A complex-valued sequence $x_e[n]$ is called conjugate-symmetric if $x_e[n] = x_e^*[-n]$ and a complex-valued seq. $x_o[n]$ is called conjugate-antisymmetric if $x_o[n] = -x_o^*[-n]$. Any arbitrary complex-valued sequence $x[n]$ can be decomposed into $x[n] = x_e[n] + x_o[n]$ where $x_e[n]$ and $x_o[n]$ are given by

$$x_e[n] = \frac{1}{2}\left(x[n] + x^*[-n]\right) \quad \text{and} \quad x_o[n] = \frac{1}{2}\left(x[n] - x^*[-n]\right).$$

Discrete-Time Signals and Systems

2

This chapter reviews some of the basic principles and definitions of digital signal processing. It uses several representations for signals, including functional expressions, graphs, and sequences of numbers. Pictorial representations of signals are particularly useful because they provide visual insight. Figure 2.1 graphically represents the sequence $x[n]$ as a function of the index n.

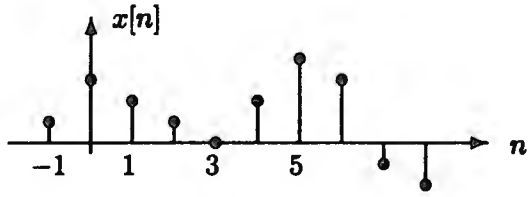


Figure 2.1. Pictorial representation of a sequence $x[n]$.

Equally important to a discussion of discrete-time signal processing is the concept of a discrete-time system. Section 2.2 discusses discrete-time systems and means for classifying them. Linear, time-invariant systems are highlighted because they have special properties that make them relatively easy to analyze, implement, and design.

2.1 DISCRETE-TIME SIGNALS

A number of elementary signals appear frequently in signal processing applications. Some of these are listed below and are pictured in Fig. 2.2.

- The *unit sample* or *unit impulse*

$$\delta[n] \triangleq \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{otherwise;} \end{cases}$$

- The *unit step*

$$u[n] \triangleq \begin{cases} 1, & \text{if } n \geq 0 \\ 0, & \text{if } n < 0; \end{cases}$$

- The *complex exponential*

$$e^{j\omega_0 n}$$

(ω_0 is called the *frequency*);

- The *sinusoid*

$$\sin[\omega_0 n + \theta_0]$$

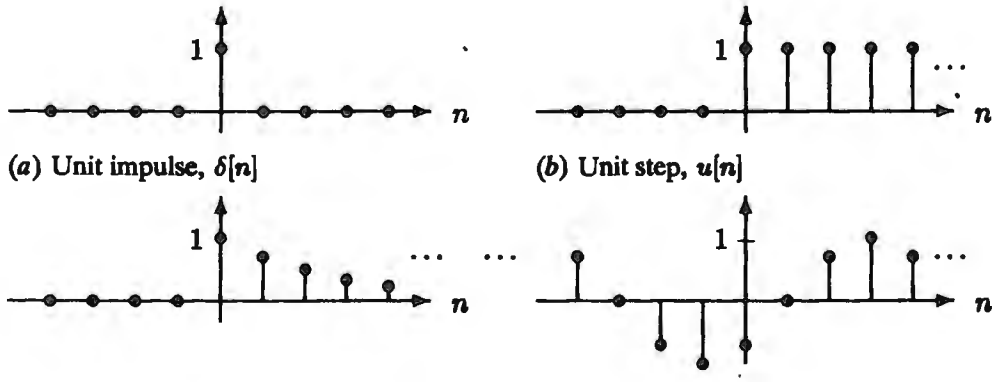
(ω_0 is the frequency and θ_0 is the *phase offset*);

- The *one-sided exponential*

$$a^n u[n],$$

where $|a| < 1$ and $u[n]$ is the step function.

These elementary signals play an important role in describing more complex signals. For example, many useful signals can be expressed as weighted sums of damped exponentials or sinusoids. These and other elementary signals are examined in the exercises at the end of this section.



(*a*) Unit impulse, $\delta[n]$ (*b*) Unit step, $u[n]$

(*c*) One-sided exponential (*d*) Sinusoid

Figure 2.2. Some common signals.

The signal processing literature is filled with many terms that describe signals. These constitute a basic DSP vocabulary. These terms and acronyms are usually based

on general characteristics of the signal. One such characteristic is whether the sequence is of *finite* or *infinite length*. A finite length sequence has a leftmost nonzero sample and a rightmost nonzero sample. The unit impulse is an example of a finite length sequence. An infinite length sequence extends infinitely far in one or both of the two directions. The unit step, exponential, and sinusoid are all examples of infinite length sequences.

Another useful signal characteristic is periodicity. A *periodic* sequence, $x[n]$, is infinite in duration and satisfies the relationship

$$x[n] = x[n+N] \qquad -\infty < n < \infty$$

for some $N > 0$ called the *period*. The period N is always restricted to be a finite positive integer. Signals that are not periodic are called *aperiodic*. Aperiodic sequences may be of either finite or infinite duration and do not repeat indefinitely.

Signals can also be classified by whether their sample values are real or complex. Although most signals encountered in applications are real, it is often useful to work with complex signals in analyzing systems. Complex sequences can be represented in terms of their real and imaginary parts, $\mathcal{R}e\{x[n]\}$ and $\mathcal{I}m\{x[n]\}$, which are both real sequences, or in terms of their magnitudes and phases, $|x[n]|$ and $\angle x[n]$. For a complex sequence, $x[n]$, these are defined by

$$\begin{aligned} x[n] &\triangleq \mathcal{R}e\{x[n]\} + j\,\mathcal{I}m\{x[n]\} \\ &\triangleq |x[n]|e^{j\angle x[n]} \end{aligned}$$

The *complex conjugate* of $x[n]$ is the sequence $x^*[n]$.

$$\begin{aligned} x^*[n] &\triangleq \mathcal{R}e\{x[n]\} - j\,\mathcal{I}m\{x[n]\} \\ &\triangleq |x[n]|e^{-j\angle x[n]} \end{aligned}$$

Chapter 6 explores complex sequences in greater detail.

Signals may also be classified as *deterministic* or *random*. Deterministic sequences are sequences that can be expressed in a functional form. These include elementary signals such as sinusoids, steps, ramps, and the like, and arbitrary sequences composed of a weighted combination of elementary signals. Random signals, on the other hand, are probabilistic and "noise-like." The individual values in the sequence are less important than their distribution. Virtually all of the signals examined here are deterministic, but a few exercises at the end of this chapter and in the next focus on the generation of random signals and their properties.

Finally, signals may be classified according to their symmetry characteristics. Not all signals are symmetric, but all signals can be decomposed into even and odd symmetric sequences. Any arbitrary real sequence, $x[n]$, can be viewed as having two additive components: an *even part*, $x_e[n] = ev\{x[n]\}$, defined as

$$x_e[n] = \tfrac{1}{2}(x[n] + x[-n]);$$

and an *odd part*, $x_o[n] = od\{x[n]\}$, defined as

$$x_o[n] = \tfrac{1}{2}(x[n] - x[-n]).$$

The sum of the even and odd parts results in the original sequence. This concept is easily generalized to include complex signals of the form

$$x[n] = a[n] + jb[n]$$

where $a[n]$ and $b[n]$ are the real and imaginary parts, respectively. All complex sequences can be expressed as the sum of the four terms shown below

$$x[n] = \underbrace{a_e[n]}_{\text{real-even}} + \underbrace{a_o[n]}_{\text{real-odd}} + j\underbrace{b_e[n]}_{\text{imaginary-even}} + j\underbrace{b_o[n]}_{\text{imaginary-odd}} \tag{2.1}$$

where the subscripts e and o denote the even and odd parts, respectively. Examining sequences in terms of their symmetry can sometimes greatly simplify their analysis and enable difficult problems to be solved in a simple way.

Summations involving sequences often arise in analyzing discrete-time systems. These summations can be expressed explicitly using the summation operator, $\sum$. However, in many cases these summations can be reduced to compact, closed-form expressions. Three of these can be derived from the geometric series:

$$\sum_{n=0}^{N-1} a^n = \frac{1 - a^N}{1 - a}, \qquad \forall\ a, \tag{2.2}$$

where the symbol $\forall$ denotes "for all";

$$\sum_{n=0}^{\infty} a^n = \frac{1}{1 - a}, \qquad \text{if } |a| < 1; \tag{2.3}$$

and

$$\sum_{n=0}^{\infty} na^n = \frac{a}{(1 - a)^2}, \qquad \text{if } |a| < 1. \tag{2.4}$$

Equation (2.2) is valid for all complex values of the parameter a. Equation (2.3) is obtained by taking the limit of (2.2) as N goes to infinity. This is subject to the condition that $|a| < 1$. Equation (2.4) is related to equation (2.3) by differentiation. The exercises that follow explore some properties of sequences in more depth.

