PART II

Fundamental Concepts and Techniques

Chapter 3

SAMPLING AND QUANTIZING

Our main purpose in this book is to study the basic techniques required to accurately simulate communication systems using digital computers. In most communications applications, waveforms are generated and processed through the system under study. The computer, of course, can only process numbers representing samples of the waveforms of interest. In addition, since the computer has finite word length, the sample values have finite precision. In other words, the sample values are quantized. Thus, sampling and quantizing are underlying operations in all digital simulations, and each of these operations give rise to errors in the simulation results. The complete elimination of these error sources is not possible and tradeoffs are often required. We will see that the best we can do is to minimize the effects of sampling and quantizing on simulation accuracy. It is worth noting that many physical systems make use of digital signal-processing (DSP) techniques and also suffer from the effects of sampling and quantizing errors.

3.1 Sampling

As illustrated in Figure 3.1, a digital signal is formed from an analog signal by the operations of sampling, quantizing, and encoding. The analog signal, denoted $x(t)$, is continuous in both time and amplitude. The result of the sampling operation is a signal that is still continuous in amplitude but discrete in time. Such signals are often referred to as sampled-data signals. A digital signal is formed from a sampled data-signal by encoding the time-sampled values onto a finite set of values. As we will see, errors are usually induced at each step of this process.

3.1.1 The Lowpass Sampling Theorem

The first step in forming a digital signal from a continuous-time signal, $x(t)$, is to sample $x(t)$ at a uniformly spaced series of points in time to produce the sample values, $x_s(t) = x(kT_s) = x[k]$.[1] The parameter T_s is known as the sampling period and is the inverse of the sampling frequency, f_s.

A model for the sampling operation is illustrated in Figure 3.2. The signal $x(t)$ is multiplied by a periodic pulse $p(t)$ to form the sampled signal $x_s(t)$. In other words

$$x_s(t) = x(t)\,p(t) \tag{3.1}$$

The signal $p(t)$ is referred to as the sampling function. The sampling function is assumed to be a narrow pulse, which is either zero or one. Thus $x_s(t) = x(t)$ when $p(t) = 1$, and $x_s(t) = 0$ when $p(t) = 0$. We will see shortly that only the period of the sampling function $p(t)$ is significant and the waveshape of $p(t)$ is arbitrary. The pulse type function illustrated in Figure 3.2 simply provides us with the intuitively pleasing notion of a switch periodically closing at the sampling instants.

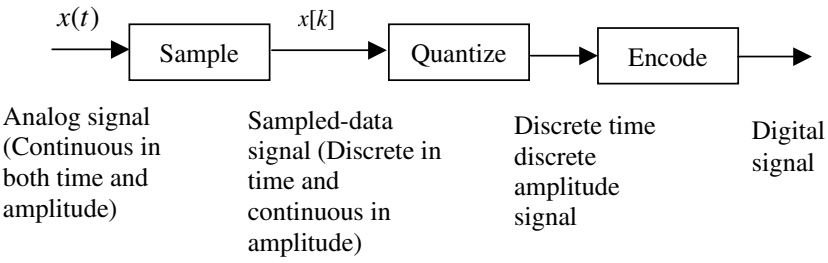
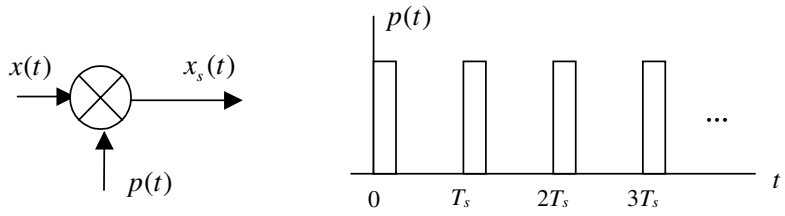


Figure 3.1 Sampling, quantizing, and encoding.

[1] Once a signal is sampled, the sample values are a function of the index k and the notation $x[k]$ is used. This notation, made popular by Oppenheim and Schafer [1], is commonly used in the DSP literature. Since the square brackets implies a sampling operation the subscript s is not needed to denote sampling. The value of $x[\cdot]$ is defined only for integer arguments.



(a) Sampling operation (b) Sampling function

Figure 3.2 Sampling operation and sampling function.

Since $p\,(t)$ is a periodic signal, it can be represented by the Fourier series

$$p\,(t) = \sum_{n=-\infty}^{\infty} C_n \exp(j2\pi n f_s t) \tag{3.2}$$

in which the Fourier coefficients are given by

$$C_n = \frac{1}{T_s} \int_{-T_s/2}^{T_s/2} p\,(t) \exp(-j2\pi n f_s t)\, dt \tag{3.3}$$

Substituting (3.2) into (3.1) gives

$$x_s\,(t) = x(t) \sum_{n=-\infty}^{\infty} C_n \exp(j2\pi n f_s t) \tag{3.4}$$

for the sampled signal.

In order to derive the sampling theorem and thereby show that under appropriate conditions $x\,(t)$ is completely represented by the samples $x\,(kT_s)$, we must derive the spectrum of $x_s\,(t)$ and show that $x(t)$ can indeed be reconstructed from $x_s(t)$. The Fourier transform of the sampled signal is

$$X_s\,(f) = \int_{-\infty}^{\infty} x(t) \sum_{n=-\infty}^{\infty} C_n \exp(j2\pi n f_s t) \exp(-j2\pi f t)\, dt \tag{3.5}$$

which, upon interchanging integration and summation, becomes

$$X_s\,(f) = \sum_{n=-\infty}^{\infty} C_n \int_{-\infty}^{\infty} x\,(t) \exp\left[-j2\pi(f - n f_s)t\right]\, dt \tag{3.6}$$

Since the Fourier transform of the continuous-time signal $x(t)$ is

$$X\,(f) = \int_{-\infty}^{\infty} x\,(t) \exp(-j2\pi f t)\, dt \tag{3.7}$$

it follows from (3.6) that the Fourier transform of the sampled signal can be written

$$X_s(f) = \sum_{n=-\infty}^{\infty} C_n X(f - nf_s) \tag{3.8}$$

We therefore see that the effect of sampling a continuous-time signal is to reproduce the spectrum of the signal being sampled about dc ($f = 0$) and all harmonics of the sampling frequency ($f = nf_s$). The translated spectra are weighted by the corresponding Fourier coefficient in the series expansion of the sampling pulse $p(t)$.

The next, and final, step in the development of the sampling theorem is to define $p(t)$. Since the samples are assumed to be taken instantaneously, a suitable definition of $p(t)$ is

$$p(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s) \tag{3.9}$$

This is known as impulse function sampling in which the sample values are represented by the weights of the impulse functions. Substitution of (3.9) into (3.3) gives

$$C_n = \frac{1}{T_s} \int_{-T_s/2}^{T_s/2} \delta(t) \exp(-j2\pi n f_s t)\, dt \tag{3.10}$$

Applying the sifting property of the delta function gives

$$C_n = \frac{1}{T_s} = f_s \tag{3.11}$$

Using this result in (3.2) shows that the Fourier transform of $p(t)$ can be represented by

$$P(f) = f_s \sum_{n=-\infty}^{\infty} \delta(f - nf_s) \tag{3.12}$$

For impulse function sampling $C_n = f_s$ for all n. Thus, using (3.8) the spectrum of the sampled signal becomes

$$X_s(f) = f_s \sum_{n=-\infty}^{\infty} X(f - nf_s) \tag{3.13}$$

Note that this result could have also been obtained from the expression

$$X_s(f) = X(f) \circledast P(f) \tag{3.14}$$

where $\circledast$ denotes convolution. The generation of $X_s(f)$ using (3.14) is illustrated in Figure 3.3 for the case of a bandlimited signal.

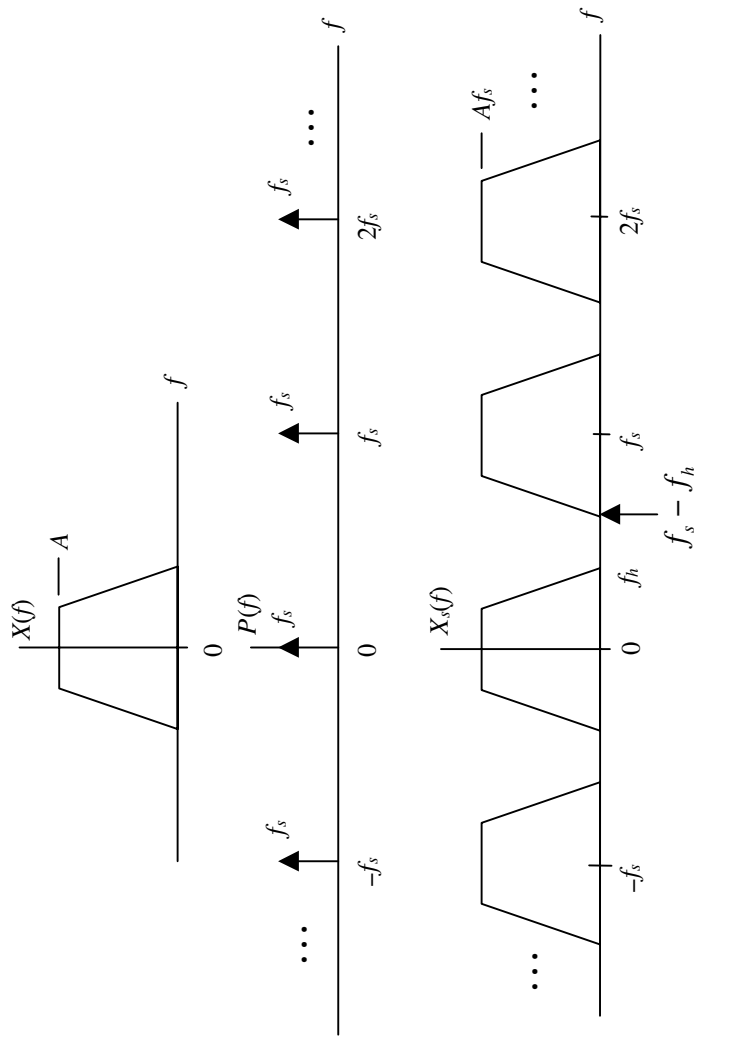


Figure 3.3 Sampling viewed in the frequency domain.

