Chapter

7

Fundamentals of Digital Signal Processing

Digital signal processing (DSP) is based on the fact that an analog signal can be digitized and input to a general-purpose digital computer or special-purpose digital processor. Once this is accomplished, we are free to perform all sorts of mathematical operations on the sequence of digital data samples inside the processor. Some of these operations are simply digital versions of classical analog techniques, while others have no counterpart in analog circuit devices or processing methods. This chapter covers digitization and introduces the various types of processing that can be performed on the sequence of digital values once they are inside the processor.

7.1 Digitization

Digitization is the process of converting an analog signal such as a time-varying voltage or current into a sequence of digital values. Digitization actually involves two distinct parts—*sampling* and *quantization*—which are usually analyzed separately for the sake of convenience and simplicity. Three basic types of sampling, shown in Fig. 7.1, are *ideal*, *instantaneous*, and *natural*. From the illustration we can see that the sampling process converts a signal that is defined over a continuous time interval into a signal that has nonzero amplitude values only at discrete instants of time (as in ideal sampling) or over a number of discretely separate but internally continuous subintervals of time (as in instantaneous and natural sampling). The signal that results from a sampling process is called a *sampled-data signal*. The signals resulting from ideal sampling are also referred to as *discrete-time signals*.

Each of the three basic sampling types occurs at different places within a DSP system. The output from a sample-and-hold amplifier or a digital-to-analog converter (DAC) is an instantaneously sampled signal. In the output

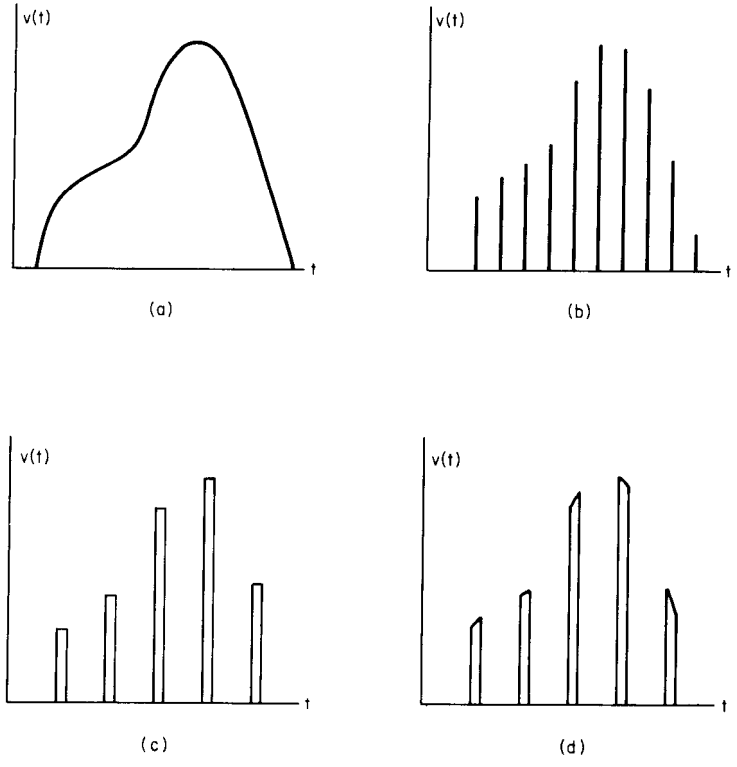

Figure 7.1 An analog signal (*a*) and three different types of sampling: (*b*) ideal, (*c*) instantaneous, and (*d*) natural.

of a practical analog-to-digital converter (ADC) used to sample a signal, each sample will of course exist for some nonzero interval of time. However, within the software of the digital processor, these values can still be interpreted as the amplitudes for a sequence of ideal samples. In fact, this is almost always the best approach since the ideal sampling model results in the simplest processing for most applications. Natural sampling is encountered in the analysis of the analog multiplexing that is often performed prior to A/D conversion in multiple-signal systems. In all three of the sampling approaches presented, the sample values are free to assume any appropriate value from the continuum of possible analog signal values.

Quantization is the part of digitization that is concerned with converting the amplitudes of an analog signal into values that can be represented by binary numbers having some finite number of bits. A quantized, or *discrete-valued*, signal is shown in Fig. 7.2. The sampling and quantization processes will introduce some significant changes in the spectrum of a digitized signal. The details of the changes will depend upon both the precision of the quantization operation and the particular sampling model that most aptly fits the actual situation.

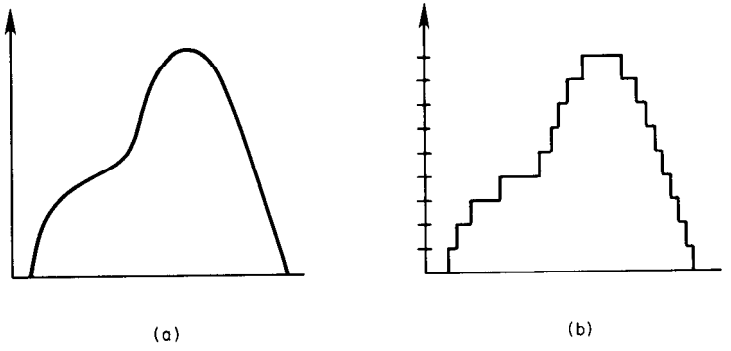

Figure 7.2 An analog signal (*a*) and the corresponding quantized signal (*b*).