EXERCISE 2.1.1. **Shifting and Reversing Sequences**

Discrete-time signals are often expressed in terms of time-shifted and time-reversed combinations of other signals.

(a) The signal, $aa[n-N]$, is said to be *time shifted* by N where N is an integer. This means that the signal, $aa[n]$, is translated by N samples to the right if $N > 0$ and/or by $|N|$ samples to the left if $N < 0$. To illustrate this relatively simple concept, consider the following exercise. Begin by displaying the 5-point sequence, $aa[n]$, that may be found in the file `aa` using the `x view` function.

(i) Sketch $aa[n]$.

(ii) Sketch $aa[n-2]$.

(iii) Sketch $aa[n+4]$.

Now check your sketches by using the computer. Use the `x lshift` function to perform the time shifting. Note that specifying positive integers shifts to the right; specifying negative integers shifts to the left.

(b) The signal, $aa[-n]$, is said to be *time reversed*. This means that the signal is flipped about the point $n = 0$.

(i) Sketch $aa[-n]$.

(ii) If $v[n] = aa[n-4]$, sketch $v[-n]$.

(iii) If $r[n] = aa[n+2]$, sketch $r[-n]$.

Again check your sketches using the computer. Use the function `x reverse` to perform the time reversal.

EXERCISE 2.1.2. **Finite Length Sequences**

The sequence

$$h[n] = u[n] - u[n-10]$$

is a finite length sequence consisting of ones for n in the range $0 \leq n < 10$ and zeros everywhere else. It begins at $n = 0$ and ends at $n = 9$. Thus its length is 10 samples. On the other hand, the sequence

$$h[n] = a^n u[n]$$

is an example of an infinite length sequence, because it contains an infinite number of nonzero samples.

Several finite length sequences are listed below. Find the sequence lengths for each of these sequences and determine their starting and ending points. This exercise should be performed initially without the aid of the computer.

(a) $x[n] = u[n-2] - u[n-12]$

(b) $v[n] = u[n+16] - u[n-7]$

(c) $y[n] = x[n] \cdot v[n]$

(d) $r[n] = x[n] + v[n]$

(e) $s[n] = x[n+2] \cdot v[n-2]$

(f) $t[n] = y[n-1] + y[n+1]$

You can check your answers on the computer by using the `x siggen`, `x add`, `x subtract`, `x lshift`, and `x multiply` functions. For parts (a) and (b) the *block* option in `x siggen` can be used to approximate a step function.

Comment. It is important to note that a true step is infinite in duration, but a long block sequence can be used to approximate the step. A true step minus a shifted step results in a block or pulse. If steps are represented as finite length blocks, the difference between the block and shifted block produces erroneous sample values at the end of the sequence. To avoid this difficulty in this exercise, use block sequences of length 100 for the step functions. Before displaying your final result use `x truncate` with $L = 30$ to remove the erroneous samples at the end of the sequence.

EXERCISE 2.1.3. **Elementary Signals**

This problem explores a number of discrete-time signals that commonly arise in digital signal processing and provides a vehicle for becoming acquainted with the signal generator that will be used throughout this text. The signal generator can be executed by typing `x siggen`. It can generate a variety of different signals.

A small set of elementary signals is needed in this exercise. To begin, create the following signals of length 16 ($0 \le n \le 15$).

- $b[n]$, a 16-point block sequence with unit amplitude.
- $r[n]$, the first 16 points of the ramp function, defined as $nu[n]$.
- $t[n]$, 16 points of a periodic triangular wave with period 8, a maximum value of one, and starting point $n = 0$.
- $e[n]$, the first 16 points of the one-sided exponential, $(5/6)^n u[n]$.

Display and sketch them using `x view`. Signals are often formulated or expressed in terms of elementary signals. Using the elementary signals just created, sketch the following new signals:

(a) $v[n] = r[n-6]\, u[n]$;

(b) $a[n] = \begin{cases} b[n] - b[n-6], & n < 10 \\ 0, & \text{otherwise;} \end{cases}$

(c) $y[n] = e[n+10]\, b[n]$;

(d) $z[n] = t[n]\, (u[n] - u[n-10])$;

(e) $e_e[n] = ev\{e[n]\}\, (u[n+5] - u[n-5])$;

(f) $e_o[n] = od\{e[n]\}\, (u[n+5] - u[n-5])$.

This should be done initially without the aid of the computer, but you may use the functions `x lshift`, `x subtract`, `x truncate`, `x reverse`, and `x view` to check your results.

– *EXERCISE 2.1.4.* **Time Axis Alteration**

The time axis of a signal, $x[n]$, can be altered to produce a new signal, $y[n]$. The method of alteration can be as simple as a time shift, or it can be a more complicated operation. A general time axis alteration can be expressed as

$$y[n] = x[f[n]]$$

where $f[n]$ is the time axis alteration function. The value of $f[n]$ must be an integer because the time index for digital sequences is only defined for integer values. As a simple example of time axis alteration, consider the problem of determining

$$y[n] = x[n-6]$$

when

$$x[n] = u[n] - u[n-5].$$

The sequence $x[n]$ is simply shifted to the right by six samples to produce the sequence $y[n]$. While this example poses no difficulty, determining the result of other alterations may not be so easy. For example, consider sketching

$$y[n] = x[3-n]$$

where $f[n] = 3-n$ and $x[n]$ is an arbitrary signal. A simple and systematic procedure for determining $y[n] = x[f[n]]$ can be summarized as follows:

1. Sketch $x[n]$.
2. Sketch another time axis underneath it for $y[n]$.
3. Write the expression for $f[n]$.
4. To find $y[0]$, evaluate $f[0]$ (which will always be an integer). Next evaluate $x[f[0]]$ and enter this value for $y[0]$ on the time axis for $y[n]$ at the point $n = 0$.
5. To find $y[1]$, evaluate $f[1]$ and $x[f[1]]$. Set $y[1] = x[f[1]]$ and enter it on the $y[n]$ time axis at the point $n = 1$.
6. Continue this procedure until all values, $-\infty < n < \infty$, of $y[n]$ have been found.

To test your understanding, consider the sequence $x[n]$ where

$$x[n] = \left(\tfrac{3}{4}\right)^n (u[n] - u[n-5]).$$

(a) Without the aid of the computer, sketch $y[n]$ for each of the cases given below using the procedure just described.

(i) $y[n] = x[-n]$.

(ii) $y[n] = x[-7-n]$.

(iii) $y[n] = x[-n+15]$.

(b) Develop a rule based on using the `x lshift` and `x reverse` operations that can be used to obtain $y[n]$ for each of the cases in part (a). Use `x siggen` to generate the damped exponential sequence segment, $x[n]$. Write a macro (as described in Section 1.3) using the `x reverse` and `x lshift` functions that will produce $y[n]$ for each of the cases in part (a). Indicate the specific shift parameter values that must be used in order for your macro to work properly. Check to verify that the answers obtained manually agree with those produced by your macro.

(c) Manually sketch the sequence

$$y[n] = x[2n].$$

Check your answer by using the function `x dnsample`. By selecting $M = 2$ in this function, you will be able to generate $y[n]$.

(d) Sketch the sequence

$$y[n] = x[5 - 2n].$$

EXERCISE 2.1.5. **Working with Summations**

Several useful closed-form expressions for summations exist, three of which are given in equations (2.2–2.4). Summations, like integrals, can be changed to alternate, but equivalent, forms through change-of-variables operations. This exercise should be performed analytically (without the aid of the computer).

(a) Consider the expressions shown below:

$$a_1 = \sum_{n=0}^{\infty} \left(\tfrac{1}{2}\right)^n$$

$$a_2 = \sum_{n=10}^{\infty} \left(\tfrac{1}{2}\right)^{n-10}$$

$$a_3 = 2^5 \sum_{n=5}^{\infty} \left(\tfrac{1}{2}\right)^n.$$

(i) Determine the numerical values of a_1, a_2, and a_3.

