Chapter 2
Discrete-Time Signals and Systems

2.1 Introduction

Continuous-time or analog signals are processed using analog devices such as amplifiers, filters, etc. It is impossible to process signals multiplexed from various sources using a single hardware system in the analog domain. On the other hand, digital signals can be processed using both special-purpose hardware and software systems. Worldwide use of Internet, mobile communications, etc. demands all kinds of data such as video, audio, graphics, etc. In order to receive this information on a single device, computer, for instance, it is impossible to use analog signals and techniques. In order to be able to design and implement digitally based systems, it is absolutely necessary to have an understanding of digital signals and systems. Digital signals are discrete in time and amplitude. However, we will assume discrete-time signals to have a continuum of amplitude in order to be able to analyze such signals and systems mathematically. In this chapter we will describe typical discrete-time signals mathematically and then use them to describe and analyze linear time-invariant discrete-time systems. To help the readers understand the mathematical details, we will work out examples followed by MATLAB-based examples. Since digital signals are obtained from analog sources, we will also discuss the conversion of continuous-time signals into digital signals using analog-to-digital converters (ADC).

Electronic supplementary material: The online version of this article (https://doi.org/10.1007/978-3-319-76029-2_2) contains supplementary material, which is available to authorized users.

K. S. Thyagarajan, *Introduction to Digital Signal Processing Using MATLAB with Application to Digital Communications*,
https://doi.org/10.1007/978-3-319-76029-2_2

2.2 Typical Discrete-Time Signals

A discrete-time signal is denoted by $x[n]$, $y[n]$, etc. and is defined over the interval $-\infty < n < \infty$, $n \in Z$. The amplitude of a discrete-time signal is a continuum, while its argument n is an integer. If a discrete-time signal is obtained from a continuous-time signal, then the argument of the discrete-time signal is an integer multiple of the sampling interval. We will discuss the sampling process later in the chapter. A discrete-time signal is also referred to as a sequence. When a discrete-time signal is processed by a computer in software or hardware, the signal amplitudes are represented by numbers, and so the signal is a digital signal. Even though discrete-time signals are processed by a computer, we will still assume their amplitudes to be a continuum in our discussion.

There are several discrete-time signals that are useful in characterizing other discrete-time signals as well as systems similar to those used in the continuous-time domain. We will describe them here briefly.

Unit Impulse Sequence A unit impulse sequence is denoted by $\delta[n]$ and is defined as

$$\delta[n] = \begin{cases} 1, n = 0 \\ 0, n \neq 0 \end{cases} \tag{2.1}$$

A unit impulse in the discrete-time is similar to the Dirac delta function in the continuous-time except that the unit impulse sequence is physically realizable.

Unit Step Sequence A unit step sequence is denoted by $u[n]$ and is defined as

$$u[n] = \begin{cases} 1, n \geq 0 \\ 0, n < 0 \end{cases} \tag{2.2}$$

The unit step sequence plays a similar part in the analysis of discrete-time systems as its continuous-time counterpart.

Exponential Sequence A real exponential sequence is defined as

$$x[n] = \alpha^n u[n], |\alpha| < 1 \tag{2.3}$$

In (2.3), α is a real constant.

Real Sinusoidal Sequence A real sinusoidal sequence is defined as

$$x[n] = A\cos(n\Omega_0 + \phi), \ -\infty < n < \infty, n \in Z \tag{2.4}$$

In Eq. (2.4), A is the amplitude, $\Omega_0 = 2\pi\frac{f_0}{F_s}$ is the normalized frequency in rad, f_0 is the frequency in Hz, F_s is the sampling frequency in Hz, and ϕ is the phase offset in rad.

Complex Exponential Sequence A complex exponential sequence is described by

$$x[n] = A\alpha^n exp(-jn\Omega_0)u[n], |\alpha| < 1 \tag{2.5}$$

In Eq. (2.5), the amplitude A may be complex, and α is a real constant.

Periodic Sequence A sequence $x[n]$ is said to be periodic with period N if $x[n + kN] = x[n]$, where k is an integer. From the definition we can easily verify that the sinusoidal sequence in (2.4) is periodic with period $N = \frac{kF_s}{f_0}$.

2.3 Discrete-Time Systems

A discrete-time system, $\mathcal{L}\{.\}$, is one that accepts an input sequence $x[n]$ to produce an output sequence $y[n]$. It can be formally written as

$$y[n] = \mathcal{L}\{x[n]\} \tag{2.6}$$

Linearity A discrete-time system is said to be linear if it satisfies the superposition rule. In other words, a discrete-time system is linear if the following condition holds:

$$y[n] = \mathcal{L}\{\alpha x_1[n] + \beta x_2[n]\} = \alpha y_1[n] + \beta y_2[n], \tag{2.7}$$

where $y_1[n] = \mathcal{L}\{x_1[n]\}$ and $y_2[n] = \mathcal{L}\{x_2[n]\}$. So, a linear discrete-time system responds to a linear combination of input sequences with the same linear combination of individual responses. Linear discrete-time systems are most useful because they can be solved analytically. If the above condition stated in (2.7) is not valid, then the discrete-time system is nonlinear. Nonlinear systems in general don't have closed-form solution and must be solved iteratively. Hence linear systems are preferred in practice though many practical systems may be nonlinear.

Time- or Shift-Invariant Discrete-Time Systems A discrete-time system is said to be time- or shift-invariant if a delayed input results in a delayed response:

$$\mathcal{L}\{x[n-m]\} = y[n-m], m \in Z \tag{2.8}$$

Impulse Response As in the continuous-time system, the response of a discrete-time system to a unit impulse sequence is called the *impulse response* and is denoted by $h[n]$. The impulse response is formally defined as

$$h[n] = \mathcal{L}\{\delta[n]\} \tag{2.9}$$

The impulse response is unique to a given discrete-time linear system and is very useful in calculating the system response to any given input. It is also very useful in the design of digital filters. We will deal with the design of digital filters later in the book.

Causality A discrete-time system is causal if it is non-anticipatory. That is to say that if the response of a discrete-time system at the current time index, n, does not depend on the input at a future time instant, then the system is causal.

Let us understand what we have discussed so far clearly by going through the following examples.

Example 2.1 A discrete-time system is defined by the following difference equation:

$$y[n] = x[n+1] - 2x[n] + x[n-1] \tag{2.10}$$

Is it (a) linear? (b) Is it time-invariant? (c) Is it causal?

Solution Let me make it clear. A discrete-time system may be characterized by a linear difference equation just as we described a continuous-time system by a differential equation.

(a) Let the input be a linear sum of two input sequences: $x[n] = ax_1[n] + bx_2[n]$, where a and b are constants. Then the response of the system can be written using (2.10) as

$$\begin{aligned} y[n] &= a(x_1[n+1] - 2x_1[n] + x_1[n-1]) \\ &\quad + b(x_2[n+1] - 2x_2[n] + x_2[n-1]) \\ &= ay_1[n] + by_2[n] \end{aligned} \tag{2.11}$$

In the above equation, $y_1[n] = \mathcal{L}\{x_1[n]\}$ and $y_2[n] = \mathcal{L}\{x_2[n]\}$. Since the superposition rule is satisfied, the given discrete-time system is linear.

(b) From the given difference equation, we notice that delaying the input sequence by an integer M produces a response, which is exactly the delayed version of the response by the same integer M. Hence the system is time- or shift-invariant. Note that if the coefficients a and b are dependent on the time index *n*, then the system will no longer be shift-invariant!
(c) We notice from the given system's input-output relationship that the response of the system at the current time index n depends on the input at the next future input. Therefore, the system is anticipatory and hence is non-causal.

Example 2.2 Determine if the discrete-time system $y[n] = K + x[n] + 0.75x[n-1]$, where K is a constant, is (a) linear, (b) time-invariant, and (c) causal?

Solution
(a) If we apply the superposition rule, we observe that it is not satisfied due to the constant K. Hence the system is not linear. However, it is piecewise linear.
(b) Since the delayed input produces the same delayed response, the system is time-invariant. It can also be inferred from the fact that the coefficients in the given input-output relationship are constants, independent of the time index.
(c) The system response at the current time index n does not depend on the input sequence at future time instants. Hence the system is causal.

Stability A discrete-time system is said to be stable if a bounded input produces a bounded output. Equivalently, we can impose the stability condition on the impulse response. We will look at it after we define the convolution sum. This definition of stability is called bounded-input bounded-output (BIBO) stability.

Example 2.3 Is the system described in Example 2.1 stable in the BIBO sense?

Solution If we assume that the absolute value of the input sequence is finite, that is, $|x[n]| \leq M\, for\ all\ n$, then we see that the output sequence value is also finite:

$$|y[n]| \leq |x[n+1] - 2x[n-1] + x[n]| \leq 4M < \infty \tag{2.12}$$

Hence the system is stable.

2.4 Convolution Sum

The response of an LTI discrete-time system to any given input sequence can be obtained in terms of its impulse response sequence and the input sequence by what is called the *convolution sum*. We first observe that a given sequence $x[n]$ can be represented as an infinite sum of unit impulses:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{2.13}$$

Equation (2.13) follows from the definition of the unit impulse. So, the right-hand side of (2.13) is zero except for $k = n$, in which case, the right-hand side is simply $x[n]$. Having expressed the input sequence in terms of the unit impulse sequence, we next determine the response of the LTI system as

$$y[n] = \mathcal{L}\{x[n]\} = \mathcal{L}\left\{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right\} \tag{2.14}$$

Since the system is linear, the system operator can be taken inside the summation as

$$y[n] = \sum_{k=-\infty}^{\infty} \mathcal{L}\{x[k]\delta[n-k]\} \tag{2.15}$$

That is, the response is the linear sum of the responses to individual impulses $\delta[n-k]$. However, $x[k]$ is a constant, and since the system is linear, the response to constant times an input is equal to the constant times the response to the input. Therefore, (2.15) can be rewritten as

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\mathcal{L}\{\delta[n-k]\} \tag{2.16}$$

We have also assumed the system to be time-invariant. Therefore, $\mathcal{L}\{\delta[n-k]\} = h[n-k]$. Hence,

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{2.17}$$

Equation (2.17) is known as the convolution sum of the sequences $x[n]$ and $h[n]$ and is usually abbreviated as $x[n] \otimes h[n]$. In (2.17), if we substitute $m = n - k$, then we can also write the convolution sum as

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m] \tag{2.18}$$

Procedure to Calculate the Convolution Sum

We can list a graphical procedure to calculate the convolution sum given in Eq. (2.17) as follows:

1. Flip the impulse response $h[n]$ about the origin, and label the abscissa with the integer variable k.
2. Multiply the input sequence and the flipped impulse response sequence point by point, and sum them over the entire interval. This sum gives the response at n = 0.
3. Slide the flipped impulse response to the right one sample at a time.
4. Multiply the input sequence and the flipped impulse response sequence point by point, and sum them over the entire interval. The sum gives the system response at subsequent time instants.
5. For negative integer values of n, repeat steps 3 and 4, except that the impulse response sequence is slid to the left instead of right.

Example 2.4 Consider the LTI discrete-time system with an impulse response $h[n] = \alpha^n u[n]$, $|\alpha| < 1$. Determine its unit step response.

Solution Using the graphical interpretation of the convolution sum, we first label the abscissa with the index k. Next we flip the impulse response about the ordinate. This is shown in black color in Fig. 2.1a with no shift. In the same plot, the input sequence in red color is shown, which is a unit step. There is only one sample that overlaps the two sequences for n = 0, and the corresponding product of the two sequences results in the response at n = 0. As can be seen from Fig. 2.1b, any shift of the impulse response sequence to the left, that is, for n < 0, will leave no overlapping of the two sequences. Hence the system response will be zero for n < 0. Figure 2.1c shows the flipped impulse response shifted to the right by 3, that is, n = 3. Now there are four overlapping samples. We multiply the overlapping samples and sum them to obtain the response at n = 3. This can be continued for each shift of the impulse response to the right. To obtain the response in closed form, we resort to the convolution sum in Eq. (2.17).

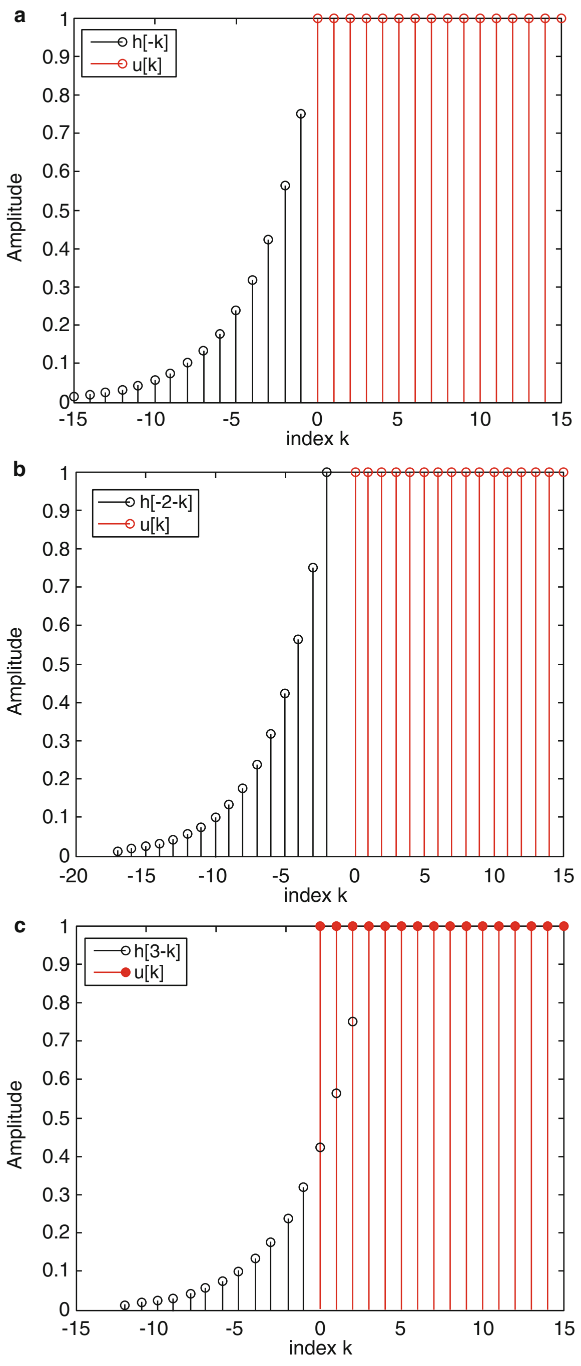


Fig. 2.1 Graphical interpretation of the convolution sum: (**a**) flipped impulse response for no shift, (**b**) flipped impulse response shifted to the left by two samples, and (**c**) flipped impulse response shifted to the right by three samples

$$y[n]=\sum_{k=-\infty}^{\infty}x[k]h[n-k]=\sum_{k=0}^{n}\alpha^{n-k}=\propto^{n}\sum_{k=0}^{n}\alpha^{-k} \tag{2.19}$$

In (2.19) since the input is zero for n < 0, the lower limit of the summation is zero. Also, the response is zero for n < 0. The upper limit is n corresponding to the current time instant. The sequence on the right-hand side of (2.19) is an exponentially decreasing sequence. After simplifying (2.19) we get the unit step response of the system,

$$y[n]=\frac{\alpha^{n+1}-1}{\alpha-1},n\geq 0 \tag{2.20}$$

The impulse response and the system response are shown in Figs. 2.2a and b, respectively. The impulse response sequence is assumed to be $(0.75)^{n}u[n]$. The rise time of the LTI system is defined as that interval in which the step response changes from 10% to 90% of its final response. The final value of the response is found to be 4.

