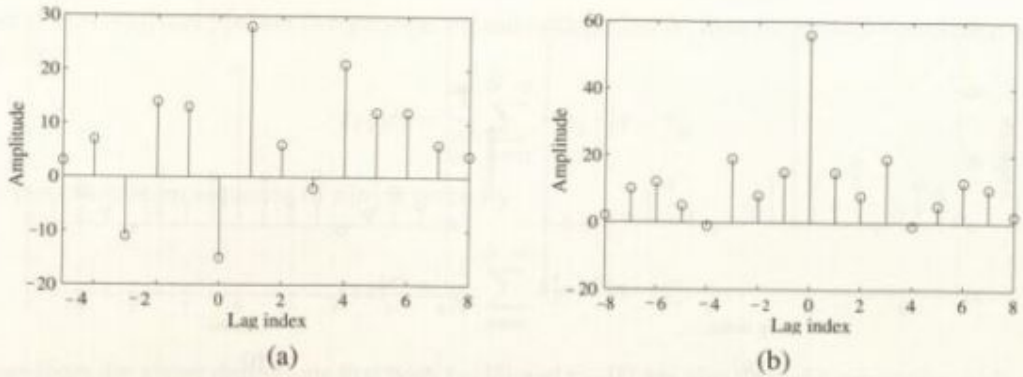
$$\sqrt{\mathcal{E}_x\mathcal{E}_y} = \sqrt{b^2\mathcal{E}_x^2} = b\mathcal{E}_x.$$

Using the above result in Eq. (2.72) we get in this case

$$-b\, r_{xx}[0] \leq r_{xy}[\ell] \leq b\, r_{xx}[0].$$

2.6.3 Correlation Computation Using MATLAB

The cross-correlation and the autocorrelation sequences can be easily computed using MATLAB, as illustrated in Examples 2.17 and 2.18.



(a) (b)

Figure 2.29: (a) Cross-correlation sequence and (b) autocorrelation sequence.

EXAMPLE 2.17 Cross-Correlation Computation Using MATLAB

Consider the two finite-length sequences

$$x[n] = [1,\ 3,\ -2,\ 1,\ 2,\ -1,\ 4,\ 4,\ 2],$$
$$y[n] = [2,\ -1,\ 4,\ 1,\ -2,\ 3].$$

Using MATLAB, we determine and plot the cross-correlation sequence $r_{xy}[\ell]$.

As indicated by Eq. (2.70), the cross-correlation sequence $r_{xy}[\ell]$ of two sequences $x[n]$ and $y[n]$ is given by the linear convolution of $x[n]$ and the time-reversed sequence $y[-n]$. Hence, the cross-correlation of two finite-length sequences can be computed using Program 2_5, which makes use of the M-file `conv`. The pertinent code fragment is

```
r = conv(x,fliplr(y));
```

Program 2_5.m

The cross-correlation sequence of the two finite-length sequences given above obtained using Program 2_5 is shown in Figure 2.29(a).

EXAMPLE 2.18 Autocorrelation Computation Using MATLAB

In this example, we evaluate and plot the autocorrelation of the sequence $x[n]$ of Example 2.17. Next, by adding a random noise $d[n]$ to $x[n]$, we compute and plot the autocorrelation of the noise-corrupted sequence.

Program 2_5 can also be used to compute the autocorrelation sequence of a finite-length sequence. The plot of the autocorrelation sequence $r_{xx}[\ell]$ generated by this program for the sequence $x[n]$ of Example 2.17 is shown in Figure 2.29(b). As expected at zero lag, $r_{xx}[0]$ is the maximum.

Program 2_5 is run again by computing the cross-correlation of $x[n]$ and $y[n] = x[n - N]$ for $N = 4$. As can be seen from the plot in Figure 2.30(a), the peak of the cross-correlation is precisely the value of N used, thus demonstrating the fact that the cross-correlation can be employed to compute the exact value of the delay N.

Next, Program 2_5 is modified to generate a sequence formed by adding a random noise computed using the function `rand` to $x[n]$, and the autocorrelation of the noise-corrupted sequence is plotted in Figure 2.30(b). As expected, the autocorrelation still exhibits a pronounced peak at zero lag.

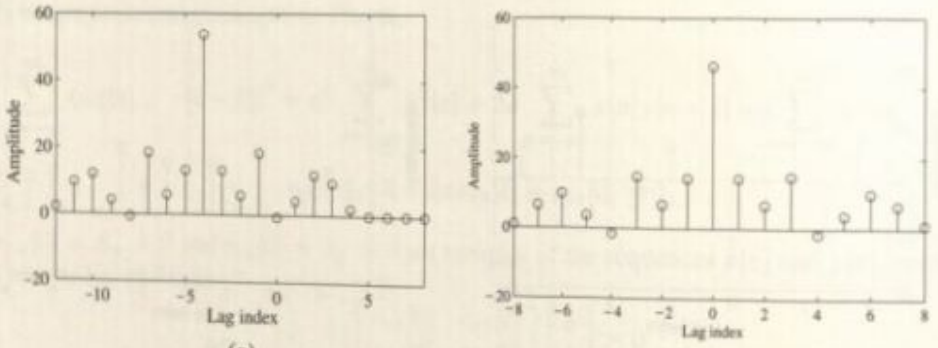


Figure 2.30: (a) Delay estimation from cross-correlation sequence and (b) autocorrelation sequence of a noise-corrupted aperiodic sequence.

