Sampling

4

Most signals originate as continuous-time waveforms. Sampling, or analog-to-digital (A/D) conversion, allows these analog signals to be represented in digital form. Ideal sampling is defined by

$$x[n] = x_a(t)|_{t=nT} = x_a(nT), \qquad -\infty < n < \infty. \tag{4.1}$$

The continuous-time (analog) signal, $x_a(t)$, is a function of the continuous (time) variable t and $x[n]$ is the sequence of sample values. The constant T, known as the *sampling period*, defines the time spacing between samples. Its reciprocal, $f_s = 1/T$, is the *sampling rate*, which is measured in samples per second.[1]

This chapter addresses many of the issues that surround A/D conversion and its dual process, D/A conversion. D/A conversion reconstructs bandlimited continuous-time signals from their sample values. The next section examines the processes of sampling and signal reconstruction when there is no quantization of the sample values. In the nonideal world sampling represents only one of two aspects of the digitization process. In addition to the discretization of the t variable, the sample values must also be quantized in amplitude. Quantization is a nonlinear operation that will be treated separately in Section 4.2. The final section of this chapter looks at sampling-rate conversion. This is the process in which a sequence of sample values that have been taken at one sampling rate are digitally mapped into another sequence of sample values that are taken at a different rate. It is often important that the process of sampling-rate conversion preserve the essential properties of the Fourier transforms of all of the signals that are involved.

[1]The term *sampling frequency* is often used in lieu of *sampling rate*, in which case the unit of measurement is Hz or cycles per second.

4.1 SAMPLING AND RECONSTRUCTION

Any discrete-time representation for continuous-time waveforms must satisfy several requirements: it must be invertible, it must preserve the essential information in the waveform, and it must be implementable. Remarkably, periodic sampling meets all of these conditions when the continuous-time waveform is bandlimited. We can verify this by looking at sampling in both the time and frequency domains.

Let $x_a(t)$ be the continuous-time signal that is sampled and let $X_a(\Omega)$ be its continuous-time Fourier transform. These two functions are related by

$$X_a(\Omega) = \int_{-\infty}^{\infty} x_a(t) e^{-j\Omega t} dt$$

where $X_a(\Omega)$ is a complex function of the frequency variable Ω with units of radians per second (rad/sec), and the subscript "a" signifies an analog or continuous-time waveform. An analogous Fourier relationship exists for digital signals as discussed in Chapter 3. The discrete-time sequence of samples, $x[n]$, and its discrete-time Fourier transform, $X(e^{j\omega})$, are related by

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

where $X(e^{j\omega})$ is a complex, continuous, and periodic function of the discrete-domain frequency variable ω, which is measured in radians. The ideal A/D converter receives as an input a continuous-time signal and produces the discrete-time signal $x[n]$ as its output, as shown in Fig. 4.1.

Figure 4.1. An ideal A/D converter.

The analog and digital signals are related by equation (4.1) in the time domain. Their Fourier transforms are related by

$$X(e^{j\omega}) = \frac{1}{T} \sum_{r=-\infty}^{\infty} X_a\left(\Omega + \frac{2\pi r}{T}\right) \qquad \text{where } \Omega = \frac{\omega}{T}. \tag{4.2}$$

This relationship states that $X(e^{j\omega})$ is a scaled superposition of shifted copies of $X_a(\Omega)$ that have been normalized in frequency. A graphical interpretation of this formula is given in Fig. 4.2. The spectrum shown at the top of Fig. 4.2 shows a signal that is confined to the frequency region between $-\Omega_0$ and Ω_0. Such signals are said to be *bandlimited* to Ω_0. The middle graph shows the resulting DTFT when $\Omega_0 < \pi/T$.

The bottom graph shows the same DTFT for the case where $\Omega_0 > \pi/T$. Equation (4.2) states that the DTFT is composed of a sum of an infinite number of identical, uniformly shifted replicas of $X_a(\Omega)$. This sum produces the periodic signal $X(e^{j\omega})$. The spacing between the midpoints of the spectral copies is 2π radians. This is consistent with the fact that the period of any DTFT is 2π radians.

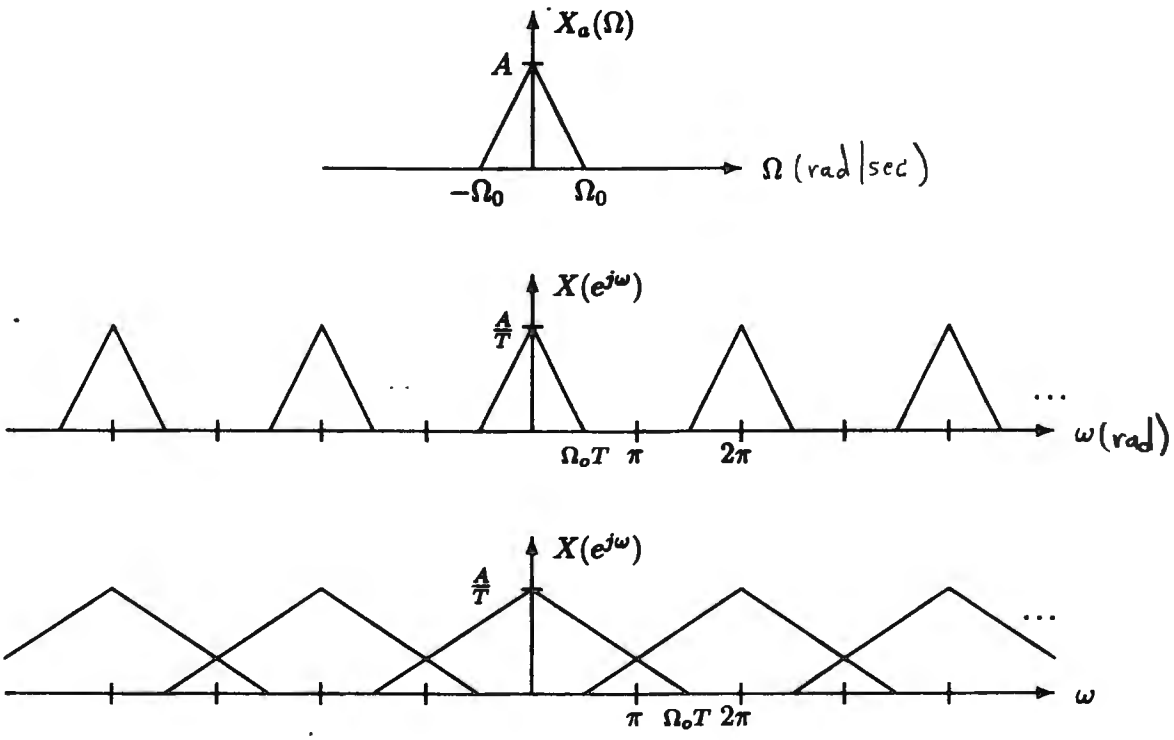

Figure 4.2. A look at sampling from the frequency domain. (Top) Spectrum of a bandlimited continuous-time signal. (Middle) Spectrum of the sampled signal when $\Omega_0 < \pi/T$. (Bottom) Spectrum of the sampled signal when $\Omega_0 > \pi/T$.