The sampling theorem can be developed from observation of Figure 3.3. In order for the samples $x(nT_s)$ to contain all of the information in the continuous-time signal $x(t)$, so that no information is lost in the sampling process, the sampling must be performed so that $x(t)$ can be reconstructed without error from the samples $x(nT_s)$. We will see that reconstruction of $x(t)$ from $x_s(t)$ is accomplished by extracting the $n = 0$ term from $X_s(f)$ by lowpass filtering. Accomplishing reconstruction without error therefore requires that the portion of the spectrum of $X_s(f)$ about $f = \pm f_s$ [the $n = \pm 1$ terms in (3.13)] not overlap the portion of the spectrum about $f = 0$ [the $n = 0$ term in (3.13)]. In other words, all translated spectra in (3.13) must be disjoint. This requires that $f_s - f_h > f_h$ or $f_s > 2f_h$, which proves the sampling theorem for lowpass signals.

Theorem 1 *A bandlimited signal may be reconstructed without error from samples of the signal if the sampling frequency f_s exceeds $2f_h$, where f_h is the highest frequency present in the signal being sampled.*

While this theorem is usually referred to as the lowpass sampling theorem, it also works for bandpass signals. However, applying the lowpass sampling theorem to bandpass signals usually results in excessively high sampling frequencies. Sampling bandpass signals is the topic of a later section.

If $f_s < 2f_h$ the spectra centered on $f = \pm f_s$ overlap the spectrum centered on $f = 0$ and the output of the reconstruction filter, as illustrated in Figure 3.4, will be a distorted version of $x(t)$. This distortion is referred to as aliasing. The effect of aliasing is also illustrated in Figure 3.4, assuming that the spectrum of $x(t)$ is real.

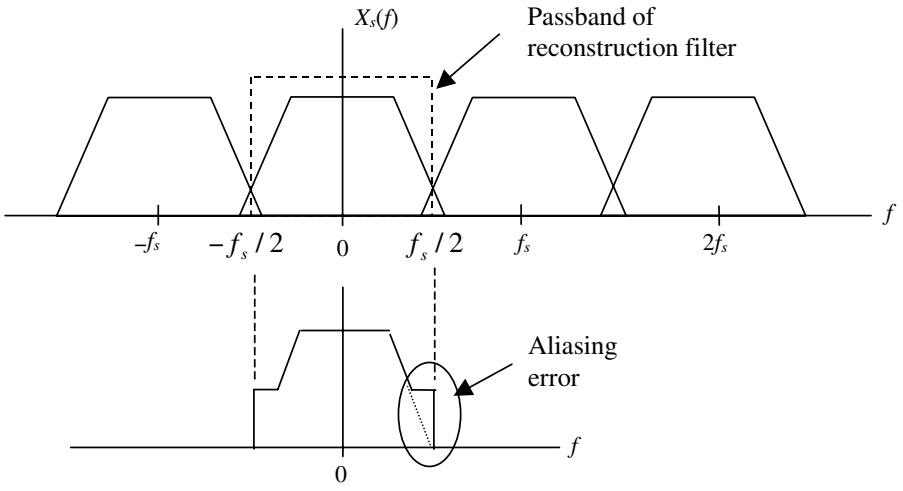


Figure 3.4 Illustration of undersampling leading to aliasing error.

3.1.2 Sampling Lowpass Random Signals

The waveform $x(t)$ in the preceding discussion was assumed to be a deterministic finite-energy signal. As a result of these assumptions, the Fourier transform exists and the sampling theorem could be based on the spectrum (the Fourier transform) of the signal. In most of our applications throughout this book it will be more natural to assume that the simulation processes sample functions of a random process. Therefore, instead of selecting a sampling frequency based on the Fourier transform of the signal to be sampled, the selection of an appropriate sampling frequency must be based on the power spectral density (PSD) of the sampling frequency.

For the case of random signals we write

$$X_s(t) = X(t)P(t) \tag{3.15}$$

where the sampling function $P(t)$ is written

$$P(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s - D) \tag{3.16}$$

in which D is a random variable independent of $X(t)$ and uniformly distributed in $(0, T_s)$. Note the similarity of (3.15) and (3.1) and the similarity of (3.16) and (3.9). There are only two essential differences. First, uppercase letters are used in the time functions $X(t)$, $P(t)$, and $X_s(t)$ to remind us that they represent random processes. The other difference is the use of the random variable D in (3.16). The effect of D is to ensure that $X_s(t)$ is a *stationary* random process. Without the inclusion of D the sampled signal is *cyclostationary*. The effect of D is to make the time origin of $P(t)$ random but fixed.

The power spectral density of $X_s(t)$ is found by first determining the autocorrelation function of

$$X_s(t) = X(t) \sum_{k=-\infty}^{\infty} \delta(t - kT_s - D) \tag{3.17}$$

The Fourier transform of the resulting autocorrelation function gives the PSD of $X_s(t)$, which is [2]

$$S_{X_s}(f) = f_s^2 \sum_{n=-\infty}^{\infty} S_X\left(f - nf_s\right) \tag{3.18}$$

where $S_X(f)$ denotes the PSD of $X(t)$. Note the similarity of (3.18) and (3.13). Also note that Figures 3.3 and 3.4 apply if the spectra are PSDs corresponding to $X(t)$ and if the axes are labeled accordingly. Note that the sampling theorem as previously derived still holds, and therefore the signal must be sampled at a frequency exceeding twice the sampling frequency if aliasing is to be avoided.

3.1.3 Bandpass Sampling

We now consider the problem of sampling bandpass signals. There are a number of strategies that can be used for representing bandpass signals by a set of samples. In the following sections we consider the two most common methods.

The Bandpass Sampling Theorem

The bandpass sampling theorem for real bandpass signals is stated as follows [2]:

Theorem 2 *If a bandpass signal has bandwidth B and highest frequency f_h, the signal can be sampled and reconstructed using a sampling frequency of $f_s = 2f_h/m$, where m is the largest integer not exceeding f_h/B. All higher sampling frequencies are not necessarily usable unless they exceed $2f_h$, which is the value of f_s dictated by the lowpass sampling theorem.*

A plot of the normalized sampling frequency f_s/B as a function of the normalized center frequency f_0/B is illustrated in Figure 3.5, where f_0 and f_h are related by $f_h = f_0 + B/2$. We see that the allowable sampling frequency always lies in the range $2B \leq f_s \leq 4B$. However, for $f_0 \gg B$, which is typically the case, the sampling frequency dictated by the bandpass sampling theorem is approximately equal to, but is lower bounded by, $2B$.

Sampling Direct/Quadrature Signals

Suppose we have a bandpass signal expressed in the form

$$x(t) = A(t)\cos\left[2\pi f_c t + \phi(t)\right] \tag{3.19}$$

The function $A(t)$ is referred to as the envelope of the bandpass signal and the function $\phi(t)$ is referred to as the phase deviation of the bandpass signal. In most communications applications both $A(t)$ and $\phi(t)$ are lowpass signal and have bandwidths roughly on the order of the bandwidth of the information-bearing signal. Using standard trigonometric identities, the bandpass signal can be written

$$x(t) = A(t)\cos\phi(t)\cos 2\pi f_c t - A(t)\sin\phi(t)\sin 2\pi f_c t \tag{3.20}$$

or

$$x(t) = x_d(t)\cos 2\pi f_c t - x_q(t)\sin 2\pi f_c t \tag{3.21}$$

In this representation

$$x_d(t) = A(t)\cos\phi(t) \tag{3.22}$$

is called the direct (or in-phase) component and

$$x_q(t) = A(t)\sin\phi(t) \tag{3.23}$$

is the quadrature component. Since $A(t)$ and $\phi(t)$ are lowpass signals, it follows that $x_d(t)$ and $x_q(t)$ are lowpass signals and therefore must be sampled in accordance with the lowpass sampling theorem. Note from (3.21) that if $x_d(t)$, $x_q(t)$ and the carrier frequency f_c are known, the bandpass signal can be reconstructed without error. The representation of bandpass signals using direct and quadature components will be covered in detail in Chapter 4.

