Discrete-Time Signals and Systems

The sinusoid is an important elementary signal that serves as a basic building block in more complex signals. However, there are other elementary signals that are important in our treatment of signal processing. These discrete-time signals are introduced in this chapter and are used as basis functions or building blocks to describe more complex signals.

The major emphasis in this chapter is the characterization of discrete-time systems in general and the class of linear time-invariant (LTI) systems in particular. A number of important time-domain properties of LTI systems are defined and developed, and an important formula, called the convolution formula, is derived which allows us to determine the output of an LTI system to any given arbitrary input signal. In addition to the convolution formula, difference equations are introduced as an alternative method for describing the input–output relationship of an LTI system, and in addition, recursive and nonrecursive realizations of LTI systems are treated.

Our motivation for the emphasis on the study of LTI systems is twofold. First, there is a large collection of mathematical techniques that can be applied to the analysis of LTI systems. Second, many practical systems are either LTI systems or can be approximated by LTI systems. Because of its importance in digital signal processing applications and its close resemblance to the convolution formula, we also introduce the correlation between two signals. The autocorrelation and crosscorrelation of signals are defined and their properties are presented.

1 Discrete-Time Signals

A discrete-time signal $x(n)$ is a function of an independent variable that is an integer. It is graphically represented as in Fig. 1.1. It is important to note that a discrete-time signal is *not defined* at instants between two successive samples. Also, it is incorrect to think that $x(n)$ is equal to zero if n is not an integer. Simply, the signal $x(n)$ is not defined for noninteger values of n.

In the sequel we will assume that a discrete-time signal is defined for every integer value n for $-\infty < n < \infty$. By tradition, we refer to $x(n)$ as the "nth sample" of the signal even if the signal $x(n)$ is inherently discrete time (i.e., not obtained by sampling an analog signal). If, indeed, $x(n)$ was obtained from sampling an analog signal $x_a(t)$, then $x(n) \equiv x_a(nT)$, where T is the sampling period (i.e., the time between successive samples).

Besides the graphical representation of a discrete-time signal or sequence as illustrated in Fig. 1.1, there are some alternative representations that are often more convenient to use. These are:

1. Functional representation, such as

$$x(n) = \begin{cases} 1, & \text{for } n = 1, 3 \\ 4, & \text{for } n = 2 \\ 0, & \text{elsewhere} \end{cases} \tag{1.1}$$

2. Tabular representation, such as

n	$\cdots$	-2	-1	0	1	2	3	4	5	$\ldots$
$x(n)$	$\cdots$	0	0	0	1	4	1	0	0	$\ldots$

3. Sequence representation

An infinite-duration signal or sequence with the time origin ($n = 0$) indicated by the symbol $\uparrow$ is represented as

$$x(n) = \{\ldots 0, \underset{\uparrow}{0}, 1, 4, 1, 0, 0, \ldots\} \tag{1.2}$$

A sequence $x(n)$, which is zero for $n < 0$, can be represented as

$$x(n) = \{\underset{\uparrow}{0}, 1, 4, 1, 0, 0, \ldots\} \tag{1.3}$$

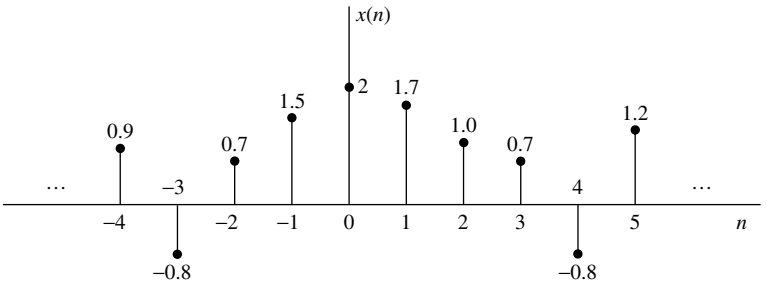


Figure 1.1 Graphical representation of a discrete-time signal.

The time origin for a sequence $x(n)$, which is zero for $n < 0$, is understood to be the first (leftmost) point in the sequence.

A finite-duration sequence can be represented as

$$x(n) = \{3, -1, \underset{\uparrow}{-2}, 5, 0, 4, -1\} \tag{1.4}$$

whereas a finite-duration sequence that satisfies the condition $x(n) = 0$ for $n < 0$ can be represented as

$$x(n) = \{\underset{\uparrow}{0}, 1, 4, 1\} \tag{1.5}$$

The signal in (1.4) consists of seven samples or points (in time), so it is called or identified as a seven-point sequence. Similarly, the sequence given by (1.5) is a four-point sequence.

1.1 Some Elementary Discrete-Time Signals

In our study of discrete-time signals and systems there are a number of basic signals that appear often and play an important role. These signals are defined below.

1. The *unit sample sequence* is denoted as $\delta(n)$ and is defined as

$$\delta(n) \equiv \begin{cases} 1, & \text{for } n = 0 \\ 0, & \text{for } n \neq 0 \end{cases} \tag{1.6}$$

In words, the unit sample sequence is a signal that is zero everywhere, except at $n = 0$ where its value is unity. This signal is sometimes referred to as a *unit impulse*. In contrast to the analog signal $\delta(t)$, which is also called a unit impulse and is defined to be zero everywhere except at $t = 0$, and has unit area, the unit sample sequence is much less mathematically complicated. The graphical representation of $\delta(n)$ is shown in Fig. 1.2.

2. The *unit step signal* is denoted as $u(n)$ and is defined as

$$u(n) \equiv \begin{cases} 1, & \text{for } n \geq 0 \\ 0, & \text{for } n < 0 \end{cases} \tag{1.7}$$

Figure 1.3 illustrates the unit step signal.

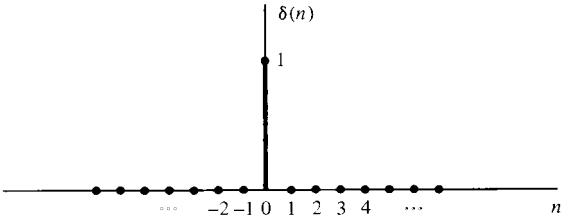


Figure 1.2
Graphical representation of the unit sample signal.

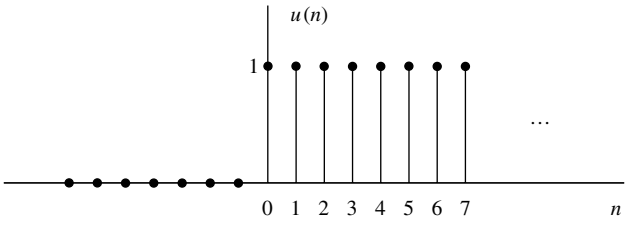


Figure 1.3
Graphical representation of the unit step signal.

3. The *unit ramp signal* is denoted as $u_r(n)$ and is defined as

$$u_r(n) \equiv \begin{cases} n, & \text{for } n \geq 0 \\ 0, & \text{for } n < 0 \end{cases} \tag{1.8}$$

This signal is illustrated in Fig. 1.4.

4. The *exponential signal* is a sequence of the form

$$x(n) = a^n \qquad \text{for all } n \tag{1.9}$$

If the parameter a is real, then $x(n)$ is a real signal. Figure 1.5 illustrates $x(n)$ for various values of the parameter a.

When the parameter a is complex valued, it can be expressed as

$$a \equiv re^{j\theta}$$

where r and θ are now the parameters. Hence we can express $x(n)$ as

$$\begin{aligned} x(n) &= r^n e^{j\theta n} \\ &= r^n(\cos\theta n + j\sin\theta n) \end{aligned} \tag{1.10}$$

Since $x(n)$ is now complex valued, it can be represented graphically by plotting the real part

$$x_R(n) \equiv r^n \cos\theta n \tag{1.11}$$

as a function of n, and separately plotting the imaginary part

$$x_I(n) \equiv r^n \sin\theta n \tag{1.12}$$

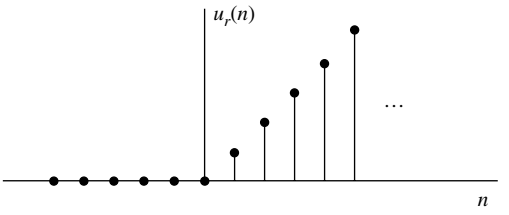


Figure 1.4
Graphical representation of the unit ramp signal.

as a function of n. Figure 1.6 illustrates the graphs of $x_R(n)$ and $x_I(n)$ for $r = 0.9$ and $\theta = \pi/10$. We observe that the signals $x_R(n)$ and $x_I(n)$ are a damped (decaying exponential) cosine function and a damped sine function. The angle variable θ is simply the frequency of the sinusoid, previously denoted by the (normalized) frequency variable ω. Clearly, if $r = 1$, the damping disappears and $x_R(n)$, $x_I(n)$, and $x(n)$ have a fixed amplitude, which is unity.

Alternatively, the signal $x(n)$ given by (1.10) can be represented graphically by the amplitude function

$$|x(n)| = A(n) \equiv r^n \tag{1.13}$$

and the phase function

$$\angle x(n) = \phi(n) \equiv \theta n \tag{1.14}$$

Figure 1.7 illustrates $A(n)$ and $\phi(n)$ for $r = 0.9$ and $\theta = \pi/10$. We observe that the phase function is linear with n. However, the phase is defined only over the interval $-\pi < \theta \leq \pi$ or, equivalently, over the interval $0 \leq \theta < 2\pi$. Consequently, by convention $\phi(n)$ is plotted over the finite interval $-\pi < \theta \leq \pi$ or $0 \leq \theta < 2\pi$. In other words, we subtract multiples of 2π from $\phi(n)$ before plotting. The subtraction of multiples of 2π from $\phi(n)$ is equivalent to interpreting the function $\phi(n)$ as $\phi(n)$, modulo 2π.

1.2 Classification of Discrete-Time Signals

The mathematical methods employed in the analysis of discrete-time signals and systems depend on the characteristics of the signals. In this section we classify discrete-time signals according to a number of different characteristics.

Energy signals and power signals. The energy E of a signal $x(n)$ is defined as

$$E \equiv \sum_{n=-\infty}^{\infty} |x(n)|^2 \tag{1.15}$$

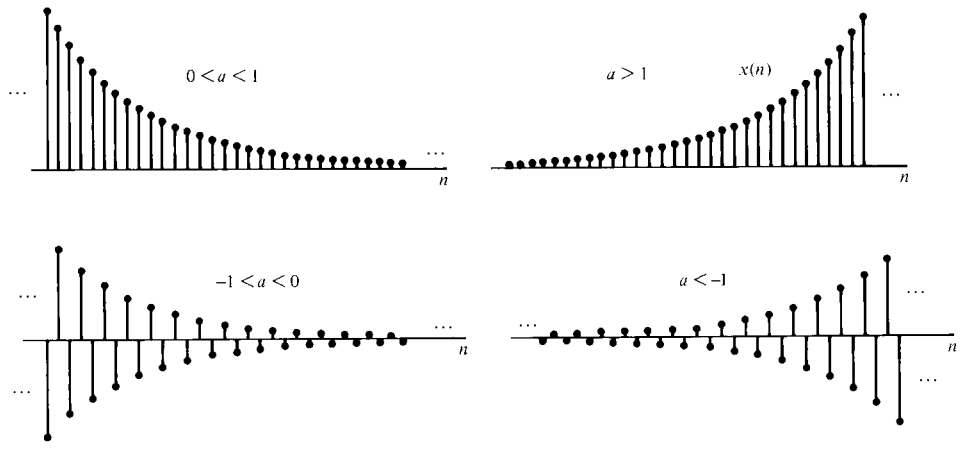


Figure 1.5 Graphical representation of exponential signals.

We have used the magnitude-squared values of $x(n)$, so that our definition applies to complex-valued signals as well as real-valued signals. The energy of a signal can be finite or infinite. If E is finite (i.e., $0 < E < \infty$), then $x(n)$ is called an *energy signal*. Sometimes we add a subscript x to E and write E_x to emphasize that E_x is the energy of the signal $x(n)$.

Many signals that possess infinite energy have a finite average power. The average power of a discrete-time signal $x(n)$ is defined as

$$P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2 \tag{1.16}$$

If we define the signal energy of $x(n)$ over the finite interval $-N \le n \le N$ as

$$E_N \equiv \sum_{n=-N}^{N} |x(n)|^2 \tag{1.17}$$

then we can express the signal energy E as

$$E \equiv \lim_{N\to\infty} E_N \tag{1.18}$$

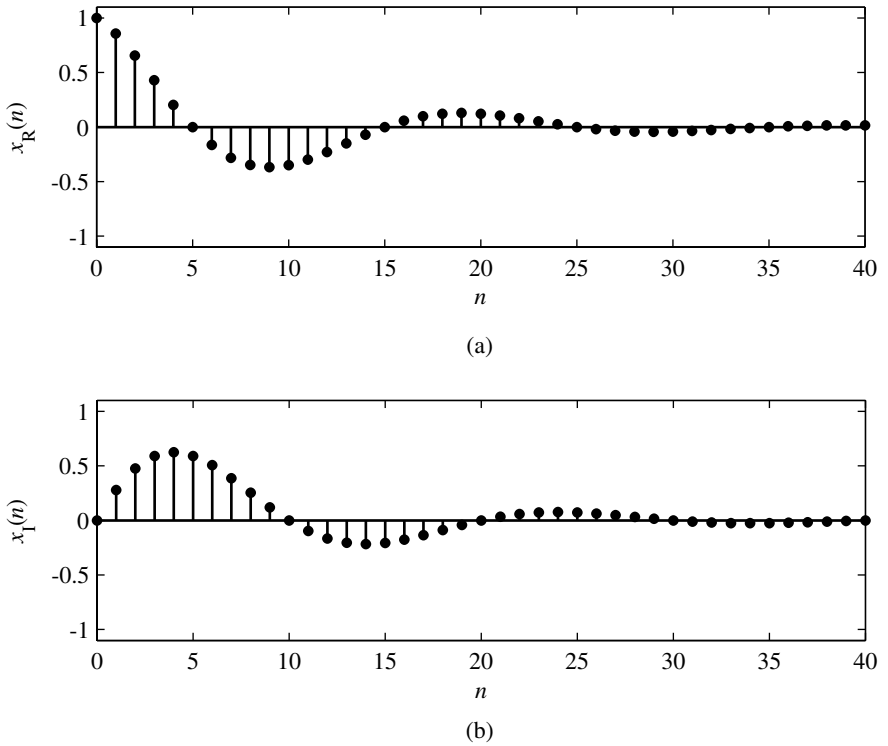


Figure 1.6 Graph of the real and imaginary components of a complex-valued exponential signal.

and the average power of the signal $x(n)$ as

$$P \equiv \lim_{N\to\infty} \frac{1}{2N+1} E_N \tag{1.19}$$

Clearly, if E is finite, $P = 0$. On the other hand, if E is infinite, the average power P may be either finite or infinite. If P is finite (and nonzero), the signal is called a *power signal*. The following example illustrates such a signal.

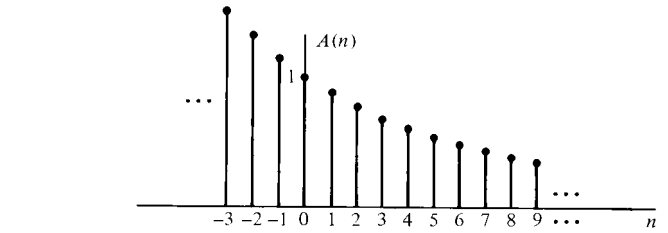
EXAMPLE 1.1

Determine the power and energy of the unit step sequence. The average power of the unit step signal is

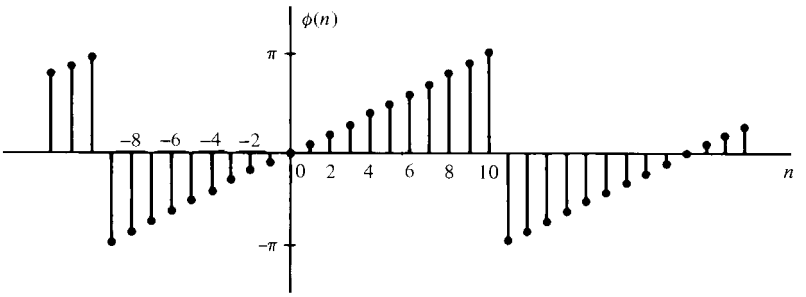
$$P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=0}^{N} u^2(n)$$

$$= \lim_{N\to\infty} \frac{N+1}{2N+1} = \lim_{N\to\infty} \frac{1+1/N}{2+1/N} = \frac{1}{2}$$

Consequently, the unit step sequence is a power signal. Its energy is infinite.



(a) Graph of $A(n) = r^n$, $r = 0.9$



(b) Graph of $\phi(n) = \frac{\pi}{10} n$, modulo 2π plotted in the range $(-\pi, \pi)$

Figure 1.7 Graph of amplitude and phase function of a complex-valued exponential signal: (a) graph of $A(n) = r^n$, $r = 0.9$; (b) graph of $\phi(n) = (\pi/10)n$, modulo 2π plotted in the range $(-\pi, \pi]$.

Similarly, it can be shown that the complex exponential sequence $x(n) = Ae^{j\omega_0 n}$ has average power A^2, so it is a power signal. On the other hand, the unit ramp sequence is neither a power signal nor an energy signal.

Periodic signals and aperiodic signals. A signal $x(n)$ is periodic with period $N (N > 0)$ if and only if

$$x(n + N) = x(n) \text{ for all } n \tag{1.20}$$

The smallest value of N for which (1.20) holds is called the (fundamental) period. If there is no value of N that satisfies (1.20), the signal is called *nonperiodic* or *aperiodic*.

We have already observed that the sinusoidal signal of the form

$$x(n) = A \sin 2\pi f_0 n \tag{1.21}$$

is periodic when f_0 is a rational number, that is, if f_0 can be expressed as

$$f_0 = \frac{k}{N} \tag{1.22}$$

where k and N are integers.

The energy of a periodic signal $x(n)$ over a single period, say, over the interval $0 \leq n \leq N - 1$, is finite if $x(n)$ takes on finite values over the period. However, the energy of the periodic signal for $-\infty \leq n \leq \infty$ is infinite. On the other hand, the average power of the periodic signal is finite and it is equal to the average power over a single period. Thus if $x(n)$ is a periodic signal with fundamental period N and takes on finite values, its power is given by

$$P = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 \tag{1.23}$$

Consequently, periodic signals are power signals.

Symmetric (even) and antisymmetric (odd) signals. A real-valued signal $x(n)$ is called symmetric (even) if

$$x(-n) = x(n) \tag{1.24}$$

On the other hand, a signal $x(n)$ is called antisymmetric (odd) if

$$x(-n) = -x(n) \tag{1.25}$$

We note that if $x(n)$ is odd, then $x(0) = 0$. Examples of signals with even and odd symmetry are illustrated in Fig. 1.8.

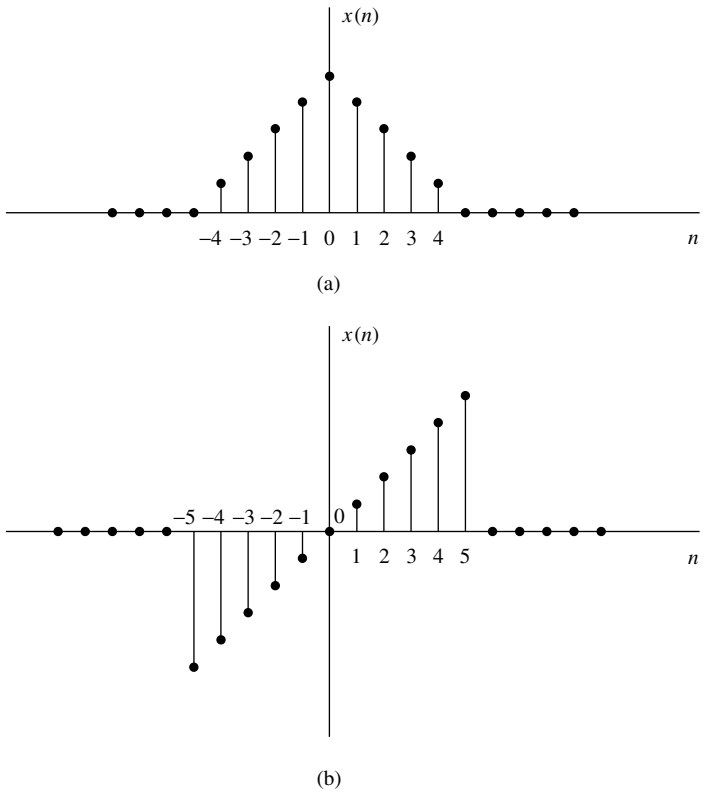


Figure 1.8 Example of even (a) and odd (b) signals.

We wish to illustrate that any arbitrary signal can be expressed as the sum of two signal components, one of which is even and the other odd. The even signal component is formed by adding $x(n)$ to $x(-n)$ and dividing by 2, that is,

$$x_e(n) = \frac{1}{2}[x(n) + x(-n)] \tag{1.26}$$

Clearly, $x_e(n)$ satisfies the symmetry condition (1.24). Similarly, we form an odd signal component $x_o(n)$ according to the relation

$$x_o(n) = \frac{1}{2}[x(n) - x(-n)] \tag{1.27}$$

Again, it is clear that $x_o(n)$ satisfies (1.25); hence it is indeed odd. Now, if we add the two signal components, defined by (1.26) and (1.27), we obtain $x(n)$, that is,

$$x(n) = x_e(n) + x_o(n) \tag{1.28}$$

Thus any arbitrary signal can be expressed as in (1.28).

1.3 Simple Manipulations of Discrete-Time Signals

In this section we consider some simple modifications or manipulations involving the independent variable and the signal amplitude (dependent variable).

Transformation of the independent variable (time). A signal $x(n)$ may be shifted in time by replacing the independent variable n by $n-k$, where k is an integer. If k is a positive integer, the time shift results in a delay of the signal by k units of time. If k is a negative integer, the time shift results in an advance of the signal by $|k|$ units in time.

EXAMPLE 1.2

A signal $x(n)$ is graphically illustrated in Fig. 1.9(a). Show a graphical representation of the signals $x(n-3)$ and $x(n+2)$.