It should be noted that the autocorrelation and cross-correlation sequences can also be computed using the MATLAB function `xcorr`. However, the correlation sequences generated using this function are the time-reversed version of those generated using Programs 2_5. The cross-correlation $r_{xy}[\ell]$ of two sequences $x[n]$ and $y[n]$ can be computed using the statement `r = xcorr(x,y)`, while the autocorrelation $r_{xx}[\ell]$ of the sequence $x[n]$ is determined using the statement `r = xcorr(x)`.

2.6.4 Normalized Forms of Correlation

For convenience in comparing and displaying, normalized forms of autocorrelation and cross-correlation given by

$$\rho_{xx}[\ell] = \frac{r_{xx}[\ell]}{r_{xx}[0]}, \tag{2.74}$$

$$\rho_{xy}[\ell] = \frac{r_{xy}[\ell]}{\sqrt{r_{xx}[0]\, r_{yy}[0]}}, \tag{2.75}$$

are often used. It follows from Eqs. (2.72) and (2.73) that $|\rho_{xx}[\ell]| \leq 1$ and $|\rho_{xy}[\ell]| \leq 1$, independent of the range of values of $x[n]$ and $y[n]$.

2.6.5 Correlation Computation for Power and Periodic Signals

In the case of power and periodic signals, the autocorrelation and cross-correlation sequences are defined slightly differently.

For a pair of power signals, $x[n]$ and $y[n]$, the cross-correlation sequence is defined as

$$r_{xy}[\ell] = \lim_{K\to\infty} \frac{1}{2K+1} \sum_{n=-K}^{K} x[n]y[n-\ell], \tag{2.76}$$

and the autocorrelation sequence of $x[n]$ is given by

$$r_{xx}[\ell] = \lim_{K\to\infty} \frac{1}{2K+1} \sum_{n=-K}^{K} x[n]x[n-\ell]. \tag{2.77}$$

Likewise, if $\tilde{x}[n]$ and $\tilde{y}[n]$ are two periodic signals with period N, then their cross-correlation sequence is given by

$$r_{\tilde{x}\tilde{y}}[\ell] = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{x}[n]\,\tilde{y}[n-\ell], \tag{2.78}$$

and the autocorrelation sequence of $x[n]$ is given by

$$r_{\tilde{x}\tilde{x}}[\ell] = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{x}[n]\,\tilde{x}[n-\ell]. \tag{2.79}$$

It follows from the above definitions that both $r_{\tilde{x}\tilde{y}}[\ell]$ and $r_{\tilde{x}\tilde{x}}[\ell]$ are also periodic sequences with a period N.

The periodicity properties of the autocorrelation sequence can be exploited to determine the period N of a periodic signal that may have been corrupted by an additive random disturbance. Let $\tilde{x}[n]$ be a positive periodic signal corrupted by the random noise $d[n]$, resulting in the signal

$$w[n] = \tilde{x}[n] + d[n],$$

which is observed for $0 \le n \le M-1$ where $M \gg N$. The autocorrelation of $w[n]$ is given by

$$\begin{aligned} r_{ww}[\ell] &= \frac{1}{M}\sum_{n=0}^{M-1} w[n]\,w[n-\ell] \\ &= \frac{1}{M}\sum_{n=0}^{M-1} (\tilde{x}[n] + d[n])\,(\tilde{x}[n-\ell] + d[n-\ell]) \\ &= \frac{1}{M}\sum_{n=0}^{M-1} \tilde{x}[n]\,\tilde{x}[n-\ell] + \frac{1}{M}\sum_{n=0}^{M-1} d[n]\,d[n-\ell] \\ &\quad + \frac{1}{M}\sum_{n=0}^{M-1} \tilde{x}[n]\,d[n-\ell] + \frac{1}{M}\sum_{n=0}^{M-1} d[n]\,\tilde{x}[n-\ell] \\ &= r_{\tilde{x}\tilde{x}}[\ell] + r_{dd}[\ell] + r_{\tilde{x}d}[\ell] + r_{d\tilde{x}}[\ell]. \end{aligned} \tag{2.80}$$

Now in the above equation, $r_{\tilde{x}\tilde{x}}[\ell]$ is a periodic sequence with a period N, and hence, it will have peaks at $\ell = 0, N, 2N, \ldots$, with the same amplitudes as ℓ approaches M. As $\tilde{x}[n]$ and $d[n]$ are not correlated, samples of cross-correlation sequences $r_{\tilde{x}d}[\ell]$ and $r_{d\tilde{x}}[\ell]$ are likely to be very small relative to the amplitudes of $r_{xx}[\ell]$. The autocorrelation of the disturbance signal $d[n]$ shows a peak at $\ell = 0$, with other samples having rapidly decreasing amplitudes with increasing values of $|\ell|$. Hence, the peaks of $r_{ww}[\ell]$ for $\ell > 0$ are essentially due to the peaks of $r_{\tilde{x}\tilde{x}}[\ell]$ and can be used to determine whether $\tilde{x}[n]$ is a periodic sequence and its period if the peaks occur at periodic intervals.