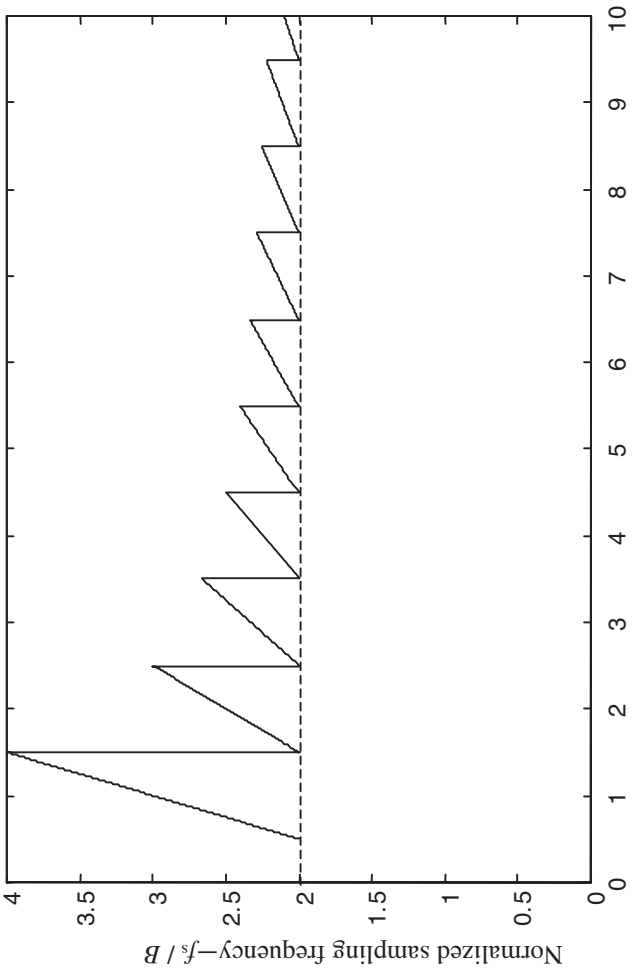


Figure 3.5 The required sampling frequency for bandpass sampling.

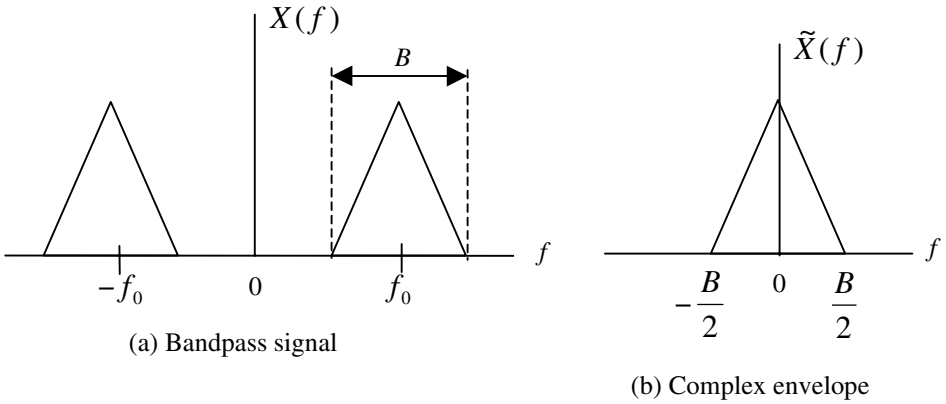


Figure 3.6 Bandpass signal and the corresponding complex envelope.

The frequency-domain representation of a bandpass signal is given in Figure 3.6(a). The complex envelope corresponding to this signal is defined by

$$\widetilde{x}(t) = x_d(t) + jx_q(t) \tag{3.24}$$

Since both $x_d(t)$ and $x_q(t)$ are lowpass signals:

$$\widetilde{X}(f) = X_d(f) + jX_q(f) \tag{3.25}$$

is lowpass as illustrated in Figure 3.6(b). In Figure 3.6 we see that $\widetilde{X}(f)$, and consequently $x_d(t)$ and $x_q(t)$ are lowpass signals. Thus $x_d(t)$ and $x_q(t)$ must be sampled according to the lowpass sampling theorem. Since the highest frequency present in $x_d(t)$ and $x_q(t)$ is $B/2$, the minimum sampling frequency for each is B. However, two lowpass signals [$x_d(t)$ and $x_q(t)$] must be sampled rather than one. As a result, a sampling rate exceeding $2B$ must be used. We therefore see that sampling the complex envelope using the lowpass sampling theorem yields the same required sampling frequency as sampling the real bandpass signal using the bandpass sampling theorem for the typical case in which $f_0 \gg B$.

Example 3.1. It follows from the preceding discussion that the bandpass signal $x(t)$ can be reconstructed without error if $x_d(t)$ and $x_q(t)$ are sampled appropriately in accordance with the lowpass sampling theorem. The advantage of representing bandpass signals by lowpass signals is obvious. Consider, for example, that we are to represent 1 second of an FM signal by a set of samples. Assume that the carrier frequency is 100 MHz (typical for the FM broadcast band) and that the highest frequency present in the modulation or information-bearing signal is 15 kHz. The bandwidth B of the modulated signal is usually approximated by Carson's rule [2], which is

$$B = 2(D+1)W \tag{3.26}$$

Assuming a deviation ratio D of 5 gives

$$B = 2(5+1)15 \quad \text{kHz} \tag{3.27}$$

which is 180 kHz (90 kHz each side of the carrier). The highest frequency present in the modulated signal is therefore 100,090 kHz. Thus, 1 second of signal requires a minimum of 200,180,000 samples according to the lowpass sampling theorem.

Now suppose we elect to represent the FM signal in direct/quadrature form. The bandwidth of both $x_d(t)$ and $x_q(t)$ is $B/2$, or 90 kHz. Thus $x_d(t)$ and $x_q(t)$ will each require at least 180,000 samples to represent 1 second of signal. This gives a total of 360,000 lowpass samples to represent 1 second of data. The savings is

$$\frac{200,180,000}{360,000} \approx 556 \tag{3.28}$$

and translates directly into a corresponding reduction in computer run time. ■

3.2 Quantizing

The quantizing process and a simple fixed-point encoding process is illustrated in Figure 3.7, which shows a continuous-time waveform and a number of samples of that waveform. The sample values are represented by the heavy dots. Each sample falls into a quantizing level. Assuming that there are n quantizing levels and that each quantizing level is represented by a b-bit binary word, it follows that

$$n = 2^b \tag{3.29}$$

In Figure 3.7 each quantizing level is mapped to three-bit ($b = 3$ and $n = 8$) digital word. After quantizing, sample values are represented by the digital word

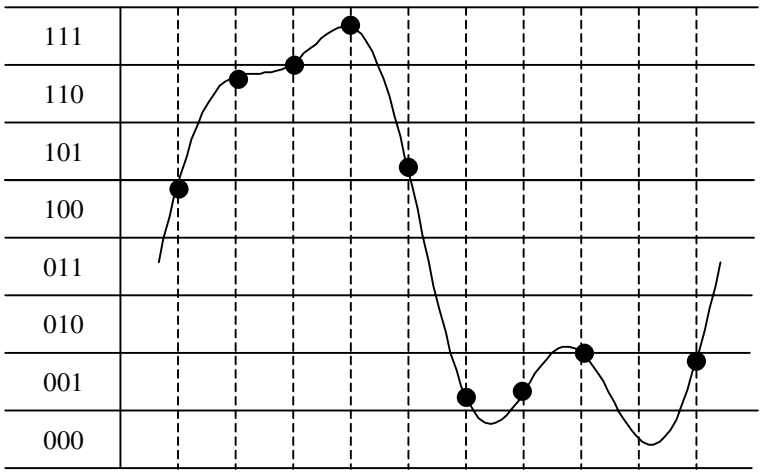


Figure 3.7 Quantizing and encoding.

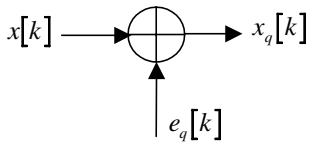


Figure 3.8 Model for quantizing error.

corresponding to the quantizing level into which the sample value falls, and digital processing of the waveform is accomplished by processing these digital words. For example, the first three sample values (from left to right) in Figure 3.7 are represented by the binary sequence 100110111.

From sampling theory we know that a continuous-time bandlimited signal, sampled at a frequency exceeding the Nyquist frequency, can be reconstructed without error from the samples. Therefore, under these conditions the sampling operation is reversible. The quantizing operation, however, is not reversible. Once sample values are quantized, only the quantizing level is maintained and therefore a random error is induced. As before, the value of the waveform at the sampling instant $t = kT_s$ is denoted $x[k]$, and the corresponding quantized value is denoted $x_q[k]$, which is

$$x_q[k] = x[k] + e_q[k] \tag{3.30}$$

where $e_q[k]$ is the error induced by the quantizing process. The quantizer model implied by (3.30) is illustrated in Figure 3.8. If the original signal is not bandlimited, the resulting digital signal contains both aliasing and quantizing errors.

The quantity of interest is the signal-to-noise ratio (SNR), where the noise is interpreted as the noise resulting from the quantizing process. The SNR due to quantizing, denoted $(SNR)_q$, is

$$(SNR)_q = \frac{S}{N_q} = \frac{E\left\{x^2[k]\right\}}{E\left\{e_q^2[k]\right\}} \tag{3.31}$$

where $E\{\cdot\}$ denotes statistical expectation and N_q is the noise power resulting from the quantizing process. In order to determine $(SNR)_q$ the probability density function of the error $e_q[k]$ must be known. The pdf of the quantizing error is a function of the format used to represent numbers in the computer. There are a wide variety of formats that can be used. The broad categories are fixed point and floating point.

Fixed-Point Arithmetic

Even though we are, for the most part, considering simulation using general-purpose computers in which numbers are represented in a floating-point format, we pause to consider quantizing errors resulting from fixed-point number representations.