Solution. The signal $x(n-3)$ is obtained by delaying $x(n)$ by three units in time. The result is illustrated in Fig. 1.9(b). On the other hand, the signal $x(n+2)$ is obtained by advancing $x(n)$ by two units in time. The result is illustrated in Fig. 1.9(c). Note that delay corresponds to shifting a signal to the right, whereas advance implies shifting the signal to the left on the time axis.

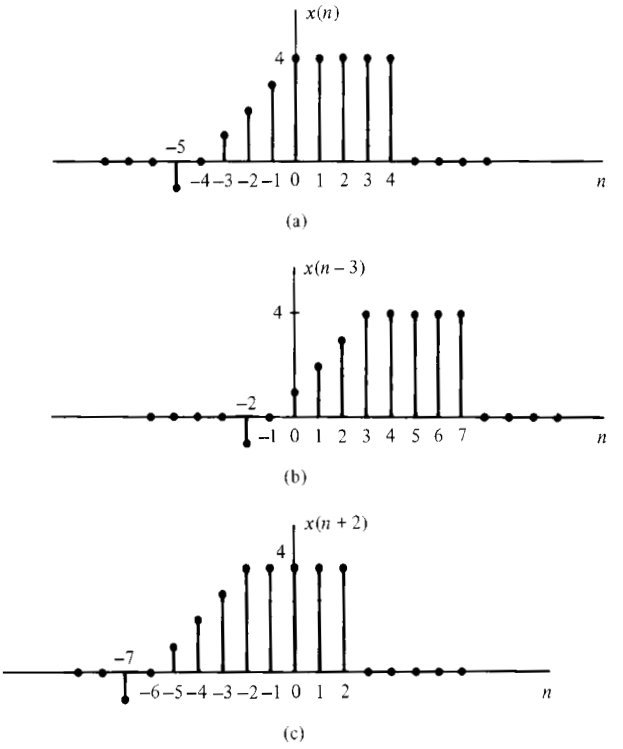


Figure 1.9
Graphical representation of a signal, and its delayed and advanced versions.

If the signal $x(n)$ is stored on magnetic tape or on a disk or, perhaps, in the memory of a computer, it is a relatively simple operation to modify the base by introducing a delay or an advance. On the other hand, if the signal is not stored but is being generated by some physical phenomenon in real time, it is not possible to advance the signal in time, since such an operation involves signal samples that have not yet been generated. Whereas it is always possible to insert a delay into signal samples that have already been generated, it is physically impossible to view the future signal samples. Consequently, in real-time signal processing applications, the operation of advancing the time base of the signal is physically unrealizable.

Another useful modification of the time base is to replace the independent variable n by $-n$. The result of this operation is a *folding* or a *reflection* of the signal about the time origin $n = 0$.

EXAMPLE 1.3

Show the graphical representation of the signals $x(-n)$ and $x(-n+2)$, where $x(n)$ is the signal illustrated in Fig. 1.10(a).

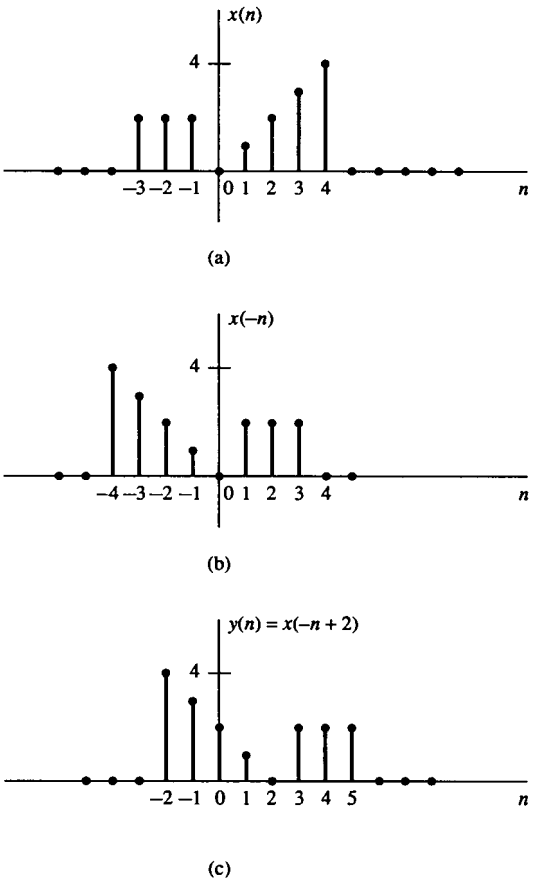


Figure 1.10
Graphical illustration of the folding and shifting operations.

Solution. The new signal $y(n) = x(-n)$ is shown in Fig. 1.10(b). Note that $y(0) = x(0)$, $y(1) = x(-1)$, $y(2) = x(-2)$, and so on. Also, $y(-1) = x(1)$, $y(-2) = x(2)$, and so on. Therefore, $y(n)$ is simply $x(n)$ reflected or folded about the time origin $n = 0$. The signal $y(n) = x(-n + 2)$ is simply $x(-n)$ delayed by two units in time. The resulting signal is illustrated in Fig. 1.10(c). A simple way to verify that the result in Fig. 1.10(c) is correct is to compute samples, such as $y(0) = x(2)$, $y(1) = x(1)$, $y(2) = x(0)$, $y(-1) = x(3)$, and so on.

It is important to note that the operations of folding and time delaying (or advancing) a signal are not commutative. If we denote the time-delay operation by TD and the folding operation by FD, we can write

$$\begin{aligned} \mathrm{TD}_k[x(n)] &= x(n-k), \qquad k > 0 \\ \mathrm{FD}[x(n)] &= x(-n) \end{aligned} \tag{1.29}$$

Now

$$\mathrm{TD}_k\{\mathrm{FD}[x(n)]\} = \mathrm{TD}_k[x(-n)] = x(-n+k) \tag{1.30}$$

whereas

$$\mathrm{FD}\{\mathrm{TD}_k[x(n)]\} = \mathrm{FD}[x(n-k)] = x(-n-k) \tag{1.31}$$

Note that because the signs of n and k in $x(n-k)$ and $x(-n+k)$ are different, the result is a shift of the signals $x(n)$ and $x(-n)$ to the right by k samples, corresponding to a time delay.

A third modification of the independent variable involves replacing n by μn, where μ is an integer. We refer to this time-base modification as *time scaling* or *down-sampling*.

EXAMPLE 1.4

Show the graphical representation of the signal $y(n) = x(2n)$, where $x(n)$ is the signal illustrated in Fig. 1.11(a).

Solution. We note that the signal $y(n)$ is obtained from $x(n)$ by taking every other sample from $x(n)$, starting with $x(0)$. Thus $y(0) = x(0)$, $y(1) = x(2)$, $y(2) = x(4), \ldots$ and $y(-1) = x(-2)$, $y(-2) = x(-4)$, and so on. In other words, we have skipped the odd-numbered samples in $x(n)$ and retained the even-numbered samples. The resulting signal is illustrated in Fig. 1.11(b).

If the signal $x(n)$ was originally obtained by sampling an analog signal $x_a(t)$, then $x(n) = x_a(nT)$, where T is the sampling interval. Now, $y(n) = x(2n) = x_a(2Tn)$. Hence the time-scaling operation described in Example 1.4 is equivalent to changing the sampling rate from $1/T$ to $1/2T$, that is, to decreasing the rate by a factor of 2. This is a *down-sampling* operation.

Addition, multiplication, and scaling of sequences. Amplitude modifications include *addition, multiplication,* and *scaling* of discrete-time signals.

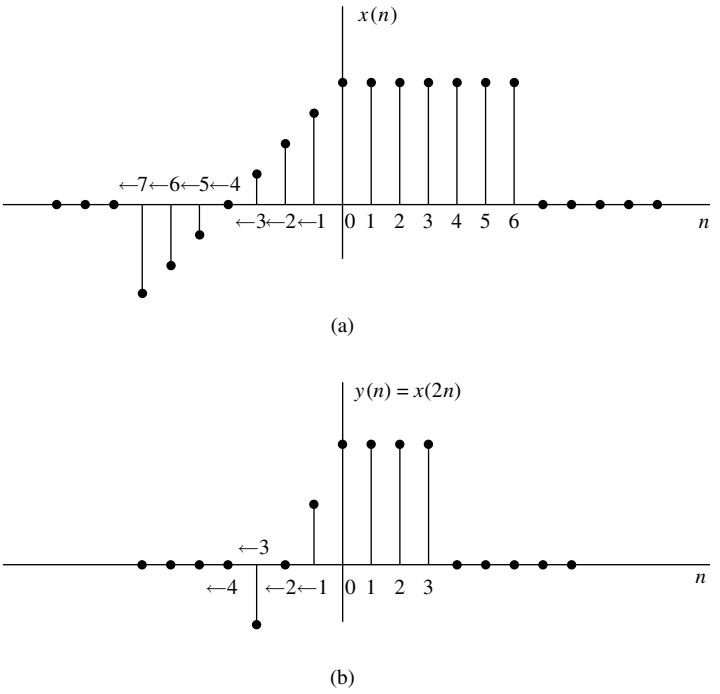


Figure 1.11 Graphical illustration of down-sampling operation.

Amplitude scaling of a signal by a constant A is accomplished by multiplying the value of every signal sample by A. Consequently, we obtain

$$y(n) = Ax(n), \qquad -\infty < n < \infty$$

The *sum* of two signals $x_1(n)$ and $x_2(n)$ is a signal $y(n)$, whose value at any instant is equal to the sum of the values of these two signals at that instant, that is,

$$y(n) = x_1(n) + x_2(n), \qquad -\infty < n < \infty$$

The *product* of two signals is similarly defined on a sample-to-sample basis as

$$y(n) = x_1(n)x_2(n), \qquad -\infty < n < \infty$$

2 Discrete-Time Systems

In many applications of digital signal processing we wish to design a device or an algorithm that performs some prescribed operation on a discrete-time signal. Such a device or algorithm is called a discrete-time system. More specifically, a *discrete-time system* is a device or algorithm that operates on a discrete-time signal, called the *input* or *excitation*, according to some well-defined rule, to produce another discrete-time

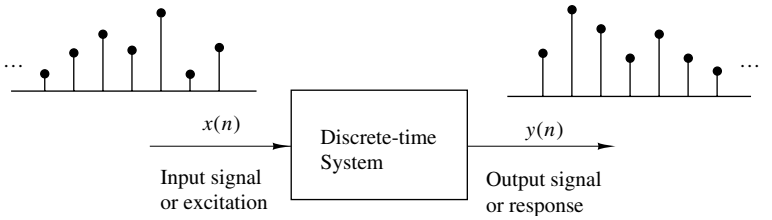


Figure 2.1 Block diagram representation of a discrete-time system.

signal called the *output* or *response* of the system. In general, we view a system as an operation or a set of operations performed on the input signal $x(n)$ to produce the output signal $y(n)$. We say that the input signal $x(n)$ is *transformed* by the system into a signal $y(n)$, and express the general relationship between $x(n)$ and $y(n)$ as

$$y(n) \equiv \mathcal{T}[x(n)] \tag{2.1}$$

where the symbol $\mathcal{T}$ denotes the transformation (also called an operator) or processing performed by the system on $x(n)$ to produce $y(n)$. The mathematical relationship in (2.1) is depicted graphically in Fig. 2.1.

There are various ways to describe the characteristics of the system and the operation it performs on $x(n)$ to produce $y(n)$. In this chapter we shall be concerned with the time-domain characterization of systems. We shall begin with an input–output description of the system. The input–output description focuses on the behavior at the terminals of the system and ignores the detailed internal construction or realization of the system.

2.1 Input–Output Description of Systems

The input–output description of a discrete-time system consists of a mathematical expression or a rule, which explicitly defines the relation between the input and output signals (*input–output relationship*). The exact internal structure of the system is either unknown or ignored. Thus the only way to interact with the system is by using its input and output terminals (i.e., the system is assumed to be a "black box" to the user). To reflect this philosophy, we use the graphical representation depicted in Fig. 2.1, and the general input–output relationship in (2.1) or, alternatively, the notation

$$x(n) \xrightarrow{\mathcal{T}} y(n) \tag{2.2}$$

which simply means that $y(n)$ is the response of the system $\mathcal{T}$ to the excitation $x(n)$. The following examples illustrate several different systems.

EXAMPLE 2.1

Determine the response of the following sytems to the input signal

$$x(n) = \begin{cases} |n|, & -3 \le n \le 3 \\ 0, & \text{otherwise} \end{cases}$$

(a) $y(n) = x(n)$ (identity system)

(b) $y(n) = x(n-1)$ (unit delay system)

(c) $y(n) = x(n+1)$ (unit advance system)

(d) $y(n) = \frac{1}{3}[x(n+1) + x(n) + x(n-1)]$ (moving average filter)

(e) $y(n) = \text{median}\{x(n+1), x(n), x(n-1)\}$ (median filter)

(f) $y(n) = \sum_{k=-\infty}^{n} x(k) = x(n) + x(n-1) + x(n-2) + \cdots$ (accumulator) (2.3)

Solution. First, we determine explicitly the sample values of the input signal

$$x(n) = \{\ldots, 0, 3, 2, 1, \underset{\uparrow}{0}, 1, 2, 3, 0, \ldots\}$$

Next, we determine the output of each system using its input–output relationship.

(a) In this case the output is exactly the same as the input signal. Such a system is known as the *identity* system.

(b) This system simply delays the input by one sample. Thus its output is given by

$$x(n) = \{\ldots, 0, 3, 2, \underset{\uparrow}{1}, 0, 1, 2, 3, 0, \ldots\}$$

(c) In this case the system "advances" the input one sample into the future. For example, the value of the output at time $n = 0$ is $y(0) = x(1)$. The response of this system to the given input is

$$x(n) = \{\ldots, 0, 3, 2, 1, 0, \underset{\uparrow}{1}, 2, 3, 0, \ldots\}$$

(d) The output of this system at any time is the mean value of the present, the immediate past, and the immediate future samples. For example, the output at time $n = 0$ is

$$y(0) = \frac{1}{3}[x(-1) + x(0) + x(1)] = \frac{1}{3}[1 + 0 + 1] = \frac{2}{3}$$

Repeating this computation for every value of n, we obtain the output signal

$$y(n) = \{\ldots, 0, 1, \frac{5}{3}, 2, 1, \underset{\uparrow}{\frac{2}{3}}, 1, 2, \frac{5}{3}, 1, 0, \ldots\}$$

(e) This system selects as its output at time n the median value of the three input samples $x(n-1)$, $x(n)$, and $x(n+1)$. Thus the response of this system to the input signal $x(n)$ is

$$y(n) = \{0, 2, 2, 1, 1, \underset{\uparrow}{1}, 2, 2, 0, 0, 0, \ldots\}$$

(f) This system is basically an *accumulator* that computes the running sum of all the past input values up to present time. The response of this system to the given input is

$$y(n) = \{\ldots, 0, 3, 5, 6, \underset{\uparrow}{6}, 7, 9, 12, 0, \ldots\}$$

We observe that for several of the systems considered in Example 2.1 the output at time $n = n_0$ depends not only on the value of the input at $n = n_0$ [i.e., $x(n_0)$], but also on the values of the input applied to the system before and after $n = n_0$. Consider, for instance, the accumulator in the example. We see that the output at time $n = n_0$ depends not only on the input at time $n = n_0$, but also on $x(n)$ at times $n = n_0 - 1$, $n_0 - 2$, and so on. By a simple algebraic manipulation the input–output relation of the accumulator can be written as

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{n} x(k) = \sum_{k=-\infty}^{n-1} x(k) + x(n) \\ &= y(n-1) + x(n) \end{aligned} \tag{2.4}$$

which justifies the term *accumulator*. Indeed, the system computes the current value of the output by adding (accumulating) the current value of the input to the previous output value.

There are some interesting conclusions that can be drawn by taking a close look into this apparently simple system. Suppose that we are given the input signal $x(n)$ for $n \geq n_0$, and we wish to determine the output $y(n)$ of this system for $n \geq n_0$. For $n = n_0, n_0 + 1, \ldots,$ (2.4) gives

$$y(n_0) = y(n_0 - 1) + x(n_0)$$

$$y(n_0 + 1) = y(n_0) + x(n_0 + 1)$$

and so on. Note that we have a problem in computing $y(n_0)$, since it depends on $y(n_0 - 1)$. However,

$$y(n_0 - 1) = \sum_{k=-\infty}^{n_0-1} x(k)$$

that is, $y(n_0 - 1)$ "summarizes" the effect on the system from all the inputs which had been applied to the system before time n_0. Thus the response of the system for $n \geq n_0$ to the input $x(n)$ that is applied at time n_0 is the combined result of this input and all inputs that had been applied previously to the system. Consequently, $y(n)$, $n \geq n_0$ is not uniquely determined by the input $x(n)$ for $n \geq n_0$.

The additional information required to determine $y(n)$ for $n \geq n_0$ is the *initial condition* $y(n_0 - 1)$. This value summarizes the effect of all previous inputs to the system. Thus the initial condition $y(n_0 - 1)$ together with the input sequence $x(n)$ for $n \geq n_0$ uniquely determine the output sequence $y(n)$ for $n \geq n_0$.

If the accumulator had no excitation prior to n_0, the initial condition is $y(n_0 - 1) = 0$. In such a case we say that the system is *initially relaxed*. Since $y(n_0 - 1) = 0$, the output sequence $y(n)$ depends only on the input sequence $x(n)$ for $n \geq n_0$.

It is customary to assume that every system is relaxed at $n = -\infty$. In this case, if an input $x(n)$ is applied at $n = -\infty$, the corresponding output $y(n)$ is *solely* and *uniquely* determined by the given input.

EXAMPLE 2.2

The accumulator described by (2.30) is excited by the sequence $x(n) = nu(n)$. Determine its output under the condition that:

(a) It is initially relaxed [i.e., $y(-1) = 0$].

(b) Initially, $y(-1) = 1$.

Solution. The output of the system is defined as

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{n} x(k) = \sum_{k=-\infty}^{-1} x(k) + \sum_{k=0}^{n} x(k) \\ &= y(-1) + \sum_{k=0}^{n} x(k) \\ &= y(-1) + \frac{n(n+1)}{2} \end{aligned}$$

(a) If the system is initially relaxed, $y(-1) = 0$ and hence

$$y(n) = \frac{n(n+1)}{2}, \qquad n \geq 0$$

(b) On the other hand, if the initial condition is $y(-1) = 1$, then

$$y(n) = 1 + \frac{n(n+1)}{2} = \frac{n^2 + n + 2}{2}, \qquad n \geq 0$$

2.2 Block Diagram Representation of Discrete-Time Systems

It is useful at this point to introduce a block diagram representation of discrete-time systems. For this purpose we need to define some basic building blocks that can be interconnected to form complex systems.

An adder. Figure 2.2 illustrates a system (adder) that performs the addition of two signal sequences to form another (the sum) sequence, which we denote as $y(n)$. Note that it is not necessary to store either one of the sequences in order to perform the addition. In other words, the addition operation is *memoryless*.

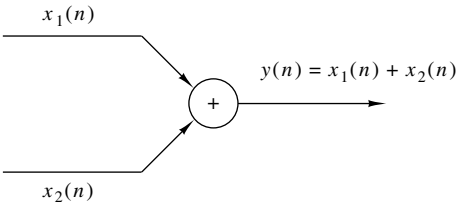


Figure 2.2
Graphical representation of an adder.

A constant multiplier. This operation is depicted by Fig. 2.3, and simply represents applying a scale factor on the input $x(n)$. Note that this operation is also memoryless.

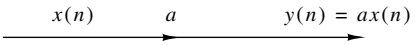


Figure 2.3
Graphical representation of a constant multiplier.

A signal multiplier. Figure 2.4 illustrates the multiplication of two signal sequences to form another (the product) sequence, denoted in the figure as $y(n)$. As in the preceding two cases, we can view the multiplication operation as memoryless.

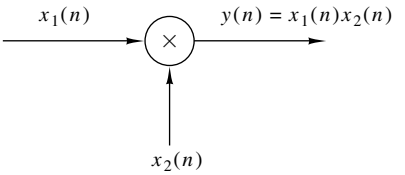


Figure 2.4
Graphical representation of a signal multiplier.

A unit delay element. The unit delay is a special system that simply delays the signal passing through it by one sample. Figure 2.5 illustrates such a system. If the input signal is $x(n)$, the output is $x(n-1)$. In fact, the sample $x(n-1)$ is stored in memory at time $n-1$ and it is recalled from memory at time n to form

$$y(n) = x(n-1)$$

Thus this basic building block requires memory. The use of the symbol z^{-1} to denote the unit of delay will become apparent when discussing the z-transform.

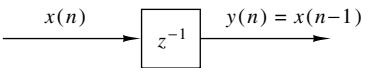


Figure 2.5
Graphical representation of the unit delay element.

A unit advance element. In contrast to the unit delay, a unit advance moves the input $x(n)$ ahead by one sample in time to yield $x(n+1)$. Figure 2.6 illustrates this operation, with the operator z being used to denote the unit advance. We observe that any such advance is physically impossible in real time, since, in fact, it involves looking into the future of the signal. On the other hand, if we store the signal in the memory of the computer, we can recall any sample at any time. In such a non-real-time application, it is possible to advance the signal $x(n)$ in time.

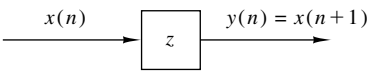


Figure 2.6
Graphical representation of the unit advance element.

EXAMPLE 2.3

Using basic building blocks introduced above, sketch the block diagram representation of the discrete-time system described by the input–output relation

$$y(n) = \frac{1}{4}y(n-1) + \frac{1}{2}x(n) + \frac{1}{2}x(n-1) \tag{2.5}$$

where $x(n)$ is the input and $y(n)$ is the output of the system.