Ideal sampling

In *ideal sampling*, the sampled-data signal, as shown in Fig. 7.3, comprises a sequence of uniformly spaced impulses, with the weight of each impulse equal to the amplitude of the analog signal at the corresponding instant in time. Although not mathematically rigorous, it is convenient to think of the sampled-data signal as the result of multiplying the analog signal $x(t)$ by a periodic train of unit impulses:

$$x_s(\cdot) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

Based upon property 11 from Table 1.5, this means that the spectrum of the sampled-data signal could be obtained by convolving the spectrum of the analog signal with the spectrum of the impulse train:

$$\mathcal{F}\left[x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)\right] = X(f) * \left[f_s \sum_{m=-\infty}^{\infty} \delta(f - mf_s)\right]$$

As illustrated in Fig. 7.4, this convolution produces copies, or *images*, of the original spectrum that are periodically repeated along the frequency axis. Each of the images is an exact (to within a scaling factor) copy of the

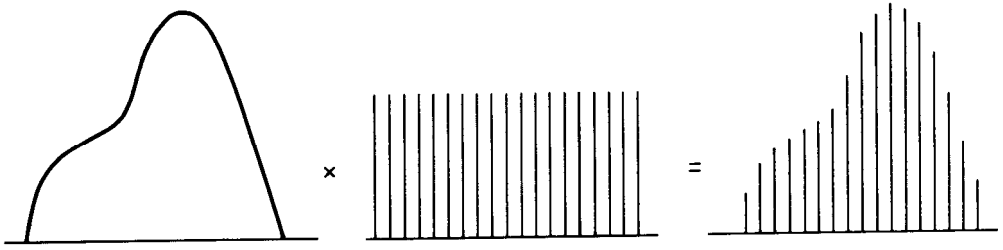

Figure 7.3 Ideal sampling.

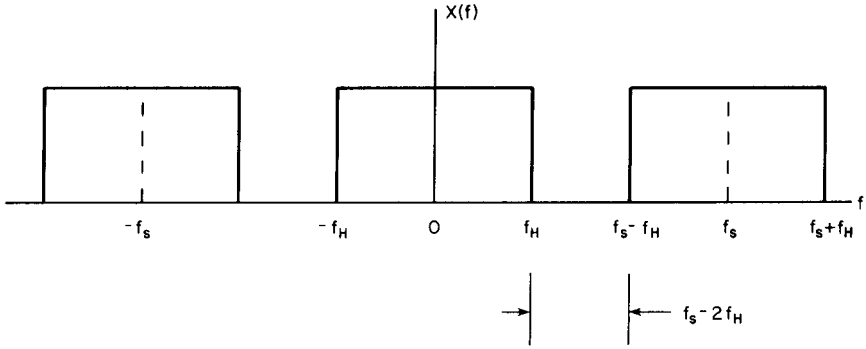

Figure 7.4 Spectrum of an ideally sampled signal.

original spectrum. The center-to-center spacing of the images is equal to the sampling rate f_s, and the edge-to-edge spacing is equal to $f_s - 2f_H$. As long as f_s is greater than 2 times f_H, the original signal can be recovered by a lowpass filtering operation that removes the extra images introduced by the sampling.

Sampling rate selection

If f_s is less than $2f_H$, the images will overlap, or *alias*, as shown in Fig. 7.5, and recovery of the original signal will not be possible. The minimum alias-free sampling rate of $2f_H$ is called the *Nyquist rate*. A signal sampled exactly at its Nyquist rate is said to be *critically sampled*.

Uniform sampling theorem. If the spectrum $X(f)$ of a function $x(t)$ vanishes beyond an upper frequency of f_H Hz or ω_H rad/s, then $x(t)$ can be completely determined by its values at uniform intervals of less than $1/(2f_H)$ or π/ω. If sampled within these constraints, the original function $x(t)$ can be reconstructed from the samples by

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \frac{\sin[2f_s(t - nT)]}{2f_s(t - nT)}$$

where T is the sampling interval.

Since practical signals cannot be strictly band-limited, sampling of a real-world signal must be performed at a rate greater than $2f_H$ where the signal is known to have negligible (that is, typically less than 1 percent) spectral energy above the frequency of f_H. When designing a signal processing system, we will rarely, if ever, have reliable information concerning the exact spectral occupancy of the noisy real-world signals that our system will eventually face. Consequently, in most practical design situations, a value is selected for f_H based upon the requirements of the particular application, and

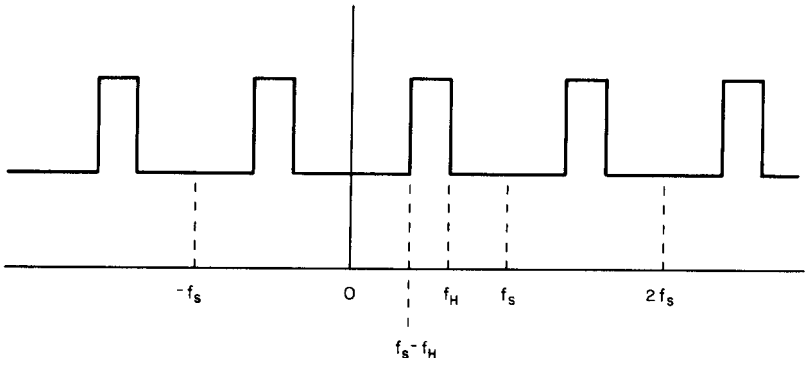

Figure 7.5 Aliasing due to overlap of spectral images.

then the signal is lowpass-filtered prior to sampling. Filters used for this purpose are called *antialiasing filters* or *guard filters*. The sample-rate selection and guard filter design are coordinated so that the filter provides attenuation of 40 dB or more for all frequencies above $f_s/2$. The spectrum of an ideally sampled practical signal is shown in Fig. 7.6. Although some aliasing does occur, the aliased components are suppressed at least 40 dB below the desired components. Antialias filtering must be performed prior to sampling. In general, there is no way to eliminate aliasing once a signal has been improperly sampled. The particular type (Butterworth, Chebyshev, Bessel, Cauer, and so on) and order of the filter should be chosen to provide the necessary stop-band attenuation while preserving the pass-band characteristics most important to the intended application.