There are several reasons for considering fixed-point arithmetic (quantizing). First, by considering fixed-point arithmetic, the basic mechanism by which quantizing errors arise is illustrated. Also, special-purpose simulators have been developed that use fixed-point arithmetic because fixed-point calculations can be executed much faster than floating-point calculations. In addition, power consumption is usually lower with fixed-point processors. Perhaps the most important reason for considering fixed-point arithmetic is that devices using fixed-point arithmetic must often be simulated. For example, software-based communications systems are becoming popular, since they can easily be reconfigured for different applications by simply downloading appropriate programs to the device. In order to be commercially attractive in a competitive environment, these systems must be available at the lowest possible cost. Cost is typically minimized by using fixed-point arithmetic and, in addition, fixed-point algorithms execute much faster than floating-point algorithms. We should point out that the design of these software-based devices usually starts with a simulation and, when the simulation shows that the device is properly designed and meets specifications, the simulation code is downloaded to the device.[2] In such applications, the simulation of the device and the physical device merge to a great extent. As previously mentioned, speed, cost, and power consumption requirements usually dictate that many commercial devices utilize fixed-point arithmetic, and simulation is an important tool for the design and performance evaluation of these devices.

Assume that the width of a quantizing level, as illustrated in Figure 3.7, is denoted Δ. Also assume that a sample value corresponding to a given quantizing level is assumed to be the value at the center of the quantizing level.[3] In this case the maximum value of $|e_q[k]|$ is $\Delta/2$. If the number of quantizing levels is large, corresponding to long digital wordlengths, and if the signal varies significantly from sample to sample, a given sample is equally likely to fall at any point in the quantizing level. For this case the errors due to quantizing can be assumed to be uniformly distributed and independent. The pdf (probability density function) of the quantizing error is therefore uniform over the range $[-\Delta/2, \Delta/2]$ as illustrated in Figure 3.9. Denoting the quantizing error of the k^{th} sample by $e_q[k]$, we have

$$E\{e_q[k]\} = \int_{-\Delta/2}^{\Delta/2} x \frac{1}{\Delta} dx = 0 \tag{3.32}$$

so that the quantizing error is zero mean. The variance of $e_q[k]$ is

$$E\left\{e_q^2[k]\right\} = \int_{-\Delta/2}^{\Delta/2} x^2 \frac{1}{\Delta} dx = \frac{\Delta^2}{12} \tag{3.33}$$

We now compute the signal-to-noise ratio due to quantizing.

[2] Recall the design cycle discussed in Chapter 1.

[3] In order to demonstrate basic principles, the pdf is assumed to be for a simple zero-mean process. In practice the pdf will depend on the manner in which fixed-point numbers are represented in the computer. The most common representations are sign-magnitude, ones-complement, and twos-complement [1].

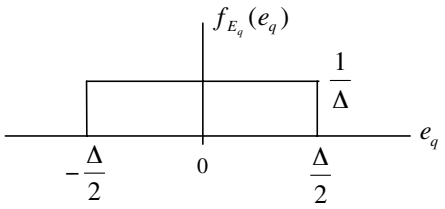


Figure 3.9 Assumed pdf of quantizing error.

Assume that a quantizer has a dynamic range D and that the word length is b. Assuming binary arithmetic, there are 2^b possible quantizing levels and the dynamic range is given by

$$D = 2^b \Delta \tag{3.34}$$

Thus

$$\Delta = D2^{-b} \tag{3.35}$$

and the noise power due to quantizing is

$$N_q = E\left\{e_q^2[k]\right\} = \frac{D^2}{12} 2^{-2b} \tag{3.36}$$

The dynamic range is determined by the peak-to-peak value of the input signal to the quantizer. If the signal power is S, the signal-to-noise ratio due to quantizing, $(SNR)_q$, is

$$(SNR)_q = \frac{S}{\left(\frac{D^2}{12}\right) 2^{-2b}} = \frac{12S}{D^2} 2^{2b} \tag{3.37}$$

Assuming the signal to be zero mean, the values of S and D are related by the *crest factor* of the underlying signal. The crest factor is defined as the ratio of the RMS, or standard deviation, of a signal to the peak value of the signal. To illustrate this relationship, assume that the underlying signal, having dynamic range (peak-to-peak value) D, lies in the range $\pm D$. Since the signal power is S, the standard deviation is $\sqrt{S}$. Therefore, the crest factor is

$$F_c = \frac{\sqrt{S}}{D/2} = \frac{2\sqrt{S}}{D} \tag{3.38}$$

Substitution of (3.38) into (3.37) gives

$$(SNR)_q = 3F_c^2 2^{2b} \tag{3.39}$$

which, in dB units, is

$$(SNR)_q = 4.7712 + 20\log_{10} F_c + 6.0206b \quad \text{dB} \tag{3.40}$$

Note that signals with a high crest factor are more immune to quantizing error than signals with a small crest factor. This result is logical, since signals with a high crest factor have a large standard deviation, which means that they are more spread out through the quantizing levels. It is, however, the word length b that has the most significant impact on $(SNR)_q$. Note that $(SNR)_q$ improves by 6 dB for each bit added to the word length.

Floating-Point Arithmetic

As mentioned previously, throughout most of this book our concern will be simulations for execution on general-purpose computers that utilize floating-point number representations. The form of a floating-point number is $\pm M * (\pm 10\hat{}E)$, where M and E are referred to as the mantissa and exponent, respectively. Where accuracy is required, 64-bit (double-precision) digital words are used and these 64 bits must be allocated between the mantissa and exponent. This allocation can have a significant effect on the result of a given computation. Fortunately, this assignment has been standardized and most, but not all, computers adhere to the standard. The ANSI/IEEE standard for floating-point arithmetic specifies that 53 bits are assigned to the mantissa and 11 bits are assigned to the exponent [3]. Fortunately, MATLAB provides a simple way to determine whether or not the IEEE standard is implemented on a given computer. One simply enters `isieee` at the MATLAB prompt, and a 1 is returned if the standard is implemented.

Since we will be using MATLAB throughout this book for developing and demonstrating simulations, it is important to consider the accuracy that can be expected. For our purposes, the most important parameters resulting from the floating-point format are the resolution (the difference between 1 and the next largest floating-point number), which is the MATLAB variable `eps`, the largest number that can be represented (`realmax` in MATLAB) and the smallest positive number that can be represented (`realmin` in MATLAB). Executing the simple MATLAB script `mparameters` tests for compliance with the IEEE floating-point standard and returns the values of each of these three important parameters. The script `mparameters` follows.

```
% File: c3_mparameters.m
format long       % display full precision
a = ['The value of isieee is ',num2str(isieee),'.'];
b = ['The value of eps is ',num2str(eps,15),'.'];
c = ['The value of realmax is ',num2str(realmax,15),'.'];
d = ['The value of realmin is ',num2str(realmin,15),'.'];
disp(a)           % display isieee
disp(b)           % display eps
disp(c)           % display realmax
disp(d)           % display realmin
```

```
format short      % restore default format
% End script file.
```

Executing the file `mparameters` on a computer that implements the IEEE floating-point standard provides the following results:

```
>> mparameters
The value of isieee is 1.
The value of eps is 2.22044604925031e-016.
The value of realmax is 1.79769313486232e+308.
The value of realmin is 2.2250738585072e-308.
```

The first result displayed (`isiee = 1`) indicates that the computer does indeed conform to the ANSI/IEEE standard for floating-point arithmetic. The next result is `eps`, which is essentially the smallest resolvable difference between two numbers. Note that `eps` is 2^{-52} (the extra bit associated with the mantissa accounts for the sign bit), which illustrates the relationship between `eps` and the word length. We see that more than 15 significant figures of accuracy are achieved. It is the value of `eps` that ties most closely to the width of the quantization level Δ that was discussed in connection with fixed-point arithmetic. Note that $\pm$`realmax` defines the dynamic range, which, in this case exceeds 600 orders of magnitude.

Example 3.2. Suppose that we use floating-point arithmetic, consistent with the ANSI/IEEE standard, to compute the value of

$$A = 1 - 0.4 - 0.3 - 0.2 - 0.1 \tag{3.41}$$

which is obviously zero. However, performing this computation in MATLAB gives the following:

```
>> a = 1-0.4-0.3-0.2-0.1
a =
 -2.7756e-017
```

We see that the error induced by floating-point arithmetic is certainly small and is probably negligible in most applications. The error is not zero, however, and the user should always keep in mind that computed results are not usually exact. ■

From this point forward we will assume that the quantizing errors resulting from floating-point calculations are negligible. While this is an appropriate (and necessary) assumption for the material contained in the remainder of this book, one should be aware that even small errors can accumulate, in certain types of calculations, to the point where the results are useless. DSP calculations in which the signal of interest is a small difference of two very large numbers are a classical example. Very large block length FFTs can give problems because of the large number of butterfly calculations that are cascaded. There are many other examples. In developing DSP algorithms care must be used to ensure that finite word length effects are minimized.

3.3 Reconstruction and Interpolation

We now consider the reconstruction of a continuous-time signal from a sequence of samples. Since a digital simulation processes only sample values, a continuous-time signal is never reconstructed from a set of samples in a simulation environment. Consideration of the reconstruction process, however, leads to the subject of interpolation, which is an important operation in a simulation environment.

The general reconstruction technique is to pass the samples through a linear filter having an impulse response $h(t)$. Thus the reconstructed waveform is given by $x_r(t) = x_s(t) \circledast h(t)$, where, as before, $\circledast$ denotes convolution. From (3.1) and (3.9) we can write

$$x_s(t) = x(t) \sum_{k=-\infty}^{\infty} \delta(t - kT_s) = \sum_{k=-\infty}^{\infty} x(kT_s)\delta(t - kT_s) \tag{3.42}$$

Thus, the reconstructed signal is given by

$$x_r(t) = \left[\sum_{k=-\infty}^{\infty} x(kT_s)\delta(t - kT_s) \right] \circledast h(t) \tag{3.43}$$

which is

$$x_r(t) = \sum_{k=-\infty}^{\infty} x(kT_s)h\left(t - kT_s\right) \tag{3.44}$$

The problem is to choose a $h(t)$ that gives satisfactory results with a reasonable level of computational burden.