The dual process in which discrete signals are converted into continuous-time waveforms is called *digital-to-analog (or D/A) conversion.* A D/A converter implements the relation

$$\begin{aligned} y_a(t) &= \sum_{n=-\infty}^{\infty} y[n] h_a(t - nT) \\ &= \underbrace{\left\{ \sum_{n=-\infty}^{\infty} y[n]\delta(t - nT) \right\}}_{y_c(t)} * h_a(t) \\ &= y_c(t) * h_a(t). \end{aligned} \tag{4.3}$$

Using the bottom equation, signal reconstruction can be viewed as a two-step process. The sequence is first converted into a weighted impulse train, $y_c(t)$. This impulse train is then passed through a continuous-time filter with the impulse response $h_a(t)$. This is shown in Fig. 4.3. In the frequency domain equation (4.3) becomes

$$Y_a(\Omega) = Y_c(\Omega)H_a(\Omega), \tag{4.4}$$

where

$$\begin{aligned} Y_c(\Omega) &= \mathcal{F}\left\{\sum_{n=-\infty}^{\infty} y[n]\delta(t-nT)\right\} \\ &= \sum_{n=-\infty}^{\infty} y[n]e^{-j\Omega Tn} \\ &= Y(e^{j\Omega T}) \end{aligned} \tag{4.5}$$

and the operator $\mathcal{F}\{\cdot\}$ denotes the (continuous-time) Fourier transform. The Fourier transform of the continuous-time impulse train $y_c(t)$ is identical to the DTFT of the sequence of samples except for the denormalization of the frequency variable. The original (analog) and normalized (discrete) frequency variables are related by

$$\omega = \Omega T. \quad \text{and} \quad \Omega = \omega \cdot f_s \tag{4.6}$$

The ideal D/A conversion process can be clearly seen in the frequency domain. The starting point is $Y(e^{j\omega})$, which is periodic in ω with period 2π, as shown in Fig. 4.4. The first step is to convert $y[n]$ into the weighted analog impulse train $y_c(t)$. In the frequency domain this corresponds to a change in the frequency variable: $\omega \Rightarrow \Omega T$. $Y_c(\Omega)$ is periodic in Ω with a period of $2\pi/T$. It contains an infinite number of periodic replicas of $Y_a(\Omega)$. The second and final step removes these replicas by filtering $Y_c(\Omega)$ with an ideal lowpass filter $H_a(\Omega)$ with a cutoff frequency of π/T and a gain of T.

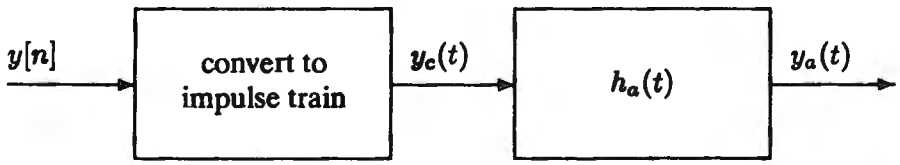

Figure 4.3. An interpretation of the D/A conversion process.

A/D and D/A converters allow for discrete-time processing of continuous-time signals and are often used together. Assume that an ideal A/D converter is cascaded with an ideal D/A, as shown in Fig. 4.5. If the spectrum of $x_a(t)$ resembles the one shown in the top portion of Fig. 4.2 with $\Omega_0 < \pi/T$, the spectrum at the input to the filter in the D/A converter is the same as the one shown in the middle of Fig. 4.2. The output of the filter is equal to $x_a(t)$ when $h_a(t)$ is an ideal lowpass with gain T and cutoff frequency π/T radians. Therefore,

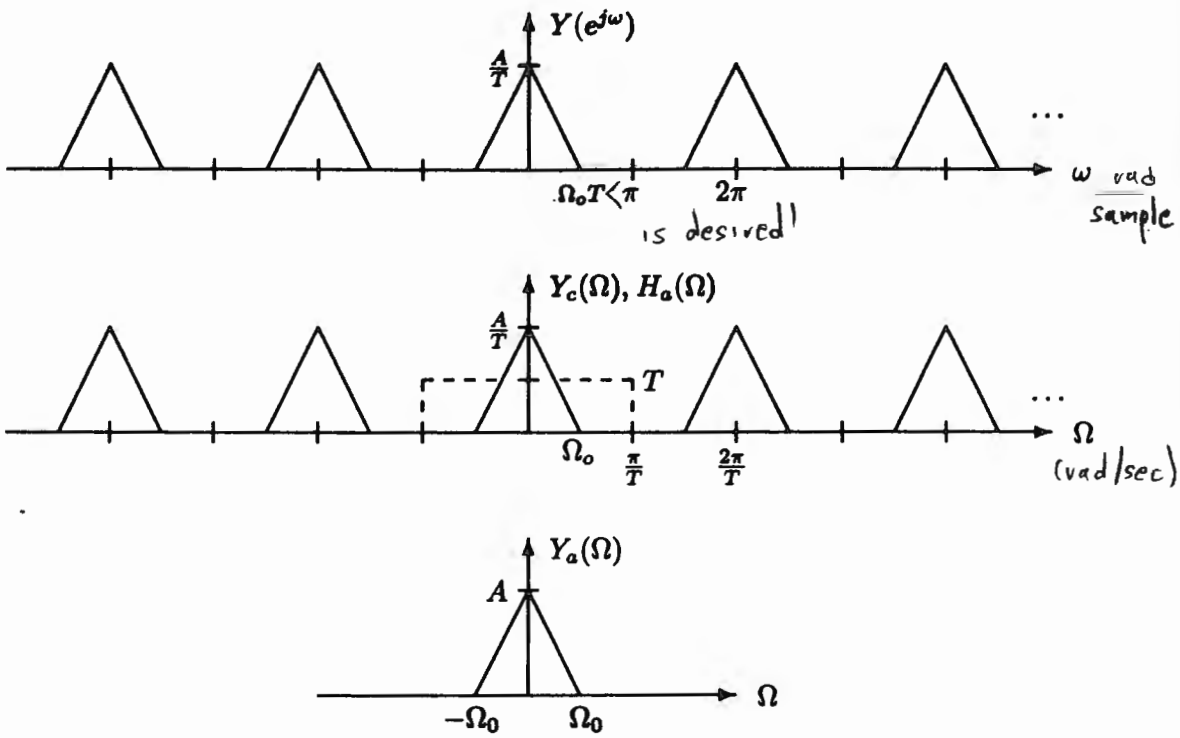

Figure 4.4. Relationships between the spectra involved in ideal D/A conversion: (top) $Y(e^{j\omega})$; (middle) $Y_c(\Omega)$; (bottom) $Y_a(\Omega)$.

$$h_a(t) = T\frac{\sin \pi t/T}{\pi t} \tag{4.7}$$

(which is the impulse response of an ideal lowpass filter) and

$$y_a(t) = T \sum_{k=-\infty}^{\infty} y[k] \frac{\sin(\pi/T)(t - kT)}{\pi(t - kT)}. \tag{4.8}$$

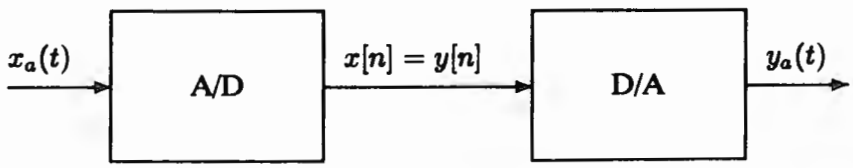

Figure 4.5. A cascade of an ideal A/D converter with an ideal D/A.

The ideal lowpass filter in equation (4.7) is not causal and it cannot be realized electronically. Therefore, practical D/A converters make use of causal substitutes

for it. Nevertheless, the ideal D/A converter is important because it represents the standard against which all practical D/A converters should be compared.