Causality Revisited Earlier we said that an LTI system is causal if it is non-anticipatory. We can also impose causality on the impulse response of the LTI system. To this end, recall that the convolution sum represents the response of an LTI discrete-time system to an input sequence. If the input to an LTI discrete-time system is assumed to be zero for n less than zero, then the response $y[n]$ in terms of the impulse response $h[n]$ is written as

$$y[n]=x[n]\otimes h[n]=\sum_{k=0}^{\infty}x[k]h[n-k] \tag{2.21}$$

For instance, let us evaluate the response at n = 1. Expanding the summation on the right-hand side of (2.21), we have

$$y[1]=x[0]h[1]+x[1]h[0]+x[2]h[-1]+x[3]h[-2]+\cdots \tag{2.21a}$$

Since the system is assumed to be causal, $y[\mathrm{n}=1]$ should not be dependent on x[n] for n > 1. However, x[n] is the input sequence and is not zero. Therefore, $h[-1]=h[-2]=\cdots=0$. That is to say that $h[n]=0$ for n < 0. Generalizing, we say that for the system to be causal, $h[n-k]=0,\ for\ k>n$. Otherwise it will be anticipatory. Let $m=n-k$. Then, for the system to be causal, $k>n\Longrightarrow m<0$. Hence for the system to be causal, $h[n]=0\, for\, n<0$, which implies that the upper limit of the summation in (2.21) is n. An LTI discrete-time system is causal if and only if its impulse response is zero for $n<0$. Otherwise it is non-causal.

Stability in Terms of the Impulse Response An LTI discrete-time system is stable in the BIBO sense if its response is finite for a finite input. Let the input sequence $x[n]$ be bounded, that is, $|x[n]|\leq M<\infty$, for all n. Using the convolution sum in (2.17), we can bound the output as given below:

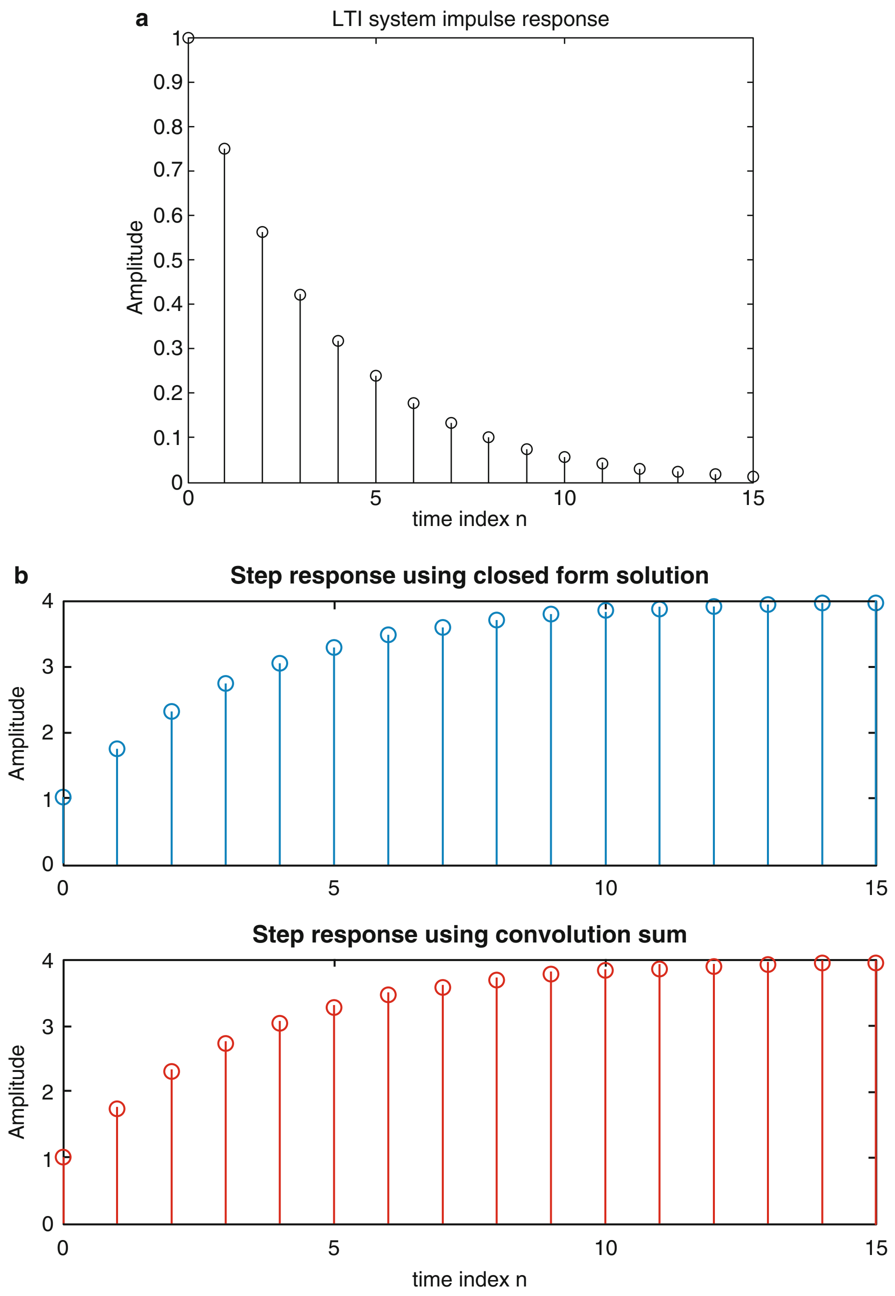


Fig. 2.2 Stem plot of the impulse response: (**a**) impulse response and (**b**) unit step response

$$|y[n]| = \left| \sum_{k=-\infty}^{\infty} x[k]h[n-k] \right| = \left| \sum_{k=-\infty}^{\infty} x[n-k]h[k] \right| \leq M \sum_{k=-\infty}^{\infty} |h[k]| \tag{2.22}$$

From (2.22) we see that the absolute value of the response is finite if and only if

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty \tag{2.23}$$

Thus, an LTI discrete-time system is stable in the BIBO sense if and only if its impulse response sequence is absolutely summable.

Example 2.5 An LTI discrete-time system is described by $y[n] = x[n] + 0.5y[n-1]$, $y[-1] = 0$. Determine if the system is stable in the BIBO sense.

Solution First, we need to find the impulse response of the system. Using $x[n] = \delta[n]$, in the above system definition and the fact that $y[-1] = 0$, we obtain the following:

$$\begin{aligned} &y[0] = \delta[0] + 0.5y[-1] = 1; \\ &y[1] = \delta[1] + 0.5y[0] = 0.5; \\ &y[2] = 0.5y[1] = 0.5^2; \\ &y[3] = 0.5y[2] = 0.5^3, \ldots \\ &y[n] = 0.5y[n-1] = 0.5^n \end{aligned} \tag{2.24}$$

Thus, we find the impulse response sequence to be $h[n] = 0.5^n u[n]$. For this system to be stable, the impulse response sequence must be absolutely summable:

$$\sum_{n=-\infty}^{\infty} |h[n]| = \sum_{k=0}^{\infty} |0.5^n| = \sum_{k=0}^{\infty} 0.5^n = \frac{1}{1-0.5} = 2 < \infty$$

Since the impulse response sequence is absolutely summable, the above system is stable.

2.5 Linear Difference Equation

So far we have defined an LTI discrete-time system in terms of its impulse response, which fully defines the system. The response to any input sequence can then be obtained by convolving the input and impulse response sequences. Alternatively, one can also describe an LTI discrete-time system by a linear *difference equation* with constant coefficients. More specifically, an LTI discrete-time system can be described by

$$\sum_{k=0}^{p} b_k y[n-k] = \sum_{j=0}^{q} a_j x[n-j], b_0 = 1 \tag{2.25}$$

The order of the difference equation (2.25) is the maximum of p and q. In Eq. (2.25), if not all b_k's are zero then the corresponding difference equation is called a *recursive* equation. It uses both feed-forward and feedback to compute the output at each time instant. On the other hand, if all but b_0 are zero, then the resulting equation is termed *non-recursive* difference equation. To make it clearer, let us rewrite (2.25) as

$$y[n] = \sum_{j=0}^{q} a_j x[n-j] - \sum_{k=1}^{p} b_k y[n-k] \tag{2.26}$$

At each time instant n, the response is obtained by finding the weighted sum of the previous p output samples and then subtracting it from the weighted sum of the input samples, which involves the current and previous q input samples. In order to use the previous input and output samples, we need to store them. More specifically, we need to store p previous output samples and q previous input samples and retrieve them to compute the present output sample. Let us clarify this by an example.

Example 2.6 Draw a signal flow diagram to compute the response of the 2nd-order LTI discrete-time system described by the following recursive equation:

$$y[n] = a_0 x[n] + a_1 x[n-1] - b_1 y[n-1] - b_2 y[n-2] \tag{2.27}$$

A signal flow diagram shows how the signals flow or propagate from the input to the output. It uses adders, multipliers, and delays. Each delay element corresponds to one sampling interval. Lines with arrows indicate the direction of signal flow. Figure 2.3 depicts the signal flow diagram to compute the output for a given input described by (2.27). An adder is depicted by a circle with a plus sign inscribed in it.

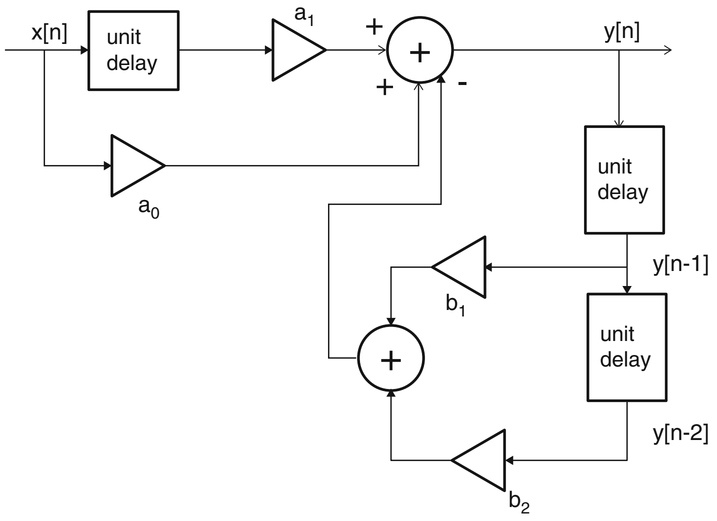


Fig. 2.3 A signal flow diagram corresponding to the difference equation (2.27)

A triangle pointing in the direction of the signal flow indicates a multiplier with the coefficient shown by the side of the triangle. A unit delay is depicted by a rectangle. The various elements are interconnected by straight lines with arrows indicating the direction of the signal flow.

The signal flow diagram shown in Fig. 2.3 is not the most efficient in terms of delay elements. It uses three delay elements. The difference equation in (2.27) uses output samples corresponding to two previous sampling intervals. Hence, it is possible to use a total of only two unit delays, which is more efficient than using three unit delays. We will discuss signal flow graphs in more detail in a later chapter.

Solving Linear Difference Equations Instead of computing the response of an LTI discrete-time system at each time instant by solving the difference equation recursively, one can also obtain the system response in closed form by means of analytical solution to the difference equation. The general solution to a constant coefficient linear difference equation consists of two parts: *complementary* solution $y_C[n]$ and *particular* solution $y_P[n]$. The complementary solution is the response to zero input, and the particular solution is the response to a specified input. Thus, the total solution to a linear difference equation with constant coefficients can be expressed as

$$y[n] = y_C[n] + y_P[n] \tag{2.28}$$

The complementary solution is obtained by (1) setting the input to zero, (2) assuming a solution of the type α^n, (3) substituting the solution in the zero input difference equation (2.25), and (4) solving for α. For a pth-order linear difference equation with constant coefficients, the complementary solution then takes the form

$$y_C[n] = \sum_{i=1}^{p} a_i {\alpha_i}^n \tag{2.29}$$

The particular solution is assumed to be some constant times the input. The constant of proportionality is determined by substituting the particular solution in the difference equation and solving the resulting equation. Finally, the constants in the complementary solution are determined using the initial conditions in the total solution. Let us illustrate the above statements by the following example.

Example 2.7 Solve the following difference equation when the input is a unit step sequence:

$$y[n] = x[n] + 0.25y[n-1] + 0.125y[n-2], \mathit{with}\ y[-1] = 1, y[-2] = -1$$

Solution Let the complementary solution be $y_C[n] = \alpha^n$. Substituting $y_C[n]$ for $y[n]$ in the above difference equation with $x[n] = 0$, we get

$$\alpha^n = 0.25\alpha^{n-1} + 0.125\alpha^{n-2}$$

Or,

$$1 = 0.25\alpha^{-1} + 0.125\alpha^{-2} \Rightarrow \alpha_1 = 0.5, \alpha_2 = -0.25$$

Therefore, the complementary solution is $y_C[n] = a(0.5)^n + b(-0.25)^n$. Since the input is a unit step sequence, the particular solution is assumed to be $y_P[n] = cu[n]$. To find the value of the constant c, substitute $y_P[n]$ for $y[n]$in the given difference equation with the input $x[n] = u[n]$. We, therefore, have

$$cu[n] = u[n] + c \times 0.25u[n-1] + c \times 0.125u[n-2] \Rightarrow c = 1.6$$

Therefore, the total solution to the given difference equation is expressed as

$$y[n] = 1.6u[n] + \mathrm{a}(0.5)^n + \mathrm{b}(-0.25)^n$$

Finally, use the initial conditions to solve for the two constants in the complementary solution. Thus, the two equations involving the constants a and b are

$$y[-1] = 1 = a(0.5)^{-1} + b(-0.25)^{-1} + 1.6 \Rightarrow 2a - 4b = -0.6$$

$$y[-2] = -1 = a(0.5)^{-2} + b(-0.25)^{-2} + 1.6 \Rightarrow 4a + 16b = -2.6$$

The solution to the above two equations gives $a = -\frac{5}{12}$ and $b = -\frac{7}{120}$. The overall solution to the given difference equation is, therefore,

$$y[n] = -\frac{5}{12}(0.5)^n - \frac{7}{120}(-0.25)^n + 1.6, n \geq 0$$

Note that the difference equation and the total solution give the same value of 1.125 at n = 0. Figure 2.4 shows stem plots of the response to a unit step sequence of the system in Example 2.7 using both the difference equation and the total solution. They appear to be identical. This shows that one can compute the response of an LTI discrete-time system either directly from the given difference equation or from the total solution obtained by analytical means.