3.3.1 Ideal Reconstruction

Assuming that a bandlimited signal is sampled at a rate exceeding $2f_h$, the signal may be reconstructed by passing the samples through an ideal lowpass filter having a bandwidth of $f_s/2$. This can be seen in Figure 3.10. If $f_s > 2f_h$ the spectra centered on $f = \pm f_s$ do not overlap the spectrum centered on $f = 0$. The output of the resconstruction filter is $f_s X(f)$ or, in the time domain, $f_s x(t)$. Amplitude scaling by $1/f_s = T_s$ yields $x(t)$.

It follows from Figure 3.10 that the impulse response of the reconstruction filter is

$$h(t) = T_s \int_{-f_s/2}^{f_s/2} \exp(j2\pi ft)df \tag{3.45}$$

where the scale factor of T_s has been included. Thus:

$$h(t) = T_s \frac{1}{j2\pi t} \left[\exp\left(j\pi f_s t\right) - \exp\left(j\pi f_s t\right)\right] \tag{3.46}$$

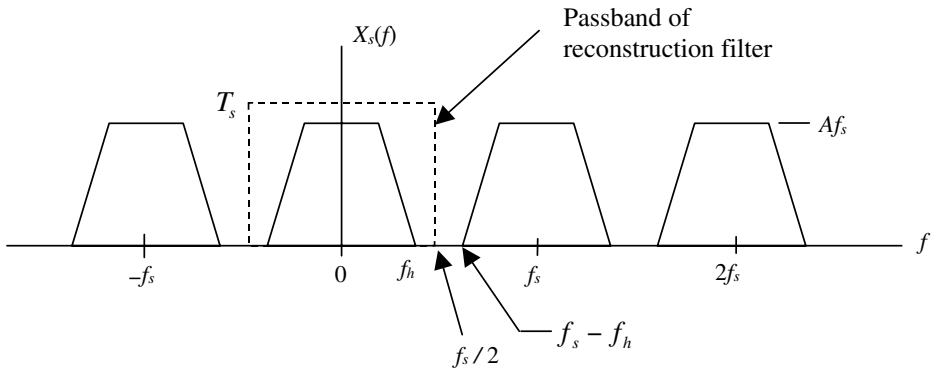


Figure 3.10 Reconstruction filter.

or

$$h(t) = T_s \frac{1}{\pi t} \sin(\pi f_s t) = \text{sinc}(f_s t) \tag{3.47}$$

Substitution into (3.44) gives

$$x_r(t) = \sum_{k=-\infty}^{\infty} x(kT_s)\,\text{sinc}\,[f_s(t - kT_s)] \tag{3.48}$$

or, in more convenient form:

$$x_r(t) = \sum_{k=-\infty}^{\infty} x(kT_s)\,\text{sinc}\left(\frac{t}{T_s} - k\right) \tag{3.49}$$

Note that since the signal $x(t)$ is assumed bandlimited and the sampling frequency is sufficiently high to ensure that aliasing errors are avoided, $x_r(t) = x(t)$. Thus, perfect reconstruction is achieved, at least in theory. Note, however, that (3.49) can never be used in practice, since the sinc(·) function is infinite in extent. Equation (3.49) will be used, however, as the building block for a practical interpolation technique in the following section.

3.3.2 Upsampling and Downsampling

Upsampling and downsampling are used in the simulation of many systems. The need for these operations is illustrated by an example. Consider the direct sequence spread-spectrum system illustrated in Figure 3.11. The data source generates a data signal having a narrowband spectrum of bandwidth W.[4] The data signal is multiplied by a wideband spreading code $c(t)$, which is represented by a binary sequence

[4]The terms *narrowband* and *wideband* are used in a relative sense.

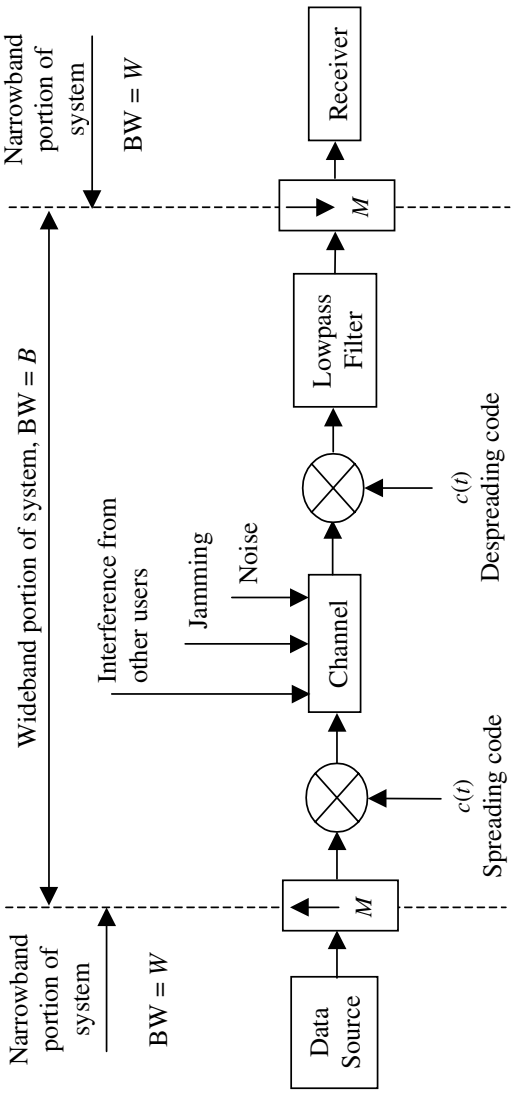


Figure 3.11 System in which upsampling and downsampling is useful.

with a symbol rate much greater than the data rate. The ratio of the spreading code rate to the data rate is called the processing gain of the system. Multiplication by the spreading code $c(t)$ generates a wideband signal, having bandwidth B. The channel imperfections may consist of interference from other users, jamming signals in military communication systems, noise, and perhaps other degradations not accounted for in Figure 3.11. The waveform at the output of the channel is multiplied by the despreading code. The spreading code is assumed to take on values ± 1 and if the spreading code and the despreading code are identical and properly synchronized, multiplying the spreading code and the despreading code gives $c^2(t) = 1$ so that the spreading and despreading codes have no effect on the channel of interest. Note that the data signal is multiplied by $c(t)$ twice and the channel impairments are multiplied by $c(t)$ only once. Thus, at the input to the lowpass filter following multiplication by the despreading code the data signal is again narrowband and all other components are wideband. The lowpass filter extracts the narrowband data signal and passes it to the receiver.

The important attribute of the system illustrated in Figure 3.11 is that both narrowband signals and wideband signals are present. If $B \gg W$, which is typically the case, sampling the narrowband signal at the sampling rate required for the wideband signal will be inefficient and will result in excessive simulation run times. Ideally, each signal should be sampled with a sampling rate appropriate for that signal.

Since signals having two different bandwidths are present in the example system, it is appropriate to use two different sampling rates. Thus the sampling rate must be increased at the boundary between the narrowband and wideband portions of the system (left-hand dashed line in Figure 3.11) and decreased at the boundary between the wideband and narrowband portions of the system (right-hand dashed line in Figure 3.11). Increasing the sampling rate is accomplished by upsampling followed by interpolation, in which new sample values are interpolated from old sample values. Reducing the sampling rate is accomplished by decimation in which unneeded samples are discarded. Upsampling is represented by a block with an upward-pointing arrow and downsampling is represented by a block with a downward-pointing arrow. The parameter M represents the factor by which the sampling period is reduced (upsampling) or increased (downsampling) by the process.

In the material to follow we will use T_s to represent the sampling period prior to the upsampling or downampling process. After upsampling or downsampling, the sampling period will be represented by T_u or T_d, respectively. The signal prior to upsampling or downsampling is denoted $x(t)$ (no subscript on x) and the signal after upsampling or downsampling will be denoted using the appropriate subscript; for example, $x_u(t)$ and $x_d(t)$.

Upsampling and Interpolation

Upsampling is the first operation illustrated in Figure 3.11 and is the process through which the sampling frequency is increased. Since upsampling reduces the sampling period by a factor of M the new sampling period T_u and the old

sampling period T_s are related by $T_u = T_s/M$. Thus, in terms of an underlying continuous-time signal $x(t)$, the upsampling process generates new sample values $x(kT_u) = x(kT_s/M)$ from the old sample values $x(kT_s)$. As an example, suppose that we construct a new set of samples by interpolating the reconstructed signal $x_r(t)$, given by (3.49) at points $t = nT_s/M$. Performing this operations gives

$$x_i(nT_u) = x(nT_s/M) = \sum_{k=-\infty}^{\infty} x(kT_s)\,\mathrm{sinc}\left(\frac{n}{M} - k\right) \tag{3.50}$$

This is not a practical interpolator, since the sinc(·) function is infinite in extent. Truncating the sinc(·) function yields

$$x_i(nT_u) = x(nT_s/M) \cong \sum_{k=-L}^{L} x(kT_s)\,\mathrm{sinc}\left(\frac{n}{M} - k\right) \tag{3.51}$$

a more practical, although not perfect, interpolator. Making L large clearly reduces the interpolation error. However, since each interpolated sample requires $2L + 1$ samples, the computational burden is often unacceptable for large L. Thus, there is a tradeoff between computational burden and accuracy. This tradeoff will be seen many times in our study of simulation. Note also that, since a causal function must be used for interpolation, a delay of LT_s is induced. This delay does not present a problem in simulation, but we must be aware of its presence.