(ii) Show by using a change of variables in the summation index or by another approach that these summation expressions are related.

(b) Now consider these finite sums:

$$b_1 = \sum_{n=0}^{10} \left(\tfrac{1}{2}\right)^n$$

$$b_2 = \alpha_1 \sum_{n=-10}^{0} 2^n$$

$$b_3 = \alpha_2 \sum_{n=6}^{16} \left(\tfrac{1}{2}\right)^n$$

$$b_4 = \alpha_3 \sum_{n=0}^{10} \left(\tfrac{1}{2}\right)^{n+2}.$$

Determine the values for α_1, α_2, and α_3 so that

$$b_1 = b_2 = b_3 = b_4.$$

— ***EXERCISE 2.1.6.*** **The Geometric Series**

Many problems become straightforward if the mathematical equations used in the problem descriptions are simplified. The signals shown below are expressed in terms of summations. Determine a simplified expression for each sequence so that it can be computed easily. Display and sketch each of these signals for the range $0 \leq n \leq 20$. Give the numerical values for the last three signal values (i.e., for $n = 18,\ 19,\ 20$) in each case. (*Hint:* Use the geometric series expressions to transform the signals into a convenient form.) Then use the computer to generate the final sequences using **x siggen, x add, x gain,** and **x multiply**. Examine the requested numerical values by either using a text editor or by typing the signal files to the screen.

(a)

$$x[n] = \left[\sum_{\ell=0}^{n} \left(\tfrac{7}{8}\right)^{\ell} - \sum_{\ell=0}^{n} \left(\tfrac{3}{4}\right)^{\ell}\right] u[n].$$

(b)

$$x[n] = \left[\sum_{\ell=0}^{\infty} (\ell+1)\left(\tfrac{1}{2}\right)^{\ell} + \sum_{\ell=0}^{n} \left(\tfrac{8}{9}\right)^{\ell}\right] u[n].$$

(c)

$$x[n] = \left[\sum_{\ell=-10}^{n-10} \left(\tfrac{7}{8}\right)^{\ell}\right] u[n].$$

(d)

$$x[n] = \left[\sum_{\ell=0}^{n} \ell\,(0.9)^{\ell}\right] u[n].$$

Part (d) may require some careful thought.

$$\sum_{k=0}^{n} k \cdot \alpha^k = \frac{\alpha\left(1-\alpha^n\right)}{(1-\alpha)^2} - \frac{n \cdot \alpha^{n+1}}{1-\alpha} = \frac{n \cdot \alpha^{n+2} - (n+1) \cdot \alpha^{n+1} + \alpha}{(1-\alpha)^2}$$

EXERCISE 2.1.7. Characteristics of Signals

In the introductory discussion, several common signal characteristics were defined. These included periodicity, signal duration, and the disposition of the sequence values with respect to being purely real, purely imaginary, or complex.

(a) Consider the signal

$$x[n] = \cos(\omega_0 n).$$

(i) Assume $\omega_0 = \pi/8$. Is this signal periodic? If so, determine its period and sketch one period of $x[n]$.

(ii) Now let $\omega_0 = 1$. Is the discrete-time signal, $x[n]$, periodic? Now consider the continuous-time signal,

$$x(t) = \cos t.$$

What is the period of this signal? Explain why the discrete-time and continuous-time cosine signals behave differently.

(b) The signal

$$x[n] = e^{j\omega_0 n}u[n]$$

is infinite in duration. It is also complex valued. Generate a 40-point segment of the sequence, $x[n]$, where $\omega_0 = \pi/10$. This can be done by using the *exponential* $Ke^{\alpha n}$ option in `x siggen` with $K = 1$, *alpha* $= 0 \quad \pi/10$, and starting point at zero. Display and sketch the real and imaginary parts and the magnitude and phase of $x[n]$ using `x view`. Note that the real and imaginary parts are cosine and sine functions. This illustrates the relationship which is attributed to Euler:

$$e^{j\omega_0 n} = \cos\omega_0 n \; + \; j \sin\omega_0 n.$$

(c) Using the 40-point signal $x[n]$ defined in part (b), manually sketch each of the signals shown below:

$$y_0[n] = |10\, x[n]|$$
$$y_1[n] = \angle x^*[n]$$
$$y_2[n] = \Re e\{jx[n]\}$$
$$y_3[n] = \Im m\{(jx[n])^*\}.$$

Now verify your results using the computer. The `x view` function will allow you to display the real and imaginary parts as well as the magnitude and phase. The `x gain` function (with *gain* $= 0 \quad 1$) can be used to generate $jx[n]$; the `x conjugate` function will generate $x^*[n]$.

2.2 DISCRETE-TIME SYSTEMS

A discrete-time system can be viewed as a set of manipulations performed on one or more sequences. This text will primarily consider single input single output systems, since these represent the most common systems in traditional digital signal processing applications. Figure 2.3 depicts a typical system with input $x[n]$ and output $y[n]$. The input/output relationship can be written as $y[n] = T\{x[n]\}$ where $T\{\cdot\}$ denotes the system operation. Systems are commonly discussed and classified in terms of specific properties, some of which are now defined:

- *Additivity.* A system is *additive* if

$$T\{x_1[n] + x_2[n]\} = T\{x_1[n]\} + T\{x_2[n]\}$$

  for all real and complex inputs $x_1[n]$ and $x_2[n]$.
- *Homogeneity.* A system is *homogeneous* if

$$T\{\alpha x[n]\} = \alpha T\{x[n]\}$$

  for all complex values of α, and for all $x[n]$.
- *Linearity.* A system is *linear* if it is both additive and homogeneous.
- *Time Invariance or Shift Invariance.* A system is *time invariant* if a time shift in the input results in time shifting the output by the same amount. In other words, given that

$$y[n] = T\{x[n]\}$$

  a time-invariant system must satisfy the condition

$$y[n - n_0] = T\{x[n - n_0]\}$$

  for all integers, n_0, and all inputs, $x[n]$.
- *Stability.* A system is *unstable* if there exists a bounded input ($|x[n]| < \infty$ for all n) that causes the output signal $y[n]$ to be unbounded, i.e., $|y[n]| = \infty$. Otherwise the system is stable.[1]
- *Causality.* A system is *causal* if the output at time n is not dependent on future input samples. In other words, in a causal system output samples are only obtained from past or present samples of the input. Future input samples cannot be required to compute the output. For example, the system represented by the difference equation

$$y[n] = x[n] + x[n + 1]$$

  is not causal since $x[n + 1]$ is a future sample of the input. To see this clearly, assume that the present time is $n = 1$. Then $y[1] = x[1] + x[2]$. We see that $y[1]$

[1] This is called bounded input bounded output (BIBO) stability. There are other definitions of stability, but they will not be treated in this text.

requires that the present value $x[1]$ and the future value $x[2]$ be known. The system

$$y[n] = x[n] + x[n-1],$$

on the other hand, is causal because $x[n]$ and $x[n-1]$ represent present and past input samples, respectively.

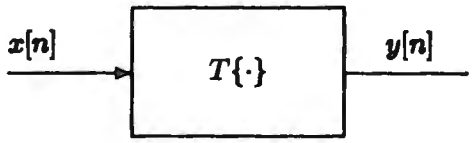


Figure 2.3. Block diagram representation of a system with input $x[n]$ and output $y[n]$.

Systems that are both linear and time invariant (LTI) are of special significance because they are well understood and can be conveniently analyzed in both the time domain and the frequency domain. The following exercises are designed to help you classify several unknown systems in terms of their properties. These systems, **system1**, **system2**, and **system3**, are provided as part of the software. Execution of these systems is best illustrated by an example. To execute **system1**, for example, you should simply type

system1 inputfile outputfile

EXERCISE 2.2.1. **Testing Linearity**

Here you will work with an unknown system $T\{\cdot\}$. Use **x siggen** to create a ramp sequence $x_1[n]$ with a duration of five points in the range $0 \leq n < 5$. Then create a 5-point sequence of ones, $x_2[n]$, also in the range $0 \leq n < 5$ using the *block* option in **x siggen**. For the first part of this exercise, use **system1** as the unknown system.

- Generate $y_1[n] = T\{x_1[n]\}$.
- Generate $y_2[n] = T\{x_2[n]\}$.
- Generate $y_3[n] = T\{ax_1[n] + bx_2[n]\}$ by using the **x gain** and **x add** functions. Select arbitrary but convenient values for a and b.