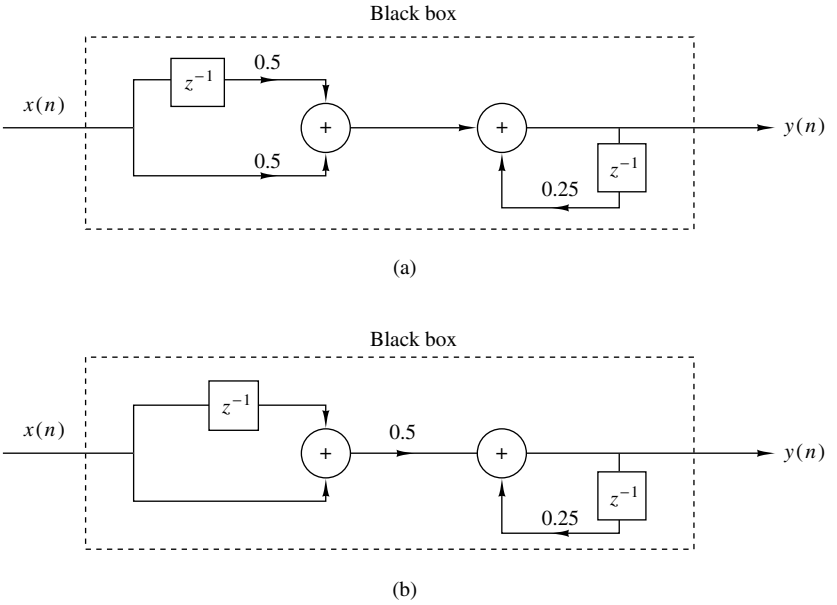


Figure 2.7 Block diagram realizations of the system $y(n) = 0.25y(n-1) + 0.5x(n) + 0.5x(n-1)$.

Solution. According to (2.5), the output $y(n)$ is obtained by multiplying the input $x(n)$ by 0.5, multiplying the previous input $x(n-1)$ by 0.5, adding the two products, and then adding the previous output $y(n-1)$ multiplied by $\frac{1}{4}$. Figure 2.7(a) illustrates this block diagram realization of the system. A simple rearrangement of (2.5), namely,

$$y(n) = \frac{1}{4}y(n-1) + \frac{1}{2}[x(n) + x(n-1)] \tag{2.6}$$

leads to the block diagram realization shown in Fig. 2.7(b). Note that if we treat "the system" from the "viewpoint" of an input–output or an external description, we are not concerned about how the system is realized. On the other hand, if we adopt an internal description of the system, we know exactly how the system building blocks are configured. In terms of such a realization, we can see that a system is *relaxed* at time $n = n_0$ if the outputs of all the *delays* existing in the system are zero at $n = n_0$ (i.e., all memory is *filled* with zeros).

2.3 Classification of Discrete-Time Systems

In the analysis as well as in the design of systems, it is desirable to classify the systems according to the general properties that they satisfy. In fact, the mathematical techniques developed in this chapter and future study for analyzing and designing discrete-time systems depend heavily on the general characteristics of the systems that are being considered. For this reason it is necessary for us to develop a number of properties or categories that can be used to describe the general characteristics of systems.

We stress the point that for a system to possess a given property, the property must hold for every possible input signal to the system. If a property holds for some

input signals but not for others, the system does not possess that property. Thus a counterexample is sufficient to prove that a system does not possess a property. However, to prove that the system has some property, we must prove that this property holds for every possible input signal.

Static versus dynamic systems. A discrete-time system is called *static* or memoryless if its output at any instant n depends at most on the input sample at the same time, but not on past or future samples of the input. In any other case, the system is said to be *dynamic* or to have memory. If the output of a system at time n is completely determined by the input samples in the interval from $n-N$ to $n(N \geq 0)$, the system is said to have *memory* of duration N. If $N = 0$, the system is static. If $0 < N < \infty$, the system is said to have *finite memory*, whereas if $N = \infty$, the system is said to have *infinite memory*.

The systems described by the following input–output equations

$$y(n) = ax(n) \tag{2.7}$$

$$y(n) = nx(n) + bx^3(n) \tag{2.8}$$

are both static or memoryless. Note that there is no need to store any of the past inputs or outputs in order to compute the present output. On the other hand, the systems described by the following input–output relations

$$y(n) = x(n) + 3x(n-1) \tag{2.9}$$

$$y(n) = \sum_{k=0}^{n} x(n-k) \tag{2.10}$$

$$y(n) = \sum_{k=0}^{\infty} x(n-k) \tag{2.11}$$

are dynamic systems or systems with memory. The systems described by (2.9) and (2.10) have finite memory, whereas the system described by (2.11) has infinite memory.

We observe that static or memoryless systems are described in general by input–output equations of the form

$$y(n) = \mathcal{T}[x(n), n] \tag{2.12}$$

and they do not include delay elements (memory).

Time-invariant versus time-variant systems. We can subdivide the general class of systems into the two broad categories, time-invariant systems and time-variant systems. A system is called time-invariant if its input–output characteristics do not change with time. To elaborate, suppose that we have a system $\mathcal{T}$ in a relaxed state

which, when excited by an input signal $x(n)$, produces an output signal $y(n)$. Thus we write

$$y(n) = \mathcal{T}[x(n)] \tag{2.13}$$

Now suppose that the same input signal is delayed by k units of time to yield $x(n-k)$, and again applied to the same system. If the characteristics of the system do not change with time, the output of the relaxed system will be $y(n-k)$. That is, the output will be the same as the response to $x(n)$, except that it will be delayed by the same k units in time that the input was delayed. This leads us to define a time-invariant or shift-invariant system as follows.

Definition. A relaxed system $\mathcal{T}$ is *time invariant* or *shift invariant* if and only if

$$x(n) \xrightarrow{\mathcal{T}} y(n)$$

implies that

$$x(n-k) \xrightarrow{\mathcal{T}} y(n-k) \tag{2.14}$$

for every input signal $x(n)$ and every time shift k.

To determine if any given system is time invariant, we need to perform the test specified by the preceding definition. Basically, we excite the system with an arbitrary input sequence $x(n)$, which produces an output denoted as $y(n)$. Next we delay the input sequence by some amount k and recompute the output. In general, we can write the output as

$$y(n,k) = \mathcal{T}[x(n-k)]$$

Now if this output $y(n,k) = y(n-k)$, for all possible values of k, the system is time invariant. On the other hand, if the output $y(n,k) \neq y(n-k)$, even for one value of k, the system is time variant.

EXAMPLE 2.4

Determine if the systems shown in Fig. 2.8 are time invariant or time variant.

Solution.

(a) This system is described by the input–output equations

$$y(n) = \mathcal{T}[x(n)] = x(n) - x(n-1) \tag{2.15}$$

Now if the input is delayed by k units in time and applied to the system, it is clear from the block diagram that the output will be

$$y(n,k) = x(n-k) - x(n-k-1) \tag{2.16}$$

On the other hand, from (2.14) we note that if we delay $y(n)$ by k units in time, we obtain

$$y(n-k) = x(n-k) - x(n-k-1) \tag{2.17}$$

Since the right-hand sides of (2.16) and (2.17) are identical, it follows that $y(n,k) = y(n-k)$. Therefore, the system is time invariant.

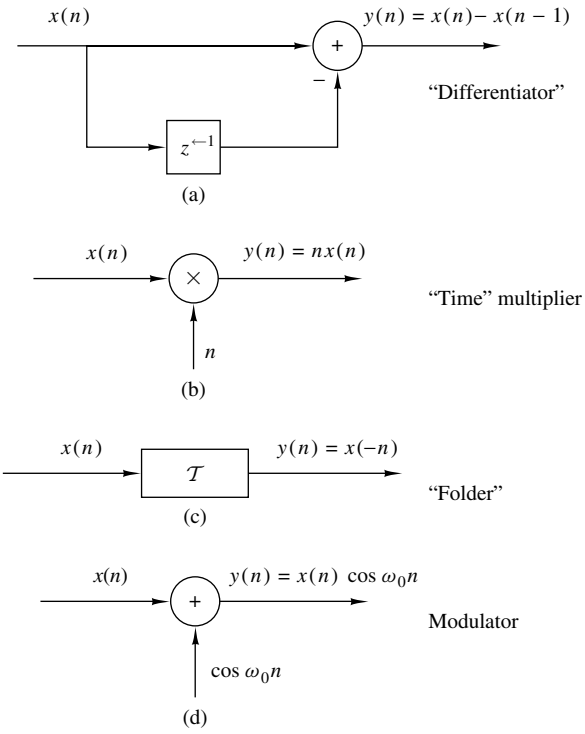


Figure 2.8
Examples of a time-invariant (a) and some time-variant systems (b)–(d).

(b) The input–output equation for this system is

$$y(n) = \mathcal{T}[x(n)] = nx(n) \tag{2.18}$$

The response of this system to $x(n-k)$ is

$$y(n,k) = nx(n-k) \tag{2.19}$$

Now if we delay $y(n)$ in (2.18) by k units in time, we obtain

$$\begin{aligned} y(n-k) &= (n-k)x(n-k) \\ &= nx(n-k) - kx(n-k) \end{aligned} \tag{2.20}$$

This system is time variant, since $y(n,k) \neq y(n-k)$.

(c) This system is described by the input–output relation

$$y(n) = \mathcal{T}[x(n)] = x(-n) \tag{2.21}$$

The response of this system to $x(n-k)$ is

$$y(n,k) = \mathcal{T}[x(n-k)] = x(-n-k) \tag{2.22}$$

Now, if we delay the output $y(n)$, as given by (2.21), by k units in time, the result will be

$$y(n-k) = x(-n+k) \tag{2.23}$$

Since $y(n,k) \neq y(n-k)$, the system is time variant.

(d) The input–output equation for this system is

$$y(n) = x(n)\cos\omega_0 n \tag{2.24}$$

The response of this system to $x(n-k)$ is

$$y(n,k) = x(n-k)\cos\omega_0 n \tag{2.25}$$

If the expression in (2.24) is delayed by k units and the result is compared to (2.25), it is evident that the system is time variant.

Linear versus nonlinear systems. The general class of systems can also be subdivided into linear systems and nonlinear systems. A linear system is one that satisfies the *superposition principle*. Simply stated, the principle of superposition requires that the response of the system to a weighted sum of signals be equal to the corresponding weighted sum of the responses (outputs) of the system to each of the individual input signals. Hence we have the following definition of linearity.

Definition. A system is linear if and only if

$$\mathcal{T}[a_1x_1(n) + a_2x_2(n)] = a_1\mathcal{T}[x_1(n)] + a_2\mathcal{T}[x_2(n)] \tag{2.26}$$

for any arbitrary input sequences $x_1(n)$ and $x_2(n)$, and any arbitrary constants a_1 and a_2. Figure 2.9 gives a pictorial illustration of the superposition principle.

The superposition principle embodied in the relation (2.26) can be separated into two parts. First, suppose that $a_2 = 0$. Then (2.26) reduces to

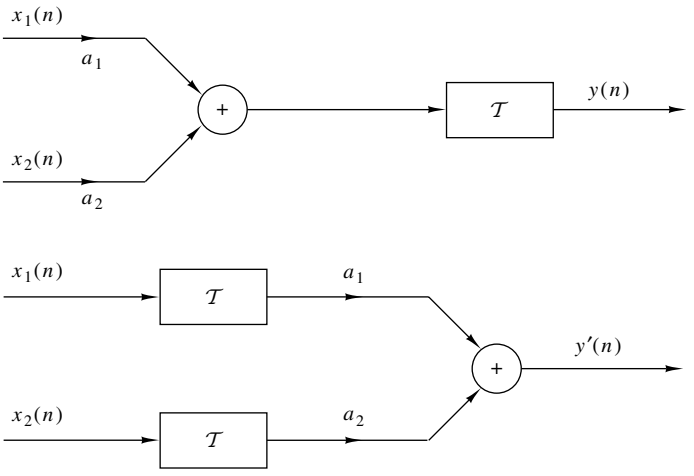


Figure 2.9 Graphical representation of the superposition principle. $\mathcal{T}$ is linear if and only if $y(n) = y'(n)$.

$$\mathcal{T}[a_1x_1(n)] = a_1\mathcal{T}[x_1(n)] = a_1y_1(n) \tag{2.27}$$

where

$$y_1(n) = \mathcal{T}[x_1(n)]$$

The relation (2.27) demonstrates the *multiplicative* or *scaling property* of a linear system. That is, if the response of the system to the input $x_1(n)$ is $y_1(n)$, the response to $a_1x_1(n)$ is simply $a_1y_1(n)$. Thus any scaling of the input results in an identical scaling of the corresponding output.

Second, suppose that $a_1 = a_2 = 1$ in (2.26). Then

$$\begin{aligned}\mathcal{T}[x_1(n) + x_2(n)] &= \mathcal{T}[x_1(n)] + \mathcal{T}[x_1(n)] \\ &= y_1(n) + y_2(n)\end{aligned} \tag{2.28}$$

This relation demonstrates the *additivity property* of a linear system. The additivity and multiplicative properties constitute the superposition principle as it applies to linear systems.

The linearity condition embodied in (2.26) can be extended arbitrarily to any weighted linear combination of signals by induction. In general, we have

$$x(n) = \sum_{k=1}^{M-1} a_kx_k(n) \xrightarrow{\mathcal{T}} y(n) = \sum_{k=1}^{M-1} a_ky_k(n) \tag{2.29}$$

where

$$y_k(n) = \mathcal{T}[x_k(n)], \qquad k = 1, 2, \ldots, M-1 \tag{2.30}$$

We observe from (2.27) that if $a_1 = 0$, then $y(n) = 0$. In other words, a relaxed, linear system with zero input produces a zero output. If a system produces a nonzero output with a zero input, the system may be either nonrelaxed or nonlinear. If a relaxed system does not satisfy the superposition principle as given by the definition above, it is called *nonlinear*.

EXAMPLE 2.5

Determine if the systems described by the following input–output equations are linear or nonlinear.
(a) $y(n) = nx(n)$ **(b)** $y(n) = x(n^2)$ **(c)** $y(n) = x^2(n)$
(d) $y(n) = Ax(n) + B$ **(e)** $y(n) = e^{x(n)}$

Solution.

(a) For two input sequences $x_1(n)$ and $x_2(n)$, the corresponding outputs are

$$\begin{aligned}y_1(n) &= nx_1(n) \\ y_2(n) &= nx_2(n)\end{aligned} \tag{2.31}$$

A linear combination of the two input sequences results in the output

$$\begin{aligned}y_3(n) &= \mathcal{T}[a_1x_1(n) + a_2x_2(n)] = n[a_1x_1(n) + a_2x_2(n)] \\ &= a_1nx_1(n) + a_2nx_2(n)\end{aligned} \tag{2.32}$$

On the other hand, a linear combination of the two outputs in (2.31) results in the output

$$a_1 y_1(n) + a_2 y_2(n) = a_1 n x_1(n) + a_2 n x_2(n) \tag{2.33}$$

Since the right-hand sides of (2.32) and (2.33) are identical, the system is linear.

(b) As in part (a), we find the response of the system to two separate input signals $x_1(n)$ and $x_2(n)$. The result is

$$\begin{aligned} y_1(n) &= x_1(n^2) \\ y_2(n) &= x_2(n^2) \end{aligned} \tag{2.34}$$

The output of the system to a linear combination of $x_1(n)$ and $x_2(n)$ is

$$y_3(n) = \mathcal{T}[a_1 x_1(n) + a_2 x_2(n)] = a_1 x_1(n^2) + a_2 x_2(n^2) \tag{2.35}$$

Finally, a linear combination of the two outputs in (2.34) yields

$$a_1 y_1(n) + a_2 y_2(n) = a_1 x_1(n^2) + a_2 x_2(n^2) \tag{2.36}$$

By comparing (2.35) with (2.36), we conclude that the system is linear.

(c) The output of the system is the square of the input. (Electronic devices that have such an input–output characteristic are called square-law devices.) From our previous discussion it is clear that such a system is memoryless. We now illustrate that this system is nonlinear.

The responses of the system to two separate input signals are

$$\begin{aligned} y_1(n) &= x_1^2(n) \\ y_2(n) &= x_2^2(n) \end{aligned} \tag{2.37}$$

The response of the system to a linear combination of these two input signals is

$$\begin{aligned} y_3(n) &= \mathcal{T}[a_1 x_1(n) + a_2 x_2(n)] \\ &= [a_1 x_1(n) + a_2 x_2(n)]^2 \\ &= a_1^2 x_1^2(n) + 2a_1 a_2 x_1(n) x_2(n) + a_2^2 x_2^2(n) \end{aligned} \tag{2.38}$$

On the other hand, if the system is linear, it will produce a linear combination of the two outputs in (2.37), namely,

$$a_1 y_1(n) + a_2 y_2(n) = a_1 x_1^2(n) + a_2 x_2^2(n) \tag{2.39}$$

Since the actual output of the system, as given by (2.38), is not equal to (2.39), the system is nonlinear.

(d) Assuming that the system is excited by $x_1(n)$ and $x_2(n)$ separately, we obtain the corresponding outputs

$$\begin{aligned} y_1(n) &= Ax_1(n) + B \\ y_2(n) &= Ax_2(n) + B \end{aligned} \tag{2.40}$$

A linear combination of $x_1(n)$ and $x_2(n)$ produces the output

$$\begin{aligned} y_3(n) &= \mathcal{T}[a_1x_1(n) + a_2x_2(n)] \\ &= A[a_1x_1(n) + a_2x_2(n)] + B \\ &= Aa_1x_1(n) + a_2Ax_2(n) + B \end{aligned} \tag{2.41}$$

On the other hand, if the system were linear, its output to the linear combination of $x_1(n)$ and $x_2(n)$ would be a linear combination of $y_1(n)$ and $y_2(n)$, that is,

$$a_1y_1(n) + a_2y_2(n) = a_1Ax_1(n) + a_1B + a_2Ax_2(n) + a_2B \tag{2.42}$$

Clearly, (2.41) and (2.42) are different and hence the system fails to satisfy the linearity test.

The reason that this system fails to satisfy the linearity test is not that the system is nonlinear (in fact, the system is described by a linear equation) but the presence of the constant B. Consequently, the output depends on both the input excitation and on the parameter $B \neq 0$. Hence, for $B \neq 0$, the system is not relaxed. If we set $B = 0$, the system is now relaxed and the linearity test is satisfied.

(e) Note that the system described by the input–output equation

$$y(n) = e^{x(n)} \tag{2.43}$$

is relaxed. If $x(n) = 0$, we find that $y(n) = 1$. This is an indication that the system is nonlinear. This, in fact, is the conclusion reached when the linearity test is applied.

Causal versus noncausal systems. We begin with the definition of causal discrete-time systems.

Definition. A system is said to be *causal* if the output of the system at any time n [i.e., $y(n)$] depends only on present and past inputs [i.e., $x(n)$, $x(n-1)$, $x(n-2)$, ...], but does not depend on future inputs [i.e., $x(n+1)$, $x(n+2)$, ...]. In mathematical terms, the output of a causal system satisfies an equation of the form

$$y(n) = F[x(n), x(n-1), x(n-2), \ldots] \tag{2.44}$$

where $F[\cdot]$ is some arbitrary function.

If a system does not satisfy this definition, it is called *noncausal*. Such a system has an output that depends not only on present and past inputs but also on future inputs.

It is apparent that in real-time signal processing applications we cannot observe future values of the signal, and hence a noncausal system is physically unrealizable (i.e., it cannot be implemented). On the other hand, if the signal is recorded so that the processing is done off-line (nonreal time), it is possible to implement a noncausal system, since all values of the signal are available at the time of processing. This is often the case in the processing of geophysical signals and images.

EXAMPLE 2.6

Determine if the systems described by the following input–output equations are causal or noncausal.
(a) $y(n) = x(n) - x(n-1)$ **(b)** $y(n) = \sum_{k=-\infty}^{n} x(k)$ **(c)** $y(n) = ax(n)$ **(d)** $y(n) = x(n) + 3x(n+4)$
(e) $y(n) = x(n^2)$ **(f)** $y(n) = x(2n)$ **(g)** $y(n) = x(-n)$

Solution. The systems described in parts (a), (b), and (c) are clearly causal, since the output depends only on the present and past inputs. On the other hand, the systems in parts (d), (e), and (f) are clearly noncausal, since the output depends on future values of the input. The system in (g) is also noncausal, as we note by selecting, for example, $n = -1$, which yields $y(-1) = x(1)$. Thus the output at $n = -1$ depends on the input at $n = 1$, which is two units of time into the future.

Stable versus unstable systems. Stability is an important property that must be considered in any practical application of a system. Unstable systems usually exhibit erratic and extreme behavior and cause overflow in any practical implementation. Here, we define mathematically what we mean by a stable system, and later, in Section 3.6, we explore the implications of this definition for linear, time-invariant systems.

Definition. An arbitrary relaxed system is said to be bounded input–bounded output (BIBO) stable if and only if every bounded input produces a bounded output.

The condition that the input sequence $x(n)$ and the output sequence $y(n)$ are bounded is translated mathematically to mean that there exist some finite numbers, say M_x and M_y, such that

$$|x(n)| \leq M_x < \infty, \qquad |y(n)| \leq M_y < \infty \tag{2.45}$$

for all n. If, for some bounded input sequence $x(n)$, the output is unbounded (infinite), the system is classified as unstable.

EXAMPLE 2.7

Consider the nonlinear system described by the input–output equation

$$y(n) = y^2(n-1) + x(n)$$

As an input sequence we select the bounded signal

$$x(n) = C\delta(n)$$

where C is a constant. We also assume that $y(-1) = 0$. Then the output sequence is

$$y(0) = C, \quad y(1) = C^2, \quad y(2) = C^4, \quad \ldots, \quad y(n) = C^{2n}$$

Clearly, the output is unbounded when $1 < |C| < \infty$. Therefore, the system is BIBO unstable, since a bounded input sequence has resulted in an unbounded output.

2.4 Interconnection of Discrete-Time Systems

Discrete-time systems can be interconnected to form larger systems. There are two basic ways in which systems can be interconnected: in cascade (series) or in parallel. These interconnections are illustrated in Fig. 2.10. Note that the two interconnected systems are different.

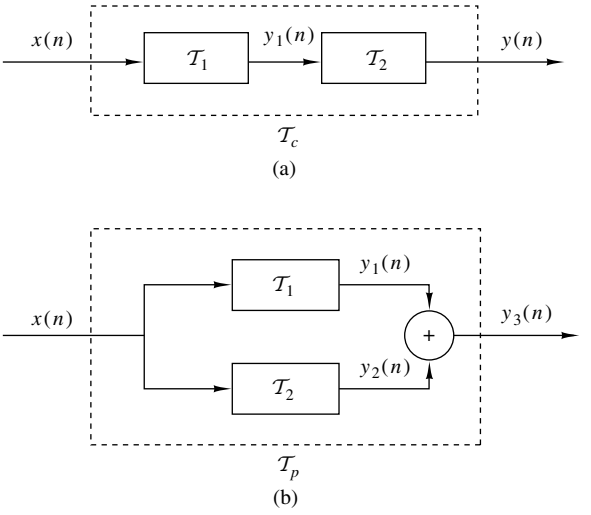


Figure 2.10
Cascade (a) and parallel (b) interconnections of systems.