It should be clear that the ideal D/A converter inverts the sampling operation only when:

1. $X_a(\Omega)$ is bandlimited to Ω_0. By definition, an analog signal, $x_a(t)$, is *bandlimited* if
$$X_a(\Omega) = 0 \qquad |\Omega| > \Omega_0.$$
The frequency Ω_0 is called the *Nyquist frequency*. An example of a bandlimited signal, $X_a(\Omega)$, is illustrated in Fig. 4.2 (top).
2. The condition $\Omega_0 < \pi/T$ is true. This prevents aliasing. When this condition is violated, the output of the D/A will differ from the input to the A/D. An example is shown in Fig. 4.6. When $\Omega_0 < \pi/T$, the reconstruction is perfect, as shown in Fig. 4.6*b*. However, when $\Omega_0 > \pi/T$, $Y_a(\Omega)$ is corrupted by aliasing, as illustrated in Fig. 4.6*c*.

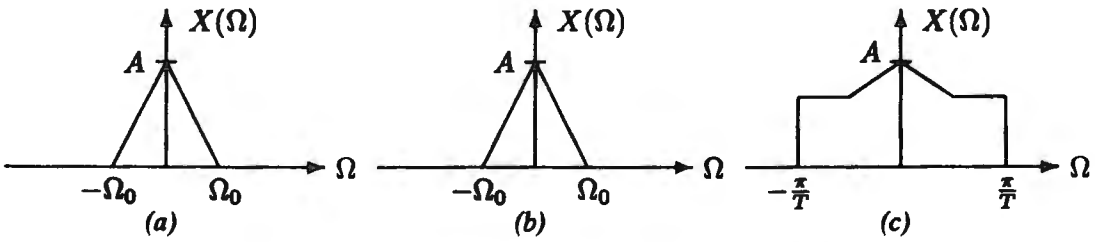

Figure 4.6. The effect of the sampling period on the performance of cascaded ideal A/D and D/A converters as shown in Fig. 5. (a) Bandlimited spectrum of the input to the A/D. (b) Output spectrum when $\Omega_0 < \frac{\pi}{T}$. (c) Output spectrum corresponding to a value of $\Omega_0 > \frac{\pi}{T}$.

The minimum sampling frequency for exact reconstruction, $f_s = \Omega_0/\pi$ samples/second, is called the *Nyquist rate*. If a signal is sampled at less than the Nyquist rate, the Fourier spectra in equation (4.2) overlap, as illustrated in Fig. 4.2 (bottom). This spectral overlap, which is called *aliasing*, prevents the signal from being recovered exactly. The reconstruction condition for a bandlimited signal $x_a(t)$ with a highest frequency component Ω_0 can be formalized as a sampling theorem and stated in several equivalent ways:

1. $x_a(t)$ may be reconstructed if $\Omega_0 < \pi/T$, or;
2. $x_a(t)$ must be sampled above its Nyquist rate, i.e., $f_s > \Omega_0/\pi$ samples/sec, or;
3. $x_a(t)$ must be sampled at a rate greater than twice the Nyquist frequency, i.e., $f_s > 2\Omega_0/(2\pi)$.

The exercises that follow provide illustrations of sampling, aliasing, and related issues associated with the discrete-time processing of continuous-time signals. Because of the difficulty of representing an analog waveform in a computer, a digital represen-

tation is used to simulate analog signals. In these exercises, several special functions will be used. These are rather restrictive and are only intended for use in the exercises in this chapter. The inputs and outputs, when appropriate, are displayed as continuous functions to give the appearance of an analog signal. A brief description of these special functions follows.

x aconvolve

The **x aconvolve** function forms the linear convolution of two simulated analog signals,

$$y_a(t) = x_a(t) * h_a(t).$$

It is executed by typing

$$\mathbf{x\ aconvolve} \quad \overbrace{\text{(input1)}}^{x_a(t)} \quad \overbrace{\text{(input2)}}^{h_a(t)} \quad \overbrace{\text{(output)}}^{y_a(t)}$$

x afilter

This special function is used to simulate analog filtering of the form

$$y_a(t) = h_a(t) * x_a(t)$$

where $x_a(t)$ and $y_a(t)$ are simulated analog input and output signals and $h_a(t)$ is the impulse response of the simulated analog filter that is provided for you in the file **hh**. To invoke the function, type

$$\mathbf{x\ afilter} \quad \overbrace{\text{(input)}}^{x_a(t)} \quad \overbrace{\mathbf{hh}}^{h_a(t)} \quad \overbrace{\text{(output)}}^{y_a(t)} \quad M$$

M is the number of output samples. Since simulated analog signals are actually represented by discrete samples, this parameter is a necessary input. In these exercises 500 is a good value to specify for this number.

x alook

This function allows you to display simulated analog signals in the time domain. The signal is displayed as a continuous-time waveform where the time axis is measured in milliseconds (msec). It is invoked by typing

$$\mathbf{x\ alook} \quad \overbrace{\text{(inputfile)}}^{x_a(t)}$$

The function always displays the simulated analog signal in a 100-millisecond time interval. Pressing the "*g*" key allows you to turn the grid of tic marks on or off to suit

your preference. The "*q*" key will terminate the display immediately and return you to DOS.

x ft

This function simulates the Fourier transform of an analog signal. It is invoked by typing

$$\texttt{x ft} \quad \overbrace{\text{(inputfile)}}^{x_a(t)}$$

A display option menu will appear that will ask for display of the magnitude, log magnitude, phase, real part, imaginary part, magnitude and phase, or real and imaginary parts. The frequency axis should be interpreted as if it extends infinitely in both directions. Once the plot is displayed on the screen, three keys are available for use. Pressing the "*g*" key allows you to turn the grid of tic marks on or off to suit your preference. The "*esc*" key can be used to return you to the display option menu. The "*q*" key will terminate the display immediately and return you to DOS.

x sample

This function allows a simulated analog signal, $x_a(t)$, to be sampled. It simulates the sampling operation

$$x[n] = x_a(nT)$$

where T is the sampling period in milliseconds. The sampling period can be included on the command line or the program will prompt you for it. The program is invoked by typing

$$\texttt{x sample} \quad \overbrace{\text{(input)}}^{x_a(t)} \quad \overbrace{\text{(output)}}^{x[n]} \quad T$$

The output sequence, $x[n]$, is truncated to 101 samples in all cases. This is done so that various sampled signals investigated in the exercises will all have the same number of samples and thus permit fair comparisons to be made. Unlike the other DSP functions, x sample is not equipped to handle input and output files with the same name.

x sti

This program simulates the "sample-to-impulse" conversion operation that transforms a sequence such as $y[n]$ into the weighted analog impulse train $y_c(t)$. It is invoked by typing

$x(t) = e^{j\Omega t}$
$x[n] = x(n \cdot T_s)$
$\Big\} x[n] = e^{j\Omega n T_s} = e^{j(\Omega \cdot T_s)n} = e^{j\omega_d n}$
where $\omega_d = \Omega \cdot T_s$ in rad

Ω : analog frequency in rad/sec

x sti $\overbrace{\text{(input)}}^{y[n]}$ $\overbrace{\text{(output)}}^{y_a(t)}$ T

T is the integer number of milliseconds between impulses. If this parameter is omitted, the function will ask you for its value.

—*EXERCISE 4.1.1.* **Ideal Sampling (Frequency Domain)**

This exercise considers the effects of sampling a continuous-time signal on the Fourier transform of the signal. The signal considered is the sine wave

$$x_a(t) = \sin \frac{2000\pi}{12} t.$$

$f = 500/6$ Hz T 1 sec

2000 n / 12

To simulate this signal, generate 4000 samples of

$$\sin(0.2618n)$$

1.5236

starting at $n = 0$, using the *sine wave* option in x **siggen**. Store the resulting sequence in a file.