(a) Examine the sequences $ay_1[n]$ and $by_2[n]$ for the same values of a and b. Does **system1** appear to be

(i) homogeneous?

(ii) additive?

(iii) linear?

Explain your reasoning.

(b) Now repeat this exercise using **system2** as the system $T\{\cdot\}$.

Note that if a counterexample is found that violates the linearity tests, one can be certain that the system is nonlinear. Proving that a system is linear based on this kind of probing is not practical since the definition requires that the linearity conditions be satisfied for all possible inputs.

EXERCISE 2.2.2. **Testing Time Invariance**

This exercise attempts to determine if a system $T\{\cdot\}$ is time invariant by observing the system's response to shifted inputs. Use a 5-point ramp sequence, $x[n]$, as the input and the unknown system, **system3**, as the system to be tested. It can be implemented by typing

system3 inputfile outputfile.

Compute $y_1[n] = T\{x[n]\}$. Then use the **x lshift** function to generate the sequence $x[n-1]$. Finally, compute $y_2[n] = T\{x[n-1]\}$, using **system3**.

(a) What is your conclusion about the time invariance of this system?

(b) Repeat this problem using **system1** as the unknown system. Can you prove that **system1** is either time invariant or time varying based on your experimentation?

EXERCISE 2.2.3. **Linear Time-Invariant (LTI) Systems**

The trial-and-error method that was used in the previous exercises can be effective in determining if a system is nonlinear or time varying if a counterexample can be identified. However, it is very difficult to prove that an unknown system is linear or time invariant following this approach. If the equation that describes the system is known, we can determine with certainty whether a system is LTI. Based on the definitions, determine if the following systems with real inputs $x[n]$ and outputs $y[n]$ are: (1) linear and (2) time invariant.

(a) $y[n] = \sqrt{x[n] \cdot x[n]}$.

(b) $y[n] = x[\sqrt{n^2}]$.

(c) $y[n] = \log_2 10^{x[n]}$.

(d) $y[n] = \cos(x[n]) + \cos n$.

(e) $y[n] = x[n] + x[n+1] + x[n-2]$.

EXERCISE 2.2.4. **Stability**

Unstable systems are characterized by the ability to produce output sample values of infinite amplitude for bounded inputs. In practice, values of infinity cannot be represented, but an unstable system will often produce an output that saturates. This, in turn, leads to erroneous outputs if processing is allowed to continue. The system

$$y[n] = \frac{1}{x[n]}$$

is reasonably well behaved except when the numerical value of $x[n]$ is close to zero. When $x[n] = 0$, the output is clearly infinite and therefore the system is unstable.

Consider the system described by the relationship

$$y[n] = e^{x[n]}.$$

(a) Is this system stable?

(b) What numerical difficulties might be encountered in using this system?

(c) What is the approximate limit on the amplitude of the input for this system to operate properly on your machine? Check this experimentally using the function **x nlinear** to generate the signal $e^{x[n]}$. One way to do this is to generate a short *block* sequence ($x[n] = 1$) using **x siggen**. Use **x gain** to increase the amplitude of $x[n]$ until $e^{x[n]}$ saturates, i.e., results in an overflow error. You may wish to begin by considering the range $600 < x[n] < 800$.

EXERCISE 2.2.5. **Stability of LTI Systems**

For an LTI system with the impulse response, $h[n]$, a simple stability criterion exists. If

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

then the LTI system is stable. The impulse responses for several LTI systems are listed below. Determine if they are stable.

(a) $h[n] = 2^n u[n]$.

(b) $h[n] = 2^n u[-n]$.

(c) $h[n] = (10n)^{-1} u[n-1]$.

(d) $h[n] = u[n] - u[-n]$.

(e) $h[n] = 6^{-|n|}$.

EXERCISE 2.2.6. **More on Stability**

In this exercise, you will determine if a system is stable by examining the equation that defines the system. The system input and output are $x[n]$ and $y[n]$, respectively. For each of the systems shown below, analytically determine if the system is stable or unstable. Note that some of these systems are not LTI. This exercise does not require the use of the computer.

(a) $y[n] = x[n-1] \cdot x[n]$. $\lim_{n \to +\infty} |x[n]| \leq 1$

(b) $y[n] = x[n-1]/x[n]$.

(c) $y[n] = \sum_{k=0}^{\infty} x[n-k]$.

(d) $y[n] = \sum_{k=1}^{\infty} x[n-k]/k$.

(e) $y[n] = |x[n]|^2$.

(f) $y[n] = \sin(x[n])$.

EXERCISE 2.2.7. **Causality**

Causality is important in many practical systems, but it is critical in *real-time* systems. These are systems in which the input samples arrive sequentially and for which one output sample must be produced immediately after each input sample is received. These systems must be causal because only present or past input samples are available to be used to obtain the current output sample. This exercise does not require the use of the computer.

(a) To appreciate the importance of causality, consider the LTI system

$$y[n] = x[n] + 2x[n-1] + 3x[n-2].$$

Assume that the input signal is zero for $n < 0$ and calculate the system output step by step by using the following steps:

(i) The first input sample (at time $n = 0$) is 1. What is the output $y[0]$?

(ii) The second input sample ($n = 1$) is 2. What is the output $y[1]$?

(iii) The next input sample ($n = 2$) is 3. What is the output $y[2]$?

Clearly, there is no problem in determining the value of each output sample as each input sample is received. This is because this system is causal. By contrast, consider repeating this exercise for the system

$$y[n] = x[n] + 2x[n+1] + 3x[n+2] + 4x[n+3].$$

It should be clear that it is not possible to determine $y[0]$ at time $n = 0$, $y[1]$ at time $n = 1$, etc., because all of the quantities on the right side of this equation are not available when they are needed.

(b) A way to circumvent the causality problem is to introduce a system delay. In particular, consider the same noncausal system of part (a) with a three sample system delay, i.e.,

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

This is identical to the earlier system except that the output appears three samples delayed in time. Moreover, note that the new system is causal. If the input to the

system is $x[n] = u[n] - u[n-4]$, determine the first four samples of the output analytically.

(c) The idea of using a system delay to make a noncausal system causal as discussed in part (b) has limitations. For example, if

$$y[n] = x[n]x[-n]$$

it is not possible to find a finite length delay to implement this system as a causal one. Find another noncausal system for which a causal implementation cannot be found by using a system delay.

2.3 CONVOLUTION

The *impulse response* of a system is the response of that system to a discrete-time impulse or unit sample. The impulse response has particular significance when a system is LTI because it completely characterizes the system. The impulse response provides all of the information that is necessary to determine the output of an LTI system in every situation. The output, $y[n]$, of an LTI system is given by the convolution of the input, $x[n]$, with the system impulse response, $h[n]$. This operation can be written as the *convolution sum*,

$$y[n] = \sum_{m=-\infty}^{\infty} x[m]\, h[n-m]. \tag{2.5}$$

It is common to use the shorthand notation

$$y[n] = x[n] * h[n] \tag{2.6}$$

where the asterisk "$*$" denotes (linear) convolution.

Convolution appears extensively in signal processing. While it may be viewed as a description of the input-output relationship for an LTI system, it can also be viewed as an operation performed between two arbitrary signals. Using this viewpoint, convolution can be used as a tool for analyzing the properties of signals or the processes that produced those signals.

Although it is not difficult to write a program to evaluate the convolution sum explicitly, an alternate method called *graphical convolution* can provide valuable insight into the convolution process. Using this approach, it is often possible to visualize the result of a convolution by inspecting the input sequences. The method involves properly aligning the two sequences on a common time axis, multiplying them, and then adding sequence values. The following exercises introduce convolution.

EXERCISE 2.3.1. The Convolution Sum

The definition of convolution is given in equation (2.5).

(a) Write a program, in any language that is supported in your computing environment, that will read in two sequences, convolve them, and write out the result.

Use the signal format described in Section 1.2 in your program so that it will be compatible with the other functions that are provided. Try to make the program as short and efficient as possible. Only a few lines of code should be needed for the program.

(b) The function x **convolve**, which is provided, also performs linear convolution. Generate two ramp functions using x **siggen**, one of length 15 and the other of length 10. Convolve these two sequences and provide a sketch of the two inputs and the output. Now use your program from part (a) to perform the same convolution and verify that your program works correctly.