In the cascade interconnection the output of the first system is

$$y_1(n) = \mathcal{T}_1[x(n)] \tag{2.46}$$

and the output of the second system is

$$\begin{aligned} y(n) &= \mathcal{T}_2[y_1(n)] \\ &= \mathcal{T}_2\{\mathcal{T}_1[x(n)]\} \end{aligned} \tag{2.47}$$

We observe that systems $\mathcal{T}_1$ and $\mathcal{T}_2$ can be combined or consolidated into a single overall system

$$\mathcal{T}_c \equiv \mathcal{T}_2\mathcal{T}_1 \tag{2.48}$$

Consequently, we can express the output of the combined system as

$$y(n) = \mathcal{T}_c[x(n)]$$

In general, the order in which the operations $\mathcal{T}_1$ and $\mathcal{T}_2$ are performed is important. That is,

$$\mathcal{T}_2\mathcal{T}_1 \neq \mathcal{T}_1\mathcal{T}_2$$

for arbitrary systems. However, if the systems $\mathcal{T}_1$ and $\mathcal{T}_2$ are linear and time invariant, then (a) $\mathcal{T}_c$ is time invariant and (b) $\mathcal{T}_2\mathcal{T}_1 = \mathcal{T}_1\mathcal{T}_2$, that is, the order in which the systems process the signal is not important. $\mathcal{T}_2\mathcal{T}_1$ and $\mathcal{T}_1\mathcal{T}_2$ yield identical output sequences.

The proof of (a) follows. The proof of (b) is given in Section 3.4. To prove time invariance, suppose that $\mathcal{T}_1$ and $\mathcal{T}_2$ are time invariant; then

$$x(n-k) \xrightarrow{\mathcal{T}_1} y_1(n-k)$$

and

$$y_1(n-k) \xrightarrow{\mathcal{T}_2} y(n-k)$$

Thus

$$x(n-k) \xrightarrow{\mathcal{T}_c=\mathcal{T}_2\mathcal{T}_1} y(n-k)$$

and therefore, $\mathcal{T}_c$ is time invariant.

In the parallel interconnection, the output of the system $\mathcal{T}_1$ is $y_1(n)$ and the output of the system $\mathcal{T}_2$ is $y_2(n)$. Hence the output of the parallel interconnection is

$$\begin{aligned} y_3(n) &= y_1(n) + y_2(n) \\ &= \mathcal{T}_1[x(n)] + \mathcal{T}_2[x(n)] \\ &= (\mathcal{T}_1 + \mathcal{T}_2)[x(n)] \\ &= \mathcal{T}_p[x(n)] \end{aligned}$$

where $\mathcal{T}_p = \mathcal{T}_1 + \mathcal{T}_2$.

In general, we can use parallel and cascade interconnection of systems to construct larger, more complex systems. Conversely, we can take a larger system and break it down into smaller subsystems for purposes of analysis and implementation. We shall use these notions later, in the design and implementation of digital filters.

3 Analysis of Discrete-Time Linear Time-Invariant Systems

In Section 2 we classified systems in accordance with a number of characteristic properties or categories, namely: linearity, causality, stability, and time invariance. Having done so, we now turn our attention to the analysis of the important class of linear, time-invariant (LTI) systems. In particular, we shall demonstrate that such systems are characterized in the time domain simply by their response to a unit sample sequence. We shall also demonstrate that any arbitrary input signal can be decomposed and represented as a weighted sum of unit sample sequences. As a consequence of the linearity and time-invariance properties of the system, the response of the system to any arbitrary input signal can be expressed in terms of the unit sample response of the system. The general form of the expression that relates the unit sample response of the system and the arbitrary input signal to the output signal, called the convolution sum or the convolution formula, is also derived. Thus we are able to determine the output of any linear, time-invariant system to any arbitrary input signal.

3.1 Techniques for the Analysis of Linear Systems

There are two basic methods for analyzing the behavior or response of a linear system to a given input signal. One method is based on the direct solution of the input–output equation for the system, which, in general, has the form

$$y(n) = F[y(n-1), y(n-2), \ldots, y(n-N), x(n), x(n-1), \ldots, x(n-M)]$$

where $F[\cdot]$ denotes some function of the quantities in brackets. Specifically, for an LTI system, we shall see later that the general form of the input–output relationship is

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{3.1}$$

where $\{a_k\}$ and $\{b_k\}$ are constant parameters that specify the system and are independent of $x(n)$ and $y(n)$. The input–output relationship in (3.1) is called a difference equation and represents one way to characterize the behavior of a discrete-time LTI system. The solution of (3.1) is the subject of Section 4.

The second method for analyzing the behavior of a linear system to a given input signal is first to decompose or resolve the input signal into a sum of elementary signals. The elementary signals are selected so that the response of the system to each signal component is easily determined. Then, using the linearity property of the system, the responses of the system to the elementary signals are added to obtain the total response of the system to the given input signal. This second method is the one described in this section.

To elaborate, suppose that the input signal $x(n)$ is resolved into a weighted sum of elementary signal components $\{x_k(n)\}$ so that

$$x(n) = \sum_{k} c_k x_k(n) \tag{3.2}$$

where the $\{c_k\}$ are the set of amplitudes (weighting coefficients) in the decomposition of the signal $x(n)$. Now suppose that the response of the system to the elementary signal component $x_k(n)$ is $y_k(n)$. Thus,

$$y_k(n) \equiv \mathcal{T}[x_k(n)] \tag{3.3}$$

assuming that the system is relaxed and that the response to $c_k x_k(n)$ is $c_k y_k(n)$, as a consequence of the scaling property of the linear system.

Finally, the total response to the input $x(n)$ is

$$\begin{aligned} y(n) = \mathcal{T}[x(n)] &= \mathcal{T}\left[\sum_{k} c_k x_k(n)\right] \\ &= \sum_{k} c_k \mathcal{T}[x_k(n)] \\ &= \sum_{k} c_k y_k(n) \end{aligned} \tag{3.4}$$

In (3.4) we used the additivity property of the linear system.

Although to a large extent, the choice of the elementary signals appears to be arbitrary, our selection is heavily dependent on the class of input signals that we wish to consider. If we place no restriction on the characteristics of the input signals,

their resolution into a weighted sum of unit sample (impulse) sequences proves to be mathematically convenient and completely general. On the other hand, if we restrict our attention to a subclass of input signals, there may be another set of elementary signals that is more convenient mathematically in the determination of the output. For example, if the input signal $x(n)$ is periodic with period N, a mathematically convenient set of elementary signals is the set of exponentials

$$x_k(n) = e^{j\omega_k n}, \qquad k = 0, 1, \ldots, N-1 \tag{3.5}$$

where the frequencies $\{\omega_k\}$ are harmonically related, that is,

$$\omega_k = \left(\frac{2\pi}{N}\right) k, \qquad k = 0, 1, \ldots, N-1 \tag{3.6}$$

The frequency $2\pi/N$ is called the fundamental frequency, and all higher-frequency components are multiples of the fundamental frequency component. This subclass of input signals is considered in more detail later.

For the resolution of the input signal into a weighted sum of unit sample sequences, we must first determine the response of the system to a unit sample sequence and then use the scaling and multiplicative properties of the linear system to determine the formula for the output given any arbitrary input. This development is described in detail as follows.

3.2 Resolution of a Discrete-Time Signal into Impulses

Suppose we have an arbitrary signal $x(n)$ that we wish to resolve into a sum of unit sample sequences. To utilize the notation established in the preceding section, we select the elementary signals $x_k(n)$ to be

$$x_k(n) = \delta(n-k) \tag{3.7}$$

where k represents the delay of the unit sample sequence. To handle an arbitrary signal $x(n)$ that may have nonzero values over an infinite duration, the set of unit impulses must also be infinite, to encompass the infinite number of delays.

Now suppose that we multiply the two sequences $x(n)$ and $\delta(n-k)$. Since $\delta(n-k)$ is zero everywhere except at $n = k$, where its value is unity, the result of this multiplication is another sequence that is zero everywhere except at $n = k$, where its value is $x(k)$, as illustrated in Fig. 3.1. Thus

$$x(n)\delta(n-k) = x(k)\delta(n-k) \tag{3.8}$$

is a sequence that is zero everywhere except at $n = k$, where its value is $x(k)$. If we repeat the multiplication of $x(n)$ with $\delta(n-m)$, where m is another delay ($m \neq k$), the result will be a sequence that is zero everywhere except at $n = m$, where its value is $x(m)$. Hence

$$x(n)\delta(n-m) = x(m)\delta(n-m) \tag{3.9}$$

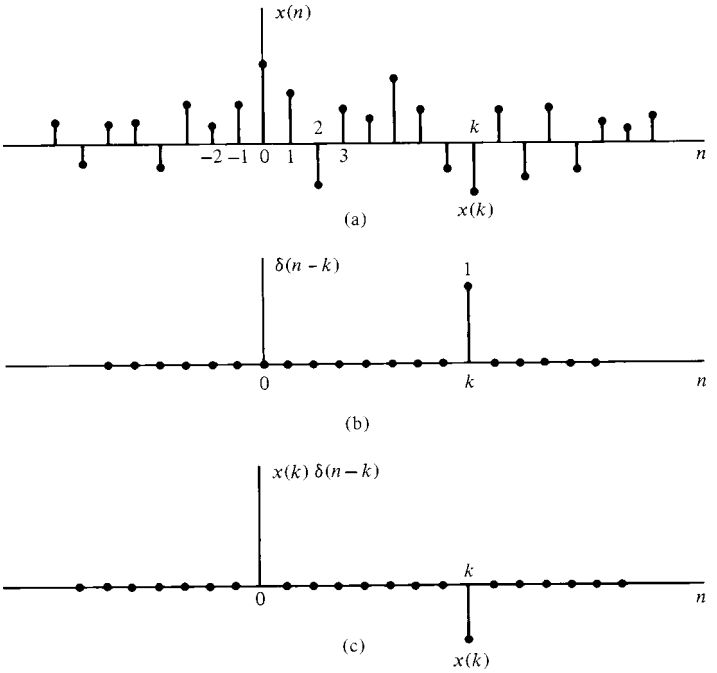


Figure 3.1 Multiplication of a signal $x(n)$ with a shifted unit sample sequence.

In other words, each multiplication of the signal $x(n)$ by a unit impulse at some delay k, [i.e., $\delta(n-k)$], in essence picks out the single value $x(k)$ of the signal $x(n)$ at the delay where the unit impulse is nonzero. Consequently, if we repeat this multiplication over all possible delays, $-\infty < k < \infty$, and sum all the product sequences, the result will be a sequence equal to the sequence $x(n)$, that is,

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \tag{3.10}$$

We emphasize that the right-hand side of (3.10) is the summation of an infinite number of scaled unit sample sequences where the unit sample sequence $\delta(n-k)$ has an amplitude value of $x(k)$. Thus the right-hand side of (3.10) gives the resolution or decomposition of any arbitrary signal $x(n)$ into a weighted (scaled) sum of shifted unit sample sequences.

EXAMPLE 3.1

Consider the special case of a finite-duration sequence given as

$$x(n) = \{2, \underset{\uparrow}{4}, 0, 3\}$$

Resolve the sequence $x(n)$ into a sum of weighted impulse sequences.

Solution. Since the sequence $x(n)$ is nonzero for the time instants $n = -1, 0, 2$, we need three impulses at delays $k = -1, 0,$ Following (3.10) we find that

$$x(n) = 2\delta(n+1) + 4\delta(n) + 3\delta(n-2)$$

3.3 Response of LTI Systems to Arbitrary Inputs: The Convolution Sum

Having resolved an arbitrary input signal $x(n)$ into a weighted sum of impulses, we are now ready to determine the response of any relaxed linear system to any input signal. First, we denote the response $y(n, k)$ of the system to the input unit sample sequence at $n = k$ by the special symbol $h(n, k)$, $-\infty < k < \infty$. That is,

$$y(n, k) \equiv h(n, k) = \mathcal{T}[\delta(n-k)] \tag{3.11}$$

In (3.11) we note that n is the time index and k is a parameter showing the location of the input impulse. If the impulse at the input is scaled by an amount $c_k \equiv x(k)$, the response of the system is the correspondingly scaled output, that is,

$$c_k h(n, k) = x(k)h(n, k) \tag{3.12}$$

Finally, if the input is the arbitrary signal $x(n)$ that is expressed as a sum of weighted impulses, that is,

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \tag{3.13}$$

then the response of the system to $x(n)$ is the corresponding sum of weighted outputs, that is,

$$\begin{aligned} y(n) = \mathcal{T}[x(n)] &= \mathcal{T}\left[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right] \\ &= \sum_{k=-\infty}^{\infty} x(k)\mathcal{T}[\delta(n-k)] \\ &= \sum_{k=-\infty}^{\infty} x(k)h(n, k) \end{aligned} \tag{3.14}$$

Clearly, (3.14) follows from the superposition property of linear systems, and is known as the *superposition summation*.

We note that (3.14) is an expression for the response of a linear system to any arbitrary input sequence $x(n)$. This expression is a function of both $x(n)$ and the responses $h(n, k)$ of the system to the unit impulses $\delta(n-k)$ for $-\infty < k < \infty$. In deriving (3.14) we used the linearity property of the system but not its time-invariance property. Thus the expression in (3.14) applies to any relaxed linear (time-variant) system.

If, in addition, the system is time invariant, the formula in (3.14) simplifies considerably. In fact, if the response of the LTI system to the unit sample sequence $\delta(n)$ is denoted as $h(n)$, that is,

$$h(n) \equiv \mathcal{T}[\delta(n)] \tag{3.15}$$

then by the time-invariance property, the response of the system to the delayed unit sample sequence $\delta(n-k)$ is

$$h(n-k) = \mathcal{T}[\delta(n-k)] \tag{3.16}$$

Consequently, the formula in (3.14) reduces to

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{3.17}$$

Now we observe that the relaxed LTI system is completely characterized by a single function $h(n)$, namely, its response to the unit sample sequence $\delta(n)$. In contrast, the general characterization of the output of a time-variant, linear system requires an infinite number of unit sample response functions, $h(n,k)$, one for each possible delay.

The formula in (3.17) that gives the response $y(n)$ of the LTI system as a function of the input signal $x(n)$ and the unit sample (impulse) response $h(n)$ is called a *convolution sum*. We say that the input $x(n)$ is convolved with the impulse response $h(n)$ to yield the output $y(n)$. We shall now explain the procedure for computing the response $y(n)$, both mathematically and graphically, given the input $x(n)$ and the impulse response $h(n)$ of the system.

Suppose that we wish to compute the output of the system at some time instant, say $n = n_0$. According to (3.17), the response at $n = n_0$ is given as

$$y(n_0) = \sum_{k=-\infty}^{\infty} x(k)h(n_0-k) \tag{3.18}$$

Our first observation is that the index in the summation is k, and hence both the input signal $x(k)$ and the impulse response $h(n_0-k)$ are functions of k. Second, we observe that the sequences $x(k)$ and $h(n_0-k)$ are multiplied together to form a product sequence. The output $y(n_0)$ is simply the sum over all values of the product sequence. The sequence $h(n_0-k)$ is obtained from $h(k)$ by, first, folding $h(k)$ about $k=0$ (the time origin), which results in the sequence $h(-k)$. The folded sequence is then shifted by n_0 to yield $h(n_0-k)$. To summarize, the process of computing the convolution between $x(k)$ and $h(k)$ involves the following four steps.

1. *Folding*. Fold $h(k)$ about $k=0$ to obtain $h(-k)$.
2. *Shifting*. Shift $h(-k)$ by n_0 to the right (left) if n_0 is positive (negative), to obtain $h(n_0-k)$.
3. *Multiplication*. Multiply $x(k)$ by $h(n_0-k)$ to obtain the product sequence $v_{n_0}(k) \equiv x(k)h(n_0-k)$.
4. *Summation*. Sum all the values of the product sequence $v_{n_0}(k)$ to obtain the value of the output at time $n = n_0$.

We note that this procedure results in the response of the system at a single time instant, say $n = n_0$. In general, we are interested in evaluating the response of the system over all time instants $-\infty < n < \infty$. Consequently, steps 2 through 4 in the summary must be repeated, for all possible time shifts $-\infty < n < \infty$.

In order to gain a better understanding of the procedure for evaluating the convolution sum, we shall demonstrate the process graphically. The graphs will aid us in explaining the four steps involved in the computation of the convolution sum.

EXAMPLE 3.2

The impulse response of a linear time-invariant system is

$$h(n) = \{1, \underset{\uparrow}{2}, 1, -1\} \tag{3.19}$$

Determine the response of the system to the input signal

$$x(n) = \{\underset{\uparrow}{1}, 2, 3, 1\} \tag{3.20}$$

Solution. We shall compute the convolution according to the formula (3.17), but we shall use graphs of the sequences to aid us in the computation. In Fig. 3.2(a) we illustrate the input signal sequence $x(k)$ and the impulse response $h(k)$ of the system, using k as the time index in order to be consistent with (3.17).

The first step in the computation of the convolution sum is to fold $h(k)$. The folded sequence $h(-k)$ is illustrated in Fig. 3.2(b). Now we can compute the output at $n = 0$, according to (3.17), which is

$$y(0) = \sum_{k=-\infty}^{\infty} x(k)h(-k) \tag{3.21}$$

Since the shift $n = 0$, we use $h(-k)$ directly without shifting it. The product sequence

$$v_0(k) \equiv x(k)h(-k) \tag{3.22}$$

is also shown in Fig. 3.2(b). Finally, the sum of all the terms in the product sequence yields

$$y(0) = \sum^{\infty} h = -\infty v_0(k) = 4$$

We continue the computation by evaluating the response of the system at $n = 1$. According to (3.17),

$$y(1) = \sum_{h=-\infty}^{\infty} x(k)h(1-k) \tag{3.23}$$

The sequence $h(1-k)$ is simply the folded sequence $h(-k)$ shifted to the right by one unit in time. This sequence is illustrated in Fig. 3.2(c). The product sequence

$$v_1(k) = x(k)h(1-k) \tag{3.24}$$

is also illustrated in Fig. 3.2(c). Finally, the sum of all the values in the product sequence yields

$$y(1) = \sum_{k=-\infty}^{\infty} v_1(k) = 8$$

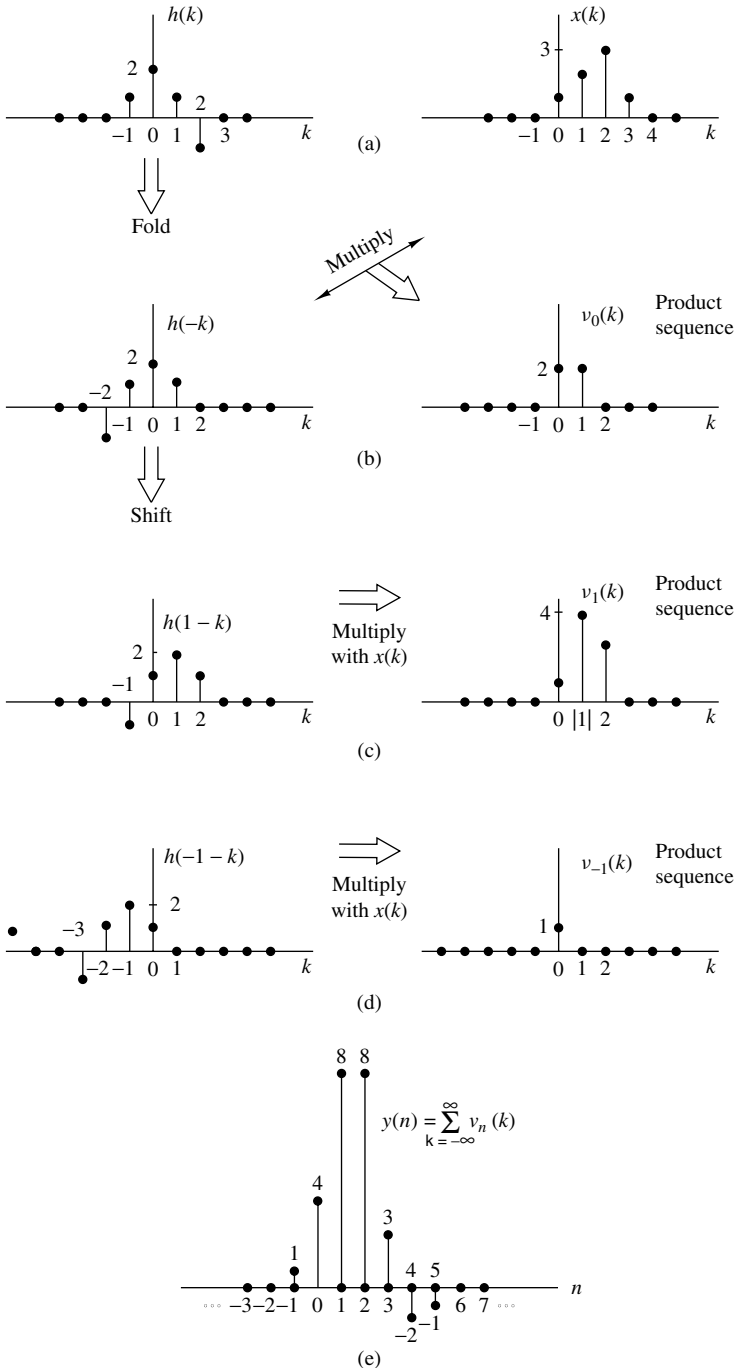


Figure 3.2 Graphical computation of convolution.

In a similar manner, we obtain $y(2)$ by shifting $h(-k)$ two units to the right, forming the product sequence $v_2(k) = x(k)h(2-k)$ and then summing all the terms in the product sequence obtaining $y(2) = 8$. By shifting $h(-k)$ farther to the right, multiplying the corresponding sequence, and summing over all the values of the resulting product sequences, we obtain $y(3) = 3$, $y(4) = -2$, $y(5) = -1$. For $n > 5$, we find that $y(n) = 0$ because the product sequences contain all zeros. Thus we have obtained the response $y(n)$ for $n > 0$.