Example 2.8 Let us consider the case where the input has the same form as one of the terms in the complementary solution. Specifically, we want to solve the difference equation of an LTI discrete-time system described by

$$y[n] - 0.8y[n-1] + 0.15y[n-2] = (0.5)^n u[n], y[-1] = 1, y[-2] = 0 \qquad (2.30)$$

Let the complementary solution be $y_C[n] = \alpha^n$. Substituting the complementary solution in (2.30) with input being zero, we have

$$\alpha^n - 0.8\alpha^{n-1} + 0.15\alpha^{n-2} = 0 \qquad (2.31a)$$

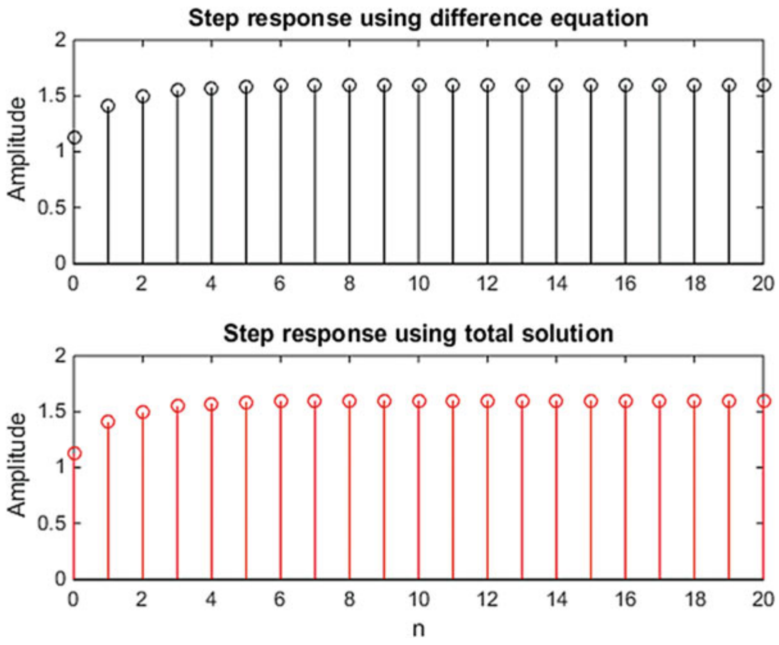


Fig. 2.4 Stem plots showing the system response to a unit step in Example 2.7. Top plot: response using the difference equation. Bottom plot: response obtained from the total solution

or,

$$\alpha^2 - 0.8\alpha + 0.15 = 0, \tag{2.31b}$$

which gives $\alpha_1 = 0.5$ and $\alpha_2 = 0.3$.

Since one of the terms in the complementary solution, namely, α_1, has the same form as the input, the particular solution must be assumed to be

$$y_P[n] = \beta n(0.5)^n u[n] \tag{2.32}$$

The complementary solution and the particular solution must be linearly independent. That is why the particular solution in (2.32) is used instead. To determine β, we solve Eq. (2.30) with $y[n]$ replaced with $\beta n(0.5)^n u[n]$:

$$\beta n(0.5)^n - 0.8\beta(n-1)(0.5)^{n-1} + 0.15\beta(n-2)(0.5)^{n-2} = 0.5^n \tag{2.33a}$$

Or,

$$\beta n - \frac{0.8\beta(n-1)}{0.5} + \frac{0.15\beta(n-2)}{0.5^2} = 1 \Rightarrow \beta = 2.5 \tag{2.33b}$$

Therefore, the total solution to the difference equation (2.30) is

$$y[n] = a(0.5)^n + b(0.3)^n + 2.5n(0.5)^n, n \geq 0 \tag{2.34}$$

To find the values for the constants in (2.34), use the initial conditions:

$$y[-1] = 1 = \frac{a}{0.5} + \frac{b}{0.3} + \frac{2.5(-1)}{0.5} \tag{2.35a}$$

$$y[-2] = 0 = \frac{a}{0.5^2} + \frac{b}{0.3^2} + \frac{2.5(-2)}{0.5^2} \tag{2.35b}$$

The solution to (2.35a) and (2.35b) results in $a = 0$ and $b = 1.8$. Thus, the solution to Eq. (2.30) is

$$y[n] = 1.8(0.3)^n + 2.5n(0.5)^n, n \geq 0 \tag{2.36}$$

Figure 2.5 shows the response calculated using the difference equation (2.30) in the top plot and the response obtained from the total solution in the bottom plot. They seem to be identical.

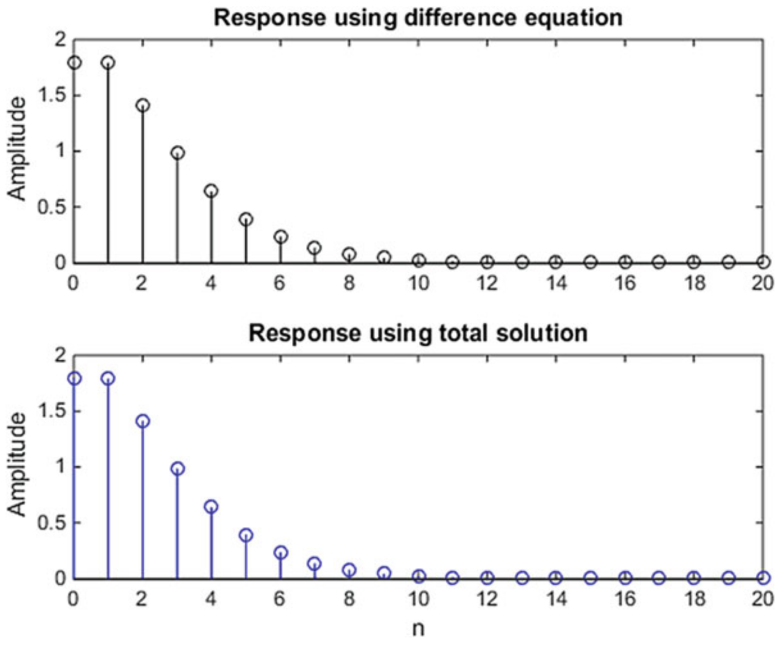


Fig. 2.5 Stem plots showing the system response to the input in Example 2.8. Top plot: response using the difference equation. Bottom plot: response obtained from the total solution

Example 2.9 An LTI discrete-time system is described by the following difference equation with initial conditions:

$$y[n] - 1.5y[n-1] + 0.5625y[n-2] = x[n], y[-1] = 1, y[-2] = 0$$

Determine the total solution to the above difference equation if $x[n] = \cos(0.2n)u[n]$.

Solution Let the complementary solution be $y_C[n] = \alpha^n$. Then, with $x[n] = 0$, the difference equation becomes

$$\alpha^n - 1.5\alpha^{n-1} + 0.5625\alpha^{n-2} = 0. \tag{2.37a}$$

Or,

$$\alpha^{n-2}\left(\alpha^2 - 1.5\alpha + 0.5625\right) = 0 \Rightarrow \alpha_1 = \alpha_2 = 0.75 \tag{2.37b}$$

Since the two roots of the characteristic equation are the same, the two terms of the complementary solution are α^n and $n\alpha^n$ and are linearly independent. Thus,

$$y_C[n] = a\alpha^n + bn\alpha^n \tag{2.38}$$

Next we assume the particular solution to be

$$y_P[n] = A\cos(0.2n + \varphi)u[n] = Re\left\{Ae^{j(0.2n+\varphi)}\right\}u[n] \tag{2.39}$$

Note that we have introduced a phase term in the argument of the cosine function of the particular solution. An LTI system will respond to a sinusoidal input of a certain frequency with the same sinusoid but with different amplitude and phase. In order to evaluate the constants of the particular solution, we substitute (2.39) in the given difference equation. Therefore, we have

$$\begin{aligned}&Re\left\{Ae^{j(0.2n+\varphi)} - 1.5Ae^{j(0.2n+\varphi-0.2)} + 0.5625Ae^{j(0.2n+\varphi-0.4)}\right\}\\&\quad = Re\left\{e^{j0.2n}\right\}\end{aligned} \tag{2.40}$$

Rearranging (2.40), we get

$$\begin{aligned}&A[\cos(\varphi) - 1.5\cos(\varphi - 0.2) + 0.5625\cos(\varphi - 0.4)]\cos(0.2n)\\&\quad - A[\sin(\varphi) - 1.5\sin(\varphi - 0.2) + 0.5625\sin(\varphi - 0.4)]\sin(0.2n)\\&\quad = \cos(0.2n)\end{aligned} \tag{2.41}$$

From (2.41), we obtain the following two equations:

$$A[\cos(\varphi) - 1.5\cos(\varphi - 0.2) + 0.5625\cos(\varphi - 0.4)] = 1 \tag{2.42a}$$

$$A[\sin(\varphi) - 1.5\sin(\varphi - 0.2) + 0.5625\sin(\varphi - 0.4)] = 0 \tag{2.42b}$$

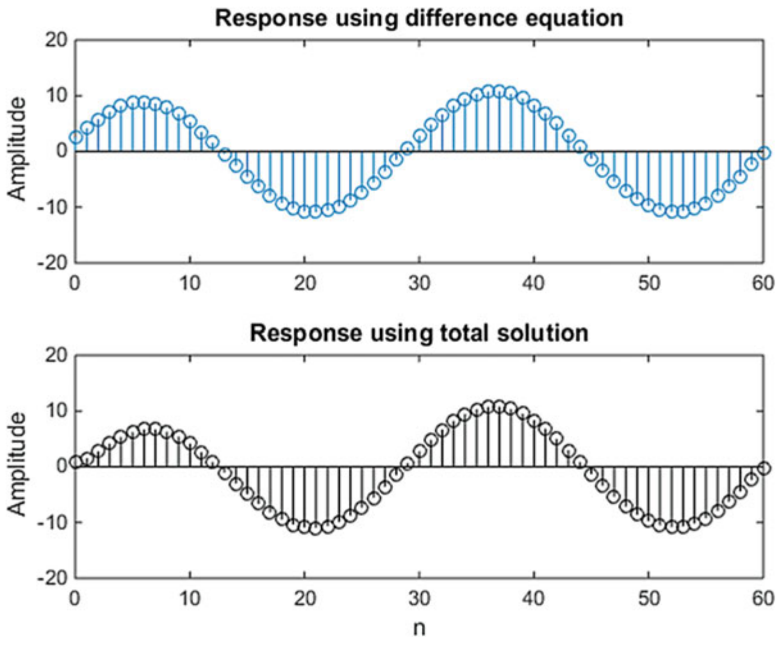


Fig. 2.6 Response of the LTI system of Example 2.9 plotted as stem plots

Solving the Eqs. (2.42a) and (2.42b) for A and φ, we get $A = 10.8225$ and $\varphi = -1.024592$ *rad*, respectively. The total solution is, therefore,

$$y[n] = a(0.75)^n + bn(0.75)^n + 10.8225\cos\left(0.2n - 1.024592^r\right) \tag{2.43}$$

Using the given initial conditions in the above equation and solving the resulting two equations, we find the system response to be

$$\begin{aligned} y[n] = &-4.8955(0.75)^n - 2.8912 \times n(0.75)^n \\ &+ 10.8225\cos\left(0.2n - 1.024592\right), n \geq 0 \end{aligned} \tag{2.44}$$

The responses obtained by recursively solving the difference equation and from the total solution to the difference equation are plotted as stem plots and are shown in Fig. 2.6. They seem to agree.

Convolution of Finite-Length Sequences Consider the two finite-length sequences $\{x[n]\}$, $0 \leq n \leq N - 1$ and $\{h[n]\}$, $0 \leq n \leq M - 1$. Since the two sequences are of finite length, the convolution of these two sequences will result in a sequence that is also of finite length. In fact, the length of the convolution of the sequences of lengths M and N is $M + N - 1$. We can demonstrate this by an example. In Fig. 2.7 top plot is shown the two sequences to be convolved. For the sake of simplicity, the sequences are shown in solid lines though they are discrete. The bottom plot of Fig. 2.7 shows $\{x[k]\}$ as well as the flipped sequences $\{h[-k]\}$,

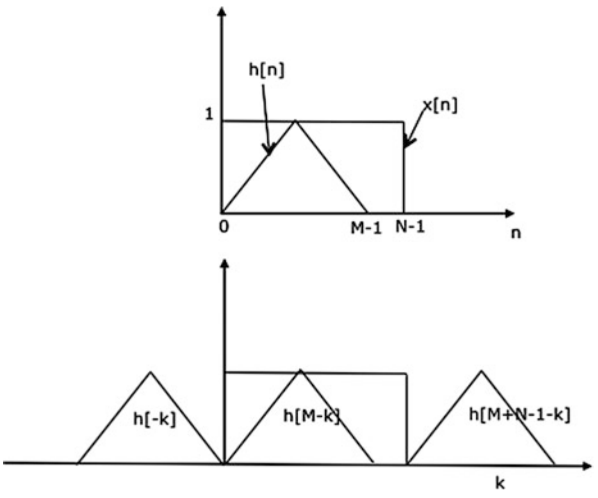


Fig. 2.7 Graphical illustration of the convolution of two finite-length sequences. Top plot: length M and N sequences. Bottom plot: flipped and shifted sequence for shifts 0, M, and M + N − 1, respectively

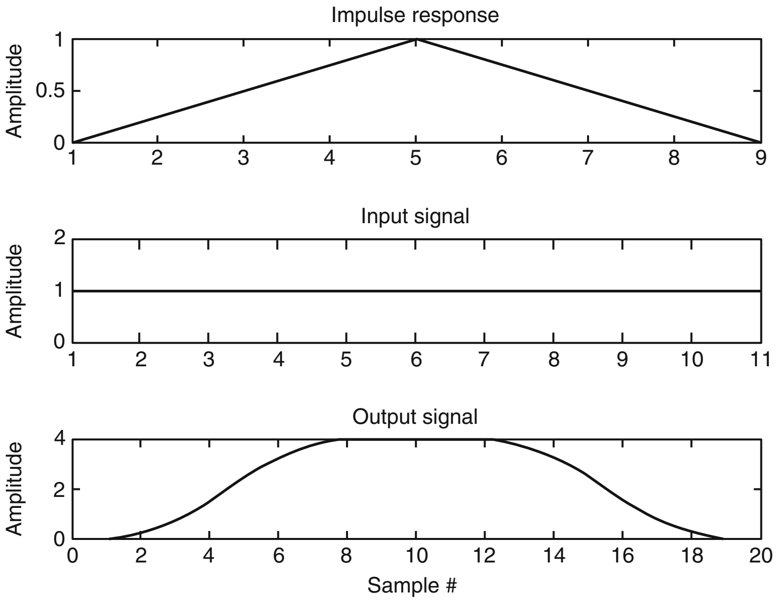


Fig. 2.8 Convolution of a length-9 triangular sequence and length-11 unit amplitude pulse

$\{h[M - k]\}$, and $\{h[M + N - 1 - k\}$. From the figure we observe that the convolution is zero for n < 0 and n > M + N − 1. Hence, the length of the convolution of two sequences of length M and N is M + N − 1.