– *EXERCISE 2.3.2.* **An Interpretation of Convolution**

This exercise presents another interpretation of the convolution operation. Carefully examine the convolution sum and observe that

$$\delta[n] * x[n] = x[n],$$

$$\delta[n-1] * x[n] = x[n-1],$$

and

$$\delta[n-2] * x[n] = x[n-2].$$

Generalizing this observation, it follows that

$$\delta[n-n_0] * x[n] = x[n-n_0]$$

where n_0 is an arbitrary integer. Now consider the simple convolution $y[n] = aa[n] * x[n]$ where

$$x[n] = \delta[n] + \delta[n-1] + \delta[n-2] + \delta[n-3].$$

(a) The sequence $aa[n]$ is provided for you in the file **aa**. Use the *block* option of the function x **siggen** to generate $x[n]$. Sketch and display both of these sequences.

(b) Now create the four individual signals, $\delta[n]$, $\delta[n-1]$, $\delta[n-2]$, and $\delta[n-3]$ using x **siggen**. This can be done using the *block* option with different starting locations for the length 1 sequences. Sketch these sequences.

(c) Using the x **convolve** function create the sequence $y[n] = aa[n] * x[n]$ and sketch the result. $y[n] = \alpha\alpha[n] + \alpha\alpha[n-1] + \alpha\alpha[n-2] + \alpha\alpha[n-3]$

(d) Again using the x **convolve** function, convolve $aa[n]$ with each of the four sequences that you generated in part (b). Sketch these individual sequences.

(e) Add the resulting sequences from part (d) together using the x **add** function and sketch the result. Use x **view2** to compare the result with the result of part (b). Observe that the sum is identical to the sequence obtained by convolution.

The important point here is that any input sequence can be represented as a superposition of delayed unit impulses. The output is then equal to the same superposition of delayed unit impulse responses.

—*EXERCISE 2.3.3.* Introduction to Graphical Convolution

Graphical convolution is a straightforward and intuitive procedure to evaluate the convolution sum without the aid of a computer. The method is motivated by recognizing that the term $h[n-m]$ in the convolution summation (2.5) can be interpreted as a signal in the variable m with n interpreted as a shift parameter. Since $h[n-m] = h[-(m-n)]$, it is a time-reversed copy of $h[m]$ that is then shifted by n samples on the m-axis. Thus, given $h[n]$, $h[n-m]$ can be obtained by first reflecting the signal on the m-axis about the point $m = 0$ to form $h[-m]$ and shifting it $|n|$ places to the left if n is negative or n places to the right if n is positive. This is illustrated in Fig. 2.4. The convolution at sample n can now be found by displaying $x[m]$ and $h[n-m]$ on a common m-axis, multiplying the two aligned sequences on a point-for-point basis, and summing the products. This exercise will walk through this procedure one step at a time.

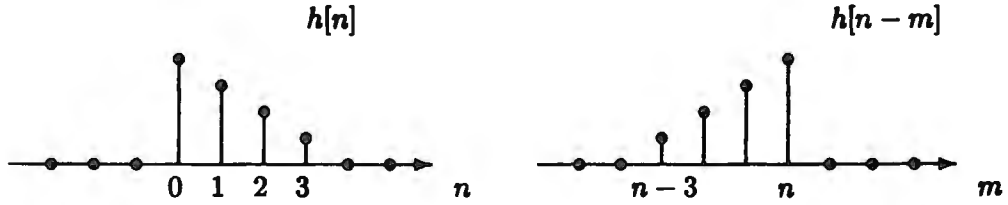


Figure 2.4. The reflected impulse response of the convolution sum expressed as a sequence.

(a) First examine the sequences $x[n]$ and $h[n]$ that are provided in the files **bb** and **cc**, respectively, using **x view**. As you go through the procedure for the first time, sketch each of the shifted signals on graphs that are aligned vertically.

- **(i)** Reflect the sequence $h[n]$ about the origin using the **x reverse** function. Display and sketch your results. This is the signal $h[-m]$.
- **(ii)** To find $h[n_0 - m]$ at an arbitrary value $n = n_0$, shift the reflected sequence n_0 places to the right for $n_0 > 0$ or $|n_0|$ places to the left for $n_0 < 0$. For this example, begin by finding the value of $y[2]$, i.e., $n_0 = 2$. Use the **x lshift** function to slide the reflected signal two places to the right. Display and sketch your result.
- **(iii)** Now that the two sequences $h[m]$ and $x[2-m]$ are aligned, multiply them pointwise using the **x multiply** function. Display and sketch your result.
- **(iv)** Add all the sample values in the product sequence together and assign that value to $y[n_0]$ (in this case $y[2]$). Do this step manually by printing the signal file to the screen. For this example, the correct value is -11. Therefore on a separate sheet of paper, record $y[2]$ graphically with a height of -11.

(v) Repeat this procedure for all values of n, $-\infty < n < \infty$. In practice it is obviously not necessary to do this an infinite number of times when finite duration sequences are being convolved. Observe that shifting the reflected sequence too far to the left or right results in nonoverlapping sequences, in which case the result of the multiplication is an all-zero sequence. It is typical to start with the shift corresponding to the leftmost nonzero value of the output and increment n until the convolution is complete.

(b) Given this step-by-step illustration of the procedure, determine and sketch the complete sequence $y[n]$. Observe that $y[n]$ is zero outside the range $0 \leq n \leq 7$. Moreover, observe that the total length of the output sequence is one less than the sum of the lengths of the two sequences that are convolved. Use the function `x convolve` to check your answer.

(c) Determine and sketch $y[n]$ using graphical convolution by reflecting $x[n]$ instead of $h[n]$. Verify that your result is the same.

After this procedure becomes familiar, you should be able to reflect and align the sequences mentally to obtain a quick estimate of output values for simple convolutions.

— ***EXERCISE 2.3.4.*** **Convolving Sequences**

In this exercise, discrete-time convolution is examined by example. Generate the following sequences:

$$x_1[n] = u[n] - u[n-4], \quad n \quad 0,1,2,3$$
$$x_2[n] = u[n] - u[n-6], \quad n = 0,1,\ldots,5$$
$$x_3[n] = \left(\tfrac{7}{8}\right)^n (u[n] - u[n-5]), \quad n = 0,1,\ldots,4$$
$$x_4[n] = \sin\left(\tfrac{\pi n}{12} + \tfrac{\pi}{4}\right) (u[n] - u[n-6]). \quad n \quad ,1 \quad \ldots 5.$$

Notice that $x_1[n]$ and $x_2[n]$ are ~~5~~ 4-point and ~~7~~ 5-point blocks, respectively, and that they can be created using `x siggen`. The signals $x_3[n]$ and $x_4[n]$ can be created using options 5 and 3, respectively. Test your intuition by performing the following convolutions in your head using the graphical convolution method.

(a) $y_1[n] = x_1[n] * x_2[n]$;

(b) $y_2[n] = x_1[n] * x_3[n-3]$;

(c) $y_3[n] = x_1[n] * x_4[n]$;

(d) $y_4[n] = x_3[n] * x_4[n]$.

In each case provide a rough and quick sketch. How long are the resulting sequences? Then check the accuracy of your result by performing the convolution operations using `x convolve` and `x lshift`.

— ***EXERCISE 2.3.5.*** **More Convolution**

Consider the sequences shown below:

$$v_1[n] = \sum_{m=0}^{2} \delta[n-6m]$$

$$v_2[n] = \sum_{k=1}^{3} \sum_{m=0}^{2} \delta[n-6m-k]$$

$$v_3[n] = \left(\tfrac{2}{3}\right)^n (u[n] - u[n-5]).$$

(a) Use `x siggen` to create these sequences. This can be done easily by using options 8, 1, and 5. Alternatively, in the case of $v_3[n]$ you may wish to convolve a block signal with an impulse train. Display and sketch each result.

(b) Using graphical convolution, sketch the following sequences as accurately as you can.

(i) $y_0[n] = v_1[n] * v_3[n]$;
(ii) $y_1[n] = v_1[n] * v_1[n] * v_1[n]$;
(iii) $y_2[n] = v_1[n] * v_2[n]$;
(iv) $y_3[n] = v_3[n] * v_2[n]$;
(v) $y_4[n] = v_3[n] * v_3[n-1]$.

(*Note:* For this case your sketch will probably be rather rough. However, you should be able to capture the general shape of the output.)

Check your results using the computer.

— ***EXERCISE 2.3.6.*** **Beginning and End Points**

The convolution of two finite length sequences always results in a sequence of finite length. Graphical convolution can be used to determine when the output sequence begins and ends.