(a) Use the function x **sample** to sample $x_a(t)$ with a sampling period, $T = 1$ msec. Call the resulting signal $x[n]$. Calculate and display the magnitude of the Fourier transform of $x[n]$ using x **dtft**. Carefully examine equation (4.2) and the plot of $|X(e^{j\omega})|$ to determine the frequency (in radians) of the discrete sinusoid. Recall that the DTFT of an ideal sinusoid is a pair of impulses. The location of these impulses on the frequency axis represents the frequency of the sinusoid.

(b) Repeat the sampling procedure described in part (a) for sampling periods of 4, 10, 15, 20, 24, 30, and 39 milliseconds. Display and sketch the plots of the magnitude response and record the frequencies of the discrete sinusoids in each case. How does the behavior of the peak locations of the DTFT magnitude change as the sampling period is increased? This behavior is due to *aliasing*.

(c) Based on the definition, determine the Nyquist rate for $x_a(t)$.

—*EXERCISE 4.1.2.* **Ideal Sampling (Time Domain)**

This exercise looks at aliasing in the time domain. To generate $x_a(t)$ create 5200 samples of

$$\sin(0.1309n)$$

$f = \frac{12}{\pi}$ Hz, $\omega_d = \pi/2$ rad

starting at $n = 0$ using the *sine wave* option in x **siggen**. Sample this waveform using x **sample** with a sampling period $T = 1$ millisecond. Use the function x **view** to display this sampled waveform.

(a) Repeat the sampling process described above for sampling periods of 2 and 4 milliseconds. Display and sketch the sampled waveforms. Determine the period and digital frequency (in radians) of each of the sampled signals. As the sampling rate is lowered, does the (digital) frequency of the signal increase or decrease? Remember that the sampling rate is the reciprocal of the sampling period.

$T_{(in\ 1)} = \frac{T_{analog}}{T_{analog}} \quad \frac{1/f}{T_s} = \frac{1}{T_s}$

(b) Now sample $x_a(t)$ with sampling periods of 50 and 51 milliseconds. Again use `x view2` to display the outputs and provide sketches of these signals. These are undersampled signals that contain aliasing. By comparing the periods to those generated in part (a), what does aliasing look like in the time domain?

EXERCISE 4.1.3. **Ideal Reconstruction (Frequency Domain)**

As we saw earlier, the process of ideal D/A conversion is equivalent to a two-step process. First, the sequence $y[n]$ is converted into a weighted continuous-time impulse train, $y_c(t)$. Then the impulse train is lowpass filtered to form $y_a(t)$. In this exercise D/A operations are examined in the frequency domain.

(a) Use the triangle wave option of `x siggen` to generate five periods of a triangle wave $y[n]$ with a period of 20 beginning at $n = 0$. Display and sketch the magnitude of the DTFT of $y[n]$.

(b) The first component of the D/A implements the "sample-to-impulse train" operation. This is simulated by the function `x sti` that converts the samples into impulses spaced T milliseconds apart. Use `x sti` with sampling period $T = 5$ to produce $y_c(t)$. Then using `x ft`, display and sketch $|Y_c(\Omega)|$. Note that $|Y_c(\Omega)|$ is periodic in Ω, but that `x ft` only evaluates it over a limited frequency range. The replication phenomenon that you observe is called *imaging*.

(c) The spectral replicas due to the sample-to-impulse conversion need to be removed. Use `x afilter` to implement the lowpass filter `hh` with the input signal $y_c(t)$ to produce $y_a(t)$. (Notice that you must specify $y_c(t)$ as the first argument of `x afilter` and `hh` as the second.) Specify 500 for the number of output points. Use `x ft` to display and sketch the magnitude of the frequency response of the lowpass filter and also $|Y_a(\Omega)|$. Recall that the sampling rate is $f_s = 1/T$ and label the frequency axis correctly in your sketch.

EXERCISE 4.1.4. **Commercial D/A Converters**

Commercially available D/A converters do not produce smooth continuous-time output signals, but instead create a piecewise-constant (staircase) output. This can be modeled by setting $h_a(t)$ in equation (4.3) equal to

$$h_a(t) = \begin{cases} 1, & 0 \le t \le T \\ 0, & \text{otherwise.} \end{cases}$$

The behavior of these D/As will be examined by simulation. Let the input to the D/A, $x[n]$, be 101 points of the sequence

$$\sin(\pi n/8 + \pi/10)$$

starting at $n = 0$. Use the *sine wave* option of `x siggen` to create $x[n]$.

(a) Let $x_s(t)$ be the output of a *sample-to-impulse* converter with $x[n]$ as the input. Use the `x sti` function with $T = 5$ to generate the simulated continuous-time

signal $x_s(t)$. Display and sketch this signal using **x alook**. Let the output of the D/A be

$$\hat{x}_a(t) = x_s(t) * h_a(t).$$

To simulate $h_a(t)$, create a sequence of 5 ones using the *block* option in **x siggen** with $n = 0$ as the starting point. Display $h_a(t)$ using **x alook**. Then use the **x aconvolve** function to evaluate the convolution. Display and sketch $\hat{x}_a(t)$ using **x alook**.

(b) Now look at $|\hat{X}_a(\Omega)|$ using **x ft**. How does the stair-step output affect the spectrum of the signal?

(c) A simple lowpass filter can be used to filter $\hat{x}_a(t)$ so that it more closely resembles

$$\sin\left(\frac{\pi}{16}t + \frac{\pi}{10}\right). \quad \sin\left(25 \cdot \frac{\pi}{10}\right)$$

Such a filter $h(t)$ is stored in the file **hh**. Use the **x ft** function to display the frequency response of this filter. Now use **x afilter** to lowpass filter $\hat{x}_a(t)$. Be sure to specify the input as the first argument and **hh** as the second argument on the command line. Specify 500 for the number of output points. Use **x alook** to display and sketch the result, $\hat{x}_a(t)$. Use **x ft** and **x alook** to display and sketch $|X_a(\Omega)|$.

Comment. In many real-world applications, the outputs of commercial D/As are passed through lowpass filters, as in this example, to smooth out the staircase-like structure and partially compensate for the D/A response.

— ***EXERCISE 4.1.5.*** **DTFT of a Sampled Signal**

Do this exercise without the aid of your computer. Consider the case of a band-limited continuous-time signal with Fourier transform $X_a(\Omega)$ that has been sampled above its Nyquist rate so that no aliasing is present. Then

$$X(e^{j\omega}) = \frac{1}{T}X_a(\omega/T), \qquad -\pi \le \omega \le \pi.$$

Recall that $\Omega = \omega/T$.

(a) Consider the case where

$$X_a(\Omega) = \begin{cases} 1, & -2\pi(5000) \le \Omega \le 2\pi(5000) \\ 0, & \text{otherwise.} \end{cases}$$

Sketch $X(e^{j\omega})$ for the following sampling rates:

(i) $f_s = 15,000$ samples/sec.

(ii) $f_s = 30,000$ samples/sec.

(iii) $f_s = 10,000$ samples/sec.

(b) When the signal is undersampled, the spectrum is aliased. In such cases, the general expression

$$X(e^{j\omega}) = \frac{1}{T}\sum_{r=-\infty}^{\infty} X_a\left(\Omega + \frac{2\pi r}{T}\right) \qquad \text{for } \Omega = \frac{\omega}{T}$$

relates the discrete-time and continuous-time Fourier transforms. Sketch $X(e^{j\omega})$ when $x_a(t)$ is sampled at the following rates:

(i) $f_s = 8000$ samples/sec.

(ii) $f_s = 5000$ samples/sec.

(iii) $f_s = 3000$ samples/sec.