Figure 2.8 shows the convolution of a length-9 triangular sequence and a length-11 unit amplitude pulse sequence resulting in a length- 11 + 9–1 = 19 sequence.

2.6 Sampling a Continuous-Time Signal

So far we have assumed explicitly the availability of discrete-time signals without reference to their origin. However, many discrete-time signals originate from their continuous-time counterparts. It is, therefore, necessary to understand how discrete-time signals are obtained from continuous-time signals and the implications thereof. It must be pointed out that the processed discrete-time signals must be converted back to their continuous-time versions. For example, one has to understand how many samples per second are necessary so that the discrete-time signal can be converted back to its continuous-time version without any impairment. Too many samples per second mean that the digital signal processor has to carry out a lot of arithmetic operations per second. This may impose undue constraints on the processor speed. On the other hand, fewer samples per second may cause serious distortions, which cannot be tolerated. Thus, one must determine the correct number of samples per second required for distortionless recovery of the continuous-time signal from the discrete-time signal. This can be achieved only by mathematical reasoning. In the following we will consider the process of sampling a continuous-time signal to obtain the discrete-time version and its implications. We will further ascertain the correct sampling interval for a given continuous-time signal.

Ideal Sampling A discrete-time signal is obtained from a continuous-time signal by sampling the continuous-time signal precisely at regular or uniform intervals of time. The sampled signal $x_s(t)$ can be expressed mathematically as

$$x_s(t) = x(t)|_{t=nT_s}, n \in Z \tag{2.45}$$

where T_s is the sampling interval. Figure 2.9 illustrates the ideal sampling process. The continuous-time signal is shown in cyan color and the sampled signal in red stems. At each sampling instant, the amplitude of the discrete-time signal corresponds to that of the continuous-time signal. Since the interval between any two samples is the same, the sampling is called *uniform sampling*. We also notice that the width of each sample is zero. Therefore, this type of sampling is called *ideal sampling* or *impulse sampling*. In practice, there is no such thing as ideal sampling. Each sample has a finite width, though very small. This type of sampling is called *non-ideal sampling* and has some implications, which we will consider later.

Our first task is to establish an upper limit on the sampling interval. In other words, what is the largest value of T_s and yet the continuous-time signal can be recovered from the sampled signal without any distortion? In order to answer this question, we must resort to the frequency domain representation of the signals under consideration. To this end we can rewrite Eq. (2.45) in terms of Dirac delta functions as

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(t)\delta(t - nT_s) \tag{2.46}$$

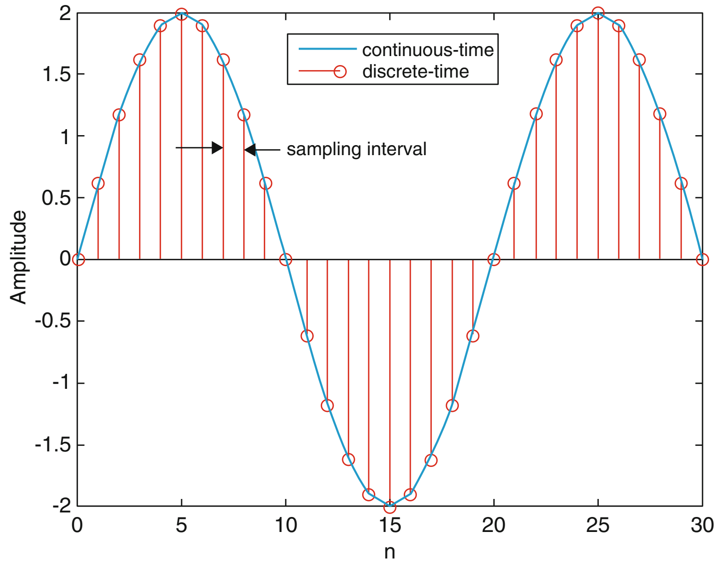


Fig. 2.9 Illustration of ideal sampling

Note that the unit impulse is zero except at time instants $t = nT_s$, at which instants its strength equals the sample value of the continuous-time signal. We can now express the Fourier transform of the sampled signal in terms of the Fourier transform of the continuous-time signal. Since the Fourier transform is the frequency domain representation of a signal, we will be able to determine the upper limit on the sampling interval. To this end, let $X(f)$ be the Fourier transform of the continuous-time signal $x(t)$. Then the Fourier transform of the sampled signal can be written as

$$X_s(f) = \mathcal{F}\{x_s(t)\} = \mathcal{F}\left\{\sum_{n=-\infty}^{\infty} x(t)\delta(t - nT_s)\right\} \tag{2.47}$$

Using the linearity and convolution in the frequency domain properties of the Fourier transform, we can express (2.47) as

$$X_s(f) = \sum_{n=-\infty}^{\infty} \mathcal{F}\{x(t)\delta(t - nT_s)\} = \sum_{-\infty}^{\infty} X(f) \otimes \delta(f - nF_s) \tag{2.48}$$

In Eq. (2.48), the sampling frequency is $F_s = \frac{1}{T_s}$. The convolution of $X(f)$ with an impulse $\delta(f - nF_s)$ results in shifting the spectrum of $X(f)$ to nF_s, that is, $X(f - nF_s)$. Therefore, we have

$$X_s(f) = \sum_{n=-\infty}^{\infty} X(f - nF_s) \tag{2.49}$$

Equation (2.49) implies that the Fourier transform of the sampled signal is an infinite sum of the Fourier transform of the continuous-time signal, replicated at integer multiples of the sampling frequency. By knowing the spectrum of the continuous-time signal, we can determine the upper limit for the sampling interval or equivalently and the lower limit on the sampling frequency. Since the continuous-time signal must be recovered from its samples, we must find a way to recover or *reconstruct* the continuous-time signal from its samples.

Sampling or Nyquist Theorem A continuous-time signal that is band limited to $|f| \le f_c$ can be recovered or reconstructed exactly from its samples taken uniformly at a rate $F_s \ge 2f_c$. The sampling frequency $F_s = 2f_c$ is called the *Nyquist frequency*. In terms of the sampling interval, the Nyquist theorem implies $\frac{1}{T_s} \ge 2f_c$ or $T_s \le \frac{1}{2f_c}$. That is, that the sampling interval must be less than or equal to the inverse of twice the maximum frequency of the continuous-time signal to be sampled.

From the statement of the sampling theorem, we notice that it pertains to continuous-time signals with finite bandwidth, that is, signals that are band limited. A continuous-time signal that is band limited to $|f| \le f_c$ is the same thing as saying that its Fourier transform satisfies the condition

$$|H(f)| = \begin{cases} K, |f| \le f_c \\ 0, otherwise \end{cases} \tag{2.50}$$

where K is a constant. This type of magnitude response is known as the *brick wall* type of response and is an ideal case. But no physically realizable system can have such a brick wall type of frequency spectrum. So in practice, to limit the frequency spectrum to a specified frequency range, one must prefilter the continuous-time signal and then sample it.

Reconstruction of an Ideally Sampled Signal The continuous-time signal can be recovered or reconstructed *exactly* from its samples by passing the samples through an *ideal* lowpass filter having a cutoff frequency equal to half the sampling frequency. In order to prove the statement, let $h(t)$ be the impulse response of the ideal lowpass filter. Then its response $y(t)$ to the sampled signal $x_s(t)$ is the convolution of the sampled signal and the impulse response:

$$y(t) = x_s(t) \otimes h(t) \tag{2.51}$$

Using (2.46) in (2.51), we have

$$\begin{aligned} y(t) &= \left\{ \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \right\} \otimes h(t) \\ &= \sum_{n=-\infty}^{\infty} x(nT_s)\{\delta(t - nT_s) \otimes h(t)\} \end{aligned} \tag{2.52}$$

Since the convolution of $\delta(t - nT_s)$ and $h(t)$ equals $h(t - nT_s)$, Eq. (2.52) results in

$$y(t) = \sum_{n=-\infty}^{\infty} x(nT_s)h(t - nT_s) \tag{2.53}$$

The impulse response of the ideal lowpass filter band limited to f_c can be shown to be

$$h(t) = 2f_c \frac{\sin(2\pi f_c t)}{2\pi f_c t} = 2f_c sinc(2f_c t) \tag{2.54}$$

The sinc function is defined as

$$sinc(x) = \frac{\sin(\pi x)}{\pi x} \tag{2.55}$$

Using Eqs. (2.54) in (2.53), the reconstructed signal is found to be

$$y(t) = 2f_c \sum_{n=-\infty}^{\infty} x(nT_s) \frac{\sin(2\pi f_c(t - nT_s))}{2\pi f_c(t - nT_s)} \tag{2.56}$$

The *sinc* function is unity at the sampling instants with an amplitude equal to the sample values of the signal. At other instants the signal amplitude is interpolated by the filter to reconstruct the continuous-time signal exactly. Thus, a continuous-time signal is recovered or reconstructed from its samples by filtering the sampled signal through an ideal lowpass filter whose cutoff frequency equals half the Nyquist frequency at most.

Aliasing Distortion What if the sampling frequency does not meet the Nyquist criterion? What happens when the sampling frequency is less than twice the maximum frequency of the continuous-time signal to be sampled? When a continuous-time signal is under-sampled, meaning the sampling frequency is below the Nyquist frequency, a distortion known as *aliasing distortion* occurs, because of which the continuous-time signal cannot be recovered from its samples. The frequencies above the *folding frequency* are aliased as lower frequencies. The folding frequency corresponds to half the sampling frequency. For instance, a frequency $f_1 + \frac{F_s}{2}$ present in the continuous-time signal will appear as a frequency $\frac{F_s}{2} - f_1$, which is lower than the frequency f_1. Thus, a frequency higher than the folding frequency if present will alias itself as a lower frequency. This is the aliasing distortion. This is depicted in Fig. 2.10. In Fig. 2.10a the spectrum of a band-limited continuous-time signal with a maximum frequency f_c is shown. Figure 2.10b shows the spectrum of the sampled signal, where the sampling frequency is much higher than $2f_c$. There is no overlap between the replicas of the spectra. Therefore, the continuous-time signal can be recovered by filtering the sampled signal by an ideal lowpass filter with a cutoff frequency f_c. Figure 2.10c depicts the case where the sampling frequency is less than twice the maximum frequency of the continuous-time signal. Because of the overlap of adjacent spectra, the spectrum in the frequency range $-f_c \leq f \leq f_c$ is distorted and is the cause for the aliasing distortion.

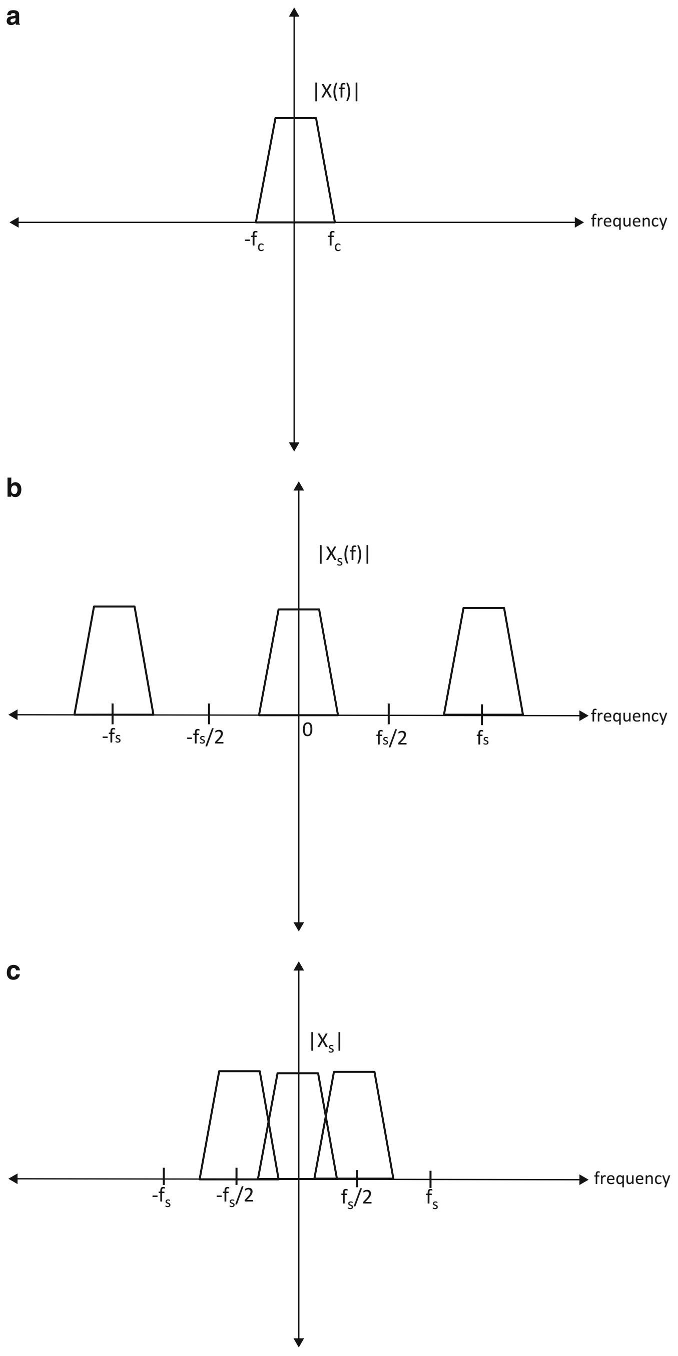


Fig. 2.10 An illustration of aliasing distortion: (**a**) spectrum of a continuous-time signal. (**b**) Oversampling case. (**c**) Undersampling case

As a second example, let us consider a continuous-time sinusoid of frequency 5 Hz. It is sampled at a rate of 20 per second. A plot of the sampled signal is shown in the top plot in Fig. 2.11a as a line plot for easier visualization. In the bottom plot of Fig. 2.11a a sinusoid at a frequency of 15 Hz sampled at 20 Hz is shown. The two plots look identical even though the two frequencies are different. The sinusoid at 15 Hz has a frequency higher than half of the sampling frequency of 20 Hz. It aliases itself as 15–10 = 5 Hz signal component. This aliasing frequency is the same as that of the sinusoid at 5 Hz. Hence, the two sampled signals look alike. This is further ascertained by the frequency spectra, which are shown in Fig. 2.11b as top and bottom plots, respectively. Similarly, a third example of undersampling and oversampling of a continuous-time signal is illustrated in Fig. 2.12 and is self-explanatory.