2.6.6 Correlation Computation of a Periodic Sequence Using MATLAB

Using MATLAB we determine the period of a noise-corrupted periodic sequence in Example 2.19.

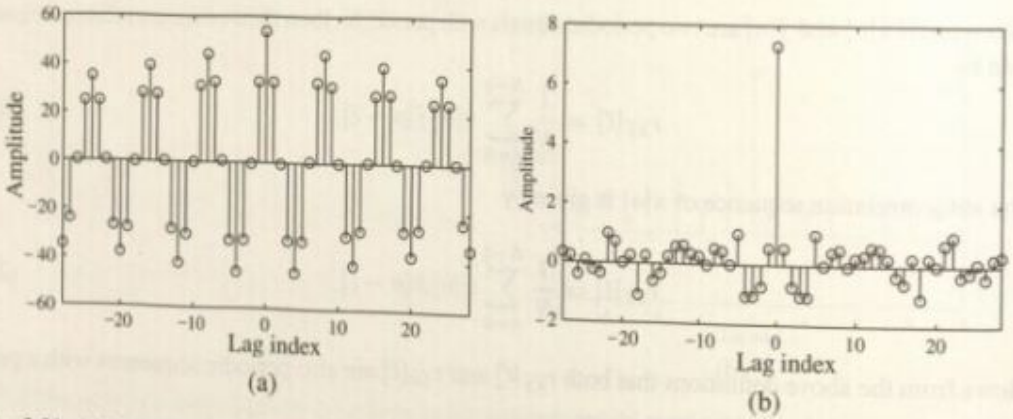


Figure 2.31: (a) Autocorrelation sequence of the noise-corrupted sinusoid and (b) autocorrelation sequence of the noise.

Program 2_6.m

EXAMPLE 2.19 Determination of the Period of a Noise-Corrupted Periodic Sequence

We determine the period of the sinusoidal sequence $x[n] = \cos(0.25\pi n), 0 \leq n \leq 95$, corrupted by an additive uniformly distributed random noise of amplitude in the range $[-0.5, 0.5]$.

To this end, we use Program 2_6 to compute the autocorrelation sequence of the noise-corrupted sinusoidal sequence. The plot generated by running this program is shown in Figure 2.31(a). As can be seen from this plot, there is a strong peak at zero lag. However, there are distinct peaks at lags that are multiples of 8, indicating the period of the sinusoidal sequence to be 8, as expected.[6] Figure 2.31(b) shows the plot of the autocorrelation sequence $r_{dd}[n]$ of the noise component. As can be seen from this plot, $r_{dd}[n]$ shows a very strong peak only at zero lag. The amplitudes are considerably smaller at other values of the lag, as sample values of the noise sequence are uncorrelated with each other.

2.7 Random Signals

Most of the book deals with the processing of signals that are deterministic in nature. The underlying assumption of this type of discrete-time signals is that they can be uniquely determined by well-defined processes such as a mathematical expression or a rule or a lookup table. Such a signal is usually called a *deterministic signal* since all sample values of the sequence are well defined for all values of the time index. For example, the sinusoidal sequence of Eq. (2.48) and the exponential sequence of Eq. (2.51) are deterministic sequences.

Signals for which each sample value is generated in a random fashion and cannot be predicted ahead of time comprise another class of signals. Such a signal, called a *random signal* or a *stochastic signal*, cannot be reproduced at will, not even using the process generating the signal, and therefore needs to be modeled using statistical information about the signal. Some common examples of random signals are models for speech, music, and seismic signals. The error signal generated by forming the difference between the ideal sampled version of a continuous-time signal and its quantized version generated by a practical analog-to-digital converter is usually modeled as a random signal for analysis purposes.[7] The

[6] The decaying amplitudes of the peaks for increasing values of the lag index ℓ are due to the finite lengths of the periodic sequences, which causes a reduction in the number of nonzero products in the computation of the convolution sum.

[7] See Section 12.5.1.

noise sequence $d[n]$ of Figure 2.6(b) generated using the `rand` function of MATLAB is also an example of a random signal.

The discrete-time *random signal* or process consists of a typically infinite collection or *ensemble* of discrete-time sequences $\{X[n]\}$. One particular sequence in this collection $\{x[n]\}$ is called a *realization* of the random process. At a given time index n, the observed sample value $x[n]$ is the value taken by the *random variable* $X[n]$. Thus, a random process is a family of random variables $\{X[n]\}$. In general, the range of sample values is a continuum. We review in Appendix C the important statistical properties of the random variable and the random process.

2.8 Summary

In this chapter, we introduced some important and fundamental concepts regarding the characterization of discrete-time signals in the time domain. Certain basic discrete-time signals that play important roles in discrete-time signal processing have been defined, along with various mathematical operations used for generating more complex signals. The relation between a continuous-time signal and the discrete-time signal generated by sampling the former at uniform time intervals has been examined. Finally, the concepts of the autocorrelation of a sequence and the cross-correlation between a pair of sequences are introduced.

Further insights can often be obtained by considering the frequency-domain representations of discrete-time signals discussed in Chapter 3. We consider the time-domain representation of discrete-time systems and the frequency-domain representation of a certain class of discrete-time systems in Chapter 4.