Instantaneous sampling

In instantaneous sampling, each sample has a nonzero width and a flat top. As shown in Fig. 7.7, the sampled-data signal resulting from instantaneous sampling can be viewed as the result of convolving a sample pulse $p(t)$ with an ideally sampled version of the analog signal. The resulting sampled-data signal can thus be expressed as

$$x_s(\cdot) = p(t) * \left[x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \right]$$

where $p(t)$ is a single rectangular sampling pulse and $x(t)$ is the original analog signal. Based upon property 10 from Table 1.5, this means that the spectrum of the instantaneous sampled-data signal can be obtained by multiplying the spectrum of the sample pulse with the spectrum of the ideally sampled signal:

$$\mathcal{F}\left\{ p(t) * \left[x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \right] \right\} = P(f) \cdot \left\{ X(f) * \left[f_s \sum_{m=-\infty}^{\infty} \delta(f - mf_s) \right] \right\}$$

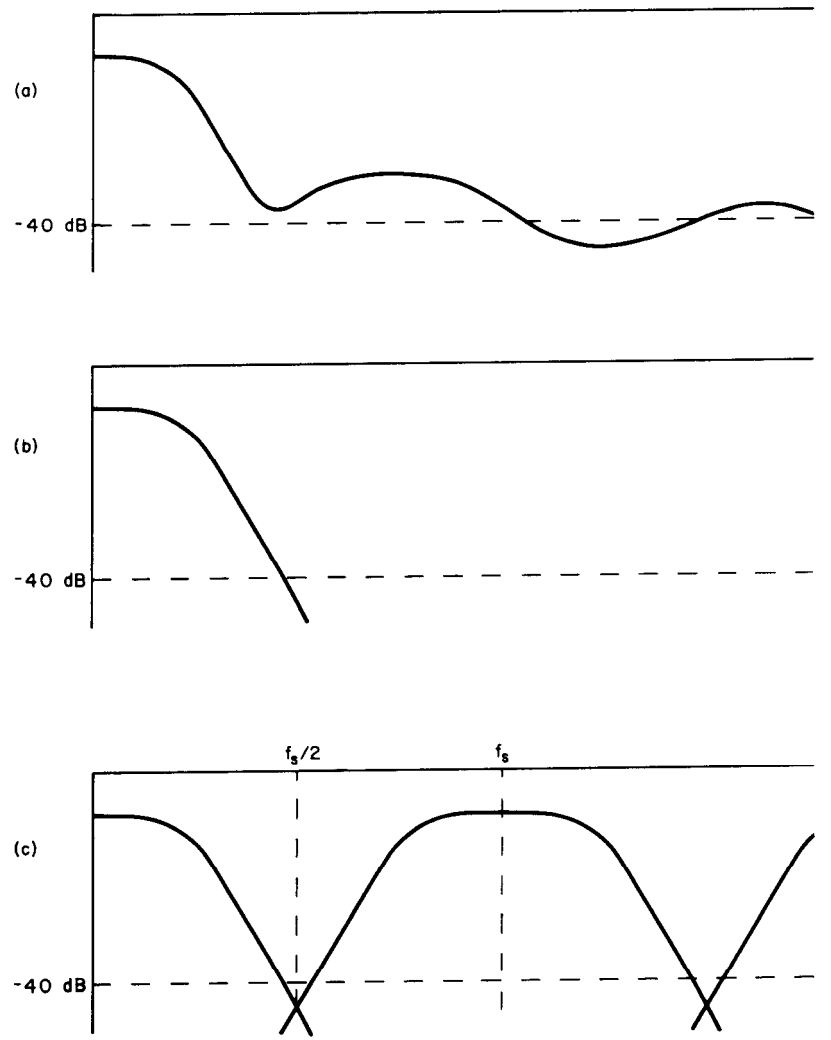

Figure 7.6 Spectrum of an ideally sampled practical signal: (*a*) spectrum of raw analog signal, (*b*) spectrum after lowpass filtering, and (*c*) spectrum after sampling.

As shown in Fig. 7.8, the resulting spectrum is similar to the spectrum produced by ideal sampling. The only difference is the amplitude distortion introduced by the spectrum of the sampling pulse. This distortion is sometimes called the *aperture effect*. Notice that distortion is present in all the images, including the one at base-band. The distortion will be less severe for narrow sampling pulses. As the pulses become extremely narrow, instantaneous sampling begins to look just like ideal sampling, and distortion due to the aperture effect all but disappears.