2.7 Conversion of Continuous-Time Signals to Digital Signals

So far in our discussion we have treated discrete-time signals as having a continuum of amplitudes, meaning that the amplitudes of the discrete-time signals have infinite accuracy in amplitude. As a result all our computations such as convolution sum, etc., were performed with infinite accuracy. This ideal scenario changes when we deal with processing discrete-time signals with computers – hardware or software. There are two issues involved here. First, we have to convert the discrete-time signals into digital signals, which are approximations to signals with continuum of amplitudes. This is known as analog-to-digital (A/D) conversion. The degree of approximation depends on the *word length* available for digital representation of the amplitudes. The larger the word length, the better the approximation. The second issue deals with the accuracy of arithmetic operations. Errors due to limited accuracy of arithmetic operations manifest as noise. This is also the case with A/D conversion. In this chapter we will deal with A/D conversion and resulting errors. In a later chapter we will analyze the effect of arithmetic errors due to finite precision arithmetic operations.

The process of converting an analog signal to digital signal is depicted in Fig. 2.13. The input continuous-time signal is first sampled, and the sampled value is held until it is converted to a digital number. The sample and hold (S/H) functional block samples the input signal at a predetermined uniform rate and holds the sample value until it is converted to a digital representation. Once the conversion is completed, the S/H acquires the next sample and so on. The process of S/H is illustrated in Fig. 2.14, where an analog signal is sampled and held constant until the next sample arrives. The second block, namely, the quantizer block, represents the sampled value to the nearest allowed level. This process is called *quantization*. There are two types of quantizers, namely, *scalar* and *vector* quantizers. A scalar quantizer accepts a single analog sample and outputs a quantized value that approximates the input analog sample. A vector quantizer, on the

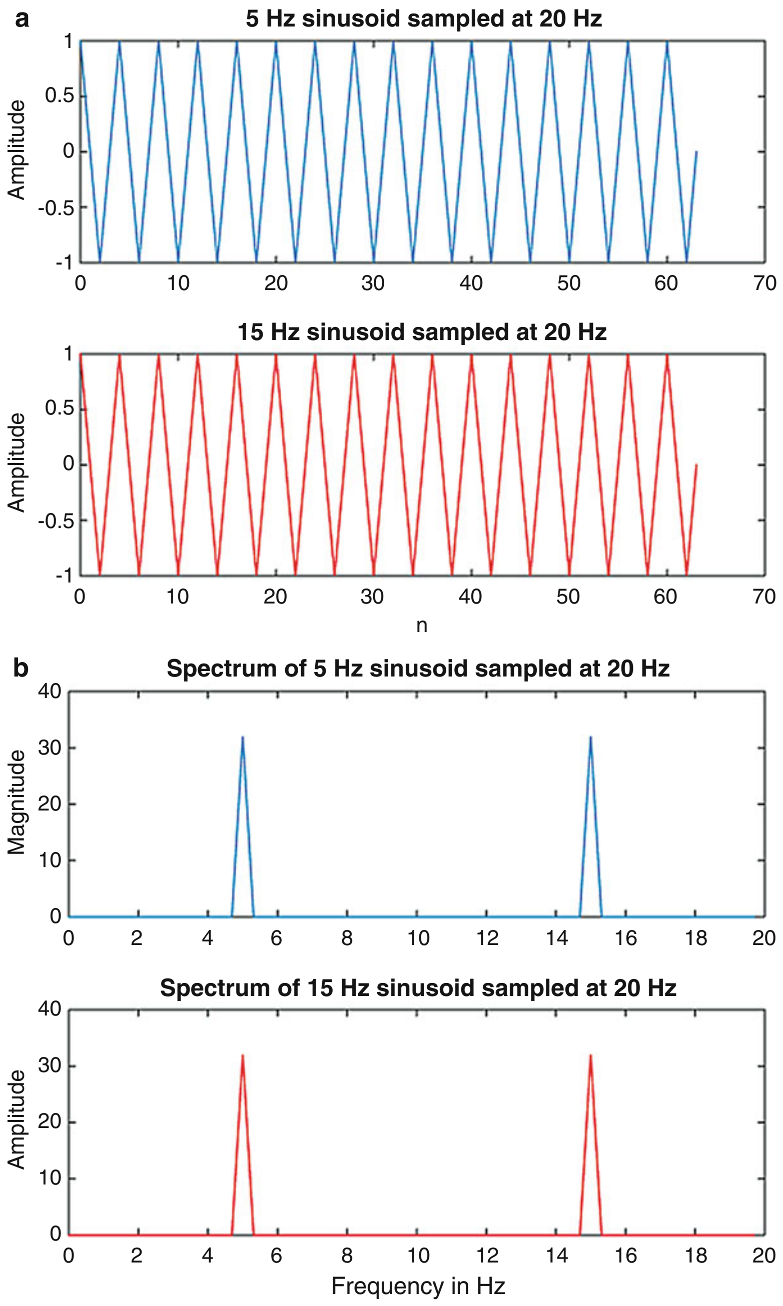
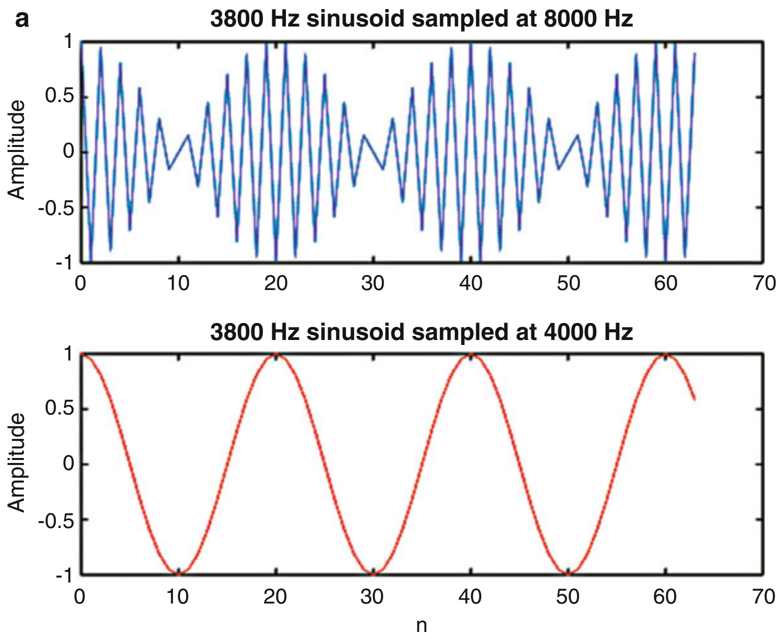


Fig. 2.11 Second example of aliasing distortion: (**a**) top plot, sampled 5 Hz sinusoid at a sampling frequency of 20 Hz; bottom plot, sampled 15 Hz sinusoid at a sampling frequency of 20 Hz. (**b**) Top plot, spectrum of 5 Hz signal at 20 Hz sampling rate; bottom plot, spectrum of 15 Hz signal sampled at 20 Hz rate. The 15 Hz signal appears as a 5 Hz signal



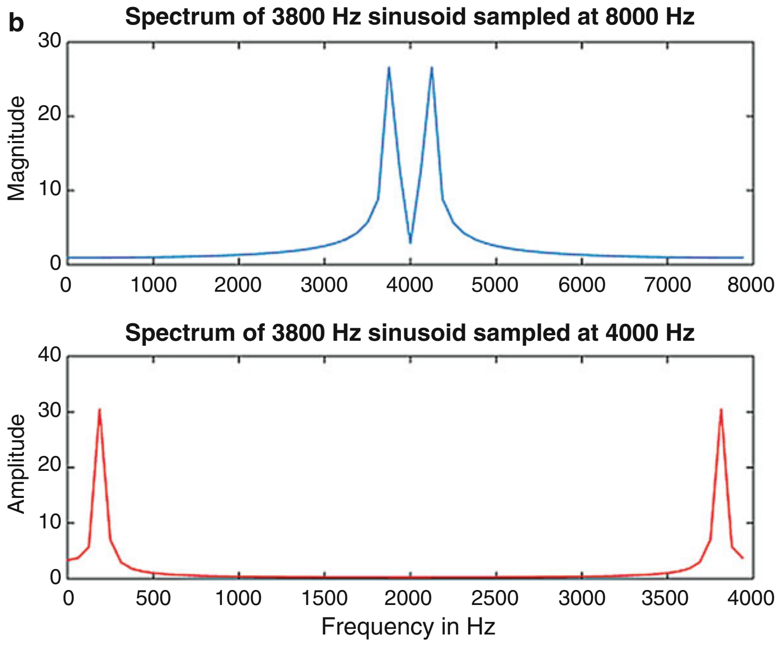


Fig. 2.12 A third example of aliasing distortion: (**a**) top plot, 3800 Hz signal at a sampling rate of 8000 Hz; bottom plot, same 3800 Hz signal sampled at a rate of 4000 Hz. (**b**) Top plot, spectrum of the signal at 8000 Hz sampling rate; bottom plot, spectrum of the same signal at 4000 Hz sampling rate. The 3800 Hz appears as 200 Hz signal

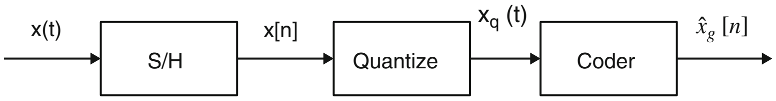


Fig. 2.13 A practical A/D converter model

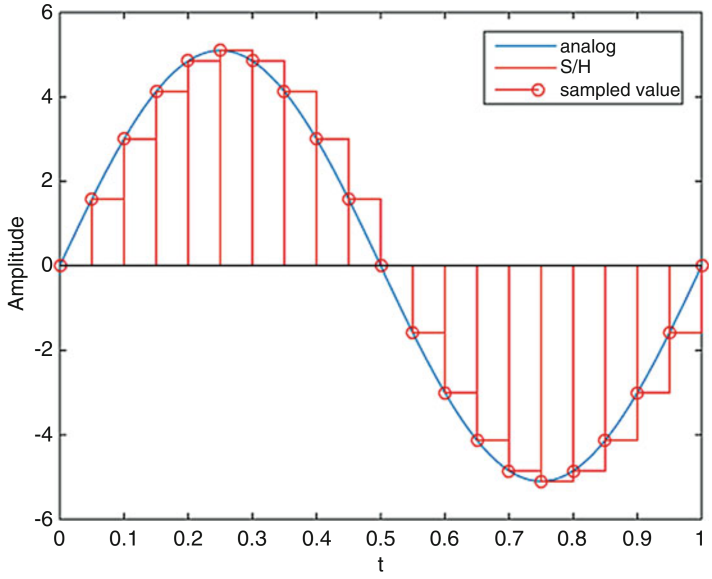


Fig. 2.14 Example of S/H function

other hand, accepts a vector of analog samples and outputs a vector of discrete samples that is close to the input vector. We will only deal with scalar quantizers here.

The design of a scalar quantizer amounts to dividing the input range of the analog signal amplitude into L + 1 levels and determining the corresponding L reconstruction or output levels. The relationship between the number of reconstruction levels L and the number of bits B of the quantizer is $L = 2^B$. The A/D converter has a fixed voltage or current amplitude limits for its input. For instance, it can have an input voltage limited to $\pm 1\ V$ or 0 to 1 volt. If the number of binary digits (bits) allowed in an A/D converter is B bits, then there are $L = 2^B$ number of levels between the minimum and maximum input amplitude range. The quantizer assigns the current sample value a level that is closest to the analog sample value. Denote the L + 1 input decision intervals by $\{D_j, 1 \le j \le L + 1\}$ and the corresponding L output levels by $\{R_k, 1 \le k \le L\}$. The quantizer maps an input analog sample x to its nearest neighbor and is formally expressed as

$$Q(x) = \widehat{x} = R_k \tag{2.57}$$

In general, the lower and upper boundaries of the input decision intervals are defined by

$$D_1 = x_{min} \& D_{L+1} = x_{max} \tag{2.58}$$

Let us assume that the output levels are chosen so that the following is satisfied.

$$R_1 < R_2 < R_3 \cdot \cdots \cdot R_L \tag{2.59}$$

For a given input analog signal, the decision boundaries and the corresponding output levels are chosen such that the mean square error (MSE) between the analog and the quantized samples is a minimum. The MSE is expressed as

$$MSE = E\left\{\left(x - \widehat{x}\right)^2\right\} = \int_{D_1}^{D_{L+1}} \left(x - \widehat{x}\right)^2 p_x(x)dx \tag{2.60}$$

In the above equation, E denotes the statistical average or expectation and $p_x(x)$ the probability density function (pdf) of the input analog samples. Because the quantized output value is constant equal to R_k over the interval $[D_k, D_{k+1})$, (2.60) can be rewritten as

$$MSE = \sum_{m=1}^{L} \int_{D_m}^{D_{m+1}} (x - R_m)^2 p_x(x)dx \tag{2.61}$$

From (2.61), we notice that the MSE is a function of both the decision boundaries and the output levels. Therefore, the minimum value of the MSE in (2.61) can be found by differentiating the MSE with respect to both the decision boundaries and the output levels and setting them to zero and then solving the two equations. Thus,

$$\frac{\partial MSE}{\partial D_i} = (D_i - R_{i-1})^2 p_x(D_i) - (D_i - R_i)^2 p_x(D_i) = 0 \tag{2.62a}$$

$$\frac{\partial MSE}{\partial R_i} = 2\int_{D_i}^{D_{i+1}} (x - R_i)p_x(x)dx = 0, 1 \le i \le L \tag{2.62b}$$

From (2.62a), we have

$$(D_i - R_{i-1})^2 = (D_i - R_i)^2 \tag{2.63}$$

Due to the fact that $R_i > R_{i-1}$, we determine the decision boundaries after simplifying (2.63) as

$$D_i = \frac{R_i + R_{i-1}}{2} \tag{2.64}$$

The output levels are obtained from (2.62b) as

$$R_i = \frac{\int_{D_i}^{D_{i+1}} x p_x(x) dx}{\int_{D_i}^{D_{i+1}} p_x(x) dx} \tag{2.65}$$

The implications of the optimal quantizer are that the decision boundaries lie at the midpoints of the corresponding output levels and the optimal output levels are the centroids of the decision intervals. Since these two quantities are interdependent, there is no closed-form solution to the two Eqs. (2.64) and (2.65). The solution is obtained by iteration. Also, note that the design of an optimal quantizer requires a priori knowledge of the pdf of the input analog samples. In other words, the optimal scalar quantizer is a function of the pdf of the input analog samples. This type of quantizer is known as the *Lloyd-Max* quantizer. The decision intervals and the corresponding output levels of a Lloyd-Max quantizer are nonuniform. Moreover, the Lloyd-Max quantizer is dependent on the input signal. Therefore, each new signal must have its own quantizer for optimal performance. However, a closed-form solution exists for a uniform quantizer, which we will describe next. The design of a uniform quantizer is simple and is the reason for its widespread use in image and video compression standards.