2.9 Problems

2.1 Determine the $\mathcal{L}_1$-, $\mathcal{L}_2$-, and $\mathcal{L}_\infty$-norms of the following finite-length sequences:

(a) $\{x_1[n]\} = \{1.21,\ -3.42,\ 10.01,\ -0.13,\ -5.11,\ -1.29,\ 3.87,\ 2.16,\ -3.21,\ 0.02\}$,

(b) $\{x_2[n]\} = \{9.81,\ 6.22,\ 1.17,\ 0.81,\ 0.12,\ -6.21,\ -12.01,\ 3.16,\ -0.14,\ 1.87\}$.

2.2 Show that $\| x \|_2 \leq \| x \|_1$.

2.3 Consider the following sequences:

$$x[n] = \{2,\ 0,\ -1,\ 6,\ -3,\ 2,\ 0\},\ -3 \leq n \leq 3,$$

$$y[n] = \{8,\ 2,\ -7,\ -3,\ 0,\ 1,\ 1\},\ -5 \leq n \leq 1,$$

$$w[n] = \{3,\ 6,\ -1,\ 2,\ 6,\ 6,\ 1\},\ -2 \leq n \leq 4.$$

The sample values of each of the above sequences outside the ranges specified are all zeros. Generate the following sequences:

(a) $c[n] = x[n+3]$, (b) $d[n] = y[n-2]$,

(c) $e[n] = x[-n]$,

(d) $u[n] = x[n-3] + y[n+3]$,

(e) $v[n] = y[n-3] \cdot w[n+2]$,

(f) $s[n] = y[n+4] - w[n-3]$, and

(g) $r[n] = 3.9w[n]$.

2.4 Figure P2.1 shows the schematic representation of four operations developed using the three basic operations: addition, multiplication, and delaying. Develop the expression for $y[n]$ for each operation as a function of $x[n]$.

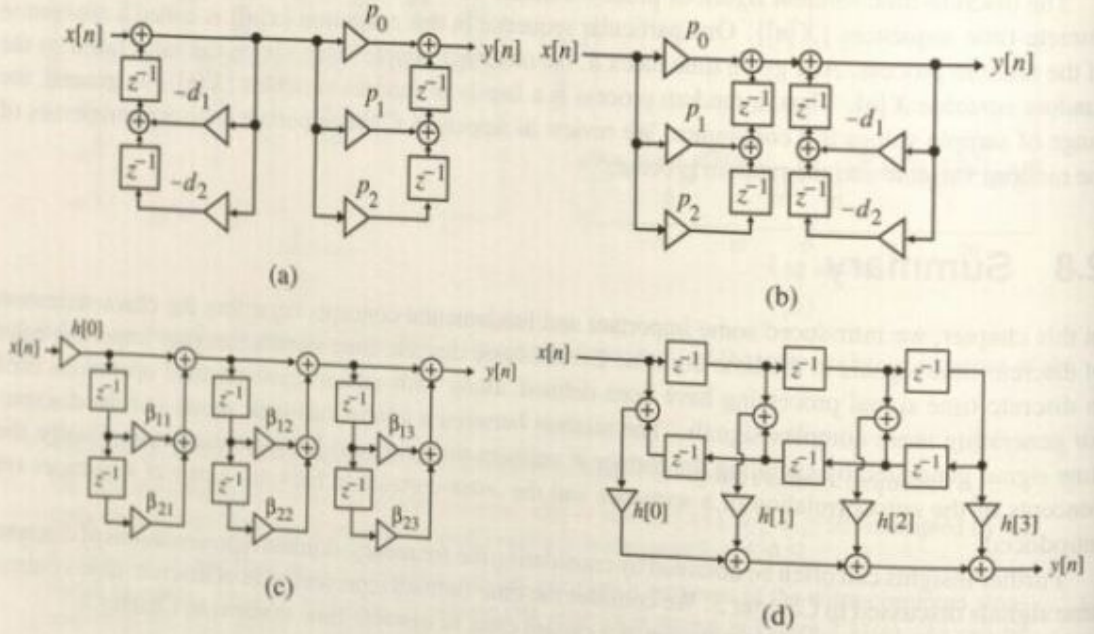


Figure P2.1

2.5 Show that the convolution of a length-M sequence with a length-N sequence leads to a sequence of length $(M + N - 1)$.

2.6 Let $x[n]$, $y[n]$, and $w[n]$ denote three finite-length sequences of lengths N, M, and L, respectively, with the first sample of each sequence occurring at $n = 0$. What is the length of the sequence $x[n] \circledast y[n] \circledast w[n]$?

2.7 Consider the following causal finite-length sequences with their first samples at $n = 0$:

(a) $\{x_1[n]\} = \{1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1\}$,

(b) $\{x_2[n]\} = \{1,\ 1\}$,

(c) $\{x_3[n]\} = \{1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1\}$,

(d) $\{x_4[n]\} = \{1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1\}$.

Show that $x_1[n] \circledast x_2[n] = x_3[n] \circledast x_4[n]$.

2.8 Evaluate the linear convolution of each of the following sequences with itself:

(a) $x_1[n] = \{1,\ -1,\ 1\}, -1 \le n \le 1$,

(b) $x_2[n] = \{1,\ -1,\ 0,\ 1,\ -1\}, 0 \le n \le 5$,

(c) $x_3[n] = \{-1,\ 2,\ 0,\ -2,\ 1\}, -3 \le n \le 1$.