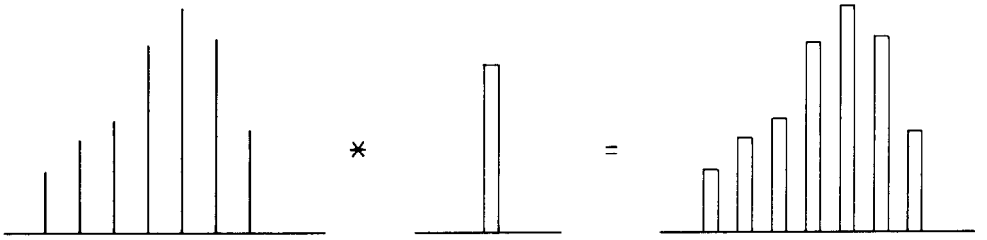

Figure 7.7 Instantaneous sampling.

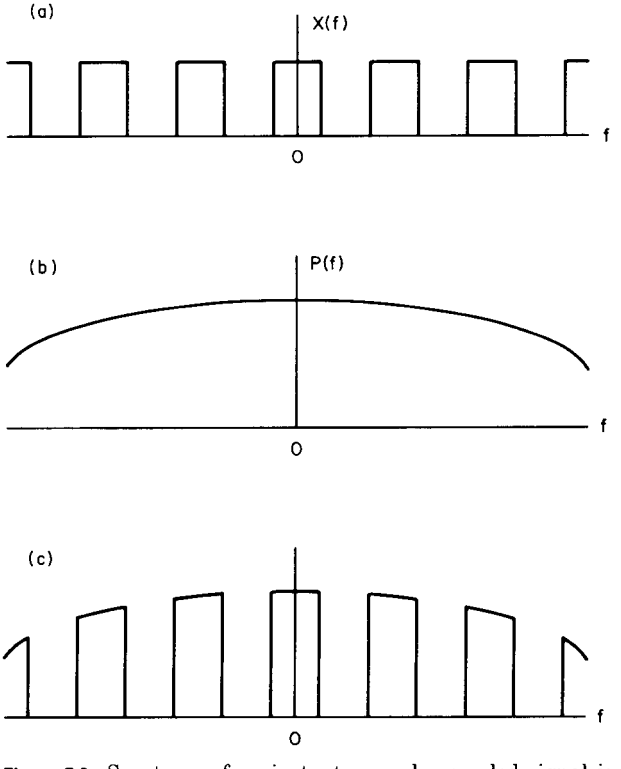

Figure 7.8 Spectrum of an instantaneously sampled signal is equal to the spectrum (*a*) of an ideally sampled signal multiplied by the spectrum (*b*) of 1 sampling pulse.

Natural sampling

In natural sampling, each sample's amplitude follows the analog signal's amplitude throughout the sample's duration. As shown in Fig. 7.9, this is mathematically equivalent to multiplying the analog signal by a periodic train of rectangular pulses:

$$x_s(\cdot) = x(t) \cdot \left\{ p(t) * \left[\sum_{n=-\infty}^{\infty} \delta(t - nT) \right] \right\}$$

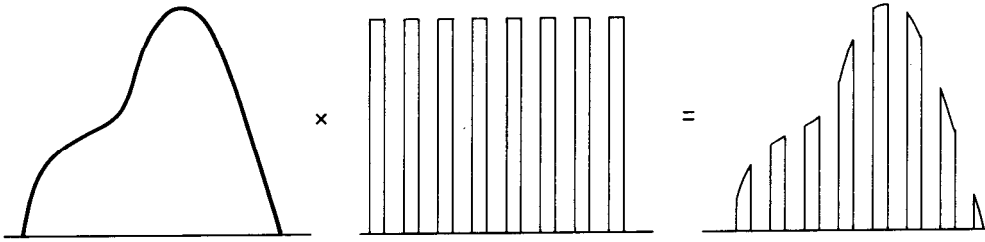

Figure 7.9 Natural sampling.

The spectrum of a naturally sampled signal is found by convolving the spectrum of the analog signal with the spectrum of the sampling pulse train:

$$\mathcal{F}[x_s(\cdot)] = X(f) * \left[P(f) f_s \sum_{m=-\infty}^{\infty} \delta(f - mf_s) \right]$$

As shown in Fig. 7.10, the resulting spectrum will be similar to the spectrum produced by instantaneous sampling. In instantaneous sampling, all frequen-

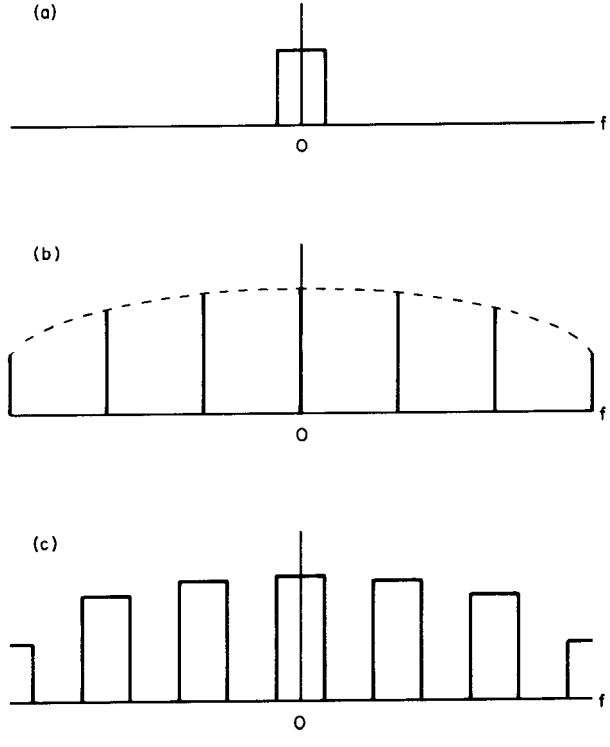

Figure 7.10 Spectrum (*c*) of a naturally sampled signal is equal to the spectum (*a*) of the analog signal multiplied by the spectrum (*b*) of the sampling pulse train.