EXERCISE 4.1.6. **Sampling Non-Bandlimited Signals**

Many continuous-time signals encountered in the real world have spectra that are not strictly bandlimited, but are bandlimited only in an approximate sense. For example, a speech signal might contain some spectral energy above 10 kHz. But, if this high-frequency energy is small, the speech will not suffer noticeable distortion if it is removed by lowpass filtering. Thus, a good strategy to apply to signals of this type that need to be sampled is to determine a frequency above which the signal has negligible energy, and use this frequency to determine the sampling rate. When the sampling frequency is sufficiently high, the aliasing introduced will be small. This exercise addresses this aspect of sampling and does not involve the use of the computer.

Consider the analog input signal with Fourier transform

$$X_a(\Omega) = e^{-.01|\Omega|}.$$

Notice that $X_a(\Omega)$ decays exponentially as $|\Omega|$ increases. When this signal is sampled with any sampling rate, f_s, aliasing will be introduced—the larger the value of f_s, the smaller the amount of aliasing.

(a) If the spectral distortion due to aliasing is not permitted to exceed 1% at any frequency, what is the minimum sampling rate that is permissible?

(b) Now assume that an analog lowpass filter (often called an antialiasing filter) is used to prefilter the signal. This is a common practice because it removes the high-frequency spectral components so that they cannot alias onto the lower frequencies. If the maximum allowable spectral distortion is still 1% at any frequency, what is the minimum sampling rate that is permissible?

4.2 QUANTIZATION

Processors for discrete-time signals must be capable of adding, multiplying, and storing sample values. This means that each sample value must be representable in a form

that requires only a finite number of bits. There are a variety of ways for doing this, but all of them share the property that only a limited number of sample amplitudes can be represented. Once a sample value has been measured, it must be approximated by one of these representable values. This process is called *quantization*.

Two examples of quantizer characteristics are shown in Fig. 4.7. The input to the quantizer is $x[n]$ and the output is $\hat{x}[n]$. These are both examples of *uniform* quantizers because the differences between successive quantization levels are equal to a constant value, Δ, which is called the *step size*. The difference between the unquantized and the quantized sample values

$$e[n] = \hat{x}[n] - x[n] \tag{4.9}$$

is called the *quantization error*. If $X_{\min} < x[n] \leq X_{\max}$, then

$$|e[n]| \leq \frac{\Delta}{2}.$$

When $x[n]$ extends beyond these limits, the error can be much larger. This condition is called *saturation* or *overload*. Saturation can be minimized by carefully controlling the amplitude of the input to the A/D converter.

In general, the larger the number of levels, the more accurate the representation or equivalently the smaller the quantization error. Minimizing quantization error is an important issue in signal processing. A common measure of quantization error (noise) is the *signal-to-noise ratio (SNR)*. Rearranging equation (4.9) gives

$$\hat{x}[n] = x[n] + e[n].$$

The quantized signal $\hat{x}[n]$ is equal to the true signal plus the quantization noise. The signal-to-noise ratio (SNR) is defined as

$$\text{SNR} = 10 \log_{10} \frac{\sum_n x^2[n]}{\sum_n e^2[n]} \qquad \text{(in dB)}$$

where *dB* denotes *decibel*. The quantizer at the top of Fig. 4.7 is called a *midriser uniform quantizer*. It does not have a level corresponding to the value zero. The lower quantizer is called a *midtread quantizer*. When $X_{\min} = -X_{\max}$ a quantizer with an even number of levels will be a midriser and one with an odd number of levels will be a midtread. In practice, midtread quantizers are popular because they include the zero-value quantization level. The quantizers associated with most A/D converters have 2^B levels, where B is the number of bits that their output registers can hold. Since 2^B is even, midtread quantizers cannot be symmetrical about zero. The common convention is for there to be one extra negative quantization level. In the exercises that follow, quantization is examined in further detail.

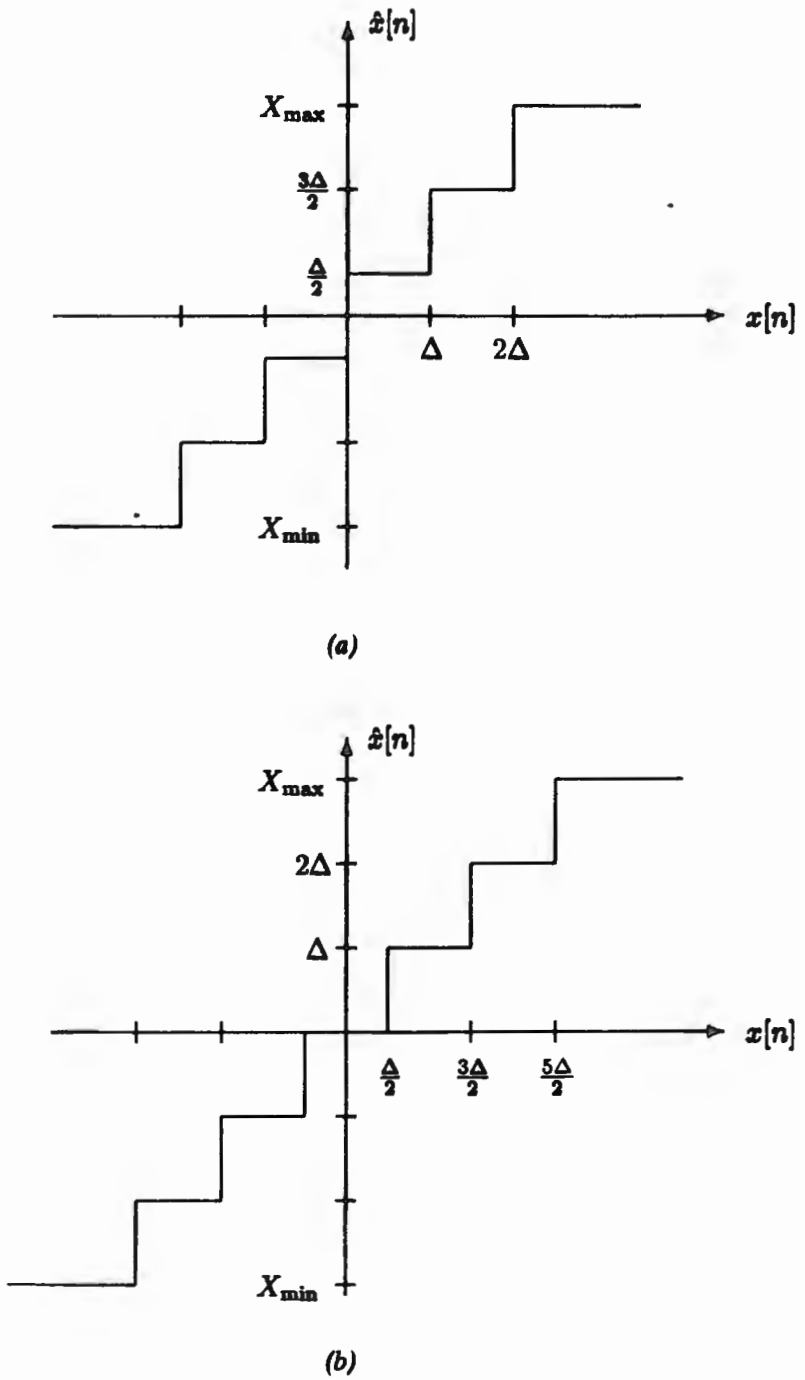

Figure 4.7. Two examples of uniform quantizers. (a) Six-level midriser quantizer. (b) Seven-level midtread quantizer.

EXERCISE 4.2.1. Quantization

To illustrate quantization generate the ramp signal $r[n] = n/999$ in the range -1000 to $+1000$. Do this by first using **x siggen** to create a ramp with slope 1/999, length 1000, and starting point zero. Next use **x reverse** to rotate the ramp about $n = 0$. Multiply the rotated ramp by -1 using **x gain** and add it to the original ramp using **x add**. Use **x look** to display $r[n]$ and verify that it is a straight line in the proper range.