This exercise will use the following sequences:

(i) $x[n] = u[n] - u[n-50]$;
(ii) $v[n] = u[n-31] - u[n-42]$;
(iii) $w[n] = u[n+51] - u[n+17]$.

(a) Determine the first and last nonzero samples for each of the following:

(i) $y_1[n] = x[n] * v[n]$;
(ii) $y_2[n] = v[n] * w[n]$;
(iii) $y_3[n] = x[n] * w[n]$.

Use the computer to check your results. Derive a rule for determining the initial and final samples for the convolution of two finite length sequences.

(b) Determine the initial and final samples for the following sequences without using the computer:

(i) $y_4[n] = x[n] * z[n]$; n = -140.820 : 261, l h(y4) -141
(ii) $y_5[n] = v[n] * z[n]$; n = -140.789 : 253, y5) = 141.043
(iii) $y_6[n] = z[n] * w[n]$. n = -140.871 : 194, length(y6) = 141.066

[n]: n=-140820: 12 where $z[n] = u[n + 140820] - u[n - 213]$. Note that the length of $z[n]$ exceeds the signal length limit of the software. Hence you cannot use the computer to z) -141.033 get the correct answer.

EXERCISE 2.3.7. **Convolution of Infinite Length Sequences**

Consider the sequences

$$x_1[n] = \left(\tfrac{1}{2}\right)^n u[n]$$
$$x_2[n] = 2^n u[-n]$$
$$x_3[n] = u[n] - u[n - 10].$$

Using graphical convolution, sketch the general shape of the following infinite length signals.

(a) $y_a[n] = x_1[n] * x_1[n]$.

(b) $y_b[n] = x_2[n] * x_2[n]$.

(c) $y_c[n] = x_2[n] * x_3[n]$.

(d) $y_d[n] = x_1[n] * x_2[n]$.

You can obtain an approximate check of your answer using **x siggen** to generate finite length approximations (of 20 samples or so) to these signals and then convolving them together using **x convolve**. To generate $x_2[n]$ first create 20 samples of $(1/2)^n u[n]$ using **x siggen** and then use **x reverse**.

EXERCISE 2.3.8. **Using Graphical Convolution**

In this exercise, you are to obtain the value of the convolution of two sequences at one specific sample location. Consider the sequences

(i) $x[n] = u[n] - u[n - 47]$;
(ii) $v[n] = u[n + 28] - u[n - 10]$;
(iii) $w[n] = \left(\tfrac{3}{4}\right)^n u[n]$;
(iv) $r[n] = \left(\tfrac{3}{4}\right)^{|n|}$;
(v) $z[n] = (n + 12)\,(u[n + 12] - u[n - 30])$.

Using the graphical convolution procedure and without the aid of the computer, determine the values as indicated below.

(a) If $y_a[n] = x[n] * v[n]$, find $y_a[10]$ and $y_a[43]$.

(b) If $y_b[n] = x[n] * w[n]$, find $y_b[40]$ and $y_b[60]$.

(c) If $y_c[n] = v[n] * r[n]$, find $y_c[-29]$ and $y_c[-2]$.

(d) If $y_d[n] = w[n] * z[n]$, find $y_d[12]$.

EXERCISE 2.3.9. **Algebraic Properties of Convolution**

Several properties of convolution that are often exploited in discrete-time signal processing are examined here. These properties will be examined by using the sequences $x[n]$ and $h[n]$, where

$$x[n] = n\,(u[n] - u[n-32])$$
$$h[n] = 2n\,(u[n] - u[n-26]).$$

(a) Show analytically that $y[n] = x[n] * h[n] = h[n] * x[n]$ by changing the variables in the summation in equation (2.5). Then verify that $x[n] * h[n] = h[n] * x[n]$ by generating these sequences with **x siggen** and using the **x convolve** function. The convolution operation is said to be *commutative*. Sketch $y[n]$.

(b) Now split $x[n]$ into two sequences of the same length (32 points), called $x_1[n]$ and $x_2[n]$, such that $x[n] = x_1[n] + x_2[n]$. Provide a sketch of these two sequences. Note that there are many choices for $x_1[n]$ and $x_2[n]$ that satisfy this condition. A simple way to do this is to use **x siggen** to design two 32-point ramp functions (perhaps with different slopes) that add to $x[n]$. Compute and sketch $y_1[n] = x_1[n] * h[n]$, $y_2[n] = x_2[n] * h[n]$, and $y[n] = y_1[n] + y_2[n]$. Observe that $y[n]$ is exactly the same as $x[n] * h[n]$ computed in part (a).

(c) Repeat the exercise performed in part (b), but with the condition that $x_1[n]$ and $x_2[n]$ are of different lengths. One way to do this is to let $x_1[n]$ be a truncated version of $x[n]$ and $x_2[n]$ be $x[n] - x_1[n]$. Is the equal length condition a necessary requirement? Convolution is said to be *distributive* with respect to addition.

EXERCISE 2.3.10. **System Impulse Response**

Knowledge of the impulse response of a system is important in digital signal processing. Here you will find the impulse response of the system $T\{\cdot\}$ that is implemented in **system1**.

(a) Find and sketch the impulse response, $h[n]$, of the system $T\{\cdot\}$. Use the *block* option of **x siggen** to create the impulse.

(b) Compute and sketch $y_1[n] = T\{x_1[n]\}$ where $x_1[n] = 2\delta[n-3]$. Use the statement

system1 inputfile outputfile

to implement **system1** and **x siggen** to generate $x_1[n]$. Compute and sketch $y_2[n] = x_1[n] * h[n]$ using **x convolve**.

(c) Do the sequences $y_1[n]$ and $y_2[n]$ suggest that $T\{\cdot\}$ is LTI? Explain.

2.4 DIFFERENCE EQUATIONS

There are many types of difference equations. The most common type in digital signal processing is the linear constant coefficient difference equation or LCCDE. These can be used to represent certain LTI systems. Assume that a system with input, $x[n]$, and output, $y[n]$, is defined by an LCCDE of the form

$$\begin{aligned} y[n] + a_1 y[n-1] + a_2 y[n-2] + \cdots + a_N y[n-N] \\ = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M]. \end{aligned}$$

This equation can be rewritten as the recursion

$$\begin{aligned} y[n] = -a_1 y[n-1] \; - a_2 y[n-2] \; - \cdots - a_N y[n-N] \\ + b_0 x[n] \; + \; b_1 x[n-1] \; + \; \cdots \; + \; b_M x[n-M] \end{aligned}$$

where $a_1, a_2, \ldots, a_N$ and $b_0, b_1, \ldots, b_M$ are the constant coefficients. To specify the output uniquely we must also specify N initial sample values of $y[n]$. If the system is causal and its output samples are to be evaluated beginning at $n = 0$ (i.e., $y[n]$ is to be evaluated for $n = 0, 1, \ldots$), then the sample values

$$y[-1], y[-2], \ldots, y[-N]$$

must be prespecified. These are called the *initial conditions.* Once the initial conditions are given, the output samples can be computed sequentially. When these initial conditions are assumed to be zero, LCCDEs can define causal, linear, time invariant systems for inputs that are zero for $n < 0$. These difference equations are commonly used to implement discrete-time LTI systems. LCCDEs are discussed further in Chapter 5.

EXERCISE 2.4.1. **Evaluating a Difference Equation**

In this exercise you will evaluate the output of a causal LTI system specified by the simple difference equation

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

with zero initial conditions. Assume that the input is

$$x[n] = \delta[n] + \delta[n-1].$$

A convenient way to evaluate the output samples without a computer is to arrange the computation in a chart.

n	$x[n]$	$x[n-1]$	$y[n-1]$	$y[n]$
$n = -1$	$x[-1] = 0$	$x[-2] = 0$	$y[-2] = 0$	$y[-1] = 0$
$n = 0$	$x[0] = 1$	$x[-1] = 0$	$y[-1] = 0$	$y[0] = 1$
$n = 1$	$x[1] = 1$	$x[0] = 1$	$y[0] = 1$	$y[1] = 3.5$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Since this system is linear and causal, no nonzero output sample is produced until a nonzero input sample is received. The input in our example begins at $n = 0$. Therefore all samples in the output sequence will be zero until this time and the chart can begin at $n = 0$. On the line corresponding to $n = 0$, $x[0] = 1$ and $x[-1] = y[-1] = 0$. Substituting these values into the difference equation gives $y[0] = 1$. This procedure can be continued by going to the next line, entering the values for $y[0]$, $x[1]$, and $x[0]$, and then evaluating $y[1]$ using the difference equation and continuing similarly until all the desired samples of $y[n]$ have been computed.