cies of the sampled signal's spectrum are attenuated by the spectrum of the sampling pulse, while in natural sampling each image of the basic spectrum will be attenuated by a factor that is equal to the value of the sampling pulse's spectrum at the center frequency of the image. In communications theory, natural sampling is called *shaped-top pulse amplitude modulation.*

Discrete-time signals

In the discussion so far, weighted impulses have been used to represent individual sample values in a discrete-time signal. This was necessary in order to use continuous mathematics to connect continuous-time analog signal representations with their corresponding discrete-time digital representations. However, once we are operating strictly within the digital or discrete-time realms, we can dispense with the Dirac delta impulse and adopt in its place the unit sample function, which is much easier to work with. The unit sample function is also referred to as a *Kronecker delta impulse* (Cadzow 1973). Figure 7.11 shows both the Dirac delta and Kronecker delta representations for a typical signal. In the function sampled using a Dirac impulse train, the independent variable is continuous time t, and integer multiples of the sampling interval T are used to explicitly define the discrete sampling instants. On the other hand, the Kronecker delta notation assumes uniform

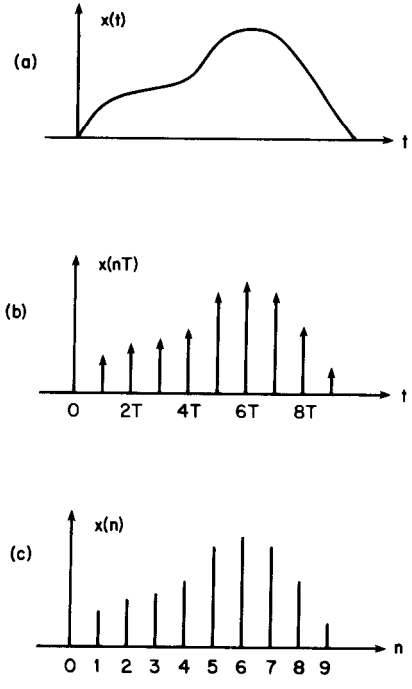

Figure 7.11 Sampling with Dirac and Kronecker impulses: (*a*) continuous signal, (*b*) sampling with Dirac impulses, and (*c*) sampling with Kronecker impulses.

sampling with an implicitly defined sampling interval. The independent variable is the integer-valued index n whose values correspond to the discrete instants at which samples can occur. In most theoretical work, the implicitly defined sampling interval is dispensed with completely by treating all the discrete-time functions as though they have been normalized by setting $T = 1$.

Notation

Writers in the field of digital-signal processing are faced with the problem of finding a convenient notational way to distinguish between continuous-time functions and discrete-time functions. Since the early 1970s, a number of different approaches have appeared in the literature, but none of the schemes advanced so far have been perfectly suited for all situations. In fact, some authors use two or more different notational schemes within different parts of the same book. In keeping with long-established mathematical practice, functions of a continuous variable are almost universally denoted with the independent variable enclosed in parentheses: $x(t)$, $H(e^{j\omega})$, $\phi(f)$ and so on. Many authors, such as Oppenheim and Schafer (1975), Rabiner and Gold (1975), and Roberts and Mullis (1987), make no real notational distinction between functions of continuous variables and functions of discrete variables, and instead rely on context to convey the distinction. This approach, while easy for the writer, can be very confusing for the reader. Another approach involves using subscripts for functions of a discrete variable:

$$x_k \triangleq x(kT)$$

$$H_n \triangleq H(e^{jn\theta})$$

$$\phi_m \triangleq \phi(mF)$$

This approach quickly becomes typographically unwieldy when the independent variable is represented by a complicated expression. A fairly recent practice (Oppenheim and Schafer 1989) uses parentheses ( ) to enclose the independent variable of continuous-variable functions and brackets [ ] to enclose the independent variable of discrete-variable functions:

$$x[k] = x(kT)$$

$$H[n] = H(e^{jn\theta})$$

$$\phi[m] = \phi(mF)$$

For the remainder of this book, we will adopt this practice and just remind ourselves to be careful in situations where the bracket notation for discrete-variable functions could be confused with the bracket notation used for arrays in the C language.

7.2 Discrete-Time Fourier Transform

The Fourier series given by Eq. (1.140) can be rewritten to make use of the discrete sequence notation that was introduced in Sec. 7.1:

$$x(t) = \sum_{n=-\infty}^{\infty} X[n]\, e^{j2\pi nFt}$$

where $F = \dfrac{1}{t_0}$ = sample spacing in the frequency domain

t_0 = period of $x(t)$

Likewise, Eq. (1.141) can be written as

$$X[n] = \frac{1}{t_0} \int_{t_0} x(t)\, e^{-jn2\pi Ft}\, dt$$

The fact that the signal $x(t)$ and sequence $F[n]$ form a Fourier series pair with a frequency domain sampling interval of F can be indicated as

$$x(t) \overset{\text{FS; } F}{\longleftrightarrow} X[n]$$

Discrete-time Fourier transform

In Sec. 7.1 the results concerning the impact of sampling upon a signal's spectrum were obtained using the *continuous-time* Fourier transform in conjunction with a periodic train of Dirac impulses to model the sampling of the continuous-time signal $x(t)$. Once we have defined a discrete-time sequence $x[n]$, the *discrete-time Fourier transform* (DTFT) can be used to obtain the corresponding spectrum directly from the sequence without having to resort to impulses and continuous-time Fourier analysis.