Next we wish to evaluate $y(n)$ for $n < 0$. We begin with $n = -1$. Then

$$y(-1) = \sum_{k=-\infty}^{\infty} x(k)h(-1-k) \tag{3.25}$$

Now the sequence $h(-1-k)$ is simply the folded sequence $h(-k)$ shifted one time unit to the left. The resulting sequence is illustrated in Fig. 3.2(d). The corresponding product sequence is also shown in Fig. 3.2(d). Finally, summing over the values of the product sequence, we obtain

$$y(-1) = 1$$

From observation of the graphs of Fig. 3.2, it is clear that any further shifts of $h(-1-k)$ to the left always result in an all-zero product sequence, and hence

$$y(n) = 0 \qquad \text{for } n \leq -2$$

Now we have the entire response of the system for $-\infty < n < \infty$, which we summarize below as

$$y(n) = \{\ldots, 0, 0, 1, \underset{\uparrow}{4}, 8, 8, 3, -2, -1, 0, 0, \ldots\} \tag{3.26}$$

In Example 3.2 we illustrated the computation of the convolution sum, using graphs of the sequences to aid us in visualizing the steps involved in the computation procedure.

Before working out another example, we wish to show that the convolution operation is commutative in the sense that it is irrelevant which of the two sequences is folded and shifted. Indeed, if we begin with (3.17) and make a change in the variable of the summation, from k to m, by defining a new index $m = n - k$, then $k = n - m$ and (3.17) becomes

$$y(n) = \sum_{m=-\infty}^{\infty} x(n-m)h(m) \tag{3.27}$$

Since m is a dummy index, we may simply replace m by k so that

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \tag{3.28}$$

The expression in (3.28) involves leaving the impulse response $h(k)$ unaltered, while the input sequence is folded and shifted. Although the output $y(n)$ in (3.28)

is identical to (3.17), the product sequences in the two forms of the convolution formula are not identical. In fact, if we define the two product sequences as

$$v_n(k) = x(k)h(n-k)$$

$$w_n(k) = x(n-k)h(k)$$

it can be easily shown that

$$v_n(k) = w_n(n-k)$$

and therefore,

$$y(n) = \sum_{k=-\infty}^{\infty} v_n(k) = \sum_{k=-\infty}^{\infty} w_n(n-k)$$

since both sequences contain the same sample values in a different arrangement. The reader is encouraged to rework Example 3.2 using the convolution sum in (3.28).

EXAMPLE 3.3

Determine the output $y(n)$ of a relaxed linear time-invariant system with impulse response

$$h(n) = a^{nu}(n), |a|1$$

when the input is a unit step sequence, that is,

$$x(n) = u(n)$$

Solution. In this case both $h(n)$ and $x(n)$ are infinite-duration sequences. We use the form of the convolution formula given by (3.28) in which $x(k)$ is folded. The sequences $h(k)$, $x(k)$, and $x(-k)$ are shown in Fig. 3.3. The product sequences $v_0(k)$, $v_1(k)$, and $v_2(k)$ corresponding to $x(-k)\,h(k)$, $x(1-k)h(k)$, and $x(2-k)h(k)$ are illustrated in Fig. 3.3(c), (d), and (e), respectively. Thus we obtain the outputs

$$y(0) = 1$$

$$y(1) = 1 + a$$

$$y(2) = 1 + a + a^2$$

Clearly, for $n > 0$, the output is

$$\begin{aligned} y(n) &= 1 + a + a^2 + \cdots + a^n \\ &= \frac{1 - a^{n+1}}{1 - a} \end{aligned} \tag{3.29}$$

On the other hand, for $n < 0$, the product sequences consist of all zeros. Hence

$$y(n) = 0, \qquad n < 0$$

A graph of the output $y(n)$ is illustrated in Fig. 3.3(f) for the case $0 < a < 1$. Note the exponential rise in the output as a function of n. Since $|a| < 1$, the final value of the output as n approaches infinity is

$$y(\infty) = \lim_{n\to\infty} y(n) = \frac{1}{1-a} \tag{3.30}$$

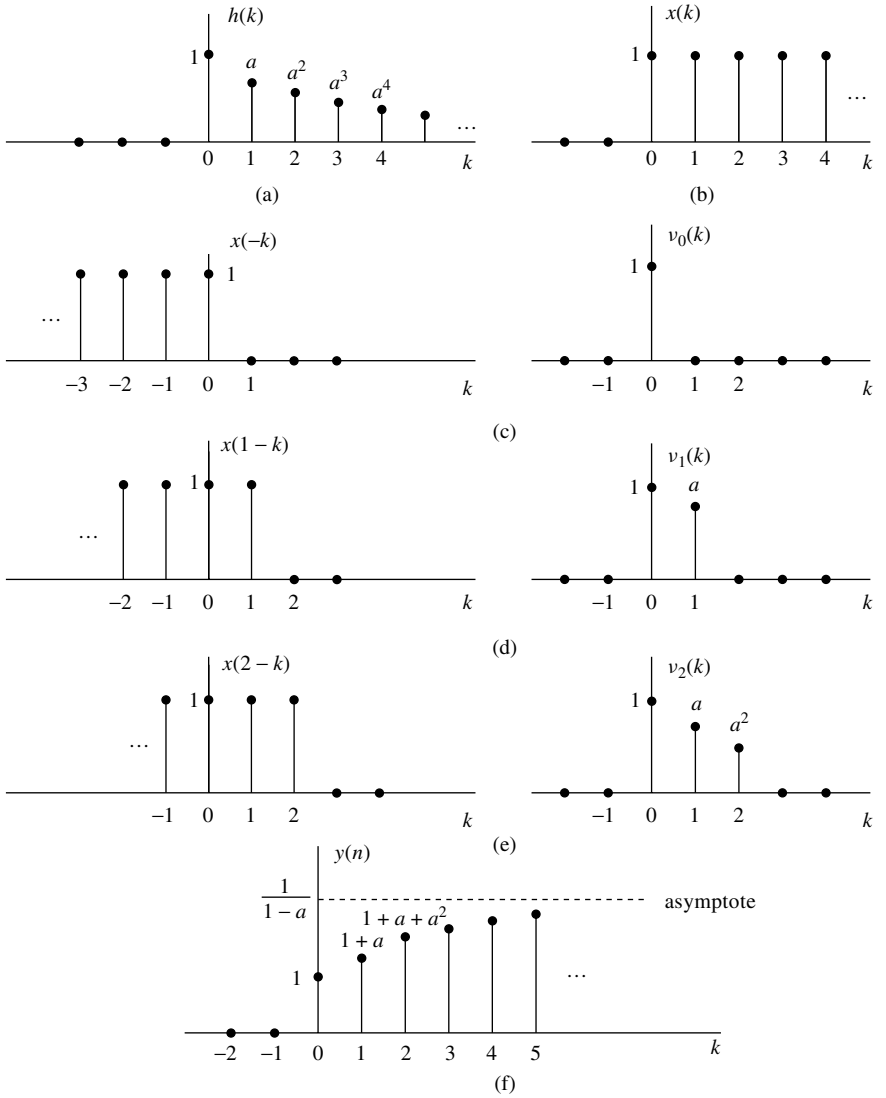


Figure 3.3 Graphical computation of convolution in Example 3.3.

To summarize, the convolution formula provides us with a means for computing the response of a relaxed, linear time-invariant system to any arbitrary input signal the input signal to the system, h(n) is the impulse response of the system, and y(n) is the output of the system in response to the input signal x(n). The evaluation of the convolution formula involves four operations, namely: folding either the impulse either h(−k) or x(−k), respectively, shifting the folded sequence by n units in time to yield either h(n−k) or x(n−k), multiplying the two sequences to yield the product Discrete-Time Signals and Systems Figure 3.3 Graphical computation of convolution in Example 3.3. x(n). It takes one of two equivalent forms, either (3.17) or (3.28), where x(n) is response as specified by (3.17) or the input sequence as specified by (3.28) to yield

sequence, either $x(k)h(n-k)$ or $x(n-k)h(k)$, and finally *summing* all the values in the product sequence to yield the output $y(n)$ of the system at time n. The folding operation is done only once. However, the other three operations are repeated for all possible shifts $-\infty < n < \infty$ in order to obtain $y(n)$ for $-\infty < n < \infty$.

3.4 Properties of Convolution and the Interconnection of LTI Systems

In this section we investigate some important properties of convolution and interpret these properties in terms of interconnecting linear time-invariant systems. We should stress that these properties hold for every input signal.

It is convenient to simplify the notation by using an asterisk to denote the convolution operation. Thus

$$y(n) = x(n) * h(n) \equiv \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{3.31}$$

In this notation the sequence following the asterisk [i.e., the impulse response $h(n)$] is folded and shifted. The input to the system is $x(n)$. On the other hand, we also showed that

$$y(n) = h(n) * x(n) \equiv \sum_{k=-\infty}^{\infty} h(k)x(n-k) \tag{3.32}$$

In this form of the convolution formula, it is the input signal that is folded. Alternatively, we may interpret this form of the convolution formula as resulting from an interchange of the roles of $x(n)$ and $h(n)$. In other words, we may regard $x(n)$ as the impulse response of the system and $h(n)$ as the excitation or input signal. Figure 3.4 illustrates this interpretation.

Identity and Shifting Properties. We also note that the unit sample sequence $\delta(n)$ is the identity element for convolution, that is

$$y(n) = x(n) * \delta(n) = x(n)$$

If we shift $\delta(n)$ by k, the convolution sequence is shifted also by k, that is

$$x(n) * \delta(n-k) = y(n-k) = x(n-k)$$

We can view convolution more abstractly as a mathematical operation between two signal sequences, say $x(n)$ and $h(n)$, that satisfies a number of properties. The property embodied in (3.31) and (3.32) is called the commutative law.

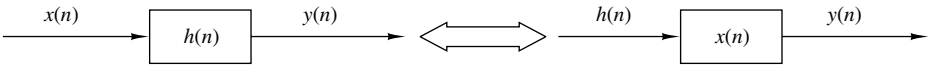


Figure 3.4 Interpretation of the commutative property of convolution.

Commutative law

$$x(n) * h(n) = h(n) * x(n) \tag{3.33}$$

Viewed mathematically, the convolution operation also satisfies the associative law, which can be stated as follows.

Associative law

$$[x(n) * h_1(n)] * h_2(n) = x(n) * [h_1(n) * h_2(n)] \tag{3.34}$$

From a physical point of view, we can interpret $x(n)$ as the input signal to a linear time-invariant system with impulse response $h_1(n)$. The output of this system, denoted as $y_1(n)$, becomes the input to a second linear time-invariant system with impulse response $h_2(n)$. Then the output is

$$\begin{aligned} y(n) &= y_1(n) * h_2(n) \\ &= [x(n) * h_1(n)] * h_2(n) \end{aligned}$$

which is precisely the left-hand side of (3.34). Thus the left-hand side of (3.34) corresponds to having two linear time-invariant systems in cascade. Now the righthhand side of (3.34) indicates that the input x(n) is applied to an equivalent system having an impulse response, say h(n), which is equal to the convolution of the two impulse responses. That is,

$$h(n) = h_1(n) * h_2(n)$$

and

$$y(n) = x(n) * h(n)$$

Furthermore, since the convolution operation satisfies the commutative property, one can interchange the order of the two systems with responses $h_1(n)$ and $h_2(n)$ without altering the overall input–output relationship. Figure 3.5 graphically illustrates the associative property.

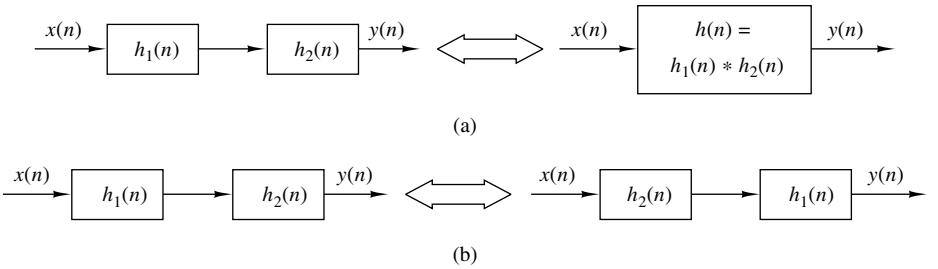


Figure 3.5 Implications of the associative (a) and the associative and commutative (b) properties of convolution.

EXAMPLE 3.4

Determine the impulse response for the cascade of two linear time-invariant systems having impulse responses

$$h_1(n) = (\frac{1}{2})^n u(n)$$

and

$$h_2(n) = (\frac{1}{4})^n u(n)$$

Solution. To determine the overall impulse response of the two systems in cascade, we simply convolve $h_1(n)$ with $h_2(n)$. Hence

$$h(n) = \sum_{k=-\infty}^{\infty} h_1(k)h_2(n-k)$$

where $h_2(n)$ is folded and shifted. We define the product sequence

$$v_n(k) = h_1(k)h_2(n-k)$$

$$= (\frac{1}{2})^k (\frac{1}{4})^{n-k}$$

which is nonzero for $k \geq 0$ and $n - k \geq 0$ or $n \geq k \geq 0$. On the other hand, for $n < 0$, we have $v_n(k) = 0$ for all k, and hence

$$h(n) = 0, n < 0$$

For $n \geq k \geq 0$, the sum of the values of the product sequence $v_n(k)$ over all k yields

$$h(n) = \sum_{k=0}^{n} (\frac{1}{2})^k (\frac{1}{4})^{n-k}$$

$$= (\frac{1}{4})^n \sum_{k=0}^{n} 2^k$$

$$= (\frac{1}{4})^n (2^{n+1} - 1)$$

$$= (\frac{1}{2})^n [2 - (\frac{1}{2})^n], \quad n \geq 0$$

The generalization of the associative law to more than two systems in cascade follows easily from the discussion given above. Thus if we have L linear time-invariant systems in cascade with impulse responses $h_1(n), h_2(n), \ldots, h_L(n)$, there is an equivalent linear time-invariant system having an impulse response that is equal to the $(L-1)$-fold convolution of the impulse responses. That is,

$$h(n) = h_1(n) * h_2(n) * \cdots * h_L(n) \tag{3.35}$$

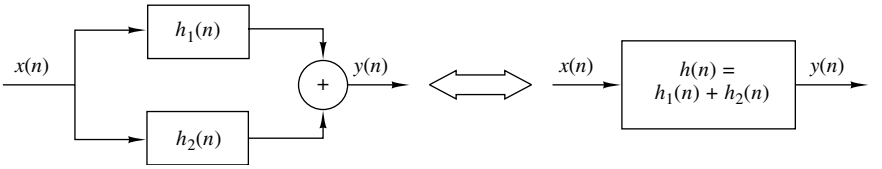


Figure 3.6 Interpretation of the distributive property of convolution: two LTI systems connected in parallel can be replaced by a single system with $h(n) = h_1(n) + h_2(n)$.

The commutative law implies that the order in which the convolutions are performed is immaterial. Conversely, any linear time-invariant system can be decomposed into a cascade interconnection of subsystems. A method for accomplishing the decomposition will be described later.

Another property that is satisfied by the convolution operation is the distributive law, which may be stated as follows.

Distributive law

$$x(n) * [h_1(n) + h_2(n)] = x(n) * h_1(n) + x(n) * h_2(n) \tag{3.36}$$

Interpreted physically, this law implies that if we have two linear time-invariant systems with impulse responses $h_1(n)$ and $h_2(n)$ excited by the same input signal $x(n)$, the sum of the two responses is identical to the response of an overall system with impulse response

$$h(n) = h_1(n) + h_2(n)$$

Thus the overall system is viewed as a parallel combination of the two linear time-invariant systems as illustrated in Fig. 3.6.

The generalization of (3.36) to more than two linear time-invariant systems in parallel follows easily by mathematical induction. Thus the interconnection of L linear time-invariant systems in parallel with impulse responses $h_1(n), h_2(n), \ldots, h_L(n)$ and excited by the same input $x(n)$ is equivalent to one overall system with impulse response

$$h(n) = \sum_{j=1}^{L} h_j(n) \tag{3.37}$$

Conversely, any linear time-invariant system can be decomposed into a parallel interconnection of subsystems.

3.5 Causal Linear Time-Invariant Systems

In Section 2.3 we defined a causal system as one whose output at time n depends only on present and past inputs but does not depend on future inputs. In other words, the output of the system at some time instant n, say $n = n_0$, depends only on values of $x(n)$ for $n \leq n_0$.

In the case of a linear time-invariant system, causality can be translated to a condition on the impulse response. To determine this relationship, let us consider a

linear time-invariant system having an output at time $n = n_0$ given by the convolution formula

$$y(n_0) = \sum_{k=-\infty}^{\infty} h(k)x(n_0 - k)$$

Suppose that we subdivide the sum into two sets of terms, one set involving present and past values of the input [i.e., $x(n)$ for $n \leq n_0$] and one set involving future values of the input [i.e., $x(n)$, $n > n_0$]. Thus we obtain

$$\begin{aligned} y(n_0) &= \sum_{k=0}^{\infty} h(k)x(n_0 - k) + \sum_{k=-\infty}^{-1} h(k)x(n_0 - k) \\ &= [h(0)x(n_0) + h(1)x(n_0 - 1) + h(2)x(n_0 - 2) + \cdots] \\ &\quad + [h(-1)x(n_0 + 1) + h(-2)x(n_0 + 2) + \cdots] \end{aligned}$$

We observe that the terms in the first sum involve $x(n_0)$, $x(n_0 - 1), \ldots$, which are the present and past values of the input signal. On the other hand, the terms in the second sum involve the input signal components $x(n_0+1)$, $x(n_0+2), \ldots$. Now, if the output at time $n = n_0$ is to depend only on the present and past inputs, then, clearly, the impulse response of the system must satisfy the condition

$$h(n) = 0, \qquad n < 0 \tag{3.38}$$

Since $h(n)$ is the response of the relaxed linear time-invariant system to a unit impulse applied at $n = 0$, it follows that $h(n) = 0$ for $n < 0$ is both a necessary and a sufficient condition for causality. Hence an *LTI system is causal if and only if its impulse response is zero for negative values of n*.

Since for a causal system, $h(n) = 0$ for $n < 0$, the limits on the summation of the convolution formula may be modified to reflect this restriction. Thus we have the two equivalent forms

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n - k) \tag{3.39}$$

$$= \sum_{k=-\infty}^{n} x(k)h(n - k) \tag{3.40}$$

As indicated previously, causality is required in any real-time signal processing application, since at any given time n we have no access to future values of the input signal. Only the present and past values of the input signal are available in computing the present output.

It is sometimes convenient to call a sequence that is zero for $n < 0$, a *causal sequence*, and one that is nonzero for $n < 0$ and $n > 0$, a *noncausal sequence*. This terminology means that such a sequence could be the unit sample response of a causal or a noncausal system, respectively.

If the input to a causal linear time-invariant system is a causal sequence [i.e., if $x(n) = 0$ for $n < 0$], the limits on the convolution formula are further restricted. In this case the two equivalent forms of the convolution formula become

$$y(n) = \sum_{k=0}^{n} h(k)x(n-k) \tag{3.41}$$

$$= \sum_{k=0}^{n} x(k)h(n-k) \tag{3.42}$$

We observe that in this case, the limits on the summations for the two alternative forms are identical, and the upper limit is growing with time. Clearly, the response of a causal system to a causal input sequence is causal, since $y(n) = 0$ for $n < 0$.

EXAMPLE 3.5

Determine the unit step response of the linear time-invariant system with impulse response

$$h(n) = a^n u(n), \qquad |a| < 1$$

Solution. Since the input signal is a unit step, which is a causal signal, and the system is also causal, we can use one of the special forms of the convolution formula, either (3.41) or (3.42). Since $x(n) = 1$ for $n \geq 0$, (3.41) is simpler to use. Because of the simplicity of this problem, one can skip the steps involved with sketching the folded and shifted sequences. Instead, we use direct substitution of the signals sequences in (3.41) and obtain

$$y(n) = \sum_{k=0}^{n} a^k$$

$$= \frac{1 - a^{n+1}}{1 - a}$$

and $y(n) = 0$ for $n < 0$. We note that this result is identical to that obtained in Example 3.3. In this simple case, however, we computed the convolution algebraically without resorting to the detailed procedure outlined previously.

3.6 Stability of Linear Time-Invariant Systems

As indicated previously, stability is an important property that must be considered in any practical implementation of a system. We defined an arbitrary relaxed system as BIBO stable if and only if its output sequence $y(n)$ is bounded for every bounded input $x(n)$.

If $x(n)$ is bounded, there exists a constant M_x such that

$$|x(n)| \leq M_x < \infty$$

Similarly, if the output is bounded, there exists a constant M_y such that

$$|y(n)| < M_y < \infty$$

for all n.

Now, given such a bounded input sequence $x(n)$ to a linear time-invariant system, let us investigate the implications of the definition of stability on the characteristics of the system. Toward this end, we work again with the convolution formula

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

If we take the absolute value of both sides of this equation, we obtain

$$|y(n)| = \left| \sum_{k=-\infty}^{\infty} h(k)x(n-k) \right|$$

Now, the absolute value of the sum of terms is always less than or equal to the sum of the absolute values of the terms. Hence

$$|y(n)| \leq \sum_{k=-\infty}^{\infty} |h(k)||x(n-k)|$$

If the input is bounded, there exists a finite number M_x such that $|x(n)| \leq M_x$. By substituting this upper bound for $x(n)$ in the equation above, we obtain

$$|y(n)| \leq M_x \sum_{k=-\infty}^{\infty} |h(k)|$$

From this expression we observe that the output is bounded if the impulse response of the system satisfies the condition

$$S_h \equiv \sum_{k=-\infty}^{\infty} |h(k)| < \infty \tag{3.43}$$

That is, *a linear time-invariant system is stable if its impulse response is absolutely summable*. This condition is not only sufficient but it is also necessary to ensure the stability of the system. Indeed, we shall show that if $S_h = \infty$, there is a bounded input for which the output is not bounded. We choose the bounded input

$$x(n) = \begin{cases} \dfrac{h^*(-n)}{|h(-n)|}, & h(n) \neq 0 \\ 0, & h(n) = 0 \end{cases}$$

where $h^*(n)$ is the complex conjugate of $h(n)$. It is sufficient to show that there is one value of n for which $y(n)$ is unbounded. For $n = 0$ we have

$$y(0) = \sum_{k=-\infty}^{\infty} x(-k)h(k) = \sum_{k=-\infty}^{\infty} \frac{|h(k)|^2}{|h(k)|} = S_h$$

Thus, if $S_h = \infty$, a bounded input produces an unbounded output since $y(0) = \infty$.