(a) Following this procedure, fill in the lines for $n = 2, 3, 4$. The column in the chart labeled $y[n]$ is the system output.

(b) A computer is ideally suited to evaluate the output samples specified by a difference equation. Use `x siggen` to create the input, then use `x lccde` to compute the output. Record the first seven values of $y[n]$ by typing the output file to the screen.

EXERCISE 2.4.2. **Convolution Versus LCCDEs**

LCCDEs with zero initial conditions are often used to implement LTI systems. Consider the causal LCCDE

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

with input $x[n]$ and output $y[n]$.

(a) Determine the impulse response, $h[n]$, of this system by evaluating the difference equation using an impulse as the input and zero for the initial condition at $n = -1$. The function `x siggen` can be used to create the impulse and the function `x lccde` will implement the difference equation. The impulse response has the form

$$h[n] = \alpha^n u[n].$$

What is the value of α? Note that we can implement this system even though its impulse response is infinite in duration.

(b) In part (a) you found the impulse response and observed that it was infinitely long. If it can be assumed that all values of $h[n]$ less than 0.001 are negligible and can be ignored, what is the effective length of $h[n]$? You may find it more convenient to read the sequence amplitude values by typing the signal file to the screen or by using a text editor rather than by using `x view`. Generate this finite length version of $h[n]$ and the sequence

$$x[n] = u[n] - u[n-6]$$

using `x siggen`. Compute and sketch $x[n] * h[n]$.

(c) Now implement the convolution in part (b) explicitly using a difference equation. Use the x lccde function with zero initial conditions. Compare your results with the sequence obtained by convolution. By using the function x subtract display and sketch the difference between the two results. How many multiplies and adds are required on average to obtain each output sample using both of these methods?

EXERCISE 2.4.3. **The First Backward Difference**

The discrete-time first backward difference operation is similar to taking the derivative of a continuous-time signal in certain respects. It takes the difference between the current sample of $x[n]$ and the previous one, i.e.,

$$y[n] = x[n] - x[n-1]. \tag{2.7}$$

It corresponds to an LTI system with the impulse response

$$h[n] = \delta[n] - \delta[n-1]. \tag{2.8}$$

To explore this operation, this exercise will consider the effects of the first backward difference operation on several sequences. To begin use the *create file* option in x siggen to create a file containing $h[n]$.

(a) Sketch the continuous-time function,

$$\mathrm{tri}_a(t) = t\,u(t) - 2(t-20)\,u(t-20) + (t-40)\,u(t-40).$$

Note that this is a continuous-time triangular function beginning at $t = 0$ and ending at $t = 40$, with a peak value at $t = 20$. In addition, sketch the continuous-time square pulse,

$$\mathrm{sq}_a(t) = u(t) - u(t-20).$$

Now compute (by inspection) and sketch the first derivative of each of these continuous-time waveforms.

(b) Generate the 41-point triangular pulse

$$\mathrm{tri}[n] = n\,u[n] - 2(n-20)\,u[n-20] + (n-40)\,u[n-40].$$

This can be done easily using the *triangular wave* option in x siggen by specifying the period to be 40 and the number of periods to be one. In addition, generate and sketch the square pulse

$$\mathrm{sq}[n] = u[n] - u[n-20].$$

Use the *square wave* option in x siggen with pulse length 20, period 40, and number of periods 1. Now compute and sketch the first backward difference

of each of these signals using either **x lccde** to implement the system using equation (2.7) or **x convolve**. Note the similarity to the continuous-time case. The function **x view2** may be used to display the signal and its first backward difference one above the other. In many cases the first backward difference is used as an approximation to a derivative operator.

EXERCISE 2.4.4. **The Running Sum**

The discrete-time running sum operation is in many ways analogous to continuous-time integration. The running sum of a sequence $x[n]$ is defined by

$$y[n] = \sum_{k=-\infty}^{n} x[k].$$

It is a linear, time-invariant operation that can be implemented as a difference equation. If the input is $x[n]$ and the running sum output is $y[n]$, then

$$y[n] = y[n-1] + x[n]$$

where zero initial conditions are assumed. This exercise will consider the performance of the running sum operation on several elementary sequences.

(a) Consider the continuous-time signals:

$$\mathrm{sq}_a(t) = u(t) - u(t-20),$$

$$\mathrm{imp}_a(t) = \sum_{k=0}^{6} \delta(t - 10k),$$

and

$$\cos(\tfrac{\pi}{5}t).$$

Sketch each of these signals and also the signals that you would expect to obtain by integrating each of them.

(b) Generate the following discrete-time signals:

(i) One period of a square wave,

$$\mathrm{sq}[n] = u[n] - u[n-20].$$

Do this by using the square wave option in **x siggen** with the pulse length specified to be 20 and the period 128.

(ii) Six periods of the impulse train,

$$\mathrm{imp}[n] = \sum_{k=0}^{6} \delta[n - 10k].$$

This can be done using the *impulse train* option in **x siggen** with period 10.

(iii) A sine wave of the form

$$\sin(\tfrac{\pi}{40}n).$$

Generate this sequence by using the *sine wave* option in **x siggen** with *alpha* $= \pi/40$, *phi* $= 0$ and length $= 128$.

Evaluate and sketch the running sum of each of these signals. Compare these results with the results found for the continuous-time case in part (a). You should observe that the running sum performs like a discrete-time integrator.

— ***EXERCISE 2.4.5.*** **Nonlinear Difference Equations**

The class of nonlinear difference equations is extremely broad, but its use is often limited by the lack of a well-developed theory for nonlinear systems. This exercise explores a nonlinear difference equation that generates a Gaussian sequence of the form

$$f[n] = \exp(-\alpha(n\tau)^2) \tag{2.9}$$

where α is positive. For this exercise, assume $\alpha = 1$ and $\tau = 0.02$.

(a) Generate the Gaussian function described by equation (2.9) for $n = 0, 1, 2, \ldots,$ 99. Do this by using the ramp option in **x siggen** and the **x gain** function to generate $n\tau$, the **x nlinear** and **x gain** functions to form $-\alpha(n\tau)^2$, and **x nlinear** to perform the exponentiation. Sketch and display your results.

(b) Write a program for evaluating this same function using the nonlinear difference equation (or recursion)

$$f[n+1] = c\frac{f^2[n]}{f[n-1]}$$

$f[0] = 1$

$f[-1] = \exp(-\alpha\tau^2)$

where $c = \exp(-2\alpha\tau^2)$. This recursion is due to Teager (see Kaiser [1]). The two required initial conditions, $f[-1]$ and $f[0]$, can be obtained directly from the Gaussian function. Evaluate the function over the range $0 \le n \le 511$ and sketch the result. Your computer program may be written in any language that you choose.

(c) Kaiser [1] has shown that Teager's difference equation can be expressed as the following pair of first-order nonlinear difference equations

$$h[n+1] = c\,h[n]$$
$$f[n+1] = h[n+1]\,f[n].$$

To begin the recursion, a value for $h[0]$ is needed. Observe from the equation that $h[0] = f[0]/f[-1]$. The initial conditions for $f[n]$ can be obtained from equation (2.9). Write a program that implements this simple pair of equations and evaluate the Gaussian as in part (b). Compare the results in all three cases. What is the maximum error?

— *EXERCISE 2.4.6.* **Autocorrelation**

In some applications, such as radar signal processing, seismic analysis, and signal enhancement, an information-bearing signal is obscured by some form of noise. The goal is to estimate certain signal characteristics from the corrupted signal. This exercise will examine autocorrelation as a procedure for determining the period of a quasi-periodic sequence that is embedded in additive noise.

The autocorrelation of a real signal $v[n]$ is defined as

$$\mathcal{A}[n] = \sum_{m=-\infty}^{\infty} v[n+m]\, v[m].$$

It is related to the convolution of $v[n]$ with itself, since

$$\mathcal{A}[n] = v[n] * v[-n].$$

$\mathcal{A}[n]$ provides a measure of similarity for different alignments of the signal.