The discrete-time Fourier transform, which links the discrete-time and continuous-frequency domain, is defined by

$$X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega nT} \tag{7.1}$$

and the corresponding inverse is given by

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega nT}\, d\omega \tag{7.2}$$

If Eqs. (7.1) and (7.2) are compared to the DTFT definitions given by certain texts (Oppenheim and Schafer 1975; Oppenheim and Schafer 1989; Rabiner and Gold 1975), an apparent disagreement will be found. The cited texts

define the DTFT and its inverse as

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n} \tag{7.3}$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega nT}\, d\omega \tag{7.4}$$

The disagreement is due to the notation used by these texts, in which ω is used to denote the *digital* frequency given by

$$\omega = \frac{\Omega}{F_s} = \Omega T$$

where Ω = analog frequency
F_s = sampling frequency
T = sampling interval

In most DSP books other than the three cited above, the analog frequency is denoted by ω rather than by Ω. Whether ω or Ω is the "natural" choice for denoting analog frequency depends upon the overall approach taken in developing Fourier analysis of sequences. Books that begin with sequences, and then proceed on to Fourier analysis of sequences, and finally tie sequences to analog signals via sampling tend to use ω for the first frequency variable encountered which is *digital* frequency. Other books that begin with analog theory and then move on to sampling and sequences, tend to use ω for the first frequency variable encountered which is *analog* frequency. In this book, we will adopt the convention used by Peled and Liu (1976) denoting analog frequency by ω and digital frequency by $\lambda = \omega T$. The function $X(e^{j\omega T})$ is periodic with a period of $\omega_p = 2\pi/T$, and $X(e^{j\lambda})$ is periodic with a period of $\lambda_p = 2\pi$.

Independent of the ω versus Ω controversy, the notation $X(e^{j\omega T})$ or $X(e^{j\lambda})$ is commonly used rather than $X(\omega)$ or $X(\lambda)$ so that the form of (7.1) remains similar to the form of the z transform given in Sec. 5.1 which is

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]\, z^{-n} \tag{7.5}$$

If $e^{j\omega}$ is substituted for z in (7.5), the result is identical to (7.1). This indicates that the DTFT is nothing more than the z transform evaluated on the unit circle. [Note: $e^{j\omega} = \cos\omega + j \sin\omega$, $0 \le \omega \le 2\pi$, does in fact define the unit circle in the z plane since $|e^{j\omega}| = (\cos^2\omega + \sin^2\omega)^{1/2} \equiv 1$].

Convergence conditions

If the time sequence $x[n]$ satisfies

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

then $X(e^{j\omega T})$ exists and the series in (7.1) converges uniformly to $X(e^{j\omega T})$. If $x[n]$ satisfies

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

then $X(e^{j\omega T})$ exists and the series in (7.1) converges in a mean-square sense to $X(e^{j\omega T})$, that is,

$$\lim_{M \to \infty} \int_{-\pi}^{\pi} |X(e^{j\omega T}) - X_M(e^{j\omega T})|^2 \, d\omega = 0$$

where $X_M(e^{j\omega T}) = \sum_{n=-M}^{M} x[n] \, e^{-j\omega nT}$

The function $X_M(e^{j\omega T})$ is a form of the Dirichlet kernel discussed in Sec. 11.2.

Relationship to Fourier series

Since the Fourier series represents a periodic continuous-time function in terms of a discrete-frequency function, and the DTFT represents a discrete-time function in terms of a periodic continuous-frequency function, we might suspect that some sort of duality exists between the Fourier series and DTFT. It turns out that such a duality does indeed exist. Specifically if

$$f[k] \overset{\text{DTFT}}{\longleftrightarrow} F(e^{j\omega T})$$

and we set

$$\omega_0 = T$$

$$x(t) = F(e^{j\omega T})\big|_{\omega = T}$$

$$X[n] = f[k]\big|_{k=-n}$$

then

$$x(t) \overset{\text{FS; } \omega_0}{\longleftrightarrow} X[n]$$

7.3 Discrete-Time Systems

In Chap. 2 we saw how continuous-time systems such as filters and amplifiers can accept analog input signals and operate upon them to produce different analog output signals. *Discrete-time systems* perform essentially the same role for digital or discrete-time signals.

Difference equations

Although I have deliberately avoided discussing differential equations and their accompanying headaches in the analysis of analog systems, *difference*

equations are much easier to work with, and they play an important role in the analysis and synthesis of discrete-time systems. A discrete-time, linear, time-invariant (or if you prefer, shift-invariant) (DTLTI or DTLSI) system, which accepts an input sequence $x[n]$ and produces an output sequence $y[n]$, can be described by a linear difference equation of the form

$$y[n] + a_1 y[n-1] + a_2 y[n-2] + \cdots + a_k y[n-k] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots + b_k x[n-k] \tag{7.6}$$

Such a difference equation can describe a DTLTI system having any initial conditions as long as they are specified. This is in contrast to the discrete-convolution and discrete-transfer function that are limited to describing digital filters that are initially relaxed (that is, all inputs and outputs are initially zero). In general, the computation of the output $y[n]$ at point n using Eq. (7.6) will involve previous outputs $y[n-1]$, $y[n-2]$, $y[n-3]$, and so on. However, in some filters, all of the coefficients $a_1, a_2, \ldots, a_k$ are equal to zero, thus yielding