A more practical interpolator, requiring much less computation than the sinc(·) function interpolator, is the linear interpolator. The linear interpolator, although much simpler than the sinc(·) function interpolator, can be used when the underlying signal is significantly oversampled. The impulse response of the linear interpolator is defined by

$$h[k] = \begin{cases} (M - |k|)/M, & k = 0, \pm 1, \pm 2, \cdots, \pm(M-1) \\ 0, & \text{otherwise} \end{cases} \tag{3.52}$$

Note that there are $2M - 1$ nonzero values of $h[k]$. A MATLAB program for developing $h[k]$ follows:

```
% File: c3_lininterp.m
function h=c3_lininterp(M)
h1 = zeros(1,(M-1));
for j=1:(M-1)
    h1(j) = j/M;
end
h = [0,h1,1,fliplr(h1),0];
% End of function file.
```

The upsampling operation is implemented on a discrete set of samples as a two-step process as illustrated in Figure 3.12. We first form $x_u[k]$ from $x[k]$ according to

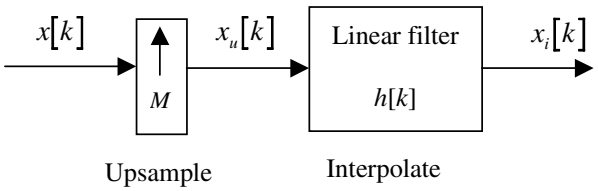


Figure 3.12 Upsampling and interpolation.

$$x_u[k] = \begin{cases} x[k/M], & k = 0, \pm M, \pm 2M, \cdots, \\ 0, & \text{otherwise} \end{cases} \tag{3.53}$$

which can be implemented with the MATLAB code

```
% File: c3_upsample.m
function out=c3_upsamp(in,M)
L = length(in);
out = zeros(1,(L-1)*M+1);
for j=1:L
 out(M*(j-1)+1)=in(j);
end
% End of function file.
```

The result of this operation is to place $M-1$ zero value samples between each sample in the original sequence $x[k]$. Interpolation is then accomplished by convolving $x_u[k]$ with $h[k]$, the impulse response of the linear interpolator. The process of linear interpolation with $M = 3$ is illustrated in Figure 3.13. Note that only two samples are used in the upsampling operation. The necessary delay is then T_s. As illustrated in Figure 3.13, the interpolated value is found by summing the contributions from

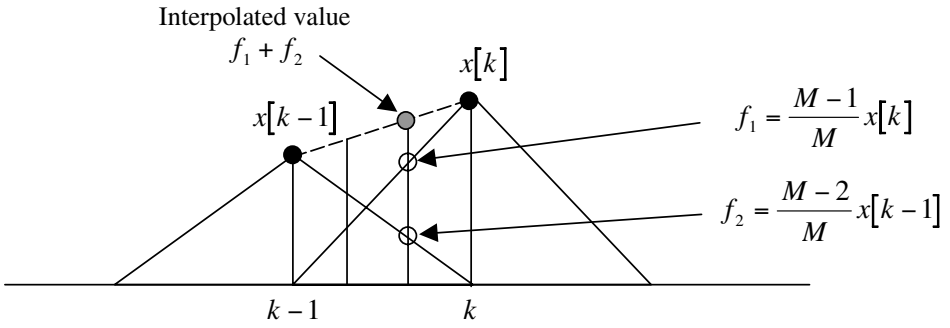


Figure 3.13 Illustration of interpolation process.

$x[k]$ and $x[k-1]$, which are $((M-1)/M)x[k]$ and $((M-2)/M)x[k-1]$, respectively. Thus, with $M = 3$, the interpolated value is

$$\frac{2}{3}x[k] + \frac{1}{3}x[k-1]$$

Since only two samples are used in the interpolation process, linear interpolation is very fast.

Example 3.3. As an illustration of upsampling and interpolation we consider interpolating the samples of a sinewave. The basic samples are illustrated in the top segment of Figure 3.14 as $x[k]$. Upsampling with $M = 6$ yields the sample values $x_u[k]$. Linear interpolation with $M = 6$ gives the sequence of samples $x_i[k]$. Note the delay of T_s. The MATLAB program used to generate Figure 3.14 follows:

```
% File: c3_upsampex.m
M = 6;                      % upsample factor
h = c3_lininterp(M);        % imp response of linear interpolator
t = 0:10;                   % time vector
tu = 0:60;                  % upsampled time vector
x = sin(2*pi*t/10);         % original samples
xu = c3_upsamp(x,M);        % upsampled sequence
subplot(3,1,1)
```

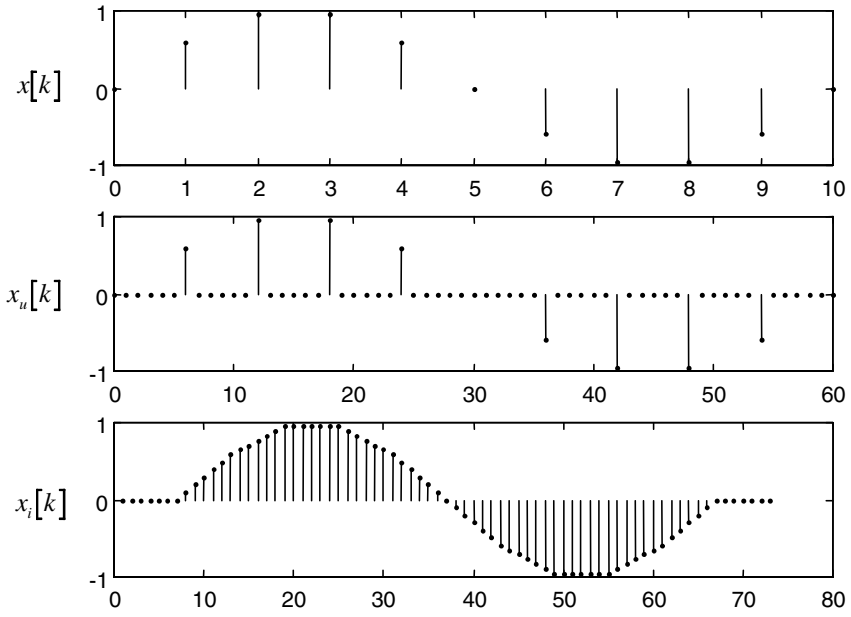


Figure 3.14 Upsampling and interpolation operations used in Example 3.3.

```
stem(t,x,'k.')
ylabel('x')
subplot(3,1,2)
stem(tu,xu,'k.')
ylabel('xu')
xi = conv(h,xu);
subplot(3,1,3)
stem(xi,'k.')
ylabel('xi')
```

■

It is clear that upsampling and downsampling involve a significant amount of overhead. If the upsampling factor M is modest, say, 2 or 3, it is usually best to develop the simulation using a single sampling frequency and therefore oversample the narrowband signals present in the system. If, however, the difference in B and W exceeds an order of magnitude, it is usually most efficient to utilize multiple sampling frequencies in the simulation and sample each signal at an appropriate sampling frequency.

Downsampling (Decimation)

Downsampling is the second operation illustrated in Figure 3.11 and is the process through which the sampling frequency is reduced. The process is accomplished by replacing a block of M samples by a single sample. Downsampling is therefore much simpler than upsampling. The functional representation for the samples at the output of a downsampler is obtained by recognizing that the downsampling process increases the sampling period by a factor of M. Thus the samples at the output of a downsampler, denoted $x_d(kT_d)$, are given by $x_d(kT_d) = x(kMT_s)$. The sample values are given by

$$x_d[k] = x[kM] \tag{3.54}$$

We need to be careful, however, to ensure that the downsampled signal does not exhibit aliasing.

3.4 The Simulation Sampling Frequency

A fundamental decision that must be made in the development of a simulation is the selection of the sampling frequency. For linear systems without feedback, the necessary sampling frequency is dictated by the allowable aliasing error, which in turn is dependent on the power spectral density of the underlying pulse shape.[5] We therefore pause to consider a common model for representing baseband signals used

[5]It will be shown in later chapters that, in addition to signal bandwidth, a number of other factors affect the required sampling frequency. For example, the presence of nonlinearities result in a requirement for higher sampling frequencies. The same is often true for systems containing feedback. In addition, multipath channels place requirements on the sampling frequency so that the multipath delays can be resolved. All of these topics will be considered in detail in later chapters.

for data transmission and to develop a technique for calculating the power spectral density of the signal corresponding to the pulse shape. Since the pulse shape plays such an important role in the selection of an appropriate sampling frequency, we consider the problem in some detail.

We know from our study of sampling that the complete elimination of aliasing errors requires an infinite sampling frequency. This is clearly a situation that cannot be achieved in practice. In addition, as the sampling frequency increases, more samples must be processed for each data symbol passed through the system. This increases the time required for executing the simulation. Since aliasing errors cannot be eliminated in practice, a natural strategy is to choose a sampling frequency for the simulation that achieves an acceptable tradeoff between aliasing errors and simulation run time. Of course, a sampling frequency must be selected so that the errors due to aliasing are negligible compared to the system degradations being investigated by the simulation.