(a) Generate 128 points of the sequence $x[n]$, where

$$x[n] = \tfrac{1}{2}\sin\left(\tfrac{\pi}{12}n\right) + \sin\left(\tfrac{\pi}{24}n\right),$$

using the *sine wave* option in `x siggen` and the `x add` function. Next, generate 128 samples of a random noise sequence $r[n]$ using `x rgen`. Use `x gain` and `x add` to generate the sequence

$$y[n] = x[n] + 3r[n]$$

that will serve as the noisy signal. Examine $x[n]$ using the `x view` function and determine the period of this signal. Now examine $y[n]$ using `x view` and observe that it is difficult to determine the period by inspection.

(b) The autocorrelation function can be used to estimate the periodicity of the signal $x[n]$. In part (a) you determined the period of $x[n]$ by inspection. Compute the autocorrelation of $x[n]$ using `x reverse` and `x convolve` and sketch your results. Using the concept of graphical convolution, explain why peaks in the autocorrelation function appear at values of n that are multiples of the period.

(c) The autocorrelation function can be used to estimate the periodicity of the signal $x[n]$, given the noisy signal $y[n]$. Specifically, the distance between the peak at the origin and either of the adjacent peaks in the autocorrelation function gives an estimate of the period of $x[n]$. Display and sketch the autocorrelation function of $y[n]$ and estimate the period of the underlying signal $x[n]$.

— *EXERCISE 2.4.7.* **Cross Correlation**

Cross correlation can be viewed as a generalization of autocorrelation. The cross correlation of two real sequences $v[n]$ and $w[n]$ is defined as

$$\Phi[n] = \sum_{m=-\infty}^{\infty} v[n+m]\, w[m].$$

If $v[n]$ and $w[n]$ are equal, the cross correlation and autocorrelation are identical. Cross correlation also has a convolutional interpretation,

$$\Phi[n] = w[n] * v[-n].$$

It can be used to determine when two sequences are best aligned in time. For example, if $x[n]$ and $v[n]$ are two different but similar sequences, one may wish to determine which value of ℓ results in the best approximation

$$x[n] \approx v[n+\ell].$$

This can be done by computing the cross correlation and looking for the time index, n, of the maximum peak of $\Phi[n]$. To examine this point further, create the sequences $x[n]$ and $v[n]$ where $x[n]$ is a random finite length sequence that is nonzero in the range $-32 \le n \le 32$. Let

$$v[n] = x[n] * h[n]$$

where $h[n]$ is a 15-point lowpass filter with cutoff frequency of $2\pi/3$ and starting point zero. Use `x rgen` to create $x[n]$ and `x lshift` to shift the starting point to -32. The *Hamming window* option in `x fdesign` can be used to create $h[n]$.

(a) Use `x view2` to display and sketch $x[n]$ and $v[n]$. These signals are not the same but are similar in terms of their distribution of samples. Such sequences have a degree of correlation as defined by their cross correlation function. Now compute and sketch $\Phi[n]$. Identify the index n corresponding to the peak in $\Phi[n]$. To simplify the task of finding the peak location, use the `x extract` function to extract the 20-point block in $\Phi[n]$ beginning at -10. Write this block to a sequence file that also begins at -10 and display it. By examining the extracted sequence file, you will be able to determine the peak location by inspecting the display. This location provides an indication of how much displacement is necessary for $v[n]$ to be most nearly aligned with $x[n]$. Based on your examination of the peak location, by how many samples should $v[n]$ be shifted to produce the best alignment between the sequences? Check your answer by shifting $v[n]$ by that amount and visually comparing it to $x[n]$. Approximately how many multiplies and adds are required to compute the cross correlation in this example?

(b) In this part, a method for increasing the efficiency of computing the cross correlation function is examined. Quantize each of the sequences into binary sequences in the following way:

$$\hat{x}[n] = \begin{cases} 1 & \text{if} \quad x[n] \ge 0 \\ -1 & \text{if} \quad x[n] < 0 \end{cases}$$

$$\hat{v}[n] = \begin{cases} 1 & \text{if} \quad v[n] \geq 0 \\ -1 & \text{if} \quad v[n] < 0. \end{cases}$$

Use the `x quantize` function to do this with $min = -1$ and $max = 1$ and the number of levels equal to two. Compute the cross correlation function of $\hat{x}[n]$ and $\hat{v}[n]$. Use `x view2` to display this cross correlation function and the one in part (a) simultaneously. What is the required shift in $\hat{v}[n]$ to produce the best alignment using this approach? How many multiplies and adds are required? How do the methods in parts (a) and (b) compare? (*Note:* Multiplication by $+1$ or -1 should not be counted as a multiply.) You will not observe any speedup in the software that is provided because those programs do not check to avoid multiplications by ± 1.

(c) Two random signals generated with different seed values should not bear any similarity to each other and are said to be uncorrelated. Consequently, they should not produce a cross correlation function with a dominant peak indicating a preferred alignment. Use `x rgen` and `x lshift` as before to create another random sequence, $y[n]$, (with a different seed) in the same range. Display and sketch the cross correlation of $x[n]$ and $y[n]$. Observe that there is no dominant peak that clearly marks a "best alignment" between the two sequences.

— *EXERCISE 2.4.8.* **Random Sequences**

It is often important to generate a random sequence of numbers that has a Gaussian distribution instead of a uniform distribution. There are many methods available for doing this. A simple one that transforms a uniform random sequence into a Gaussian random sequence is investigated here.

The central limit theorem states that the sum of N identically distributed, independent random variables approaches a Gaussian distribution in the limit as N goes to infinity.

(a) Generate a uniform random sequence $x_1[n]$ of length 128 using `x rgen`. Use the `x gain` function to multiply the sequence by 100 to produce a random sequence, $y[n]$, with amplitude values in the range -50 to 50. To examine the distribution of $y[n]$, compute and display its histogram using `x histogram` and `x view`. The histogram is a plot of sample amplitude on the x-axis versus the number of occurrences of that amplitude on the y-axis. If the histogram is roughly flat, the sequence is said to be uniformly distributed.

(b) Now generate eight additional random sequences $x_2[n]$, $x_3[n]$, ..., $x_9[n]$. Add the sequences $x_1[n], \ldots, x_9[n]$ together using `x add` to form $y_2[n]$. Multiply $y_2[n]$ by 11 so that its amplitude range will be from -50 to $+50$. Display and sketch the histogram of the new sequence. Does this distribution more closely resemble that of a Gaussian random sequence?

— *EXERCISE 2.4.9.* **Gaussian Random Sequences**

This problem considers a different method for creating a Gaussian sequence [2] with a specific standard deviation σ. With this method, a uniform random sequence

$0 < x_1[n] \leq 1$ is first transformed into a Rayleigh-distributed random sequence $r[n]$ by the relation

$$r[n] = \sqrt{2\sigma^2 \ln\left(\frac{1}{x_1[n]}\right)}.$$

Two uncorrelated (and therefore independent) Gaussian random variables, $g_1[n]$ and $g_2[n]$ can then be formed using the equations

$$\begin{aligned} g_1[n] &= r[n]\cos(2\pi x_1[n+1]), \\ g_2[n] &= r[n]\sin(2\pi x_1[n+1]). \end{aligned}$$

The standard deviation is the value of σ used to generate $r[n]$.

(a) Write a macro that will accept a uniform random sequence as its input and produce $g_1[n]$ and $g_2[n]$ as its outputs. The functions **x divide**, **x log**, **x multiply**, **x lshift**, and **x nlinear** will be helpful in computing $r[n]$. Use your macro to generate $g_1[n]$ and $g_2[n]$ with $\sigma = 1$ and length 1024. List all necessary functions and intermediate signal files in your macro. Note that the random signal $x_1[n]$ should have an amplitude range between 0 and 1. Therefore you should add a *block* sequence with amplitude .5 to the random sequence generated by **x rgen** in constructing $x_1[n]$.

(b) Determine the maximum absolute value A for $g_1[n]$ and $g_2[n]$. Multiply each by $1000/A$ so that the amplitude range of each signal will be between -500 and $+500$. Now compute the histogram of $g_1[n]$ and $g_2[n]$ and sketch the results. Does this distribution appear to be Gaussian?

(c) Cross correlation was introduced in Exercise 2.4.7. Compute the cross correlation function for $g_1[n]$ and $g_2[n]$. Do these sequences appear to be uncorrelated? Justify your conclusion.

2.5 REFERENCES

[1] J. F. Kaiser, "On the Fast Generation of Equally Spaced Values of the Gaussian Function A exp $(-at * t)$," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, pp. 1480–1481, Oct. 1987.

[2] L. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.