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots + b_k x[n-k] \tag{7.7}$$

in which the computation of $y[n]$ does not involve previous output values. Difference equations involving previous output values are called *recursive difference equations*, and equations in the form of (7.7) are called *nonrecursive difference equations.*

Example 7.1 Determine a nonrecursive difference equation for a simple moving-average lowpass filter in which the output at $n = i$ is equal to the arithmetic average of the five inputs from $n = i - 4$ through $n = i$.

solution The desired difference equation is given by

$$y[n] = \frac{x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4]}{5}$$

$$= 0.2x[n] + 0.2x[n-1] + 0.2x[n-2] + 0.2x[n-3] + 0.2x[n-4] \tag{7.8}$$

Relating this to the standard form of Eq. (7.7), we find $k = 4$, $b_i = 0$ for all i, and $a_0 = a_1 = a_2 = a_3 = a_4 = 0.2$.

Discrete convolution

A discrete-time system's impulse response is the output response produced when a unit sample function is applied to the input of the previously relaxed system. As we might expect from our experiences with continuous systems, we can obtain the output $y[n]$ due to any input by performing a *discrete convolution* of the input signal $x[n]$ and the impulse response $h[n]$. This discrete convolution is given by

$$y[n] = \sum_{m=0}^{\infty} h[m]\, x[n-m]$$

If the impulse response has nonzero values at an infinite number of points along the discrete-time axis, a digital filter having such an impulse response is called an *infinite-impulse response* (IIR) filter. On the other hand, if $h[m] = 0$ for all $m \geq M$, the filter is called a *finite-impulse response* (FIR) filter, and the convolution summation can be rewritten as

$$y[n] = \sum_{m=0}^{M-1} h[m]\, x[n-m]$$

FIR filters are also called *transversal filters.*

Example 7.2 For the moving-average filter described in Example 7.1, obtain the filter's impulse response.

solution The filter's impulse response $h[n]$ can be obtained by direct evaluation of Eq. (7.8) for the case of $x[n]$ equal to the unit sample function:

$$h[n] = y[n] \qquad \text{for} \qquad x[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

Thus,

$$h[n] = \begin{cases} 0.2 & 0 \leq n \leq 4 \\ 0 & \text{otherwise} \end{cases}$$

The following summation identities will often prove useful in the evaluation of convolution summations:

$$\sum_{n=0}^{N} \alpha^n = \frac{1-\alpha^{N+1}}{1-\alpha} \qquad \alpha \neq 1 \tag{7.9}$$

$$\sum_{n=0}^{N} n\alpha^n = \frac{\alpha}{(1-\alpha)^2}\left(1-\alpha^N - N\alpha^N + N\alpha^{N+1}\right) \qquad \alpha \neq 1 \tag{7.10}$$

$$\sum_{n=0}^{N} n^2\alpha^n = \frac{\alpha}{(1-\alpha)^3}\left[(1+\alpha)(1-\alpha^N) - 2(1-\alpha)N\alpha^N - (1-\alpha)^2 N^2 \alpha^N\right] \qquad \alpha \neq 1 \tag{7.11}$$

7.4 Diagramming Discrete-Time Systems

Block diagrams

As is the case for continuous-time systems, block diagrams are useful in the design and analysis of discrete-time systems. Construction of block diagrams for discrete time systems involves three basic building blocks: the unit-delay element, multiplier, and summer.

Unit-delay element. As its name implies, a *unit-delay element* generates an output that is identical to its input delayed by 1 sample interval:

$$y[k] = x[k-1]$$

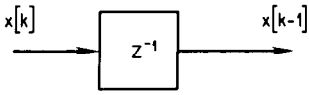

Figure 7.12 Block diagram representation of a unit-delay element.

The unit-delay element is usually drawn as shown in Fig. 7.12. The term z^{-1} is used to denote a unit delay because delaying a discrete-time signal by 1 sample time multiplies the signal's z transform by z^{-1}. (See property 5 in Table 9.4.) Delays of p sample times may be depicted as p unit delays in series or as a box enclosing z^{-p}.

Multiplier. A *multiplier* generates as output the product of a fixed constant and the input signal

$$y[k] = a\, x[k]$$

A multiplier can be drawn in any of the ways shown in Fig. 7.13. The form shown in Fig. 7.13*c* is usually reserved for adaptive filters and other situations where the factor a is not constant. [Note that a system containing multiplication by a nonconstant factor would not be a *linear time-invariant* (LTI) system!]

Summer. A *summer* adds two or more discrete-time signals to generate the discrete-time output signal:

$$y[k] = x_1[k] + x_2[k] + \cdots + x_n[k]$$

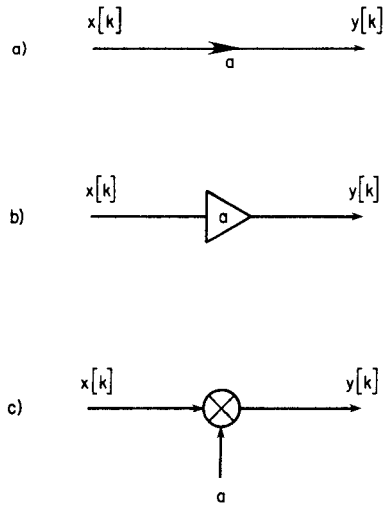

Figure 7.13 Block diagram representations of a multiplier.