3.4.1 General Development

A common model for the transmitted signal in a digital communication system is

$$x(t) = A\sum_{k=0}^{\infty} a_k p(t - kT - \Delta) \tag{3.55}$$

where

$$\cdots, a_{-2}, a_{-1}, a_0, a_1, a_2, \cdots, a_k, \cdots$$

is a sequence of a random variables representing the data. The values of a_k are typically denoted +1 or -1 in a binary digital system, $p(t)$ is the pulse shape function, T is the symbol period (bit period for binary transmission), and Δ is a random variable uniformly distributed over the sampling period.[6] The parameter A is a scaling constant used to establish the power in the transmitted signal. By incorporating this parameter, we can scale the pulse shape function so that the peak value is unity. We assume that $E\{a_k\} = 0$ and $E\{a_k a_{k+m}\} = R_m$ represent the mean and the autocorrelation of the data sequence, respectively.

It is easily shown [2] that the autocorrelation function of the transmitted signal is given by

$$R_{XX}(\tau) = A^2 \sum_{m=-\infty}^{\infty} R_m r(\tau - mT) \tag{3.56}$$

in which

$$r(\tau) = \frac{1}{T}\int_{-\infty}^{\infty} p(t)p(t+\tau)dt \tag{3.57}$$

[6]Note that we are now using $p(t)$ for the pulse shape rather than for the sampling function as in the preceding section. The meaning of $p(t)$ will be clear from the context in which it is used.

The required sampling frequency is determined from the PSD of the transmitted signal. Applying the Weiner-Khintchine theorem to (3.56) gives

$$S_X(f) = A^2 \int_{-\infty}^{\infty} \left(\sum_{m=-\infty}^{\infty} R_m r(\tau - mT) \right) \exp\left(-j2\pi f\tau\right)\, d\tau \tag{3.58}$$

or

$$S_X(f) = A^2 \sum_{m=-\infty}^{\infty} R_m \int_{-\infty}^{\infty} r(\tau - mT) \exp\left(-j2\pi f\tau\right)\, d\tau \tag{3.59}$$

We now put this in a form more useful for computation.

The first step is to apply the change of variables $\alpha = \tau - mT$ to (3.59). This gives

$$S_X(f) = A^2 \sum_{m=-\infty}^{\infty} R_m \int_{-\infty}^{\infty} r(\alpha) \exp\left[-j2\pi f(\alpha + mT)\right]\, d\alpha \tag{3.60}$$

Denoting

$$S_r(f) = \int_{-\infty}^{\infty} r(\alpha) \exp(-j2\pi f\alpha)\, d\alpha \tag{3.61}$$

gives

$$S_X(f) = A^2 \sum_{m=-\infty}^{\infty} R_m S_r(f) \exp(-j2\pi fmT) \tag{3.62}$$

We now determine $S_r(f)$.

Fourier transforming (3.57) gives

$$S_r(f) = \int_{-\infty}^{\infty} \left(\frac{1}{T} \int_{-\infty}^{\infty} p(t)p(t+\alpha)\, dt \right) \exp(-j2\pi f\alpha)\, d\alpha \tag{3.63}$$

Applying the change of variables $\beta = t + \alpha$ allows (3.63) to be expressed in the form

$$S_r(f) = \frac{1}{T} \left(\int_{-\infty}^{\infty} p(t) \exp(j2\pi ft)\, dt \right) \left(\int_{-\infty}^{\infty} p(\beta) \exp(-j2\pi f\beta)\, d\beta \right) \tag{3.64}$$

The second term in (3.64) is the Fourier transform of the pulse shape function $p(t)$ and the first term is the complex conjugate of the first term. This gives

$$S_r(f) = \frac{|P(f)|^2}{T} = \frac{G(f)}{T} \tag{3.65}$$

where $G(f)$ is the energy spectral density of the pulse shape function $p(t)$. Substitution of (3.65) into (3.62) gives the general result

$$S_X(f) = A^2 \frac{G(f)}{T} \sum_{m=-\infty}^{\infty} R_m \exp(-j2\pi fmT) \tag{3.66}$$

In many applications the data symbols can be assumed independent. This assumption results in significant simplifications.

3.4.2 Independent Data Symbols

If the data symbols $\{a_k\}$ are independent the autocorrelation becomes

$$R_m = E\{a_k a_{k+m}\} = E\{a_k\} E\{a_{k+m}\} = a_k^2 \delta[m] \tag{3.67}$$

so that $R_m = a_k^2$ for $m = 0$ and $R_m = 0$ otherwise. The PSD of $x(t)$ as defined by (3.66) takes a very simple form for this case:

$$S_X(f) = A^2 \frac{a_k^2 G(f)}{T} \tag{3.68}$$

If the data symbols are assumed to be $a_k = \pm 1$ for all k, $a_k^2 = 1$ and

$$S_X(f) = A^2 \frac{G(f)}{T} \tag{3.69}$$

which is independent of the data.[7] For the case in which the data symbols are not independent, the underlying autocorrelation function R_m must be determined and (3.66) must be evaluated term by term. See [2] for an example.

Example 3.4. Consider the rectangular pulse shape illustrated in Figure 3.15. It follows from Figure 3.15 that

$$P(f) = \int_0^T \exp(-j2\pi ft)\, dt = \frac{1}{j2\pi f}\left[1 - \exp(-j2\pi fT)\right] \tag{3.70}$$

This can be placed in the form

$$\begin{aligned} P(f) &= \frac{1}{j2\pi f}\left[\exp(j\pi fT) - \exp(-j\pi fT)\right]\exp(-j\pi fT) \\ &= \frac{\sin(\pi fT)}{\pi f}\exp(-j\pi fT) \end{aligned} \tag{3.71}$$

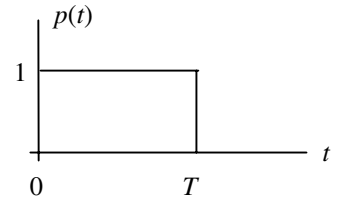


Figure 3.15 Rectangular pulse shape.

[7] We make the assumption that the data samples are +1 and −1 rather than +1 and 0 to be consistent with the assumption that $E\{a_k\} = 0$.

or, in terms of the sinc(·) function

$$P(f) = T\,\mathrm{sinc}(fT)\exp(-j\pi fT) \tag{3.72}$$

Therefore

$$G(f) = |P(f)|^2 = T^2\,\mathrm{sinc}^2(fT)$$

Substitution into (3.69) gives

$$S_X(f) = A^2T\,\mathrm{sinc}^2(fT) \tag{3.73}$$

for the power spectral density of $x(t)$.

The transmitted power is, from (3.69)

$$P = \int_{-\infty}^{\infty} S_X(f)\,df = A^2\frac{1}{T}\int_{-\infty}^{\infty} G(f)\,df \tag{3.74}$$

From Parseval's theorem and Figure 3.15 we know that

$$\int_{-\infty}^{\infty} G(f)df = \int_{-\infty}^{\infty} |P(f)|^2\,df = \int_{-\infty}^{\infty} p^2(t)\,dt = T \tag{3.75}$$

Substitution into (3.74) gives

$$P = A^2 \tag{3.76}$$

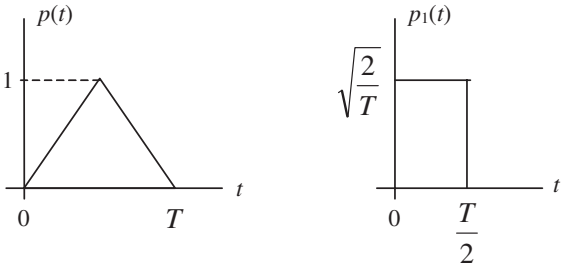
as expected. This simple result arises from the fact that $p(t)$ is a unit amplitude pulse. Thus $\sum a_k p(t - kT - \Delta)$ has unit power. Multiplication by A as shown in (3.55) simply scales the power by A^2. For other pulse shapes the relationship between power and A must be computed using the technique just illustrated. Remember also the assumed data sequence $\{a_k\}$ is a unit power (variance) process. ■

Example 3.5. An interesting pulse shape, which will be needed later, is illustrated in Figure 3.16(a). The basic pulse shape $p(t)$ can be expressed $p_1(t) \circledast p_1(t)$ where $\circledast$ denotes convolution and $p_1(t)$ is illustrated in Figure 3.16(b). Taking the Fourier transform of $p_1(t)$ gives

$$P_1(f) = \sqrt{\frac{T}{2}}\,\mathrm{sinc}\left(\frac{T}{2}f\right)\exp(-j\pi fT/2) \tag{3.77}$$

Since convolution in the time domain is equivalent to multiplication in the frequency domain we have

$$|P(f)| = |P_1(f)|^2 = \frac{T}{2}\,\mathrm{sinc}^2\left(\frac{T}{2}f\right) \tag{3.78}$$



(a) Triangular pulse shape (b) Basic shape for convolution

Figure 3.16 Triangular pulse shape.

Thus

$$G(f) = |P(f)|^2 = \frac{T^2}{4}\,\mathrm{sinc}^4\left(\frac{T}{2}f\right) \tag{3.79}$$

Substitution into (3.69) gives

$$S_X(f) = \frac{A^2T}{4}\,\mathrm{sinc}^4\left(\frac{T}{2}f\right) \tag{3.80}$$

This result will be used later in this chapter and in the problems. ■

3.4.3 Simulation Sampling Frequency

We now return to the problem of relating the simulation sampling frequency to a given pulse shape. This is accomplished by considering the signal-to-noise ratio of the sampling process where the noise power arises from aliasing. The goal is to select a sampling frequency so that the errors due to aliasing are negligible compared to the system degradations being investigated by the simulation. It will be shown that the required sampling frequency is dependent upon the waveshapes present in the simulation model.