Uniform Quantizer If the pdf of the input analog samples is uniform, there exists a closed-form solution to the decision boundaries and the output levels for the Lloyd-Max quantizer. Under this condition, we will find the decision intervals and the output levels will all be equal. Hence the quantizer is known as the *uniform quantizer*. Obviously, a uniform scalar quantizer is optimum for analog samples that have uniform pdf. A uniform pdf implies

$$p_x(x) = \frac{1}{x_{max} - x_{min}} = \frac{1}{D_{L+1} - D_1} \tag{2.66}$$

Substituting (2.66) for the pdf in (2.65), the output levels of a scalar uniform quantizer are found to be

$$R_i = \frac{D_{i+1} + D_i}{2} \tag{2.67}$$

Using (2.67) in (2.64), the following relationship is found

$$D_{i+1} - D_i = D_i - D_{i-1} = \Delta, 2 \le i \le L \tag{2.68}$$

From the above equation, it is clear that the interval between two consecutive decision boundaries is the same and equals the quantization step size Δ. For instance, if the input amplitude range is between x_{min} and x_{max}, then for a B-bit quantizer, the step size is $\Delta = \frac{x_{max} - x_{min}}{2^B}$. The value assigned by the quantizer to an analog sample then lies half way between two consecutive levels. Therefore, the error or difference between a sample and its quantized value ranges between $\pm\frac{\Delta}{2}$. Larger the value of B smaller is the quantization error. Finally, the coder block assigns a B-bit binary code to the quantized level. Thus, a sample is converted to a digital number.

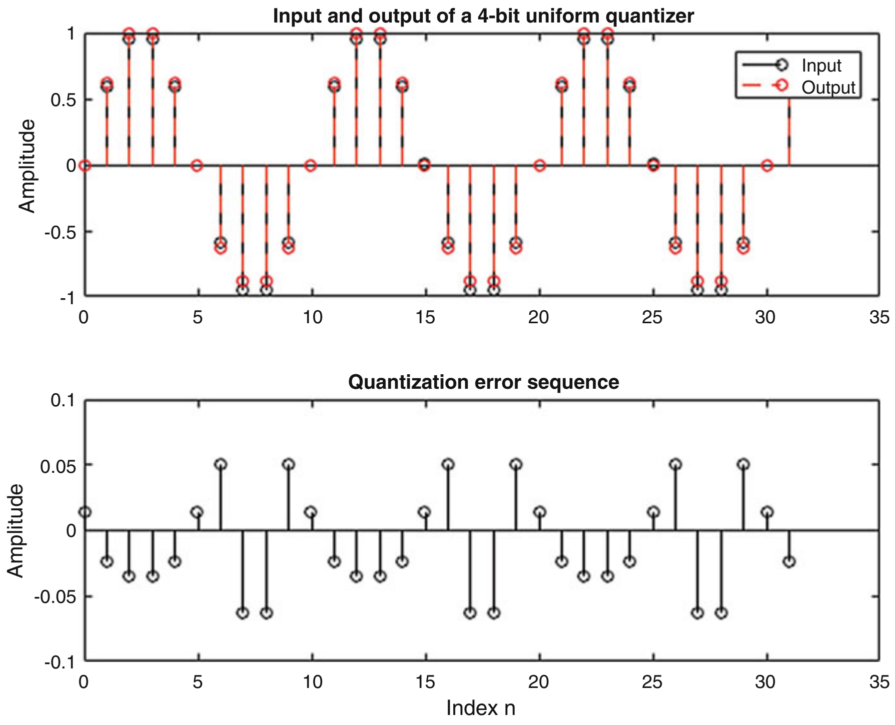


Fig. 2.15 Actual and quantized values of the sinusoid in Example 2.10 using a 4-bit uniform quantizer: top, input sinusoid in black stems and the quantized samples in red dashed stems; bottom, corresponding error sequence

Example 2.10 This example shows how to design a B-bit uniform quantizer based on an input sinusoid of specified amplitude, frequency, and sampling frequency. It then quantizes the sinusoid and calculates the resulting SNR in dB. The MATLAB M-file for this example is named *Example2_10.m*. The number of bits of quantization used in this example is 4. The amplitude range is ± 1. The resulting SNR is found to be 23.07 dB. The actual input and the corresponding quantized values are shown in the top plot in Fig. 2.15. The quantization error sequence is shown in the bottom plot of Fig. 2.15. The input-output characteristic of the 4-bit uniform quantizer for the sinusoid in this example is shown in Fig. 2.16. As required, there are 16 steps between the amplitude range of ± 1.

Coding the Quantized Values There are two ways to code the assigned level into a binary number. One way is to use what is called the *sign-magnitude* representation. In this method the magnitude of the sample value is represented by a b-bit binary number and a sign bit as the most significant bit (MSB) to indicate its sign. Thus, there are $B = b + 1$ bits in this digital representation. In the second method called *two's complement*, positive fractions are represented as in the sign-magnitude form. A negative fraction is represented in two's complement form as follows: first the

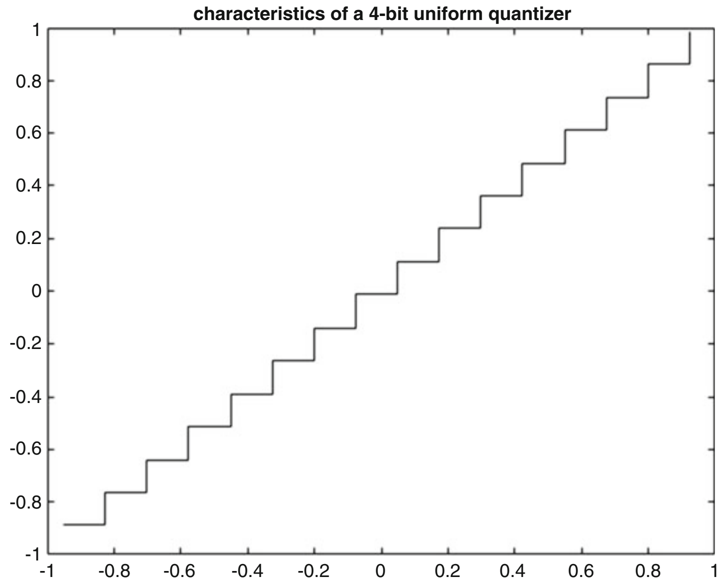


Fig. 2.16 Input-output characteristic of the 4-bit uniform quantizer of Example 2.10

magnitude is represented in binary number, the bits are complemented, and then a 1 is added to the least significant bit (LSB) to get the two's complement representation. For instance, let us represent the decimal fraction -0.875 in the two abovementioned formats. In sign-magnitude format, we first represent the magnitude of the decimal fraction in binary fraction, which is $0.875_{10} = 0.1110_2$. Then a "1" bit is inserted in the MSB position to get the sign-magnitude representation: $-0.875_{10} = 1.1110_2$. The "1" in the MSB corresponds to a negative value, and a "0" in the MSB corresponds to a positive value. The same decimal value in two's complement form is obtained by complementing each bit of the magnitude and then adding a "1" to the LSB. So, the complement of 0.875_{10} is 1.0001_2. Adding a "1" to the LSB gives the number $1.0001_2 + 0.0001_2 = 1.0010_2$. Thus, the two's complement representation of -0.875_{10} is 1.0010_2. As an example of a 3-bit A/D converter, Fig. 2.17 shows the input-output characteristics using two's complement representation. As can be seen from the figure, all input values greater than or equal to $\frac{7\Delta}{2}$ are assigned the same value of 3Δ. Similarly, all input values less than or equal to $-\frac{9\Delta}{2}$ are assigned the same value -4Δ.

2.8 Performance of A/D Converters

A/D converters come with different bit widths. Some are 8-bit converters, some are 12-bit converters, and others are 14- or 16-bit converters. We mentioned earlier that the output of an A/D converter is an approximation to the input samples. The degree

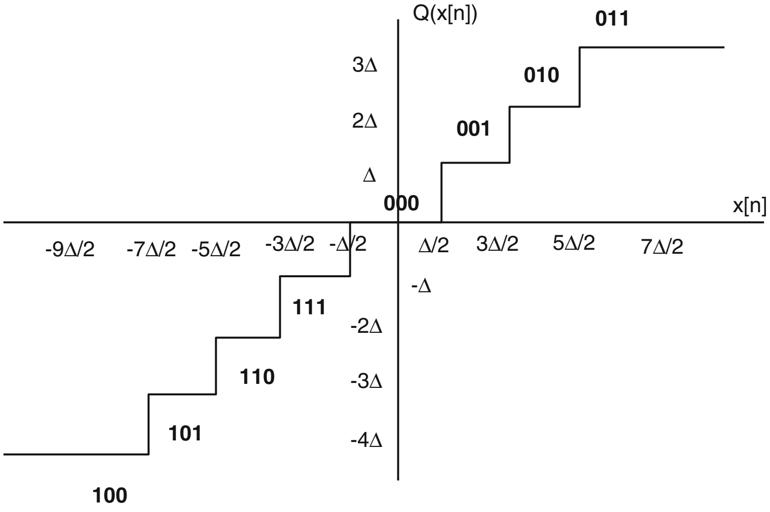


Fig. 2.17 Input-output characteristic of a 3-bit analog-to-digital converter

of approximation is a function of the bit width of the A/D converter. Because of the approximation carried out by an A/D converter, errors occur between the analog samples and the digital samples. This type of error is random and so is considered as noise. That is, there is no definite analytical expression to describe the errors sample by sample. It is, therefore, convenient and proper to describe the error due to quantization in terms of its averages. The most commonly used measure to describe noise is the variance. In order to estimate the variance of the noise due to quantization, one has to know its distribution. Here, by distribution we mean the probability of occurrence of the amplitudes of the noise due to quantization. In practice it is found that the quantization error is uniformly distributed between the range $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$. That is to say that all amplitudes in this interval have the same probability of occurrence. Mathematically speaking, the uniform distribution is expressed as

$$p(e) = \frac{1}{\Delta}, \; -\frac{\Delta}{2} \le e \le \frac{\Delta}{2} \tag{2.69}$$

with e corresponding to the possible amplitude of the quantization error. The mean or average value of the quantization error μ_e is obtained from

$$\mu_e = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} ep(e)de = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} ede = \frac{1}{2\Delta}e^2\Big|_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} = 0 \tag{2.70}$$

So, the mean value of the quantization error is zero. The variance σ_e^2 of the quantization error is obtained from

$$\sigma_e^2 = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} (e-\mu_e)^2 p(e)de = \frac{1}{\Delta}\int_{\frac{-}{\Delta 2}}^{\frac{\Delta}{2}} e^2 de = \frac{1}{3\Delta}e^3\Big|_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} = \frac{\Delta^2}{12} \tag{2.71}$$

From Eq. (2.71), we notice that the variance of the noise due to quantization is proportional to the square of the step size. In terms of the word length B of the A/D converter, Eq. (2.71) amounts to

$$\sigma_e^2 = \frac{(x_{max} - x_{min})^2 2^{-2B}}{12} \tag{2.72}$$

The quantization noise variance of the A/D converter decreases exponentially with increasing bit width. The quantization noise variance alone is not enough to judge its effect on the processed signal. It depends on the power or variance of the analog signal being processed digitally. In other words, the effect of the noise due to quantization depends on its variance relative to the signal variance. This is defined by the signal-to-noise ratio (SNR) and is usually expressed in decibel or dB for short. If the amplitude of the analog signal is assumed to be *uniformly distributed* in the range $\{x_{min}, x_{max}\}$, then its variance is found to be

$$\sigma_x^2 = \frac{(x_{max} - x_{min})^2}{12} \tag{2.73}$$

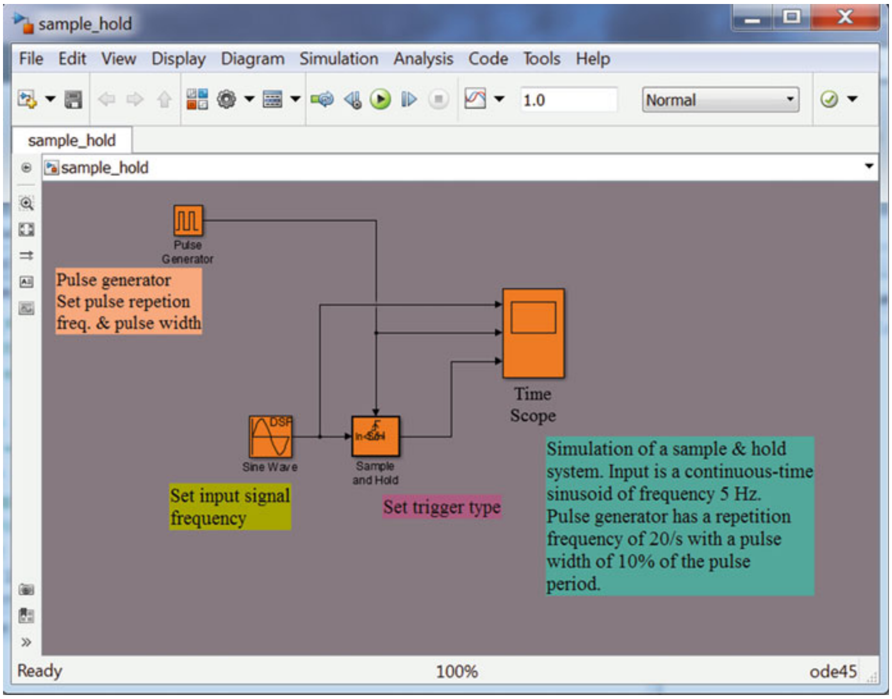
Then the SNR in dB of the A/D converter with B-bit bit width is defined as

$$SNR = 10log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10log_{10}\left(\frac{\frac{(x_{max}-x_{min})^2}{12}}{\frac{(x_{max}-x_{min})2}{12}2^{-2B}}\right) \approx 6.02B, dB \tag{2.74}$$

From Eq. (2.74), we observe that the SNR in dB increases linearly with B. If B is increased by 1 bit, then the SNR increases by approximately 6 dB. That is to say that each additional bit in the A/D converter yields an improvement of about 6 dB in the resulting SNR.