2.9 Determine the following sequences obtained by a linear convolution of the sequences given in Problem 2.3:

(a) $u[n] = x[n] \circledast y[n]$, (b) $v[n] = x[n] \circledast w[n]$, (c) $g[n] = w[n] \circledast y[n]$.

2.10 Let $y[n] = x_1[n] \circledast x_2[n]$ and $v[n] = x_1[n - N_1] \circledast x_2[n - N_2]$. Express $v[n]$ in terms of $y[n]$.

2.11 Let $g[n] = x_1[n] \circledast x_2[n] \circledast x_3[n]$ and $h[n] = x_1[n - N_1] \circledast x_2[n - N_2] \circledast x_3[n - N_3]$. Express $h[n]$ in terms of $g[n]$.

2.12 Consider the following finite-length sequences:

(i) $\{h[n]\}, -M \leq n \leq N$,

(ii) $\{g[n]\}, K \leq n \leq N$,

(iii) $\{w[n]\}, -L \leq n \leq -R$,

where M, N, K, L, and R are positive integers with $K < N$ and $L > R$. Define

(a) $y_1[n] = h[n] \circledast h[n]$,

(b) $y_2[n] = g[n] \circledast g[n]$,

(c) $y_3[n] = w[n] \circledast w[n]$,

(d) $y_4[n] = h[n] \circledast g[n]$,

(e) $y_5[n] = h[n] \circledast w[n]$.

What is the length of each of the convolved sequences? What is the range of the index n for which each of the above convolved sequences is defined?

2.13 Let $y[n]$ denote the linear convolution of the two sequences $\{x[n]\} = \{2, -3, 4, 1\}$, $-1 \leq n \leq 2$, and $\{h[n]\} = \{-3, 5, -6, 4,\}$ $-2 \leq n \leq 1$. Determine the value of $y[-1]$ without computing the convolution sum.

2.14 Let $\{x[n]\} = \{n\}$ be a length-N sequence defined for $0 \leq n \leq N - 1$, and $\{h[n]\} = \{n\}$ be a length-$(N - 2)$ sequence defined for $0 \leq n \leq N - 3$, respectively. Determine the location and the value of the largest positive sample of $y[n] = x[n] \circledast h[n]$ without performing the convolution operation.

2.15 Let $y[n]$ be the sequence obtained by a linear convolution of two causal finite-length sequences $h[n]$ and $x[n]$. For each pair of $y[n]$ and $h[n]$ listed below, determine $x[n]$. The first sample in each sequence is at time index $n = 0$.

(a) $\{y[n]\} = \{2, 5, -5, 0, 12, 10, 3\}, \{h[n]\} = \{2, -3, 1, 3\}$,

(b) $\{y[n]\} = \{0, -15, -7, 9, -4, 2\}, \{h[n]\} = \{5, -1, 1\}$,

(c) $\{y[n]\} = \{6, 8, -12, -11, 14, -22, -12, 3, -18\}, \{h[n]\} = \{-3, 2, 2, 0, 3\}$.

2.16 Consider the sequence $\{x[n]\} = \{2, -5, 6, -3, 4, -4, 0, -7, 8\}, -5 \leq n \leq 3$.

(a) Let $\{y[n]\}$ denote the sequence obtained by a left circular shift of $\{x[n]\}$ by 12 sample periods. Determine the value of the sample $y[-3]$.

(b) Let $\{z[n]\}$ denote the sequence obtained by a right circular shift of $\{x[n]\}$ by 15 sample periods. Determine the value of the sample $z[2]$.

2.17 Consider the sequence $\{g[n]\} = \{-3, 0, 4, 9, 2, 0, -2, 5\}, -4 \leq n \leq 3$.

(a) Determine the sequence $\{h[n]\}$ obtained by a right circular shift of $\{g[n]\}$ by 5 sample periods.

(b) Determine the sequence $\{w[n]\}$ obtained by a left circular shift of $\{g[n]\}$ by 4 sample periods.

2.18 Show that the average power $\mathcal{P}_x$ of a real-valued sequence $x[n]$ is given by the sum of the average powers, $\mathcal{P}_{x_{ev}}$ and $\mathcal{P}_{x_{od}}$, of the even and odd parts of $x[n]$, respectively.

2.19 Compute the energy of the length-N sequence $x[n] = \sin(2\pi kn/N)$, $0 \leq n \leq N - 1$.

2.20 Determine the even and odd parts of the sequences $x[n]$, $y[n]$, and $w[n]$ of Problem 2.3.

2.21 Determine the conjugate symmetric and conjugate antisymmetric parts of the following sequences:

(a) $x_1[n] = \{-1 + j3,\ 2 - j7,\ 4 - j5,\ 3 + j5,\ -2 - j\}, -2 \le n \le 2$,

(b) $x_2[n] = e^{j2\pi n/5} + e^{j\pi n/3}$,

(c) $x_3[n] = j\cos(2\pi n/7) - \sin(2\pi n/4)$.

2.22 Let $x[n]$ and $y[n]$ be conjugate-symmetric and conjugate-antisymmetric sequences, respectively. Which one of the following sequences is a conjugate-symmetric sequence, and which one is a conjugate-antisymmetric sequence?