If $r[n]$ is quantized to 3 bits/sample, then 8 unique amplitude levels can be represented. The first of these might be assigned the binary code 000, the second 001, and so on with the last being assigned the code 111.

(a) Use the function **x quantize** to quantize $r[n]$ to eight levels. Set the minimum and maximum quantizer levels (X_{min} and X_{max}) to -1 and $+1$, respectively. The quantized signal will be called $\hat{r}[n]$. Display $r[n]$ and $\hat{r}[n]$ one above the other using **x look2**. Now use **x slook2** to display these plots superimposed on each other and provide a sketch.

Examine the quantizer relationship shown in Fig. 4.7. Has $r[n]$ been quantized by a midriser or by a midtread quantizer? What is the step size, Δ?

(b) Examine the distortion that was introduced in part (a) as a result of quantization and sketch the result. Do this by explicitly computing the error signal,

$$e[n] = \hat{r}[n] - r[n]$$

using **x subtract** and displaying $e[n]$ using **x look**. What is the maximum value of $e[n]$?

(c) The quantization error in our example can be reduced by setting $X_{\text{min}} = -.875$ and $X_{\text{max}} = .875$. Requantize $r[n]$ as before using these limits; display and sketch the results. Compute and display the error signal, $e[n]$, and determine the maximum error.

This illustrates that the best performance is *not* achieved when the minimum and maximum amplitudes of the signal and quantizer are the same. If the minimum and maximum amplitudes of a signal $x[n]$ are $-|A|$ and $+|A|$, respectively, and L is the number of desired quantization levels, what should X_{min} and X_{max} be to achieve the lowest quantization error? Express your answer in terms of L.

(d) Quantize $r[n]$ to nine levels instead of eight. Determine and record the values of X_{min} and X_{max} that will yield the minimum quantization error. What is the value of the maximum quantization error? Display and sketch $r[n]$ and $\hat{r}[n]$ as before.

Next create a 21-point sequence $x[n] = 5 \sin \pi/10$ in the range $0 \leq n < 21$ using **x siggen**. Quantize $x[n]$ to eight levels using **x quantize**. Again select X_{min} and X_{max} to minimize quantization error. Use **x view2** to simultaneously display $x[n]$ and the quantized signal; sketch these plots. What is the value of Δ?

Real A/D converters must quantize the sampled analog input so that it can be represented on a computer. In practice, 12- to 16-bit quantization provides a more than adequate signal representation for most applications. As shown in the previous exercise, the effects of quantizing with as few as 3 bits (or 8 levels) still preserves most of the waveform's time structure.

EXERCISE 4.2.2. Quantization Effects

The number of quantization levels in an A/D converter varies somewhat depending on the application. This exercise will investigate the effects of varying the number of quantization levels.

A system composed of cascading A/D and D/A converters can be simulated by the following macro, which will be called **sys.bat**:

```
echo off
x sample  %1  dummy  1
x quantize  dummy  dummy1  %3  %4  %5
x sti  dummy1  dummy2  1
x afilter  dummy2  hh  dummy  150
x extract  dummy  %2  100  10  0
del dummy
del dummy1
del dummy2
echo on
```

Using your text editor, copy this macro onto your computer. The macro has five command-line arguments: the input, the output, the minimum quantizer limit, the maximum quantizer limit, and the number of quantizer levels. The macro receives, as its input, a simulated analog signal file, $x_a(t)$, and returns another simulated analog signal file, $y_a(t)$. The macro is executed by typing:

$$\underbrace{\text{sys}}\quad \overbrace{\text{(inputfile)}}^{x_a(t)}\quad \overbrace{\text{(outputfile)}}^{y_a(t)}\quad \text{min}\quad \text{max}\quad \text{levels}$$

Since no processing is performed, the input and output should be the same except for the distortion introduced by aliasing, quantization, and the non-ideal phase response of the filter. The purpose of the **x extract** statement is to remove a time shift that is introduced by the filter. The system assumes that time is measured in milliseconds and that the sampling period is 1 msec. The output signal length is assumed to be 150.

In this exercise, you will work with a simulated linearly increasing analog triangular wave, $x_a(t)$. To create this signal, use **x siggen** to generate five periods of a triangle wave with a period of 20 samples, amplitude 0.01, and starting point $n = 0$. Next multiply this signal by a 100-point ramp function that can be created using **x siggen**. This produces a triangular wave, $x_a(t)$, with a linearly increasing envelope.

(a) Using **x alook**, display and sketch $x_a(t)$ and verify that it is correct. What are the appropriate values of min and max for the **sys.bat** macro given the amplitude range of $x_a(t)$?

(b) In this part, the number of levels will be varied in each experiment. Using `x alook`, display and sketch the first 100 points of the `sys` output when the number of levels is 1024. How many bits are required for this representation? Sketch the output when 16 levels, 4 levels, and 2 levels are used. Notice that the greatest distortion is due to aliasing and filtering errors from sampling. However, distortions due to quantization are also evident. For this particular signal, what seems to be the smallest number of bits needed to yield acceptable performance?

— *EXERCISE 4.2.3.* **Pulse Code Modulation (PCM)**

PCM is the simplest method for discrete-time signal representation. It quantizes each of the sampled values to a finite number of levels and then represents each of these levels with a unique binary codeword. PCM serves as the basis for digitally encoding waveforms for transmission or storage. If the waveform is sampled $1/T$ times per second and if the samples can be adequately represented using a B-bit quantizer, then the waveform can be transmitted digitally over a communication channel that has a capacity of B/T bits/sec.

The quantization inherent in PCM degrades the quality of the signal and thus lowers its SNR. The impact of different numbers of quantization levels on the SNR is investigated in this exercise.

(a) Let $x[n]$ be the signal contained in file `sig1`. Examine the signal using `x look` to determine maximum and minimum values that will avoid any saturation, then use `x quantize` to quantize $x[n]$ to 4 bits (16 levels) to form $\hat{x}[n]$. Compute the SNR using the `x snr` function and record the result. Note that $x[n]$ and $\hat{x}[n]$ are the arguments of the function. Next quantize $x[n]$ to 6, 8, and 10 bits and record the SNRs. Determine a functional (equation) relationship between the SNR and the number of bits used for the quantization.

(b) Using your result from (a), what value would you predict for the SNR if a 16-bit quantizer were used?

— *EXERCISE 4.2.4.* **Differential PCM (DPCM)**

There are a number of alternatives to PCM (which was introduced in the previous exercise) that may permit less quantization error at the same bit rate, or, alternatively may allow the bit rate to be reduced without degrading the SNR. One of these alternatives is illustrated in this exercise. It has some of the flavor of a waveform coding system known as differential PCM (DPCM). The simple differential system considered here consists of three subsystems as shown in Fig. 4.8: an FIR filter (on the left); a uniform quantizer, shown in the center; and an inverse filter that compensates for the response of $h[n]$, shown on the right. The linear and time-invariant FIR filter is defined by the difference equation

$$y[n] = x[n] - ax[n-1].$$

Quantization is performed on $y[n]$ resulting in $\hat{y}[n]$. This signal may be thought of as the compressed signal or encoded signal. The goal is to represent this signal with as few bits as possible.