MATLAB Examples We can simulate A/D converters using the MATLAB Simulink system. Before building a hardware system, it is wise to first simulate it to assess its performance. If the system does not meet the target performance, one can fine-tune the design parameters and rerun the simulation to verify its performance. Thus, simulation not only saves time and energy but also guarantees performance. Simulink is a very useful tool in simulating algorithms and hardware systems. We will first demonstrate the sample and hold operation, using Simulink.

S/H Example Using Simulink Figure 2.18 shows the block diagram simulating the S/H function. It consists of a continuous-time signal source, a pulse generator as the sampler, S/H block, and a time scope to display the signals. In order to draw the block diagram, first start the Simulink by clicking the *Simulink Library* on the toolbar menu on the MATLAB window or type *simulink* in the workspace. A Simulink Library Browser appears. Click the arrow pointing downward next to the icon showing simulink diagram on the toolbar, and select *New Model*. A new window appears. Select the *DSP System Toolbox* and then *Sources*. A list of sources appears on the right side as shown in Fig. 2.19. Let us choose the *Sine Wave* source and drag it to new untitled window. Double-click the Sine Wave block, and specify



Pulse generator
Set pulse repetion
freq. & pulse width

Set input signal frequency

Set trigger type

Simulation of a sample & hold system. Input is a continuous-time sinusoid of frequency 5 Hz. Pulse generator has a repetition frequency of 20/s with a pulse width of 10% of the pulse period.

Fig. 2.18 Block diagram of sample and hold function using Simulink

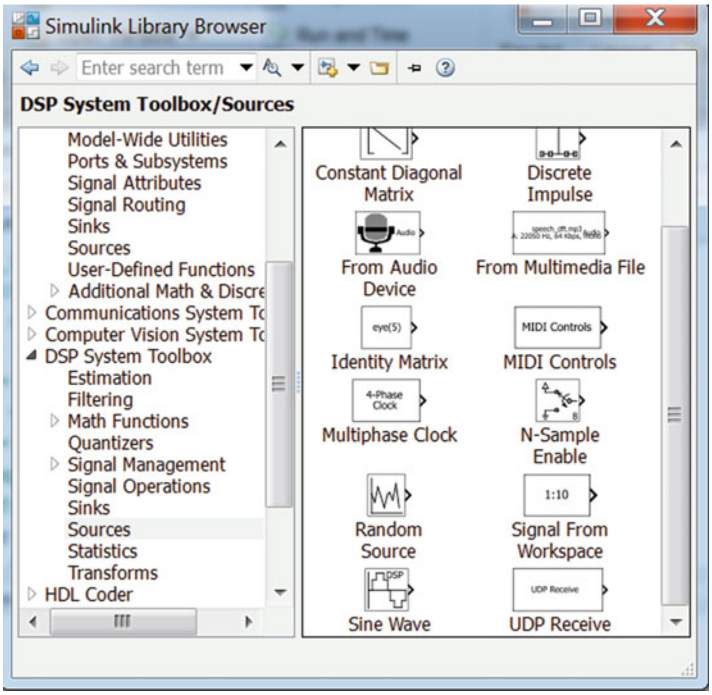
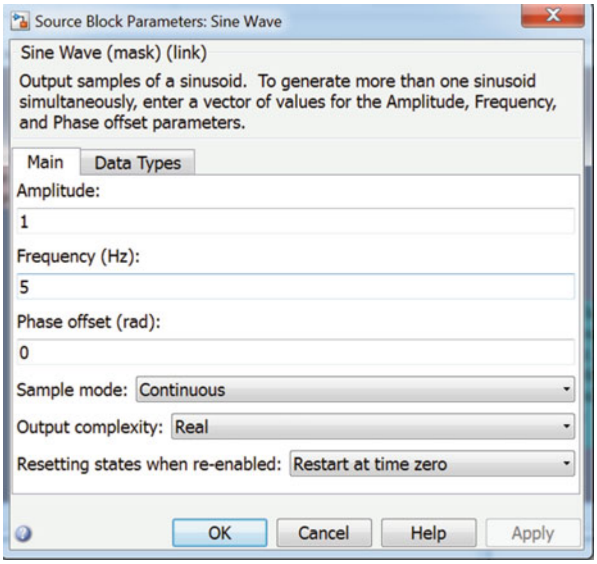


Fig. 2.19 Simulink Library Browser


Source Block Parameters: Sine Wave

Sine Wave (mask) (link)

Output samples of a sinusoid. To generate more than one sinusoid simultaneously, enter a vector of values for the Amplitude, Frequency, and Phase offset parameters.

Amplitude: 1

Frequency (Hz): 5

Phase offset (rad): 0

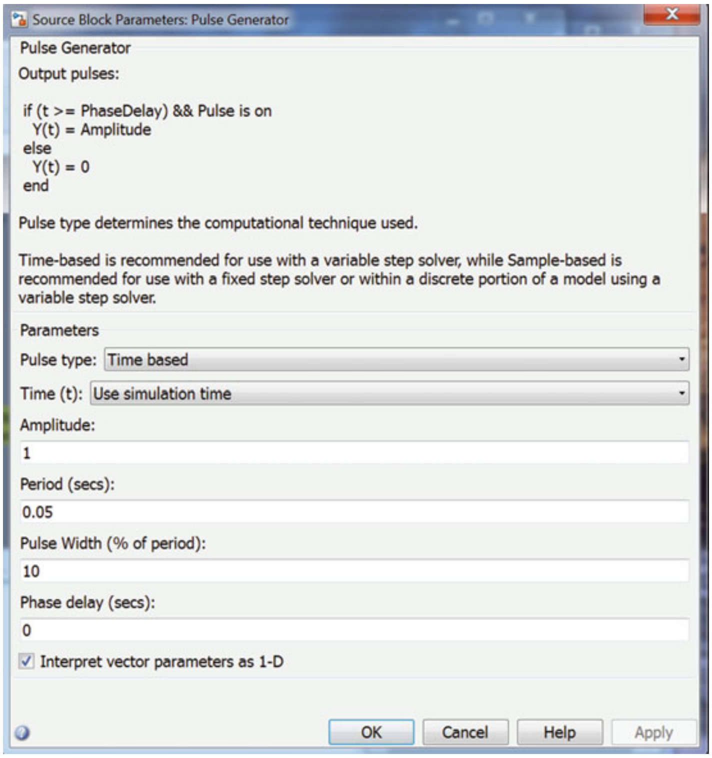
Sample mode: Continuous

Output complexity: Real

Resetting states when re-enabled: Restart at time zero

Fig. 2.20 Block parameters for sine wave source

the parameters as shown in Fig. 2.20. Next, under *Simulink*, choose *Sources* and then *Pulse Generator* and drag it to the untitled window. Double-click the *Pulse Generator* and fill in the blanks as shown in Fig. 2.21. Next, choose *DSP System Toolbox* and then *Signal Operations*. From the list of blocks that appear on the right side, choose the *Sample and Hold* block, and drag it to the untitled window. Double-click the block, and fill in the blanks as shown in Fig. 2.22. Finally, choose *Sinks* under *DSP System Toolbox*. From the list of sinks, choose *Time Scope*, and drag it to the untitled window. Double-click the *Time Scope*. A time scope appears as shown in Fig. 2.23. Figure 2.23 also shows the three time displays. This window was captured after simulation. To display three signals in three rows, click *View*, and choose *Layout* from the time scope window. By choosing *Configuration Properties* under *View* in the toolbar, we can label the three displays. Now we have all the sub-blocks and the corresponding parameters. We need to connect them in the order shown in Fig. 2.18. To connect the output of one block to the input of a second block, first click the output node, and then move the mouse to the input node of the second block while pressing the left mouse button. Thus, we have the complete system. Next, save the diagram with a name. To start the simulation, click the green button with an arrow pointing to the right on the toolbar. If everything is syntactically correct, MATLAB performs the simulation, and the various signals are displayed on the time scope, as indicated in Fig. 2.23. In this example the simulation time is chosen to be 1 s. One can change the simulation time to suit the needs. In this example the input continuous-time signal is a sine wave with a frequency of 5 Hz and amplitude unity. Since the simulation time is chosen to be 1 s, there are five cycles of the sine wave, as displayed on the topmost display. The middle display displays the pulse train of the pulse generator. The pulse period is 0.05 s. Therefore, there are 20 pulses in 1 s, as can be seen in the display. The bottommost display is the sample and hold signal.


Source Block Parameters: Pulse Generator

Pulse Generator

Output pulses:

```
if (t >= PhaseDelay) && Pulse is on
  Y(t) = Amplitude
else
  Y(t) = 0
end
```

Pulse type determines the computational technique used.

Time-based is recommended for use with a variable step solver, while Sample-based is recommended for use with a fixed step solver or within a discrete portion of a model using a variable step solver.

Parameters

Pulse type: Time based

Time (t): Use simulation time

Amplitude: 1

Period (secs): 0.05

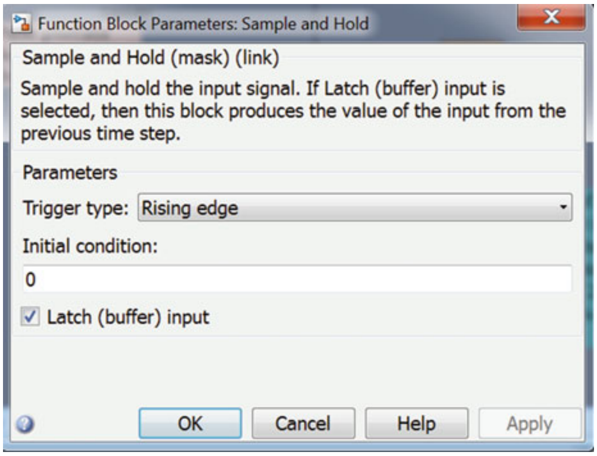
Pulse Width (% of period): 10

Phase delay (secs): 0

✓ Interpret vector parameters as 1-D

Fig. 2.21 Block parameters for pulse generator source

Fig. 2.22 Block parameters for sample and hold function


Function Block Parameters: Sample and Hold

Sample and Hold (mask) (link)

Sample and hold the input signal. If Latch (buffer) input is selected, then this block produces the value of the input from the previous time step.

Parameters

Trigger type: Rising edge

Initial condition: 0

✓ Latch (buffer) input

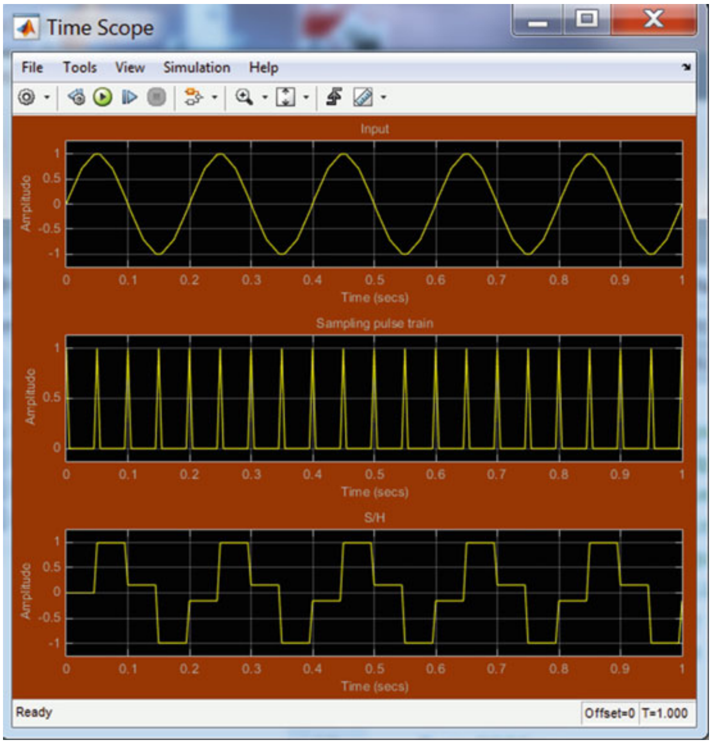


Fig. 2.23 Time scope

The analog signal is sampled at the rising edge of the pulse and held for about 45 ms duration. The pulse period is 50 ms, and the pulse width is 5 ms. So the S/H block holds the sample value for approximately 45 ms. Since there are 20 pulses in 1 s, there are 20 samples in that duration, as seen in Fig. 2.23.

Simulation of A/D Converter Using Simulink An A/D converter as shown in Fig. 2.13 first samples the input continuous-time signal and holds until the conversion is completed. We have simulated this sample and hold process as described above. Next, we will expand on this and include a quantizer to complete the A/D conversion process. Figure 2.24 shows the block diagram of an A/D converter using Simulink. As can be seen from the figure, we have included two input signal sources: a sine wave signal generator and a random signal generator. The sine wave function generates a continuous-time signal with an amplitude unity and a frequency of 5 Hz as shown in Fig. 2.25. The random signal generator produces a uniformly distributed random signal with amplitudes between -1 and $+1$ and is also a continuous function of time as seen in Fig. 2.26, which lists the parameters of the random signal generator. A manual switch is used to switch the input source between sine wave and random signal. Before starting the simulation, we have to double-click the switch to change the input

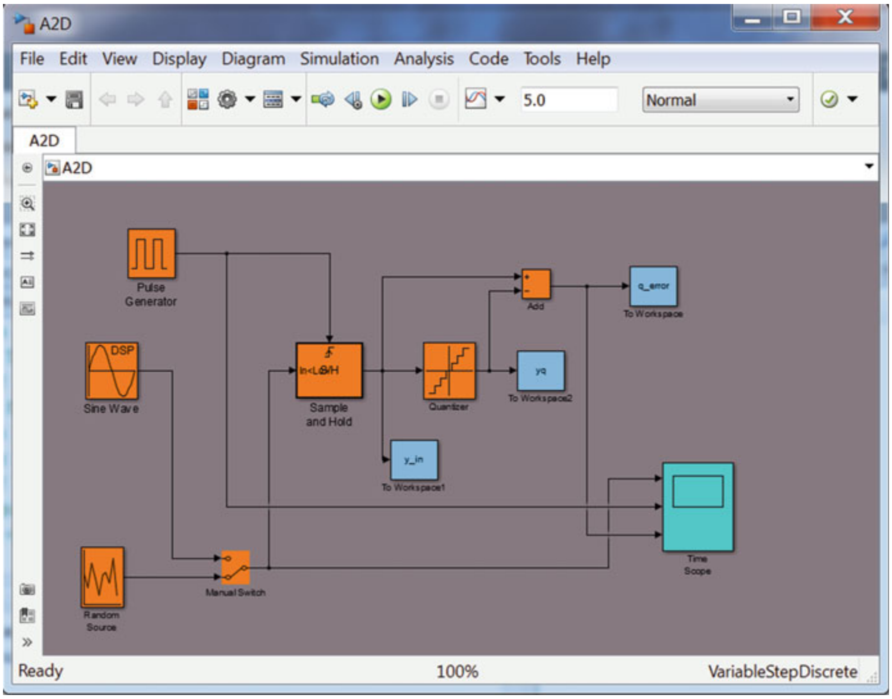
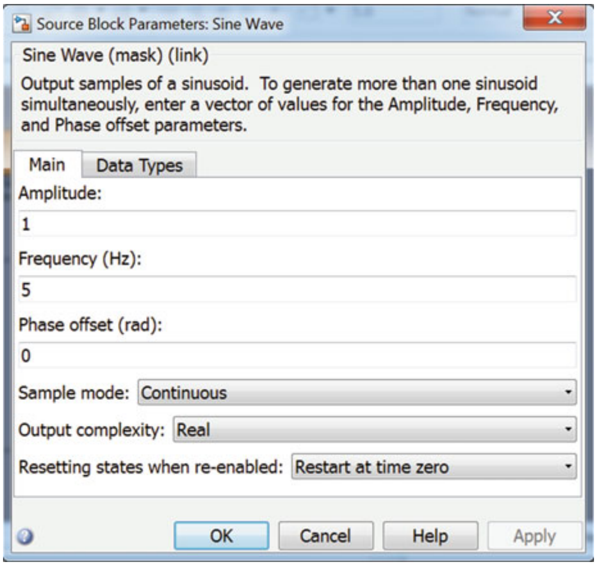


Fig. 2.24 Block diagram of an A/D converter using Simulink

Fig. 2.25 Parameters of the sine wave generator of Fig. 2.24


Source Block Parameters: Sine Wave

Sine Wave (mask) (link)

Output samples of a sinusoid. To generate more than one sinusoid simultaneously, enter a vector of values for the Amplitude, Frequency, and Phase offset parameters.