(a) $g[n] = x[n]x[n]$, (b) $u[n] = x[n]y[n]$, (c) $v[n] = y[n]y[n]$.

2.23 Is an absolutely summable sequence a bounded sequence? Justify your answer.

2.24 (a) Show that a causal real sequence $x[n]$ can be fully recovered from its even part $x_{ev}[n]$ for all $n \ge 0$, whereas it can be recovered from its odd part $x_{od}[n]$ only for all $n > 0$.

(b) Is it possible to fully recover a causal complex sequence $y[n]$ from its conjugate antisymmetric part $y_{ca}[n]$? Can $y[n]$ be fully recovered from its conjugate symmetric part $y_{cs}[n]$? Justify your answers.

2.25 Let $x[n]$ be an absolutely summable sequence. Show that the sequence $\tilde{y}[n]$ formed by an N-periodic extension according to Eq. (2.44) is a periodic sequence with a period N.

2.26 Verify the relation between the unit sample sequence $\delta[n]$ and the unit step sequence $\mu[n]$ given in Eq. (2.47a).

2.27 Determine the even and odd parts of the following real sequences:

(a) $x_1[n] = \mu[n - 3]$,

(b) $x_2[n] = \alpha^n \mu[n - 1]$,

(c) $x_3[n] = n\alpha^n \mu[n + 1]$,

(d) $x_4[n] = \alpha^{|n|}$.

2.28 Which ones of the following sequences are bounded sequences?

(a) $x[n] = A\alpha^{|n|}$, where A and α are complex numbers, and $|\alpha| < 1$,

(b) $h[n] = \frac{1}{2^n}\mu[n]$,

(c) $y[n] = \alpha^n \mu[n - 1]$, where $|\alpha| < 1$,

(d) $g[n] = 4ne^{j\omega_o n}\mu[n]$,

(e) $w[n] = 3\cos((\omega_o)^2 n)$,

(f) $v[n] = \left(1 - \frac{1}{n^2}\right)\mu[n - 1]$.

2.29 Show that the sequence $x[n] = \frac{(-1)^{n+1}}{n}\mu[n - 1]$ is not absolutely summable even though $\sum_{n=1}^{\infty} x[n] = \ln 2$.

2.30 Show that the following sequences are absolutely summable:

(a) $x_1[n] = \alpha^n \mu[n - 1]$,

(b) $x_2[n] = n\alpha^n \mu[n - 1]$,

(c) $x_3[n] = n^2\alpha^n \mu[n - 1]$,

where $|\alpha| < 1$.

2.31 Show that the following sequences are absolutely summable.

(a) $x_a[n] = \frac{1}{4^n}\mu[n]$, (b) $x_b[n] = \frac{1}{(n+2)(n+3)}\mu[n]$.

2.32 Show that an absolutely summable sequence has finite energy, but a finite energy sequence may not be absolutely summable.

2.33 Show that the square-summable sequence $x_1[n]$ of Eq. (2.35) is not absolutely summable.

2.34 Show that the sequence $x_2[n] = \frac{\cos \omega_c n}{\pi n}\mu[n-1]$ is square-summable but not absolutely summable.

2.35 Compute the energy of the following sequences:

(a) $x_a[n] = A\alpha^n\mu[n]$, $|\alpha| < 1$, (b) $x_b[n] = \frac{1}{n^2}\mu[n-1]$.

2.36 Determine the average power and the energy of the following sequences:

(a) $x_1[n] = (-1)^n$,

(b) $x_2[n] = \mu[n]$,

(c) $x_3[n] = n\mu[n]$,

(d) $x_4[n] = A_o e^{j\omega_o n}$,

(e) $x_5[n] = A\cos(\frac{2\pi n}{M} + \phi)$.

2.37 Determine the samples of one period of the periodic sequences obtained by an N-periodic extension of the sequences of Problem 2.3 for the following values of N: (a) $N = 6$, and (b) $N = 8$.

2.38 The following sequences represent one period of a sinusoidal sequence of the form $\tilde{x}[n] = A\sin(\omega_o n + \phi)$:

(a) $\{1, \ 1, \ -1, \ -1\}$,

(b) $\{0.5, \ -0.5, \ 0.5, \ -0.5\}$,

(c) $\{0, \ 0.5878, \ -0.9511, \ 0.9511, \ -0.5878\}$,

(d) $\{2 \ \ 0 \ \ -2 \ \ 0\}$.

Determine the values of the parameters A, ω_o, and ϕ for each case.

2.39 Determine the fundamental period of the following periodic sequences:

(a) $\tilde{x}_a[n] = e^{j0.25\pi n}$,

(b) $\tilde{x}_b[n] = \cos(0.6\pi n + 0.3\pi)$,

(c) $\tilde{x}_c[n] = \mathrm{Re}\left(e^{j\pi n/8}\right) + \mathrm{Im}\left(e^{j\pi n/5}\right)$,

(d) $\tilde{x}_d[n] = 6\sin(0.15\pi n) - \cos(0.12\pi n + 0.1\pi)$,

(e) $\tilde{x}_e[n] = \sin(0.1\pi n + 0.75\pi) - 3\cos(0.8\pi n + 0.2\pi) + \cos(1.3\pi n)$.