The condition in (3.43) implies that the impulse response $h(n)$ goes to zero as n approaches infinity. As a consequence, the output of the system goes to zero as n approaches infinity if the input is set to zero beyond $n > n_0$. To prove this, suppose that $|x(n)| < M_x$ for $n < n_0$ and $x(n) = 0$ for $n \geq n_0$. Then, at $n = n_0 + N$, the system output is

$$y(n_0 + N) = \sum_{k=-\infty}^{N-1} h(k)x(n_0 + N - k) + \sum_{k=N}^{\infty} h(k)x(n_0 + N - k)$$

But the first sum is zero since $x(n) = 0$ for $n \geq n_0$. For the remaining part, we take the absolute value of the output, which is

$$|y(n_0 + N)| = \left| \sum_{k=N}^{\infty} h(k)x(n_0 + N - k) \right| \leq \sum_{k=N}^{\infty} |h(k)||x(n_0 + N - k)|$$

$$\leq M_x \sum_{k=N}^{\infty} |h(k)|$$

Now, as N approaches infinity,

$$\lim_{N\to\infty} \sum_{k=N}^{\infty} |h(n)| = 0$$

and hence

$$\lim_{N\to\infty} |y(n_0 + N)| = 0$$

This result implies that any excitation at the input to the system, which is of a finite duration, produces an output that is "transient" in nature; that is, its amplitude decays with time and dies out eventually, when the system is stable.

EXAMPLE 3.6

Determine the range of values of the parameter a for which the linear time-invariant system with impulse response

$$h(n) = a^n u(n)$$

is stable.

Solution. First, we note that the system is causal. Consequently, the lower index on the summation in (3.43) begins with $k = 0$. Hence

$$\sum_{k=0}^{\infty} |a^k| = \sum_{k=0}^{\infty} |a|^k = 1 + |a| + |a|^2 + \cdots$$

Clearly, this geometric series converges to

$$\sum_{k=0}^{\infty} |a|^k = \frac{1}{1 - |a|}$$

provided that $|a| < 1$. Otherwise, it diverges. Therefore, the system is stable if $|a| < 1$. Otherwise, it is unstable. In effect, $h(n)$ must decay exponentially toward zero as n approaches infinity for the system to be stable.

EXAMPLE 3.7

Determine the range of values of a and b for which the linear time-invariant system with impulse response

$$h(n) = \begin{cases} a^n, & n \geq 0 \\ b^n, & n < 0 \end{cases}$$

is stable.

Solution. This system is noncasual. The condition on stability given by (3.43) yields

$$\sum_{n=-\infty}^{\infty} |h(n)| = \sum_{n=0}^{\infty} |a|^n + \sum_{n=-\infty}^{-1} |b|^n$$

From Example 3.6 we have already determined that the first sum converges for $|a| < 1$. The second sum can be manipulated as follows:

$$\sum_{n=-\infty}^{-1} |b|^n = \sum_{n=1}^{\infty} \frac{1}{|b|^n} = \frac{1}{|b|}\left(1 + \frac{1}{|b|} + \frac{1}{|b|^2} + \cdots\right)$$

$$= \beta(1 + \beta + \beta^2 + \cdots) = \frac{\beta}{1-\beta}$$

where $\beta = 1/|b|$ must be less than unity for the geometric series to converge. Consequently, the system is stable if both $|a| < 1$ and $|b| > 1$ are satisfied.

3.7 Systems with Finite-Duration and Infinite-Duration Impulse Response

Up to this point we have characterized a linear time-invariant system in terms of its impulse response $h(n)$. It is also convenient, however, to subdivide the class of linear time-invariant systems into two types, those that have a finite-duration impulse response (FIR) and those that have an infinite-duration impulse response (IIR). Thus an FIR system has an impulse response that is zero outside of some finite time interval. Without loss of generality, we focus our attention on causal FIR systems, so that

$$h(n) = 0, \qquad n < 0 \text{ and } n \geq M$$

The convolution formula for such a system reduces to

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

A useful interpretation of this expression is obtained by observing that the output at any time n is simply a weighted linear combination of the input signal samples $x(n)$, $x(n-1), \ldots, x(n-M+1)$. In other words, the system simply weights, by the values of the impulse response $h(k)$, $k = 0, 1, \ldots, M-1$, the most recent M signal samples

and sums the resulting M products. In effect, the system acts as a *window* that views only the most recent M input signal samples in forming the output. It neglects or simply "forgets" all prior input samples [i.e., $x(n-M)$, $x(n-M-1), \ldots$]. Thus we say that an FIR system has a finite memory of length-M samples.

In contrast, an IIR linear time-invariant system has an infinite-duration impulse response. Its output, based on the convolution formula, is

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

where causality has been assumed, although this assumption is not necessary. Now, the system output is a weighted [by the impulse response $h(k)$] linear combination of the input signal samples $x(n)$, $x(n-1)$, $x(n-2), \ldots$. Since this weighted sum involves the present and all the past input samples, we say that the system has an infinite memory.

We investigate the characteristics of FIR and IIR systems in more detail.

4 Discrete-Time Systems Described by Difference Equations

Up to this point we have treated linear and time-invariant systems that are characterized by their unit sample response $h(n)$. In turn, $h(n)$ allows us to determine the output $y(n)$ of the system for any given input sequence $x(n)$ by means of the convolution summation,

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \tag{4.1}$$

In general, then, we have shown that any linear time-invariant system is characterized by the input–output relationship in (4.1). Moreover, the convolution summation formula in (4.1) suggests a means for the realization of the system. In the case of FIR systems, such a realization involves additions, multiplications, and a finite number of memory locations. Consequently, an FIR system is readily implemented directly, as implied by the convolution summation.

If the system is IIR, however, its practical implementation as implied by convolution is clearly impossible, since it requires an infinite number of memory locations, multiplications, and additions. A question that naturally arises, then, is whether or not it is possible to realize IIR systems other than in the form suggested by the convolution summation. Fortunately, the answer is yes, there is a practical and computationally efficient means for implementing a family of IIR systems, as will be demonstrated in this section. Within the general class of IIR systems, this family of discrete-time systems is more conveniently described by difference equations. This family or subclass of IIR systems is very useful in a variety of practical applications, including the implementation of digital filters, and the modeling of physical phenomena and physical systems.

4.1 Recursive and Nonrecursive Discrete-Time Systems

As indicated above, the convolution summation formula expresses the output of the linear time-invariant system explicitly and only in terms of the input signal. However, this need not be the case, as is shown here. There are many systems where it is either necessary or desirable to express the output of the system not only in terms of the present and past values of the input, but also in terms of the already available past output values. The following problem illustrates this point.

Suppose that we wish to compute the *cumulative average* of a signal $x(n)$ in the interval $0 \leq k \leq n$, defined as

$$y(n) = \frac{1}{n+1} \sum_{k=0}^{n} x(k), \qquad n = 0, 1, \ldots \tag{4.2}$$

As implied by (4.2), the computation of $y(n)$ requires the storage of all the input samples $x(k)$ for $0 \leq k \leq n$. Since n is increasing, our memory requirements grow linearly with time.

Our intuition suggests, however, that $y(n)$ can be computed more efficiently by utilizing the previous output value $y(n-1)$. Indeed, by a simple algebraic rearrangement of (4.2), we obtain

$$\begin{aligned}(n+1)y(n) &= \sum_{k=0}^{n-1} x(k) + x(n) \\ &= ny(n-1) + x(n)\end{aligned}$$

and hence

$$y(n) = \frac{n}{n+1} y(n-1) + \frac{1}{n+1} x(n) \tag{4.3}$$

Now, the cumulative average $y(n)$ can be computed recursively by multiplying the previous output value $y(n-1)$ by $n/(n+1)$, multiplying the present input $x(n)$ by $1/(n+1)$, and adding the two products. Thus the computation of $y(n)$ by means of (4.3) requires two multiplications, one addition, and one memory location, as illustrated in Fig. 4.1. This is an example of a *recursive system*. In general, a system whose output $y(n)$ at time n depends on any number of past output values $y(n-1)$, $y(n-2), \ldots$ is called a recursive system.

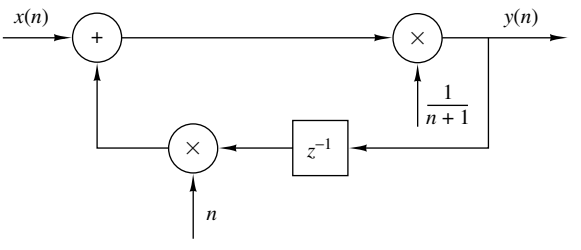


Figure 4.1
Realization of a recursive cumulative averaging system.

To determine the computation of the recursive system in (4.3) in more detail, suppose that we begin the process with $n = 0$ and proceed forward in time. Thus, according to (4.3), we obtain

$$y(0) = x(0)$$

$$y(1) = \frac{1}{2}y(0) + \frac{1}{2}x(1)$$

$$y(2) = \frac{2}{3}y(1) + \frac{1}{3}x(2)$$

and so on. If one grows fatigued with this computation and wishes to pass the problem to someone else at some time, say $n = n_0$, the only information that one needs to provide his or her successor is the past value $y(n_0 - 1)$ and the new input samples $x(n), x(n + 1), \ldots$. Thus the successor begins with

$$y(n_0) = \frac{n_0}{n_0 + 1}y(n_0 - 1) + \frac{1}{n_0 + 1}x(n_0)$$

and proceeds forward in time until some time, say $n = n_1$, when he or she becomes fatigued and passes the computational burden to someone else with the information on the value $y(n_1 - 1)$, and so on.

The point we wish to make in this discussion is that if one wishes to compute the response (in this case, the cumulative average) of the system (4.3) to an input signal $x(n)$ applied at $n = n_0$, we need the value $y(n_0 - 1)$ and the input samples $x(n)$ for $n \geq n_0$. The term $y(n_0 - 1)$ is called the *initial condition* for the system in (4.3) and contains all the information needed to determine the response of the system for $n \geq n_0$ to the input signal $x(n)$, independent of what has occurred in the past.

The following example illustrates the use of a (nonlinear) recursive system to compute the square root of a number.

EXAMPLE 4.1 Square-Root Algorithm

Many computers and calculators compute the square root of a positive number A, using the iterative algorithm

$$s_n = \frac{1}{2}\left(s_{n-1} + \frac{A}{s_{n-1}}\right), \qquad n = 0, 1, \ldots$$

where s_{n-1} is an initial guess (estimate) of $\sqrt{A}$. As the iteration converges we have $s_n \approx s_{n-1}$. Then it easily follows that $s_n \approx \sqrt{A}$.

Consider now the recursive system

$$y(n) = \frac{1}{2}\left[y(n - 1) + \frac{x(n)}{y(n - 1)}\right] \tag{4.4}$$

which is realized as in Fig. 4.2. If we excite this system with a step of amplitude A [i.e., $x(n) = Au(n)$] and use as an initial condition $y(-1)$ an estimate of $\sqrt{A}$, the response $y(n)$ of the system will tend toward $\sqrt{A}$ as n increases. Note that in contrast to the system (4.3), we do not need to specify exactly the initial condition. A rough estimate is sufficient for the proper performance of the system. For example, if we let $A = 2$ and $y(-1) = 1$, we obtain $y(0) = \frac{3}{2}$, $y(1) = 1.4166667$, $y(2) = 1.4142157$. Similarly, for $y(-1) = 1.5$, we have $y(0) = 1.416667$, $y(1) = 1.4142157$. Compare these values with the $\sqrt{2}$, which is approximately 1.4142136.

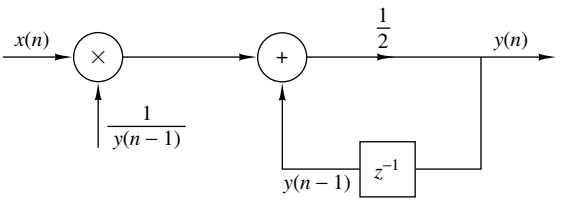


Figure 4.2
Realization of the square-root system.

We have now introduced two simple recursive systems, where the output $y(n)$ depends on the previous output value $y(n-1)$ and the current input $x(n)$. Both systems are causal. In general, we can formulate more complex causal recursive systems, in which the output $y(n)$ is a function of several past output values and present and past inputs. The system should have a finite number of delays or, equivalently, should require a finite number of storage locations to be practically implemented. Thus the output of a causal and practically realizable recursive system can be expressed in general as

$$y(n) = F[y(n-1), y(n-2), \ldots, y(n-N), x(n), x(n-1), \ldots, x(n-M)] \tag{4.5}$$

where $F[\cdot]$ denotes some function of its arguments. This is a recursive equation specifying a procedure for computing the system output in terms of previous values of the output and present and past inputs.

In contrast, if $y(n)$ depends only on the present and past inputs, then

$$y(n) = F[x(n), x(n-1), \ldots, x(n-M)] \tag{4.6}$$

Such a system is called *nonrecursive*. We hasten to add that the causal FIR systems described in Section 3.7 in terms of the convolution sum formula have the form of (2.4.6). Indeed, the convolution summation for a causal FIR system is

$$\begin{aligned} y(n) &= \sum_{k=0}^{M} h(k)x(n-k) \\ &= h(0)x(n) + h(1)x(n-1) + \cdots + h(M)x(n-M) \\ &= F[x(n), x(n-1), \ldots, x(n-M)] \end{aligned}$$

where the function $F[\cdot]$ is simply a linear weighted sum of present and past inputs and the impulse response values $h(n)$, $0 \le n \le M$, constitute the weighting coefficients. Consequently, the causal linear time-invariant FIR systems described by the convolution formula in Section 3.7 are nonrecursive. The basic differences between nonrecursive and recursive systems are illustrated in Fig. 4.3. A simple inspection of this figure reveals that the fundamental difference between these two systems is the feedback loop in the recursive system, which feeds back the output of the system into the input. This feedback loop contains a delay element. The presence of this delay is crucial for the realizability of the system, since the absence of this delay would force the system to compute $y(n)$ in terms of $y(n)$, which is not possible for discrete-time systems.

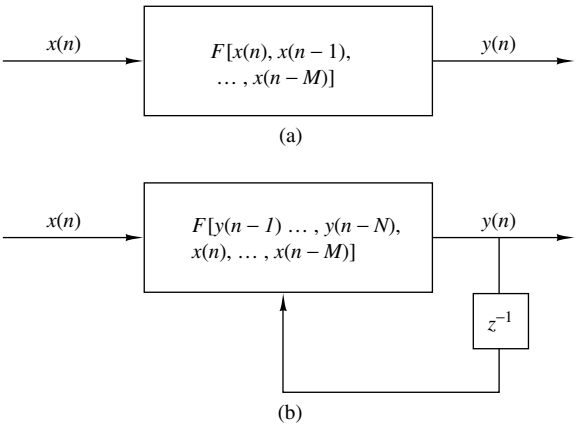


Figure 4.3
Basic form for a causal and realizable (a) nonrecursive and (b) recursive system.

The presence of the feedback loop or, equivalently, the recursive nature of (4.5) creates another important difference between recursive and nonrecursive systems. For example, suppose that we wish to compute the output $y(n_0)$ of a system when it is excited by an input applied at time $n = 0$. If the system is recursive, to compute $y(n_0)$, we first need to compute all the previous values $y(0), y(1), \ldots, y(n_0-1)$. In contrast, if the system is nonrecursive, we can compute the output $y(n_0)$ immediately without having $y(n_0-1), y(n_0-2), \ldots$. In conclusion, the output of a recursive system should be computed in order [i.e., $y(0), y(1), y(2), \ldots$], whereas for a nonrecursive system, the output can be computed in any order [i.e., $y(200), y(15), y(3), y(300)$, etc.]. This feature is desirable in some practical applications.

4.2 Linear Time-Invariant Systems Characterized by Constant-Coefficient Difference Equations

In Section 3 we treated linear time-invariant systems and characterized them in terms of their impulse responses. In this subsection we focus our attention on a family of linear time-invariant systems described by an input–output relation called a difference equation with constant coeffficients. Systems described by constant-coefficient linear difference equations are a subclass of the recursive and nonrecursive systems introduced in the preceding subsection. To bring out the important ideas, we begin by treating a simple recursive system described by a first-order difference equation.

Suppose that we have a recursive system with an input–output equation

$$y(n) = ay(n-1) + x(n) \tag{4.7}$$

where a is a constant. Figure 4.4 shows a block diagram realization of the system. In comparing this system with the cumulative averaging system described by the input–output equation (4.3), we observe that the system in (4.7) has a constant coefficient (independent of time), whereas the system described in (4.3) has timevariant coefficients. As we will show, (4.7) is an input–output equation for a linear time-invariant system, whereas (4.3) describes a linear time-variant system.

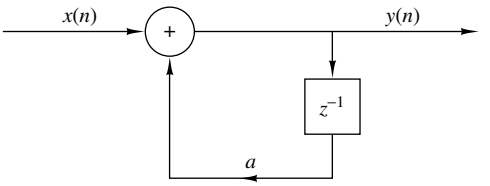


Figure 4.4
Block diagram realization of a simple recursive system.

Now, suppose that we apply an input signal $x(n)$ to the system for $n \geq 0$. We make no assumptions about the input signal for $n < 0$, but we do assume the existence of the initial condition $y(-1)$. Since (4.7) describes the system output implicitly, we must solve this equation to obtain an explicit expression for the system output. Suppose that we compute successive values of $y(n)$ for $n \geq 0$, beginning with $y(0)$. Thus

$$\begin{aligned}
y(0) &= ay(-1) + x(0) \\
y(1) &= ay(0) + x(1) = a^2 y(-1) + ax(0) + x(1) \\
y(2) &= ay(1) + x(2) = a^3 y(-1) + a^2 x(0) + ax(1) + x(2) \\
&\vdots \quad \vdots \\
y(n) &= ay(n-1) + x(n) \\
&= a^{n+1} y(-1) + a^n x(0) + a^{n-1} x(1) + \cdots + ax(n-1) + x(n)
\end{aligned}$$

or, more compactly,

$$y(n) = a^{n+1} y(-1) + \sum_{k=0}^{n} a^k x(n-k), \qquad n \geq 0 \tag{4.8}$$

The response $y(n)$ of the system as given by the right-hand side of (4.8) consists of two parts. The first part, which contains the term $y(-1)$, is a result of the initial condition $y(-1)$ of the system. The second part is the response of the system to the input signal $x(n)$.

If the system is initially relaxed at time $n = 0$, then its memory (i.e., the output of the delay) should be zero. Hence $y(-1) = 0$. Thus a recursive system is relaxed if it starts with zero initial conditions. Because the memory of the system describes, in some sense, its "state," we say that the system is at zero state and its corresponding output is called the *zero-state response*, and is denoted by $y_{zs}(n)$. Obviously, the zero-state response of the system (4.7) is given by

$$y_{zs}(n) = \sum_{k=0}^{n} a^k x(n-k), \qquad n \geq 0 \tag{4.9}$$

It is interesting to note that (4.9) is a convolution summation involving the input signal convolved with the impulse response

$$h(n) = a^n u(n) \tag{4.10}$$

We also observe that the system described by the first-order difference equation in (4.7) is causal. As a result, the lower limit on the convolution summation in (4.9) is $k = 0$. Furthermore, the condition $y(-1) = 0$ implies that the input signal can be assumed causal and hence the upper limit on the convolution summation in (4.9) is n, since $x(n-k) = 0$ for $k > n$. In effect, we have obtained the result that the relaxed recursive system described by the first-order difference equation in (4.7) is a linear time-invariant IIR system with impulse response given by (4.10).

Now, suppose that the system described by (4.7) is initially nonrelaxed [i.e., $y(-1) \neq 0$] and the input $x(n) = 0$ for all n. Then the output of the system with zero input is called the *zero-input response* or *natural response* and is denoted by $y_{zi}(n)$. From (4.7), with $x(n) = 0$ for $-\infty < n < \infty$, we obtain

$$y_{zi}(n) = a^{n+1} y(-1), \qquad n \geq 0 \tag{4.11}$$

We observe that a recursive system with nonzero initial condition is nonrelaxed in the sense that it can produce an output without being excited. Note that the zero-input response is due to the memory of the system.

To summarize, the zero-input response is obtained by setting the input signal to zero, making it independent of the input. It depends only on the nature of the system and the initial condition. Thus the zero-input response is a characteristic of the system itself, and it is also known as the *natural* or *free response* of the system. On the other hand, the zero-state response depends on the nature of the system and the input signal. Since this output is a response forced upon it by the input signal, it is usually called the *forced response* of the system. In general, the total response of the system can be expressed as $y(n) = y_{zi}(n) + y_{zs}(n)$.

The system described by the first-order difference equation in (4.7) is the simplest possible recursive system in the general class of recursive systems described by linear constant-coefficient difference equations. The general form for such an equation is

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{4.12}$$

or, equivalently,

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k), \qquad a_0 \equiv 1 \tag{4.13}$$

The integer N is called the *order* of the difference equation or the order of the system. The negative sign on the right-hand side of (4.12) is introduced as a matter of convenience to allow us to express the difference equation in (4.13) without any negative signs.

Equation (4.12) expresses the output of the system at time n directly as a weighted sum of past outputs $y(n-1),\ y(n-2), \ldots, y(n-N)$ as well as past and present input signals samples. We observe that in order to determine $y(n)$ for $n \geq 0$, we need the input $x(n)$ for all $n \geq 0$, and the initial conditions $y(-1)$,

$y(-2), \ldots, y(-N)$. In other words, the initial conditions summarize all that we need to know about the past history of the response of the system to compute the present and future outputs. The general solution of the N-order constant-coefficient difference equation is considered in the following subsection.

At this point we restate the properties of linearity, time invariance, and stability in the context of recursive systems described by linear constant-coefficient difference equations. As we have observed, a recursive system may be relaxed or nonrelaxed, depending on the initial conditions. Hence the definitions of these properties must take into account the presence of the initial conditions.