Main | Data Types

Amplitude:
1

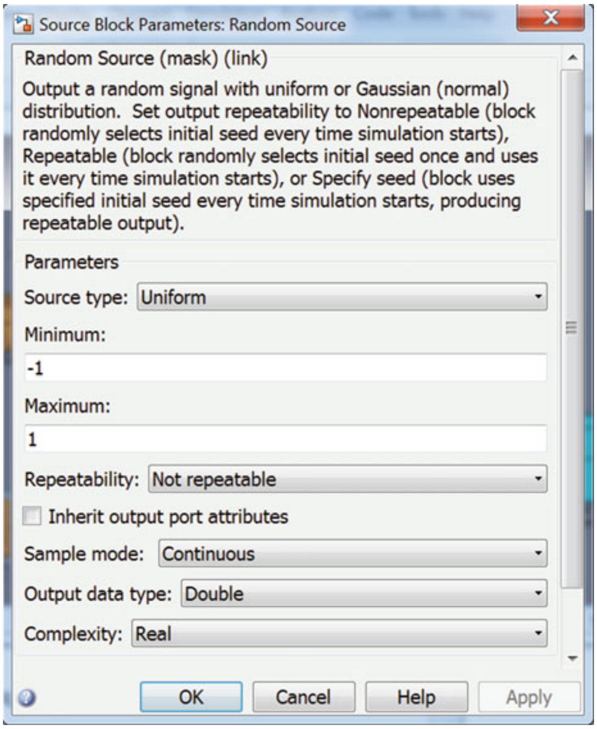
Frequency (Hz):
5

Phase offset (rad):
0

Sample mode: Continuous

Output complexity: Real

Resetting states when re-enabled: Restart at time zero


Source Block Parameters: Random Source

Random Source (mask) (link)

Output a random signal with uniform or Gaussian (normal) distribution. Set output repeatability to Nonrepeatable (block randomly selects initial seed every time simulation starts), Repeatable (block randomly selects initial seed once and uses it every time simulation starts), or Specify seed (block uses specified initial seed every time simulation starts, producing repeatable output).

Parameters

Source type: Uniform

Minimum: -1

Maximum: 1

Repeatability: Not repeatable

☐ Inherit output port attributes

Sample mode: Continuous

Output data type: Double

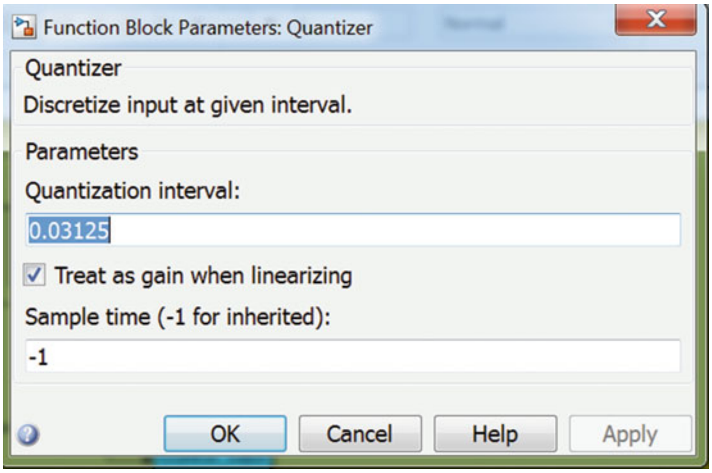
Complexity: Real

Fig. 2.26 Parameters of the random signal generator of Fig. 2.24

source. The S/H block has the same parameters as that used in the S/H example. The quantizer has 6 bits of quantization, and its parameters are shown in Fig. 2.27. Three outputs are generated in the simulation, namely, the output of S/H, the output of the quantizer, and the difference between these two signals. They are stored as vectors in the workspace with names as indicated in Fig. 2.24. The time scope displays the chosen input signal on the top plot, the pulse generator output on the middle plot, and the quantization error on the bottom plot, as shown in Fig. 2.28a. In this figure the input source is the sine wave. In Fig. 2.28b the random input signal is shown. With 1 min of simulation, the SNR due to quantization is found to be 36.36 dB for the sine wave and 36.11 dB for the random signal. These numbers agree with the SNRs obtained from analysis when the A/D bit width is 6 bits.

2.9 Summary

In this chapter we started with the mathematical description of discrete-time signals in general and some of the useful discrete-time signals in particular. Next we described linear time-invariant (LTI) discrete-time systems. Specifically, we


Function Block Parameters: Quantizer

Quantizer

Discretize input at given interval.

Parameters

Quantization interval:

0.03125

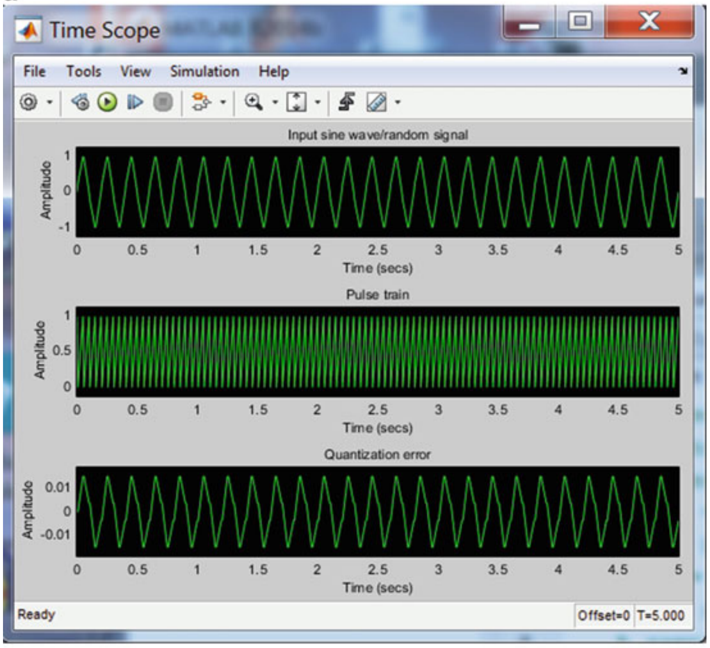
[✓] Treat as gain when linearizing

Sample time (-1 for inherited):

-1

Fig. 2.27 Parameters of the uniform quantizer of Fig. 2.24

described how an LTI system is characterized in terms of its impulse response and how its response to any given input discrete-time signal is computed via convolution sum. We then established the condition that the impulse response must satisfy for the LTI system to be stable. Next we introduced an alternative method of describing an LTI discrete-time system, namely, the linear difference equation with constant coefficients. We showed how the response of an LTI discrete-time system to a specified input sequence could be obtained in closed-form solution. Several examples were included to nail the concept. Since discrete-time signals are mostly obtained from continuous-time or analog signals, we stated the sampling theorem also called Nyquist's theorem and showed how the analog signals can be recovered or reconstructed from their discrete-time counterparts. If the Nyquist sampling criterion is not satisfied, aliasing distortion will occur, and it cannot be removed. We exemplified this notion using a few examples. The sampling theorem that we talked about pertains to lowpass signals. Often, bandpass signals are encountered, especially in the field of communications. These signals are centered at a very high frequency with a narrow bandwidth. The sampling rate of these bandpass signals will be very high if Nyquist's condition is used. Instead, one can sample a bandpass signal at a much lower rate without incurring aliasing distortion. We verified this statement using an example. The next logical thing to do is to describe the process of converting an analog signal to digital signal. We described the A/D converter function by function with plots to illustrate the results. Since A/D conversion involves the approximation of analog samples using fixed number bits of representation, errors occur between the analog and digital values. These errors propagate through the discrete-time system and manifest as noise in the output. We, therefore, derived mathematical formula to measure the performance of an A/D converter. This formula expresses the signal-to-noise ratio of an A/D converter in terms of the number of bits used in the A/D converter. Finally we simulated the S/H operation a



b

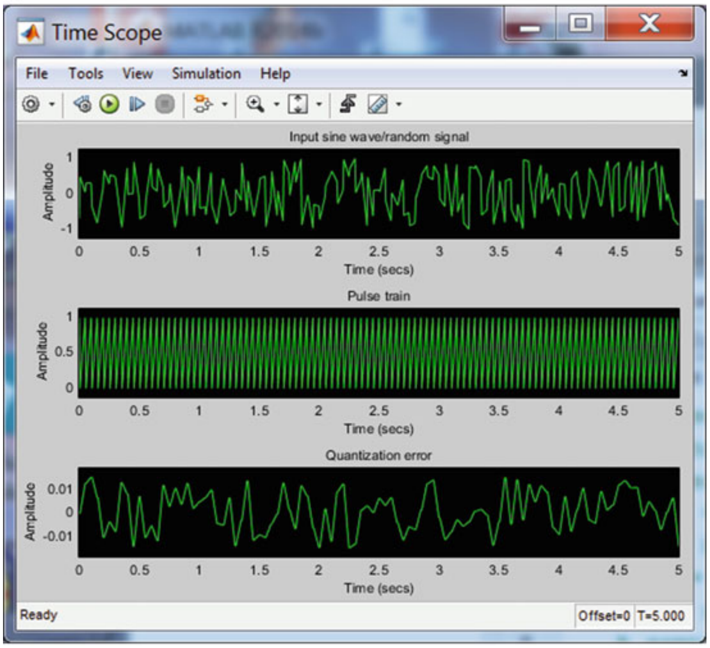


Fig. 2.28 Time scope display of signals used in Fig. 2.24: (**a**) sine wave, (**b**) random signal

as well as the complete A/D conversion using MATLAB's Simulink system. In the next chapter, we will deal with Z-transform and its use in describing LTI discrete-time systems. We will show what role the Z-transform plays in the analysis and design of discrete-time systems.

2.10 Problems

1. Give a few good reasons why we always deal with linear systems even if the actual systems are nonlinear.
2. Is the sequence, $x[n] = \alpha^n u[n]$, $| \alpha | \prec 1$ absolutely summable?
3. Is the sequence $x[n] = n^2 \alpha^n u[n]$, $| \alpha | \prec 1$ absolutely summable?
4. Consider the discrete-time system described by $y[n] = \alpha x^2[n]$ with α a real constant. Will you use this system to amplify a sinusoidal signal? If not, why? Explain.
5. Find the fundamental period of the sequence $x[\mathrm{n}] = \cos(0.8\mathrm{n}\pi + 0.25\pi)$.
6. What is the period of the sequence $x[n] = \sin\left[\frac{2\pi x 100}{1000} n\right] + \sin\left[\frac{2\pi x 150}{1000} n\right]$?
7. If x[n], y[n], and g[n] represent three finite-length sequences of lengths N, M, and L, respectively, with the first sample of each sequence occurring at n = 0, what is the length of the sequence $x[n] \otimes y[n] \otimes g[n]$?
8. Evaluate the linear convolution of $x[n]$ with itself, where $x[\mathrm{n}] = \{1, -1, 0, 1, -1\}$, $0 \le \mathrm{n} \le 4$.
9. Determine if the system described by $\mathrm{y[n]} = \alpha + \mathrm{x[n+1]} + \mathrm{x[n]} + \mathrm{x[n-1]} + \mathrm{x}[n-2]$ is (a) linear, (b) causal, (c) shift-invariant, and (d) stable.
10. Determine if the system described by $\mathrm{y[n]} = \mathrm{x[n+1]} + \mathrm{x[n]} + \mathrm{x[n-1]} + \mathrm{x[n-2]}$ is causal.
11. Consider the two discrete-time LSI systems whose impulse responses are described by $h_1[n] = \begin{cases} 1, 0 \le n \le N-1 \\ 0, \textit{otherwise} \end{cases}$ and $h_2[n] = \begin{cases} 1, -(N-1) \le n \le 0 \\ 0, \textit{otherwise} \end{cases}$. If a unit step sequence is applied to both systems, what will be their responses?
12. Obtain the total solution for $n \ge 0$ of the discrete-time system described by $y[n] - 0.4y[n-1] - 0.05y[n-2] = 1.5\cos(10\pi n)u[n]$, with initial conditions $y[-1] = 1$, $y[-2] = 0$.
13. Determine the impulse response of the system described in Problem 12 above.
14. Determine the rise time of the discrete-time system described in Problem 12. You may use MATLAB to solve the problem.
15. Find the impulse response and step response of the discrete-time system described by $y[n] - 0.7y[n-1] + 0.1y[n-2] = x[n] - 0.7x[n-1]$.

References

1. McGillem CD, Cooper GR (1974) Continuous and discrete signal and system analysis. Holt, Rhinehart and Winston, New York
2. Mitra SK (2011) Digital signal processing: a computer-based approach, 4th edn. McGraw Hill, New York
3. Oppenheim AV, Schafer RW (1975) Digital signal processing. Prentice-Hall, Englewood Cliffs
4. Papamichalis P (1990) Digital signal processing applications with the TMS320 family: theory, algorithms, and implementations, vol 3. Texas Instruments, Dallas
5. Proakis JG, Manolakis DG (1996) Digital signal processing: principles, Algorithms and Applications, 3rd edn. Prentice-Hall, Upper Saddle River