2.40 Determine the fundamental period of the sinusoidal sequence $\tilde{x}[n] = A\sin(\omega_o n)$ for the following values of the angular frequency ω_o:

(a) 0.3π, (b) 0.48π, (c) 0.45π, (d) 0.525π, (e) 0.75π, (f) 0.75π.

2.41 Determine the period of the sinusoidal sequence $\tilde{x}_1[n] = \sin(0.06\pi n)$. Determine at least two other distinct sinusoidal sequences having the same period as $\tilde{x}_1[n]$.

2.42 Determine the fundamental period and the average power of the following periodic sequences:

(a) $\tilde{x}_1[n] = 5\cos(\pi n/3)$,

(b) $\tilde{x}_2[n] = 2\cos(2\pi n/5)$,

(c) $\tilde{x}_3[n] = 2\cos(2\pi n/7)$,

(d) $\tilde{x}_4[n] = 3\cos(5\pi n/7)$,

(e) $\tilde{x}_5[n] = 4\cos(2\pi n/5) + 3\cos(3\pi n/5)$,

(f) $\tilde{x}_6[n] = 4\cos(5\pi n/3) + 3\cos(3\pi n/5)$.

2.43 Express the sequences:

(a) $x[n]$, $y[n]$, and $w[n]$ of Problem 2.3 as a linear combination of delayed unit sample sequences.

(b) $x[n]$, $y[n]$, and $w[n]$ of Problem 2.3 as a linear combination of delayed unit step sequences.

2.44 Express the length-4 sequence $x[n] = \{2.1, \quad -3.3, \quad -1.7, \quad 5.2\}$, $-1 \leq n \leq 2$, in terms of the unit step sequence $\mu[n]$.

2.45 Express the sequence

$$x[n] = \begin{cases} 2, & n \geq 2, \\ 1, & n < 2, \end{cases}$$

in terms of the unit step sequence $\mu[n]$.

2.46 Develop closed-form expressions for the following convolution sums:

(a) $\alpha^n \mu[n] \circledast \mu[n]$. (b) $n\alpha^n \mu[n] \circledast \mu[n]$.

2.47 Consider the following sequences:

(i) $x_1[n] = 2\delta[n-1] - 2\delta[n+1]$,

(ii) $x_2[n] = 3\delta[n-2] - \delta[n]$,

(iii) $h_1[n] = \delta[n+3] - 1.5\delta[n] - \delta[n-2]$,

(iv) $h_2[n] = -\delta[n+1] + 2\delta[n-1] + 3\delta[n-3]$.

Determine the following sequences obtained by a linear convolution of a pair of the above sequences:

(a) $y_1[n] = x_1[n] \circledast h_1[n]$,

(b) $y_2[n] = x_2[n] \circledast h_2[n]$,

(c) $y_3[n] = x_1[n] \circledast h_2[n]$,

(d) $y_4[n] = x_2[n] \circledast h_1[n]$.

2.48 Consider the following sequences:

(i) $x[n] = \mu[n] - \mu[n-N]$,

(ii) $h[n] = \mu[n] - \mu[n-M]$,

(iii) $g[n] = \mu[n-N] - \mu[n-N]$,

(iv) $w[n] = \mu[n+M] - \mu[n-N]$,

where N and M are positive integers with $N > M$.

Determine the location and the value of the largest sample of the following sequences without performing the convolution operation:

(a) $y_1[n] = x[n] \circledast x[n]$,

(b) $y_2[n] = h[n] \circledast h[n]$,

(c) $y_3[n] = g[n] \circledast g[n]$,

(d) $y_4[n] = g[n] \circledast g[n]$,

(e) $y_5[n] = x[n] \circledast h[n]$,

(f) $y_6[n] = x[n] \circledast g[n]$,

(g) $y_7[n] = x[n] \circledast w[n]$.

2.49 Consider two complex-valued sequences $h[n]$ and $g[n]$ expressed as a sum of their respective conjugate symmetric and conjugate antisymmetric parts, i.e., $h[n] = h_{cs}[n] + h_{ca}[n]$, and $g[n] = g_{cs}[n] + g_{ca}[n]$. For each of the following sequences, determine if it is conjugate symmetric or conjugate antisymmetric.

(a) $h_{cs}[n] \circledast g_{cs}[n]$, (b) $h_{ca}[n] \circledast g_{cs}[n]$, (c) $h_{ca}[n] \circledast g_{ca}[n]$.

2.50 Show that the continuous-time signal $x_a(t) = A\cos(\Omega_o t + \phi)$ can be uniquely recovered from its sampled version $x[n] = x_a(nT), -\infty < n < \infty$, if the sampling frequency $\Omega_T = 2\pi/T > 2\Omega_o$.

2.51 A continuous-time sinusoidal signal $x_a(t) = \cos \Omega_o t$ is sampled at $t = nT, -\infty < n < \infty$, generating the discrete-time sequence $x[n] = x_a(nT) = \cos(\Omega_o nT)$. For what values of T is $x[n]$ a periodic sequence? What is the fundamental period of $x[n]$ if $\Omega_o = 30$ radians and $T = \pi/6$ seconds?

2.52 (a) Evaluate the autocorrelation sequence of each of the sequences of Problem 2.3.