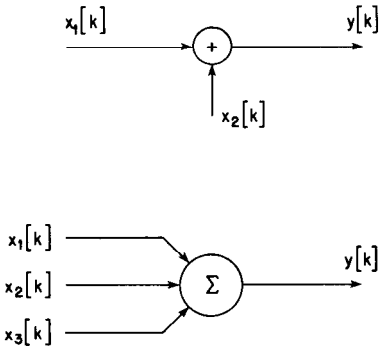

Figure 7.14 Block diagram representations of a summer.

A summer is depicted using one of the forms shown in Fig. 7.14. A negative sign can be placed next to a summer's input paths as required to indicate a signal that is to be subtracted rather than added.

Example 7.3 Draw a block diagram for a simple moving-average lowpass filter in which the output at $k = i$ is equal to the arithmetic average of the three inputs for $k = i - 2$ through $k = i$.

solution The difference equation for the desired filter is

$$y[k] = \tfrac{1}{3}x[k] + \tfrac{1}{3}x[k-1] + \tfrac{1}{3}x[k-2]$$

The block diagram for this filter will be as shown in Fig. 7.15. It should be noted that block diagram representations are in general not unique and that a given system can be represented in several different ways.

Example 7.4 Draw alternative block diagrams for the filter of Example 7.3.

solution Since multiplication distributes over addition, the difference equation can be rewritten as

$$y[k] = \tfrac{1}{3}\{x[k] + x[k-1] + x[k-2]\}$$

and the block diagram can be redrawn as shown in Fig. 7.16.

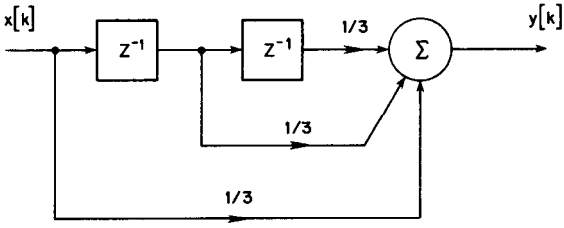

Figure 7.15 Block diagram for Example 7.3.

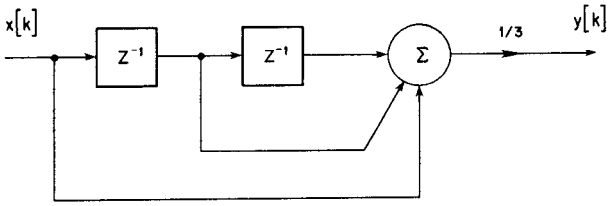

Figure 7.16 Block diagram for Example 7.4.

Signal flow graphs

A modified form of a directed graph, called a *signal flow graph* (SFT), can be used to depict all the same information as a block diagram but in a more compact form. Consider the block diagram in Fig. 7.17 which has some labeled points added for ease of reference. The *oriented graph*, or *directed graph*, for this system is obtained by replacing each multiplier, each connecting branch, and each delay element with a directed line segment called an *edge*. Furthermore, each branching point and each adder is replaced by a point called a *node*. The resulting graph is shown in Fig. 7.18. A signal flow graph is obtained by associating a signal with each node and a linear operation with each edge of the directed graph. The node weights correspond to signals present within the discrete-time system. Associated with each edge is the linear operation (delay or constant gain) that must be performed upon the signal associated with the edge's *from* node in order to obtain the signal associated with the edge's *to* node. For a node which is the *to* node for two or more edges, the signal associated with the node is the sum of all the signals produced by the incoming edges. For the graph shown in Fig. 7.18, the

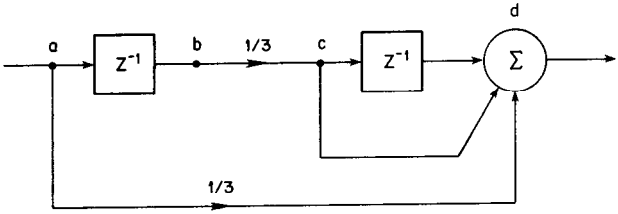

Figure 7.17 Block diagram of a discrete-time system.

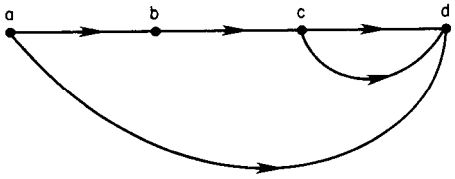

Figure 7.18 Directed graph corresponding to the block diagram of Fig. 7.17.

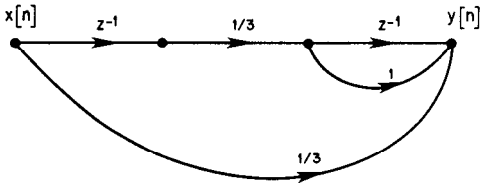

Figure 7.19 Signal flow graph derived from the directed graph of Fig. 7.18.

following correpondences can be identified:

Node a: $x[k]$

Node b: $x[k-1]$

Node c: $\frac{1}{3}x[k-1]$

Node d: summer producing $y[k]$

Edge (a, b): first delay element

Edge (c, d): second delay element

Edge (a, d): bottom multiplier

Edge (b, c): top multiplier

Edge (c, d): unity gain connection from point c to summer

The resulting signal flow graph is shown in Fig. 7.19. It is customary to use multiplication by z^{-1} as a shorthand notation for unit delay, even though the signals in an SFG are time domain signals, and multiplication by z^{-1} is a frequency domain operation.