We begin with the definition of linearity. A system is linear if it satisfies the following three requirements:

1. The total response is equal to the sum of the zero-input and zero-state responses [i.e., $y(n) = y_{zi}(n) + y_{zs}(n)$].
2. The principle of superposition applies to the zero-state response (*zero-state linear*).
3. The principle of superposition applies to the zero-input response (*zero-input linear*).

A system that does not satisfy *all three* separate requirements is by definition nonlinear. Obviously, for a relaxed system, $y_{zi}(n) = 0$, and thus requirement 2, which is the definition of linearity given in Section 2.4, is sufficient.

We illustrate the application of these requirements by a simple example.

EXAMPLE 4.2

Determine if the recursive system defined by the difference equation

$$y(n) = ay(n-1) + x(n)$$

is linear.

Solution. By combining (4.9) and (4.11), we obtain (4.8), which can be expressed as

$$y(n) = y_{zi}(n) + y_{zs}(n)$$

Thus the first requirement for linearity is satisfied.

To check for the second requirement, let us assume that $x(n) = c_1x_1(n) + c_2x_2(n)$. Then (4.9) gives

$$\begin{aligned} y_{zs}(n) &= \sum_{k=0}^{n} a^k [c_1x_1(n-k) + c_2x_2(n-k)] \\ &= c_1 \sum_{k=0}^{n} a^k x_1(n-k) + c_2 \sum_{k=0}^{n} a^k x_2(n-k) \\ &= c_1 y_{zs}^{(1)}(n) + c_2 y_{zs}^{(2)}(n) \end{aligned}$$

Hence $y_{zs}(n)$ satisfies the principle of superposition, and thus the system is zero-state linear.

Now let us assume that $y(-1) = c_1 y_1(-1) + c_2 y_2(-1)$. From (4.11) we obtain

$$\begin{aligned} y_{zi}(n) &= a^{n+1}[c_1 y_1(-1) + c_2 y_2(-1)] \\ &= c_1 a^{n+1} y_1(-1) + c_2 a^{n+1} y_2(-1) \\ &= c_1 y_{zi}^{(1)}(n) + c_2 y_{zi}^{(2)}(n) \end{aligned}$$

Hence the system is zero-input linear.

Since the system satisfies all three conditions for linearity, it is linear.

Although it is somewhat tedious, the procedure used in Example 4.2 to demonstrate linearity for the system described by the first-order difference equation carries over directly to the general recursive systems described by the constant-coefficient difference equation given in (4.13). Hence, a recursive system described by the linear difference equation in (4.13) also satisfies all three conditions in the definition of linearity, and therefore it is linear.

The next question that arises is whether or not the causal linear system described by the linear constant-coefficient difference equation in (4.13) is time invariant. This is fairly easy, when dealing with systems described by explicit input–output mathematical relationships. Clearly, the system described by (4.13) is time invariant because the coefficients a_k and b_k are constants. On the other hand, if one or more of these coefficients depends on time, the system is time variant, since its properties change as a function of time. Thus we conclude that *the recursive system described by a linear constant-coefficient difference equation is linear and time invariant.*

The final issue is the stability of the recursive system described by the linear, constant-coefficient difference equation in (4.13). In Section 3.6 we introduced the concept of bounded input–bounded output (BIBO) stability for relaxed systems. For nonrelaxed systems that may be nonlinear, BIBO stability should be viewed with some care. However, in the case of a linear time-invariant recursive system described by the linear constant-coefficient difference equation in (4.13), it suffices to state that such a system is BIBO stable if and only if for every bounded input and every bounded initial condition, the total system response is bounded.

EXAMPLE 4.3

Determine if the linear time-invariant recursive system described by the difference equation given in (4.7) is stable.

Solution. Let us assume that the input signal $x(n)$ is bounded in amplitude, that is, $|x(n)| \leq M_x < \infty$ for all $n \geq 0$. From (4.8) we have

$$\begin{aligned} |y(n)| &\leq |a^{n+1} y(-1)| + \left| \sum_{k=0}^{n} a^k x(n-k) \right|, && n \geq 0 \\ &\leq |a|^{n+1} |y(-1)| + M_x \sum_{k=0}^{n} |a|^k, && n \geq 0 \\ &\leq |a|^{n+1} |y(-1)| + M_x \frac{1 - |a|^{n+1}}{1 - |a|} = M_y, && n \geq 0 \end{aligned}$$

If n is finite, the bound M_y is finite and the output is bounded independently of the value of a. However, as $n \to \infty$, the bound M_y remains finite only if $|a| < 1$ because $|a|^n \to 0$ as $n \to \infty$. Then $M_y = M_x/(1 - |a|)$.

Thus the system is stable only if $|a| < 1$.

For the simple first-order system in Example 4.3, we were able to express the condition for BIBO stability in terms of the system parameter a, namely $|a| < 1$. We should stress, however, that this task becomes more difficult for higher-order systems. Fortunately, other simple and more efficient techniques exist for investigating the stability of recursive systems.

4.3 Solution of Linear Constant-Coefficient Difference Equations

Given a linear constant-coefficient difference equation as the input–output relationship describing a linear time-invariant system, our objective in this subsection is to determine an explicit expression for the output $y(n)$. The method that is developed is termed the *direct method.* An alternative method based on the z-transform is beyond the scope of this chapter. The z-transform approach is called the *indirect method*.

Basically, the goal is to determine the output $y(n)$, $n \geq 0$, of the system given a specific input $x(n)$, $n \geq 0$, and a set of initial conditions. The direct solution method assumes that the total solution is the sum of two parts:

$$y(n) = y_h(n) + y_p(n)$$

The part $y_h(n)$ is known as the *homogeneous* or *complementary* solution, whereas $y_p(n)$ is called the *particular* solution.

The homogeneous solution of a difference equation. We begin the problem of solving the linear constant-coefficient difference equation given by (4.13) by obtaining first the solution to the *homogeneous difference equation*

$$\sum_{k=0}^{N} a_k y(n-k) = 0 \tag{4.14}$$

The procedure for solving a linear constant-coefficient difference equation directly is very similar to the procedure for solving a linear constant-coefficient differential equation. Basically, we assume that the solution is in the form of an exponential, that is,

$$y_h(n) = \lambda^n \tag{4.15}$$

where the subscript h on $y(n)$ is used to denote the solution to the homogeneous difference equation. If we substitute this assumed solution in (4.14), we obtain the polynomial equation

$$\sum_{k=0}^{N} a_k \lambda^{n-k} = 0$$

or

$$\lambda^{n-N}(\lambda^N + a_1\lambda^{N-1} + a_2\lambda^{N-2} + \cdots + a_{N-1}\lambda + a_N) = 0 \tag{4.16}$$

The polynomial in parentheses is called the *characteristic polynomial* of the system. In general, it has N roots, which we denote as $\lambda_1, \lambda_2, \ldots, \lambda_N$. The roots can be real or complex valued. In practice the coefficients $a_1, a_2, \ldots, a_N$ are usually real. Complex-valued roots occur as complex-conjugate pairs. Some of the N roots may be identical, in which case we have multiple-order roots.

For the moment, let us assume that the roots are distinct, that is, there are no multiple-order roots. Then the most general solution to the homogeneous difference equation in (4.14) is

$$y_h(n) = C_1\lambda_1^n + C_2\lambda_2^n + \cdots + C_N\lambda_N^n \tag{4.17}$$

where $C_1, C_2, \ldots, C_N$ are weighting coefficients.

These coefficients are determined from the initial conditions specified for the system. Since the input $x(n) = 0$, (4.17) can be used to obtain the *zero-input response* of the system. The following examples illustrate the procedure.

EXAMPLE 4.4

Determine the homogeneous solution of the system described by the first-order difference equation

$$y(n) + a_1 y(n-1) = x(n) \tag{4.18}$$

Solution. The assumed solution obtained by setting $x(n) = 0$ is

$$y_h(n) = \lambda^n$$

When we substitute this solution in (4.18), we obtain [with $x(n) = 0$]

$$\lambda^n + a_1\lambda^{n-1} = 0$$

$$\lambda^{n-1}(\lambda + a_1) = 0$$

$$\lambda = -a_1$$

Therefore, the solution to the homogeneous difference equation is

$$y_h(n) = C\lambda^n = C(-a_1)^n \tag{4.19}$$

The zero-input response of the system can be determined from (4.18) and (4.19). With $x(n) = 0$, (4.18) yields

$$y(0) = -a_1 y(-1)$$

On the other hand, from (4.19) we have

$$y_h(0) = C$$

and hence the zero-input response of the system is

$$y_{zi}(n) = (-a_1)^{n+1} y(-1), \qquad n \geq 0 \tag{4.20}$$

With $a = -a_1$, this result is consistent with (4.11) for the first-order system, which was obtained earlier by iteration of the difference equation.

EXAMPLE 4.5

Determine the zero-input response of the system described by the homogeneous second-order difference equation

$$y(n) - 3y(n-1) - 4y(n-2) = 0 \tag{21}$$

Solution. First we determine the solution to the homogeneous equation. We assume the solution to be the exponential

$$y_h(n) = \lambda^n$$

Upon substitution of this solution into (4.21), we obtain the characteristic equation

$$\lambda^n - 3\lambda^{n-1} - 4\lambda^{n-2} = 0$$

$$\lambda^{n-2}(\lambda^2 - 3\lambda - 4) = 0$$

Therefore, the roots are $\lambda = -1, 4$, and the general form of the solution to the homogeneous equation is

$$\begin{aligned} y_h(n) &= C_1\lambda_1^n + C_2\lambda_2^n \\ &= C_1(-1)^n + C_2(4)^n \end{aligned} \tag{4.22}$$

The zero-input response of the system can be obtained from the homogenous solution by evaluating the constants in (4.22), given the initial conditions $y(-1)$ and $y(-2)$. From the difference equation in (4.21) we have

$$y(0) = 3y(-1) + 4y(-2)$$

$$\begin{aligned} y(1) &= 3y(0) + 4y(-1) \\ &= 3[3y(-1) + 4y(-2)] + 4y(-1) \\ &= 13y(-1) + 12y(-2) \end{aligned}$$

On the other hand, from (4.22) we obtain

$$y(0) = C_1 + C_2$$

$$y(1) = -C_1 + 4C_2$$

By equating these two sets of relations, we have

$$C_1 + C_2 = 3y(-1) + 4y(-2)$$

$$-C_1 + 4C_2 = 13y(-1) + 12y(-2)$$

The solution of these two equations is

$$C_1 = -\frac{1}{5}y(-1) + \frac{4}{5}y(-2)$$

$$C_2 = \frac{16}{5}y(-1) + \frac{16}{5}y(-2)$$

Therefore, the zero-input response of the system is

$$\begin{aligned} y_{zi}(n) &= [-\frac{1}{5}y(-1) + \frac{4}{5}y(-2)](-1)^n \\ &\quad + [\frac{16}{5}y(-1) + \frac{16}{5}y(-2)](4)^n, \qquad n \geq 0 \end{aligned} \tag{4.23}$$

For example, if $y(-2) = 0$ and $y(-1) = 5$, then $C_1 = -1$, $C_2 = 16$, and hence

$$y_{zi}(n) = (-1)^{n+1} + (4)^{n+2}, \qquad n \geq 0$$

These examples illustrate the method for obtaining the homogeneous solution and the zero-input response of the system when the characteristic equation contains distinct roots. On the other hand, if the characteristic equation contains multiple roots, the form of the solution given in (4.17) must be modified. For example, if λ_1 is a root of multiplicity m, then (4.17) becomes

$$\begin{aligned} y_h(n) &= C_1\lambda_1^n + C_2 n\lambda_1^n + C_3 n^2\lambda_1^n + \cdots + C_m n^{m-1}\lambda_1^n \\ &\quad + C_{m+1}\lambda_{m+1}^n + \cdots + C_N\lambda_n \end{aligned} \tag{4.24}$$

The particular solution of the difference equation. The particular solution $y_p(n)$ is required to satisfy the difference equation (4.13) for the specific input signal $x(n)$, $n \geq 0$. In other words, $y_p(n)$ is any solution satisfying

$$\sum_{k=0}^{N} a_k y_p(n-k) = \sum_{k=0}^{M} b_k x(n-k), \qquad a_0 = 1 \tag{4.25}$$

To solve (4.25), we assume for $y_p(n)$, a form that depends on the form of the input $x(n)$. The following example illustrates the procedure.

EXAMPLE 4.6

Determine the particular solution of the first-order difference equation

$$y(n) + a_1 y(n-1) = x(n), \qquad |a_1| < 1 \tag{4.26}$$

when the input $x(n)$ is a unit step sequence, that is,

$$x(n) = u(n)$$

Solution. Since the input sequence $x(n)$ is a constant for $n \geq 0$, the form of the solution that we assume is also a constant. Hence the assumed solution of the difference equation to the forcing function $x(n)$, called the *particular solution* of the difference equation, is

$$y_p(n) = Ku(n)$$

where K is a scale factor determined so that (4.26) is satisfied. Upon substitution of this assumed solution into (4.26), we obtain

$$Ku(n) + a_1 Ku(n-1) = u(n)$$

To determine K, we must evaluate this equation for any $n \geq 1$, where none of the terms vanish. Thus

$$K + a_1 K = 1$$

$$K = \frac{1}{1 + a_1}$$

Therefore, the particular solution to the difference equation is

$$y_p(n) = \frac{1}{1 + a_1} u(n) \tag{4.27}$$

In this example, the input $x(n)$, $n \geq 0$, is a constant and the form assumed for the particular solution is also a constant. If $x(n)$ is an exponential, we would assume that the particular solution is also an exponential. If $x(n)$ were a sinusoid, then $y_p(n)$ would also be a sinusoid. Thus our assumed form for the particular solution takes the basic form of the signal $x(n)$. Table 1 provides the general form of the particular solution for several types of excitation.

EXAMPLE 4.7

Determine the particular solution of the difference equation

$$y(n) = \frac{5}{6} y(n-1) - \frac{1}{6} y(n-2) + x(n)$$

when the forcing function $x(n) = 2^n$, $n \geq 0$ and zero elsewhere.

TABLE 1 General Form of the Particular Solution for Several Types of Input Signals

Input Signal, $x(n)$	Particular Solution, $y_p(n)$
A (constant)	K
AM^n	KM^n
An^M	$K_0 n^M + K_1 n^{M-1} + \cdots + K_M$
$A^n n^M$	$A^n(K_0 n^M + K_1 n^{M-1} + \cdots + K_M)$
$\left\{ \begin{matrix} A\cos\omega_0 n \\ A\sin\omega_0 n \end{matrix} \right\}$	$K_1 \cos\omega_0 n + K_2 \sin\omega_0 n$

Solution. The form of the particular solution is

$$y_p(n) = K2^n, \qquad n \geq 0$$

Upon substitution of $y_p(n)$ into the difference equation, we obtain

$$K2^n u(n) = \frac{5}{6}K2^{n-1}u(n-1) - \frac{1}{6}K2^{n-2}u(n-2) + 2^n u(n)$$

To determine the value of K, we can evaluate this equation for any $n \geq 2$, where none of the terms vanish. Thus we obtain

$$4K = \frac{5}{6}(2K) - \frac{1}{6}K + 4$$

and hence $K = \frac{8}{5}$. Therefore, the particular solution is

$$y_p(n) = \frac{8}{5}2^n, \qquad n \geq 0$$

We have now demonstrated how to determine the two components of the solution to a difference equation with constant coefficients. These two components are the homogeneous solution and the particular solution. From these two components, we construct the total solution from which we can obtain the zero-state response.

The total solution of the difference equation. The linearity property of the linear constant-coefficient difference equation allows us to add the homogeneous solution and the particular solution in order to obtain the *total solution*. Thus

$$y(n) = y_h(n) + y_p(n)$$

The resultant sum $y(n)$ contains the constant parameters $\{C_i\}$ embodied in the homogeneous solution component $y_h(n)$. These constants can be determined to satisfy the initial conditions. The following example illustrates the procedure.

EXAMPLE 4.8

Determine the total solution $y(n)$, $n \geq 0$, to the difference equation

$$y(n) + a_1 y(n-1) = x(n) \tag{4.28}$$

when $x(n)$ is a unit step sequence [i.e., $x(n) = u(n)$] and $y(-1)$ is the initial condition.

Solution. From (4.19) of Example 4.4, the homogeneous solution is

$$y_h(n) = C(-a_1)^n$$

and from (4.26) of Example 4.6, the particular solution is

$$y_p(n) = \frac{1}{1 + a_1}u(n)$$

Consequently, the total solution is

$$y(n) = C(-a_1)^n + \frac{1}{1+a_1}, \qquad n \geq 0 \tag{4.29}$$

where the constant C is determined to satisfy the initial condition $y(-1)$.

In particular, suppose that we wish to obtain the zero-state response of the system described by the first-order difference equation in (4.28). Then we set $y(-1) = 0$. To evaluate C, we evaluate (4.28) at $n = 0$, obtaining

$$y(0) + a_1 y(-1) = 1$$

Hence,

$$y(0) = 1 - a_1 y(-1)$$

On the other hand, (4.29) evaluated at $n = 0$ yields

$$y(0) = C + \frac{1}{1+a_1}$$

By equating these two relations, we obtain

$$C + \frac{1}{1+a_1} = -a_1 y(-1) + 1$$

$$C = -a_1 y(-1) + \frac{a_1}{1+a_1}$$

Finally, if we substitute this value of C into (4.29), we obtain

$$\begin{aligned} y(n) &= (-a_1)^{n+1} y(-1) + \frac{1-(-a_1)^{n+1}}{1+a_1}, \qquad n \geq 0 \\ &= y_{zi}(n) + y_{zs}(n) \end{aligned} \tag{4.30}$$

We observe that the system response as given by (4.30) is consistent with the response $y(n)$ given in (4.8) for the first-order system (with $a = -a)_1$, which was obtained by solving the difference equation iteratively. Furthermore, we note that the value of the constant C depends both on the initial condition $y(-1)$ and on the excitation function. Consequently, the value of C influences both the zero-input response and the zero-state response.

We further observe that the particular solution to the difference equation can be obtained from the zero-state response of the system. Indeed, if $|a_1| < 1$, which is the condition for stability of the system, as will be shown in Section 4.4, the limiting value of $y_{zs}(n)$ as n approaches infinity is the particular solution, that is,

$$y_p(n) = \lim_{n\to\infty} y_{zs}(n) = \frac{1}{1+a_1}$$

Since this component of the system response does not go to zero as n approaches infinity, it is usually called the *steady-state response* of the system. This response persists as long as the input persists. The component that dies out as n approaches infinity is called the *transient response* of the system.

The following example illustrates the evaluation of the total solution for a second-order recursive system.

EXAMPLE 4.9

Determine the response $y(n)$, $n \geq 0$, of the system described by the second-order difference equation

$$y(n) - 3y(n-1) - 4y(n-2) = x(n) + 2x(n-1) \tag{4.31}$$

when the input sequence is

$$x(n) = 4^n u(n)$$

Solution. We have already determined the solution to the homogeneous difference equation for this system in Example 4.5. From (4.22) we have

$$y_h(n) = C_1(-1)^n + C_2(4)^n \tag{4.32}$$

The particular solution to (4.31) is assumed to be an exponential sequence of the same form as $x(n)$. Normally, we could assume a solution of the form

$$y_p(n) = K(4)^n u(n)$$

However, we observe that $y_p(n)$ is already contained in the homogeneous solution, so that this particular solution is redundant. Instead, we select the particular solution to be linearly independent of the terms contained in the homogeneous solution. In fact, we treat this situation in the same manner as we have already treated multiple roots in the characteristic equation. Thus we assume that

$$y_p(n) = Kn(4)^n u(n) \tag{4.33}$$

Upon substitution of (4.33) into (4.31), we obtain

$$Kn(4)^n u(n) - 3K(n-1)(4)^{n-1}u(n-1) - 4K(n-2)(4)^{n-2}u(n-2) = (4)^n u(n) + 2(4)^{n-1}u(n-1)$$

To determine K, we evaluate this equation for any $n \geq 2$, where none of the unit step terms vanish. To simplify the arithmetic, we select $n = 2$, from which we obtain $K = \frac{6}{5}$. Therefore,

$$y_p(n) = \frac{6}{5}n(4)^n u(n) \tag{4.34}$$

The total solution to the difference equation is obtained by adding (4.32) to (4.34). Thus

$$y(n) = C_1(-1)^n + C_2(4)^n + \frac{6}{5}n(4)^n, \qquad n \geq 0 \tag{4.35}$$

where the constants C_1 and C_2 are determined such that the initial conditions are satisfied. To accomplish this, we return to (4.31), from which we obtain

$$\begin{aligned} y(0) &= 3y(-1) + 4y(-2) + 1 \\ y(1) &= 3y(0) + 4y(-1) + 6 \\ &= 13y(-1) + 12y(-2) + 9 \end{aligned}$$

On the other hand, (4.35) evaluated at $n = 0$ and $n = 1$ yields

$$\begin{aligned} y(0) &= C_1 + C_2 \\ y(1) &= -C_1 + 4C_2 + \frac{24}{5} \end{aligned}$$

We can now equate these two sets of relations to obtain C_1 and C_2. In so doing, we have the response due to initial conditions $y(-1)$ and $y(-2)$ (the zero-input response), and the zero-state response.

Since we have already solved for the zero-input response in Example 4.5, we can simplify the computations above by setting $y(-1) = y(-2) = 0$. Then we have

$$C_1 + C_2 = 1$$

$$-C_1 + 4C_2 + \frac{24}{5} = 9$$

Hence $C_1 = -\frac{1}{25}$ and $C_2 = \frac{26}{25}$. Finally, we have the zero-state response to the forcing function $x(n) = (4)^n u(n)$ in the form

$$y_{zs}(n) = -\frac{1}{25}(-1)^n + \frac{26}{25}(4)^n + \frac{6}{5}n(4)^n, \qquad n \geq 0 \tag{4.36}$$

The total response of the system, which includes the response to arbitrary initial conditions, is the sum of (4.23) and (4.36).

4.4 The Impulse Response of a Linear Time-Invariant Recursive System

The impulse response of a linear time-invariant system was previously defined as the response of the system to a unit sample excitation [i.e., $x(n) = \delta(n)$]. In the case of a recursive system, $h(n)$ is simply equal to the zero-state response of the system when the input $x(n) = \delta(n)$ and the system is initially relaxed.