(b) Evaluate the cross-correlation sequence $r_{xy}[\ell]$ between the sequences $x[n]$ and $y[n]$, and the cross-correlation sequence $r_{xw}[\ell]$ between the sequences $x[n]$ and $w[n]$ of Problem 2.3.

2.53 Determine the autocorrelation sequence of each of the following sequences, and show that it is an even sequence in each case. What is the location of the maximum value of the autocorrelation sequence in each case?

(a) $x_1[n] = \alpha^n \mu[n], |\alpha| < 1$,

(b) $x_2[n] = \begin{cases} 1, & 0 \le n \le N-1, \\ 0, & \text{otherwise.} \end{cases}$

2.54 Determine the autocorrelation sequence and its period of each of the following periodic sequences.

(a) $\tilde{x}_1[n] = \cos(\pi n/M)$, where M is a positive integer,

(b) $\tilde{x}_2[n] = n$ modulo 6,

(c) $\tilde{x}_3[n] = (-1)^n$.

2.10 MATLAB Exercises

M 2.1 Write a MATLAB program to generate the conjugate-symmetric and conjugate-antisymmetric parts of a finite-length complex sequence. Using this program verify the results of Example 2.11.

M 2.2 (a) Using Program 2_2, generate the sequences shown in Figures 2.23 and 2.24.

(b) Generate and plot the complex exponential sequence $-2.7e^{(-0.4+j\pi/6)n}$ for $0 \le n \le 82$ using Program 2_2.

M 2.3 Generate the periodic sequences of Problem 2.39(b) to 2.39(e) using MATLAB.

M 2.4 (a) Write a MATLAB program to generate a sinusoidal sequence $x[n] = A\sin(\omega_o n + \phi)$, and plot the sequence using the `stem` function. The input data specified by the user are the desired length L, amplitude A, the angular frequency ω_o, and the phase ϕ where $0 < \omega_o < \pi$ and $0 \leq \phi \leq 2\pi$. Using this program, generate the sinusoidal sequences shown in Figure 2.22.

(b) Generate sinusoidal sequences with the angular frequencies given in Problem 2.40. Determine the period of each sequence from the plot, and verify the result theoretically.

M 2.5 Using MATLAB verify the result of Example 2.15.

M 2.6 Write a MATLAB program to plot a continuous-time sinusoidal signal and its sampled version, and verify Figure 2.28. You need to use the `hold` function to keep both plots.

M 2.7 Using the program developed in the previous problem, verify experimentally that the family of continuous-time sinusoids given by Eq. (2.65) lead to identical sampled signals.

M 2.8 Using Program 2_5(new), determine the autocorrelation and the cross-correlation sequences of Problem 2.52. Are your results the same as those determined in Problem 2.52?

M 2.9 Modify Program 2_5(new) to determine the autocorrelation sequence of a sequence corrupted with a uniformly distributed random signal generated using the M-function `rand`. Using the modified program, demonstrate that the autocorrelation sequence of a noise-corrupted signal exhibits a peak at zero lag.

M 2.10 Determine and plot using MATLAB the autocorrelation sequence of the causal exponentially decaying sequence $x[n] = \alpha^n \mu[n]$, for the following values of α: (a) $\alpha = 0.6$, and (b) $\alpha = 0.8$.

Chapter 3

Discrete-Time Signals in the Frequency Domain

In Section 2.4.3, we pointed out that any arbitrary sequence can be represented in the time domain as a weighted linear combination of delayed unit sample sequences $\{\delta[n-k]\}$. An important consequence of this representation, derived in Section 4.4.1, is the input–output characterization of a certain class of discrete-time systems in the time domain. In many applications, it is convenient to consider an alternate description of a sequence in terms of complex exponential sequences of the form $\{e^{j\omega n}\}$, where ω is the normalized frequency variable in radians. This leads to a particularly useful representation of discrete-time sequences and certain discrete-time systems in the frequency domain.[1]

The frequency-domain representation of a discrete-time sequence discussed in this chapter is the discrete-time Fourier transform by which a time-domain sequence is mapped into a continuous function of the frequency variable ω. Because of the periodicity of the discrete-time Fourier transform, the corresponding discrete-time sequence can be simply obtained by computing its Fourier series representation. Since the representation is in terms of an infinite series, existence of the discrete-time Fourier transform is examined along with its properties.

We next derive the conditions under which a continuous-time signal can be represented uniquely by a discrete-time signal, and also show how the original continuous-time signal can be recovered from its discrete-time equivalent. The conversion of a continuous-time signal into an equivalent discrete-time signal is performed by a sample-and-hold circuit followed by an analog-to-digital converter. The effect of a practical sample-and-hold circuit on the generation of the discrete-time equivalent is discussed.

In this chapter also, we make extensive use of MATLAB to illustrate through computer simulations the various concepts introduced.

3.1 The Continuous-Time Fourier Transform

We begin with a brief review of the continuous-time Fourier transform, a frequency-domain representation of a continuous-time signal, and its properties, as it will provide a better understanding of the frequency-domain representation of the discrete-time signals and systems, in addition to pointing out the major differences between these two transforms.

[1] Periodic sequences can be represented in the frequency domain by means of the discrete Fourier series (see Problem 5.3).