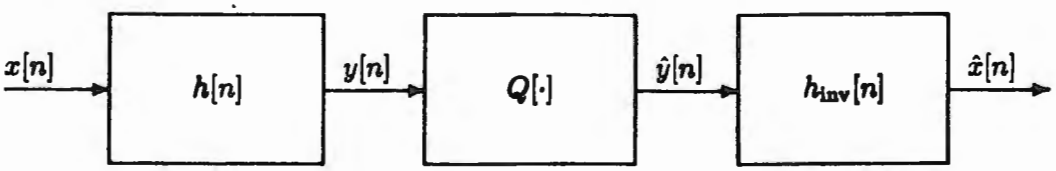

Figure 4.8. A simple differential PCM system.

The purpose of the third subsystem in the DPCM coder expands the coded signal $\hat{y}[n]$ into a signal $\hat{x}[n]$ that approximates the original input $x[n]$ as closely as possible. This system is the first-order IIR filter that is the inverse of the FIR filter. It is defined by the difference equation

$$\hat{x}[n] = a\hat{x}[n-1] + \hat{y}[n].$$

As you will see, the overall system can encode $x[n]$ more accurately than a PCM system.

Note that since both digital filters can be implemented as difference equations, the `x lccde` function can be used to evaluate both $y[n]$ and $\hat{x}[n]$. Let the length for the `x lccde` output file be 128 samples.

(a) Calculate the sequence $y[n]$ when $x[n]$ is the signal in **sig2** for $a = .8$. Quantize it to 4-bits using `x quantize`. You will need to examine the signal $y[n]$ with `x look` to set the minimum and maximum levels properly to avoid saturation. Reconstruct the signal $\hat{x}[n]$ and evaluate the signal-to-noise ratio (SNR) of the overall system using `x snr`. For comparison determine the SNR of a 16-level PCM (see previous exercise) encoder applied to $x[n]$.

(b) Determine how many quantization levels would be required in a PCM encoder that has the same SNR as the DPCM coder designed in part (a).

(c) Write a macro to implement your coder. Let the parameter a be variable and measure the SNR for different values of a using the signal **sig2**. For what a is the SNR maximized?

The structure of actual DPCM systems is somewhat different than that described here, and the filters ($h[n]$) are often optimized to the statistics of the signals being encoded.

4.3 SAMPLING-RATE CONVERSION

The higher the sampling rate the more samples must be processed. Consequently, for processing efficiency the sampling rate should be reduced whenever possible. Such a situation arises when preliminary digital processing, involving operations such as low-pass filtering, reduces a signal's bandwidth, producing an output signal that is over-sampled. One way to reduce the sampling rate is to convert the digital signal back

into analog form using a D/A converter and then resample it, but this is expensive. It requires additional components and the A/D and D/A converters are often the least accurate components in a complete realization. A simpler approach is to do the sampling-rate conversion in the discrete-time domain.

The sampling rates of a discrete-time signal are changed using operators called *decimators* and *interpolators*. A system for decimating the sampling rate by a factor of N is shown in Fig. 4.9. A decimator consists of a cascade of a lowpass filter, $H(e^{j\omega})$, and a downsampler (or subsampler), denoted by $\downarrow N$ in Fig. 4.9. The downsampler operates on successive blocks of N samples by retaining the first sample in each block and discarding the rest, i.e.,

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

This reduces the number of samples in the sequence by a factor of N. The usage of the term *decimator* is not consistent in the signal processing literature. Very often it is used to refer to the downsampling operation. However, we will consistently use *decimation* to refer to the combination of lowpass filtering and downsampling as shown in Fig. 4.9.

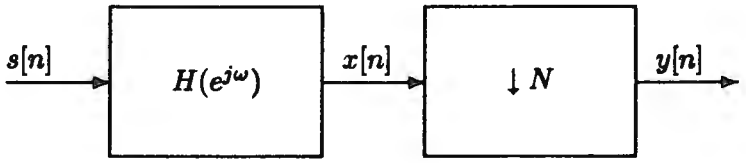

Figure 4.9. A decimator for reducing the sampling rate by a factor of N.

In the frequency domain, the input and the output of the downsampler are related by

$$Y(e^{j\omega}) = \frac{1}{N}\sum_{r=0}^{N-1} X(e^{j(\omega/N+(2\pi r)/N)}). \tag{4.10}$$

Downsampling introduces aliasing into the DTFT of $x[n]$. It is therefore important to bandlimit the input of the downsampler to the frequency range $|\omega| < \pi/N$. This is the role of the lowpass filter $H(e^{j\omega})$ that precedes the downsampler. It has unity gain and a cutoff frequency of π/N.

The dual operation is called *interpolation*. It is the process by which a signal sampled at one sampling rate is converted to one with a rate that is L times higher. Figure 4.10 shows an implementation of a rate L interpolator. It is also a two-stage operation. The first subsystem, which is called an *upsampler*, inserts $L-1$ zeros between each sample in the input. The second subsystem is a lowpass filter with a gain of L and a cutoff frequency of π/L. As in the case of decimation, the term *interpolation* is sometimes used in the literature to refer to just the upsampling operation, but we will adhere to the definition of Fig. 4.10 in this text.

The frequency-domain relationships that describe interpolation are

$$R(e^{j\omega}) = X(e^{jN\omega})$$
$$Y(e^{j\omega}) = R(e^{j\omega})H(e^{j\omega}) = X(e^{jN\omega})H(e^{j\omega}).$$

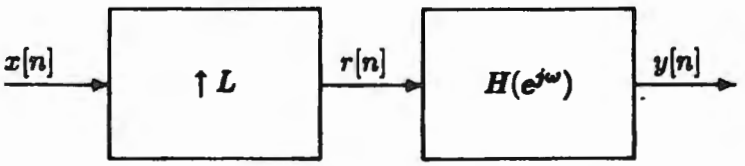

Figure 4.10. A system for interpolating by a factor of L.

—*EXERCISE 4.3.1.* **Downsampling**

The downsampler is the basic component for sampling rate reduction. To examine its operation more closely, consider the input signal $x[n]$ where

$$x[n] = \cos\left(\frac{\pi}{32}n\right).$$

Let this signal be the input to a downsampler with a decimation factor of $M = 3$ and let the output be $y[n]$.

(a) Based on the definition of decimation, determine an explicit expression for $y[n]$.

(b) The DTFT of a cosine is a pair of impulses in the frequency domain. Using equation (4.10) sketch $|Y(e^{j\omega})|$. This does not involve the use of a computer.

(c) Now consider a time-limited input sequence $x_T[n]$ consisting of 512 samples of $x[n]$ taken over the range $0 \leq n < 512$. Create $x_T[n]$ using **x siggen**. Using the **x dnsample** function compute the output $y_T[n]$ of the decimator. Use the **x truncate** function to limit both $x_T[n]$ and $y_T[n]$ to 128 samples so that you are comparing signals of the same length. Use **x dtft** to display and sketch the discrete-time Fourier transform magnitudes of these signals and compare them with the results found in part (b). Note that for finite length segments of sinusoids the spectrum is not truly impulsive, but is approximately so.

(d) Is the downsampling operation linear? Justify your answer analytically.

(e) Is the downsampling operation shift invariant? Justify your answer analytically.

—*EXERCISE 4.3.2.* **Aliasing in Downsampling**

This problem examines the effects of downsampling in the frequency domain. Use **x siggen** to create 2450 samples of $x[n]$ beginning at $n = 0$ where

$$x[n] = \cos\frac{\pi n}{11}.$$

(a) Downsample $x[n]$ by the following factors and truncate the results to 100 samples using **x dnsample** and **x truncate**.

(i) $N = 4$
(ii) $N = 18$
(iii) $N = 20$
(iv) $N = 24$.

For each case display and sketch the transform magnitude using **x dtft**. In which downsampled signals is aliasing present? Assume that $x[n]$ was obtained by taking samples of the output of an ideal A/D with a sampling rate $f_s = 100$ Hz. What sampling rates could have been used with the analog signal in order to produce each of the downsampled signals?