For example, in the simple first-order recursive system given in (4.7), the zero-state response given in (4.8), is

$$y_{zs}(n) = \sum_{k=0}^{n} a^k x(n-k) \tag{4.37}$$

When $x(n) = \delta(n)$ is substituted into (4.37), we obtain

$$y_{zs}(n) = \sum_{k=0}^{n} a^k \delta(n-k)$$

$$= a^n, \qquad n \geq 0$$

Hence the impulse response of the first-order recursive system described by (4.7) is

$$h(n) = a^n u(n) \tag{4.38}$$

as indicated in Section 4.2.

In the general case of an arbitrary, linear time-invariant recursive system, the zero-state response expressed in terms of the convolution summation is

$$y_{zs}(n) = \sum_{k=0}^{n} h(k)x(n-k), \qquad n \geq 0 \tag{4.39}$$

When the input is an impulse [i.e., $x(n) = \delta(n)$], (4.39) reduces to

$$y_{zs}(n) = h(n) \tag{4.40}$$

Now, let us consider the problem of determining the impulse response $h(n)$ given a linear constant-coefficient difference equation description of the system. In terms of our discussion in the preceding subsection, we have established the fact that the total response of the system to any excitation function consists of the sum of two solutions of the difference equation: the solution to the homogeneous equation plus the particular solution to the excitation function. In the case where the excitation is an impulse, the particular solution is zero, since $x(n) = 0$ for $n > 0$, that is,

$$y_p(n) = 0$$

Consequently, the response of the system to an impulse consists only of the solution to the homogeneous equation, with the $\{C_k\}$ parameters evaluated to satisfy the initial conditions dictated by the impulse. The following example illustrates the procedure for obtaining $h(n)$ given the difference equation for the system.

EXAMPLE 4.10

Determine the impulse response $h(n)$ for the system described by the second-order difference equation

$$y(n) - 3y(n-1) - 4y(n-2) = x(n) + 2x(n-1) \tag{4.41}$$

Solution. We have already determined in Example 4.5 that the solution to the homogeneous difference equation for this system is

$$y_h(n) = C_1(-1)^n + C_2(4)^n, \qquad n \geq 0 \tag{4.42}$$

Since the particular solution is zero when $x(n) = \delta(n)$, the impulse response of the system is simply given by (4.42), where C_1 and C_2 must be evaluated to satisfy (4.41).

For $n = 0$ and $n = 1$, (4.41) yields

$$y(0) = 1$$

$$y(1) = 3y(0) + 2 = 5$$

where we have imposed the conditions $y(-1) = y(-2) = 0$, since the system must be relaxed. On the other hand, (4.42) evaluated at $n = 0$ and $n = 1$ yields

$$y(0) = C_1 + C_2$$

$$y(1) = -C_1 + 4C_2$$

By solving these two sets of equations for C_1 and C_2, we obtain

$$C_1 = -\frac{1}{5}, \qquad C_2 = \frac{6}{5}$$

Therefore, the impulse response of the system is

$$h(n) = \left[-\frac{1}{5}(-1)^n + \frac{6}{5}(4)^n\right] u(n)$$

When the system is described by an Nth-order linear difference equation of the type given in (4.13), the solution of the homogeneous equation is

$$y_h(n) = \sum_{k=1}^{N} C_k \lambda^n k$$

when the roots $\{\lambda_k\}$ of the characteristic polynomial are distinct. Hence the impulse response of the system is identical in form, that is,

$$h(n) = \sum_{k=1}^{N} C_k \lambda_k^n \tag{4.43}$$

where the parameters $\{C_k\}$ are determined by setting the initial conditions $y(-1) = \cdots = y(-N) = 0$.

This form of $h(n)$ allows us to easily relate the stability of a system, described by an Nth-order difference equation, to the values of the roots of the characteristic polynomial. Indeed, since BIBO stability requires that the impulse response be absolutely summable, then, for a causal system, we have

$$\sum_{n=0}^{\infty} |h(n)| = \sum_{n=0}^{\infty} \left| \sum_{k=1}^{N} C_k \lambda^n k \right| \leq \sum_{k=1}^{N} |C_k| \sum_{n=0}^{\infty} |\lambda_k|^n$$

Now if $|\lambda_k| < 1$ for all k, then

$$\sum_{n=0}^{\infty} |\lambda_k|^n < \infty$$

and hence

$$\sum_{n=0}^{\infty} |h(n)| < \infty$$

On the other hand, if one or more of the $|\lambda_k| \geq 1$, $h(n)$ is no longer absolutely summable, and consequently, the system is unstable. Therefore, a necessary and sufficient condition for the stability of a causal IIR system described by a linear constant-coefficient difference equation is that all roots of the characteristic polynomial be less than unity in magnitude. The reader may verify that this condition carries over to the case where the system has roots of multiplicity m.

Finally we note that any recursive system described by a linear constant-coefficient difference equation is an IIR system. The converse is not true, however. That is, not every linear time-invariant IIR system can be described by a linear constant-coefficient difference equation. In other words, recursive systems described by linear constant-coefficient difference equations are a subclass of linear time-invariant IIR systems.

5 Implementation of Discrete-Time Systems

Our treatment of discrete-time systems has been focused on the time-domain characterization and analysis of linear time-invariant systems described by constant-coefficient linear difference equations. Additional analytical methods are developed, where we characterize and analyze LTI systems in the frequency domain. Two other important topics that will be treated later are the design and implementation of these systems.

In practice, system design and implementation are usually treated jointly rather than separately. Often, the system design is driven by the method of implementation and by implementation constraints, such as cost, hardware limitations, size limitations, and power requirements. At this point, we have not as yet developed the necessary analysis and design tools to treat such complex issues. However, we have developed sufficient background to consider some basic implementation methods for realizations of LTI systems described by linear constant-coefficient difference equations.

5.1 Structures for the Realization of Linear Time-Invariant Systems

In this subsection we describe structures for the realization of systems described by linear constant-coefficient difference equations.

As a beginning, let us consider the first-order system

$$y(n) = -a_1 y(n-1) + b_0 x(n) + b_1 x(n-1) \tag{5.1}$$

which is realized as in Fig. 5.1(a). This realization uses separate delays (memory) for both the input and output signal samples and it is called a *direct form I structure*. Note that this system can be viewed as two linear time-invariant systems in cascade. The first is a nonrecursive system described by the equation

$$v(n) = b_0 x(n) + b_1 x(n-1) \tag{5.2}$$

whereas the second is a recursive system described by the equation

$$y(n) = -a_1 y(n-1) + v(n) \tag{5.3}$$

However, as we have seen in Section 3.4, if we interchange the order of the cascaded linear time-invariant systems, the overall system response remains the same. Thus if we interchange the order of the recursive and nonrecursive systems, we obtain an alternative structure for the realization of the system described by (5.1). The resulting system is shown in Fig. 5.1(b). From this figure we obtain the two difference equations

$$w(n) = -a_1 w(n-1) + x(n) \tag{5.4}$$

$$y(n) = b_0 w(n) + b_1 w(n-1) \tag{5.5}$$

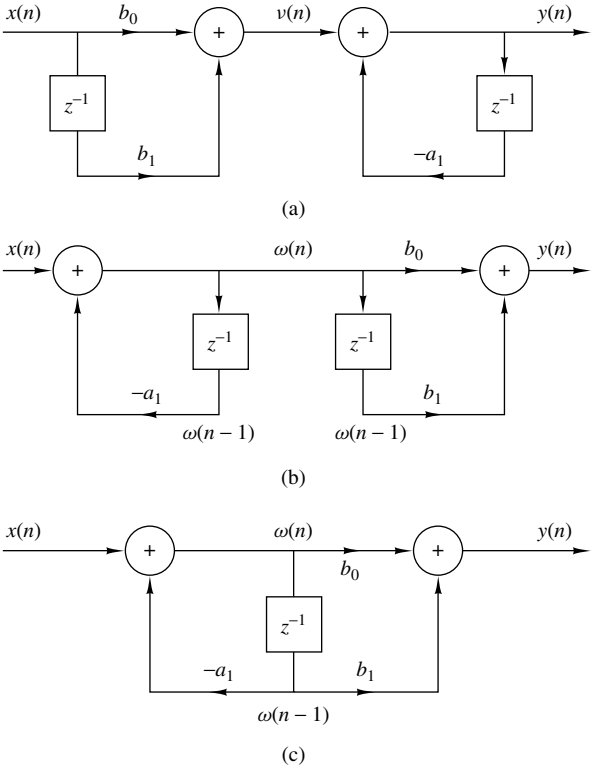


Figure 5.1
Steps in converting from the direct form I realization in (a) to the direct form II realization in (c).

which provide an alternative algorithm for computing the output of the system described by the single difference equation given in (5.1). In other words, the two difference equations (5.4) and (5.5) are equivalent to the single difference equation (5.1).

A close observation of Fig. 5.1 reveals that the two delay elements contain the same input $w(n)$ and hence the same output $w(n-1)$. Consequently, these two elements can be merged into one delay, as shown in Fig. 5.1(c). In contrast to the direct form I structure, this new realization requires only one delay for the auxiliary quantity $w(n)$, and hence it is more efficient in terms of memory requirements. It is called the *direct form II structure* and it is used extensively in practical applications.

These structures can readily be generalized for the general linear time-invariant recursive system described by the difference equation

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{5.6}$$

Figure 5.2 illustrates the direct form I structure for this system. This structure requires $M + N$ delays and $N + M + 1$ multiplications. It can be viewed as the

cascade of a nonrecursive system

$$v(n) = \sum_{k=0}^{M} b_k x(n-k) \tag{5.7}$$

and a recursive system

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + v(n) \tag{5.8}$$

By reversing the order of these two systems, as was previously done for the first-order system, we obtain the direct form II structure shown in Fig. 5.3 for $N > M$. This structure is the cascade of a recursive system

$$w(n) = -\sum_{k=1}^{N} a_k w(n-k) + x(n) \tag{5.9}$$

followed by a nonrecursive system

$$y(n) = \sum_{k=0}^{M} b_k w(n-k) \tag{5.10}$$

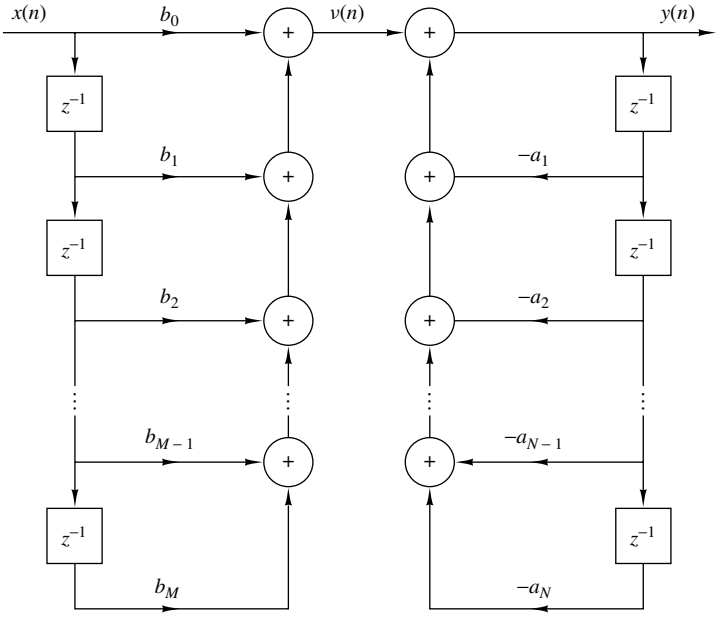


Figure 5.2 Direct form I structure of the system described by (5.6).

We observe that if $N \geq M$, this structure requires a number of delays equal to the order N of the system. However, if $M > N$, the required memory is specified by M. Figure 5.3 can easily be modified to handle this case. Thus the direct form II structure requires $M + N + 1$ multiplications and $\max\{M, N\}$ delays. Because it requires the minimum number of delays for the realization of the system described by (5.6), it is sometimes called a *canonic form*.

A special case of (5.6) occurs if we set the system parameters $a_k = 0$, $k = 1, \ldots, N$. Then the input–output relationship for the system reduces to

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) \tag{5.11}$$

which is a nonrecursive linear time-invariant system. This system views only the most recent $M + 1$ input signal samples and, prior to addition, weights each sample by the appropriate coefficient b_k from the set $\{b_k\}$. In other words, the system output is basically a *weighted moving average* of the input signal. For this reason it is sometimes called a *moving average (MA) system*. Such a system is an FIR system

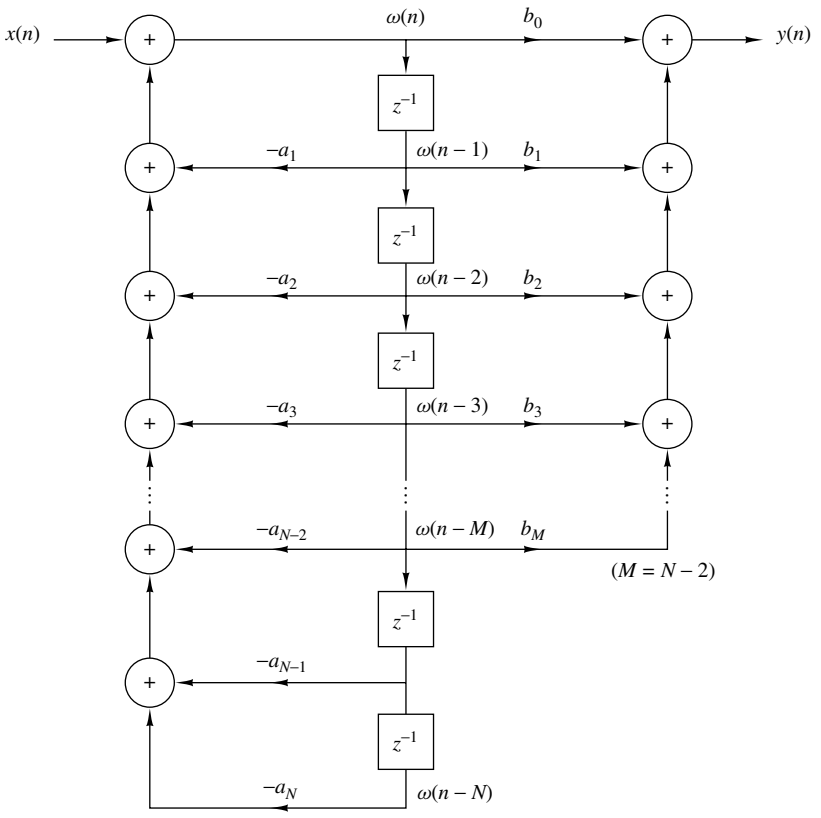


Figure 5.3 Direct form II structure for the system described by (5.6).

with an impulse response $h(k)$ equal to the coefficients b_k, that is,

$$h(k) = \begin{cases} b_k, & 0 \le k \le M \\ 0, & \text{otherwise} \end{cases} \tag{5.12}$$

If we return to (5.6) and set $M = 0$, the general linear time-invariant system reduces to a "purely recursive" system described by the difference equation

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + b_0 x(n) \tag{5.13}$$

In this case the system output is a weighted linear combination of N past outputs and the present input.

Linear time-invariant systems described by a second-order difference equation are an important subclass of the more general systems described by (5.6) or (5.10) or (5.13). The reason for their importance will be explained later when we discuss quantization effects. Suffice to say at this point that second-order systems are usually used as basic building blocks for realizing higher-order systems.

The most general second-order system is described by the difference equation

$$\begin{aligned} y(n) = &- a_1 y(n-1) - a_2 y(n-2) + b_0 x(n) \\ &+ b_1 x(n-1) + b_2 x(n-2) \end{aligned} \tag{5.14}$$

which is obtained from (5.6) by setting $N = 2$ and $M = 2$. The direct form II structure for realizing this system is shown in Fig. 5.4(a). If we set $a_1 = a_2 = 0$, then (5.14) reduces to

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \tag{5.15}$$

which is a special case of the FIR system described by (5.11). The structure for realizing this system is shown in Fig. 5.4(b). Finally, if we set $b_1 = b_2 = 0$ in (5.14), we obtain the purely recursive second-order system described by the difference equation

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_0 x(n) \tag{5.16}$$

which is a special case of (5.13). The structure for realizing this system is shown in Fig. 5.4(c).

5.2 Recursive and Nonrecursive Realizations of FIR Systems

We have already made the distinction between FIR and IIR systems, based on whether the impulse response $h(n)$ of the system has a finite duration, or an infinite duration. We have also made the distinction between recursive and nonrecursive systems. Basically, a causal recursive system is described by an input–output equation of the form

$$y(n) = F[y(n-1), \ldots, y(n-N), x(n), \ldots, x(n-M)] \tag{5.17}$$

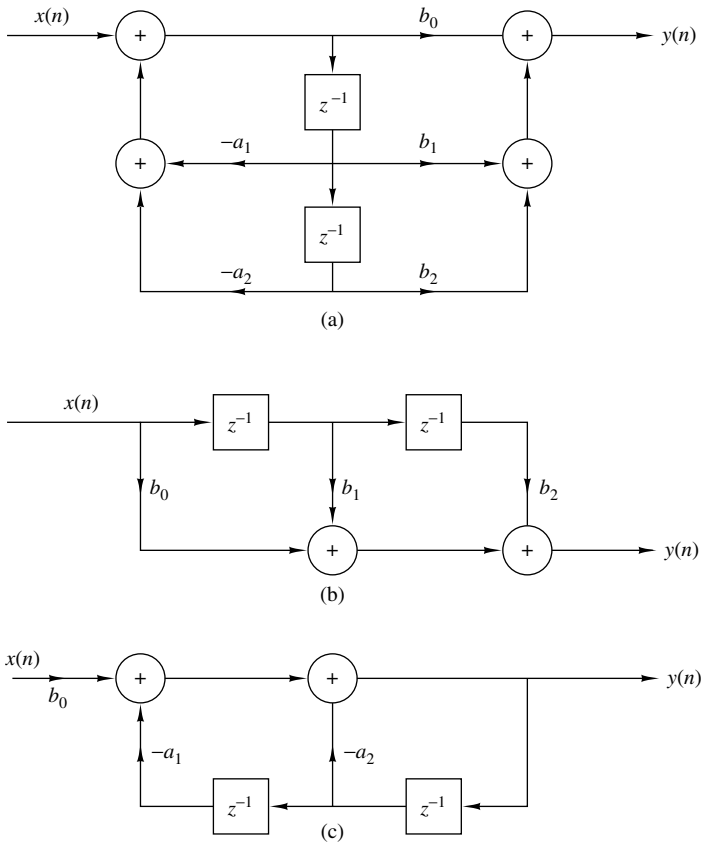


Figure 5.4 Structures for the realization of second-order systems: (a) general second-order system; (b) FIR system; (c) "purely recursive system."

and for a linear time-invariant system specifically, by the difference equation

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{5.18}$$

On the other hand, causal nonrecursive systems do not depend on past values of the output and hence are described by an input–output equation of the form

$$y(n) = F[x(n), x(n-1), \ldots, x(n-M)] \tag{5.19}$$

and for linear time-invariant systems specifically, by the difference equation in (5.18) with $a_k = 0$ for $k = 1, 2, \ldots, N$.

In the case of FIR systems, we have already observed that it is always possible to realize such systems nonrecursively. In fact, with $a_k = 0, k = 1, 2, \ldots, N$, in (5.18),

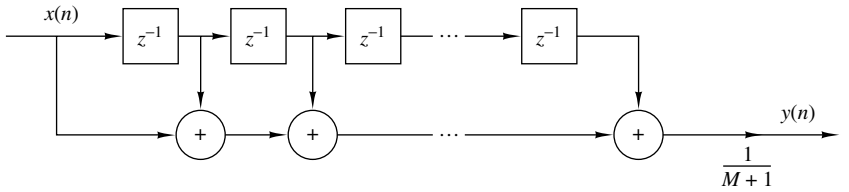


Figure 5.5 Nonrecursive realization of an FIR moving average system.

we have a system with an input–output equation

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) \tag{5.20}$$

This is a nonrecursive and FIR system. As indicated in (5.12), the impulse response of the system is simply equal to the coefficients $\{b_k\}$. Hence every FIR system can be realized nonrecursively. On the other hand, any FIR system can also be realized recursively. Although the general proof of this statement is given later, we shall give a simple example to illustrate the point.

Suppose that we have an FIR system of the form

$$y(n) = \frac{1}{M+1} \sum_{k=0}^{M} x(n-k) \tag{5.21}$$

for computing the *moving average* of a signal $x(n)$. Clearly, this system is FIR with impulse response

$$h(n) = \frac{1}{M+1}, \qquad 0 \le n \le M$$

Figure 5.5 illustrates the structure of the nonrecursive realization of the system. Now, suppose that we express (5.21) as

$$\begin{aligned} y(n) &= \frac{1}{M+1} \sum_{k=0}^{M} x(n-1-k) \\ &\quad + \frac{1}{M+1}[x(n) - x(n-1-M)] \\ &= y(n-1) + \frac{1}{M+1}[x(n) - x(n-1-M)] \end{aligned} \tag{5.22}$$

Now, (5.22) represents a recursive realization of the FIR system. The structure of this recursive realization of the moving average system is illustrated in Fig. 5.6.

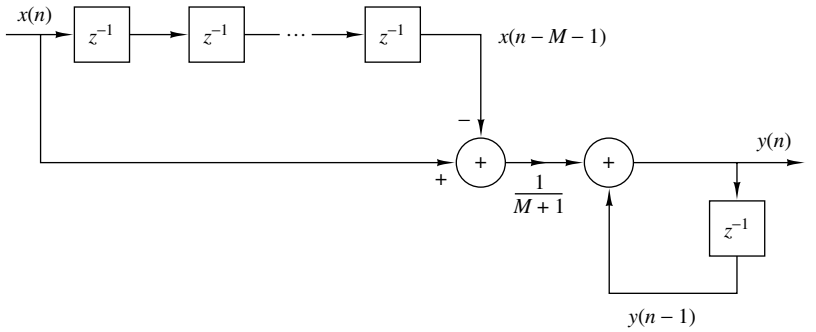


Figure 5.6 Recursive realization of an FIR moving average system.

In summary, we can think of the terms FIR and IIR as general characteristics that distinguish a type of linear time-invariant system, and of the terms *recursive* and *nonrecursive* as descriptions of the structures for realizing or implementing the system.