— ***EXERCISE 4.3.3.*** **Upsampling**

An upsampler inserts zeros between the sample values of a sequence. To explore the properties of this operation, use **x siggen** to generate 66 points of the sequence $x[n]$ in the range $0 \leq n \leq 65$ where

$$x[n] = \cos \frac{\pi}{5} n.$$

(a) Using **x upsample**, examine the sequences obtained by
   (i) upsampling $x[n]$ by a factor of 2,
   (ii) upsampling $x[n]$ by a factor of 10.

   Sketch these sequences in the time domain.

(b) Truncate each of the sequences that you generated in (a) to 128 points using **x truncate**. Display and sketch these sequences in the time domain and the frequency domain using **x view** and **x dtft**. This phenomenon that you observe whereby spectral copies are replicated is called *imaging*.

(c) Is the upsampling operation linear? Explain.

(d) Is it shift invariant? Explain.

EXERCISE 4.3.4. **Commutativity of Downsampling and Upsampling**

Systems are said to be *commutative* if the result of applying multiple operations to a signal is independent of the order in which they are applied. In general the downsampling and upsampling operators are not commutative. Downsampling by a factor of N followed by upsampling by a factor of M is not equivalent to upsampling by M followed by downsampling by N. For certain values of M and N, however, downsampling and upsampling are commutative.

(a) Determine a pair of factors, M and N, such that downsampling and upsampling are commutative or order independent. Exclude the trivial case for which $N = M = 1$.

(b) What condition must N and M satisfy in order for the upsampling and downsampling operations to be commutative? Use the computer to test examples that support your conclusion. └ N and M must be relatively prime.

— *EXERCISE 4.3.5.* **Decimation and Aliasing**

You have seen that aliasing can occur as a result of downsampling. If a signal is not bandlimited to the range $-\pi/N < \omega < \pi/N$, then downsampling by N will cause undesired aliasing. To illustrate this point consider the signal $gg[n]$ that is stored in the file `gg`.

(a) Display and sketch the DTFT of $gg[n]$. Now downsample the signal by a factor of 3 using `x dnsample` and sketch the DTFT of the result using `x dtft`. Verify this result graphically by adding together the appropriate spectral replicas of $GG(e^{j\omega})$ and scaling the result.

(b) Now decimate the signal by a factor of 3 by first filtering it with a 201-point FIR lowpass filter with a cutoff frequency $\pi/3$ that you design using the `x fdesign` function with the Hamming window option. Convolve the impulse response of the filter with the signal $gg[n]$ using `x convolve` and downsample the result using `dnsample`. Display and sketch the DTFT magnitude using `dtft`.

— *EXERCISE 4.3.6.* **Interpolation**

Interpolation is achieved by upsampling and filtering with a lowpass filter. To illustrate this, use `x siggen` to generate the 64-point chirp signal with *amplitude* = 1, $\alpha_1 = 0.4$, $\alpha_2 = 0.01$, *phi* = 0, and starting point $n = 0$.

(a) Display and sketch the DTFT magnitude of the chirp signal.

(b) Upsample and interpolate the chirp signal by a factor of 3. The functions `x upsample`, `x convolve`, `x gain`, `x extract`, and `x fdesign` may be used for implementation and design of the system. Try using a 64-point lowpass filter designed with the *Hamming window* option and cutoff frequency $\pi/3$. Look at the signal in the frequency domain and describe its relationship to the original.

— *EXERCISE 4.3.7.* **Fractional Sampling Rate Conversion**

Sometimes sampling-rate conversions are needed between sampling rates that are not integer multiples of one another. One solution to this problem is to first interpolate by a factor of M and then decimate by a factor of N. This permits a fractional sampling-rate change by a factor of M/N to be made. A system for doing this is shown in Fig. 4.11. The lowpass filters from the interpolator and the decimator can be replaced by a single filter with a gain of M and a cutoff frequency of $\pi/\max(M, N)$, i.e., the cutoff is the smaller of π/M or π/N.

(a) Create a macro for implementing a sample-rate conversion by a general factor of M/N. The function `x fdesign` with the Hamming window option can be used to design the lowpass filter. Use a filter length of 31. In addition use `x lshift` to shift the impulse response by 15 samples so that it will have zero phase.

(b) Verify the operation of your system by generating the signal

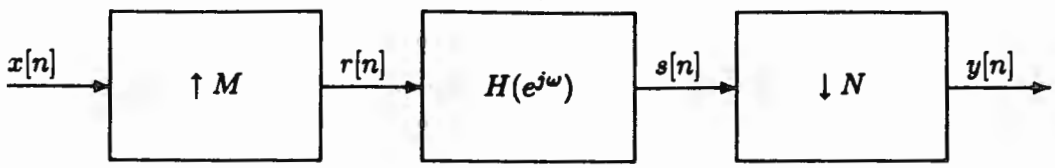

Figure 4.11. A cascade of an interpolator and a decimator to achieve a sample-rate conversion of M/N.

$$x[n] = \cos\frac{\pi}{30}n + \cos\frac{\pi}{5}n$$

over the range $0 \leq n < 128$ using **x siggen**. Use your macro to change the sampling rate by a factor of 3/4, and denote the resulting sequence by $y[n]$. Use **x look** to display $x[n]$ and $y[n]$ as continuous waveforms. Sketch these plots. Also examine $x[n]$ and $y[n]$ using **slook2**, which will superimpose the waveforms on the same plot. Observe the change in frequency due to the rate change.

(c) Repeat part (b) for the case where $M = 6$ and $N = 8$. Use a 31-point lowpass filter designed as before. Note that the ratio is the same as in part (b), but the filter, upsampler, and downsampler are different. Use **x look2** to display the outputs from parts (b) and (c) one above the other and sketch these plots. Redesign the 31-point filter and perform the same rate change using $M = 24$, $N = 32$. Sketch these results as before. Explain why the results are different.

(d) Apply your macro to the signal, $y[n]$, that you produced in (b) with a rate change factor of 4/3. Compute and sketch the DTFT magnitude of the new signal and compare it to that of the original. Was the inverse rate change successful in restoring the signal to its original sampling rate?

— ✳ *EXERCISE 4.3.8.* **Fractional Sample Delay**

The system shown in Fig. 4.12 implements a delay of $1/N$, which is less than one sample.

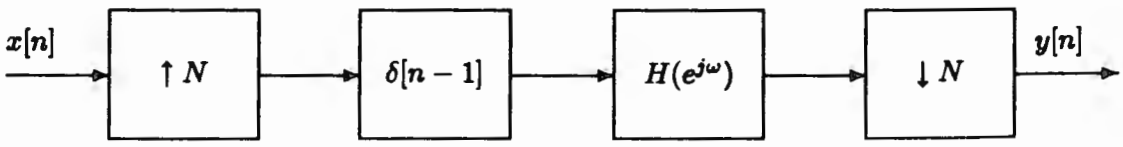

Figure 4.12. A system for implementing a delay of $1/N$ samples.

(a) Create a macro for implementing a fractional sample delay of 1/2 sample. Use the function **fdesign** with the *Hamming window* option to design the 31-point lowpass filter.

(b) Apply your macro to 64 samples of the sinusoid

$$x[n] = \cos \frac{\pi}{16} n$$

in the range $0 \leq n < 64$. Sketch the resulting time signal.

(c) Generate a unit sample, $\delta[n]$, using `x siggen` and delay it by 1/2 sample. Sketch and explain your result.

(d) Repeat part (c) for a 64-point pulse using the *block* option in `x siggen`.