Consider a waveform defined by (3.55), having the rectangular pulse shape illustrated in Figure 3.15, to be sampled as illustrated in Figure 3.17. In drawing

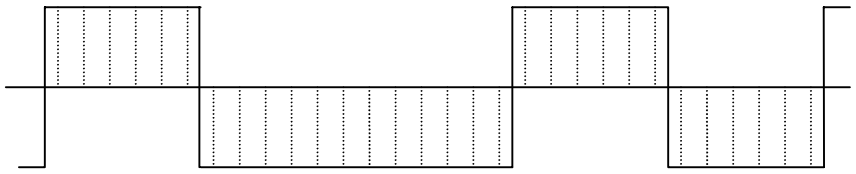


Figure 3.17 Sequence of rectangular pulses sampled at six samples per symbol.

Figure 3.17 a sampling frequency of six times the symbol frequency was assumed. The power spectral density of the data sequence is given by (3.73). Combining this result with (3.18) gives

$$S_{X_s}(f) = f_s^2 \sum_{n=-\infty}^{\infty} A^2 T \operatorname{sinc}^2[(f - nf_s)T] \tag{3.81}$$

Sampling the data sequence at m samples per symbol ($f_s = m/T$) gives

$$S_{X_s}(f) = f_s^2 \sum_{n=-\infty}^{\infty} A^2 T \operatorname{sinc}^2 \left[(f - \frac{nm}{T})T\right] \tag{3.82}$$

which is

$$S_{X_s}(f) = f_s^2 \sum_{n=-\infty}^{\infty} A^2 T \operatorname{sinc}^2 (fT - nm) \tag{3.83}$$

The next task is to compute the signal-to-noise ratio due to aliasing.

The signal-to-noise ratio due to aliasing can be expressed $(SNR)_a = S/N_a$ where the signal power is

$$S = \int_{-f_s/2}^{f_s/2} f_s^2 A^2 T \operatorname{sinc}^2(fT)\, df = 2 f_s^2 A^2 T \int_0^{f_s/2} \operatorname{sinc}^2(fT)\, df \tag{3.84}$$

and the noise power due to aliasing is

$$\begin{aligned} N_a &= \int_{-f_s/2}^{f_s/2} f_s^2 \sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} A^2 T \operatorname{sinc}^2 (fT - nm)\ df \\ &= 2 f_s^2 A^2 T \sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} \int_0^{f_s/2} \operatorname{sinc}^2 (fT - nm)\ df \end{aligned} \tag{3.85}$$

Note that we have made use of the fact that PSD is an even function of frequency. The signal power is determined by integrating the $n = 0$ term in (3.83) over the simulation bandwidth $|f| < f_s/2$. The noise power due to aliasing is the power from all of the frequency translated terms ($n \neq 0$) that fall in the simulation bandwidth. Thus, the noise due to aliasing is found by integrating over all terms in (3.83) with the $n = 0$ term excepted. This is made clear by Figure 3.18, which is drawn (not to scale) for $m = 6$. Figure 3.18 illustrates the positive frequency portion of the $n = 0$ term of (3.83) in the range $0 < f < f_s$. The translated spectra for $n = \pm 1$ and $n = 2$ are also shown.

The next step in the determination of $(SNR)_a$ is to show that both S and N_a can be determined using only the $n = 0$ term in (3.83). The lobes of the sinc$(\cdot)$ function, each having width $1/T$, are illustrated and numbered in Figure 3.18. Note that for $m = 6$ and $n = 0$ lobes 1, 2, and 3 fall in the range $0 < f < f_s/2$ and

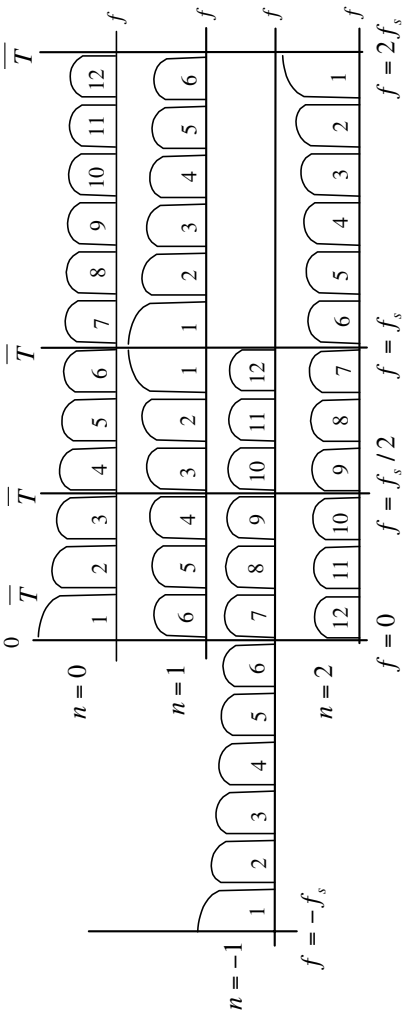


Figure 3.18 Spectra and translated spectra for $n = 0$, $n = \pm 1$, and $n = 2$.

therefore represent signal power. Lobes 4, 5, and 6 fall into the range $0 < f < f_s/2$ for the $n = 1$ term and therefore represents aliasing noise. In a similar manner lobes 7, 8, and 9 result from the $n = -1$ term in (3.83) and lobes 10, 11, and 12 result from the $n = 2$ term in (3.83). Continuing this line of thought shows that

$$\sum_{\substack{n=-\infty \\ n\neq 0}}^{\infty} \int_0^{f_s/2} \text{sinc}^2\left(fT - nm\right)\, df = \int_{f_s/2}^{\infty} \text{sinc}^2\left(fT - nm\right)\, df \tag{3.86}$$

Therefore

$$(SNR)_a = \frac{S}{N_a} = \frac{\int_0^{f_s/2} \text{sinc}^2(fT)\, df}{\int_{f_s/2}^{\infty} \text{sinc}^2(fT)\, df} \tag{3.87}$$

As can be seen by comparing Examples 3.4 and 3.5, the form of the integrand will be different depending on the pulse shape.

We will frequently find it necessary to use numerical integration in order to evaluate $(SNR)_a$. In order to accomplish this a second sampling operation is introduced in which the continuous frequency variable f is sampled at points $f = jf_1$. For accuracy we clearly require $f_1 \ll 1/T$ so that many samples are taken in each lobe of the sinc$(\cdot)$ function. Frequency sampling in this way allows the integrals in (3.87) to be replaced by sums. In order to satisfy $f_1 \ll 1/T$ let $f_1 = 1/(kT)$ where k is large so that the error induced by the numerical integration is small. With $f = jf_1$ and $f_1 = 1/(kT)$ we have

$$fT = \frac{j}{k} \tag{3.88}$$

The next step is to compute the folding frequency $f_s/2$ in terms of the discrete parameters k and m. From Figure 3.18 we see that, for m samples per symbol, the folding frequency $f_s/2$ is $m/(2T)$. Since k samples are taken for every frequency interval of width $1/T$, the folding frequency corresponds to the index $km/2$. Using (3.88) and the fact that $f_s/2$ corresponds to $km/2$ in (3.87) gives

$$(SNR)_a \cong \frac{\sum_{j=0}^{km/2} \text{sinc}^2(j/k)}{\sum_{j=km/2}^{\infty} \text{sinc}^2(j/k)} \tag{3.89}$$

The preceding is an approximation because numerical integration is used to approximate the true value of the integral.

The MATLAB program to evaluate (3.89) follows.

```
% File: c3_sna.m
k = 50;                              % samples per lobe
nsamp = 50000;                       % total frequency samples
snrdb = zeros(1,17);                 % initialize memory
x = 4:20;                            % vector for plotting
for m = 4:20                         % iterate samples per symbol
```

```
    signal = 0; noise = 0;          % initialize sum values
    f_fold = k*m/2;                 % folding frequency
    for j = 1:f_fold
        term = (sin(pi*j/k)/(pi*j/k))^2;
        signal = signal+term;
    end
    for j = (f_fold+1):nsamp
        term = (sin(pi*j/k)/(pi*j/k))^2;
        noise = noise+term;
    end
    snrdb(m-3) = 10*log10(signal/noise);
end
plot(x,snrdb)                       % plot results}
xlabel('Samples per symbol')}
ylabel('Signal-to-aliasing noise ratio')}
% End script file.
```

Note that 50 frequency samples are taken in each lobe of the sinc(·) function and that a total of 50,000 frequency samples are taken. Thus, the summation in the denominator of (3.89) spans 1,000 lobes of the sinc(·) function, after which the PSD is assumed negligible. This assumption may be verified by experimenting with the parameter `nsamp`.

Executing the preceding program yields the result illustrated in Figure 3.19. Note that $(SNR)_a$ is slightly less than 17 dB for $m = 10$ samples per symbol and that $(SNR)_a$ continues to increase as m increases. However, the impact on $(SNR)_a$ decreases for increasing m. Also note that the PSD of the sampled signal decreases as $1/f^2$ for a rectangular pulse shape. Example 3.5 shows that the PSD of the sampled signal decreases as $1/f^4$ for a triangular pulse shape. Thus, for a given value of m, the value of $(SNR)_a$ will be greater for the triangular pulse shape than for the rectangular pulse shape. The rectangular pulse shape represents is a worst-case situation. Other pulse shapes are considered in the Problems.

In a practical communications system the pulse shape $p(t)$ is chosen to give a required bandwidth efficiency [2]. High bandwidth efficiency implies that the spectrum of $x(t)$ as defined by (3.55) is compact about $f = 0$.[8] Thus, signals that exhibit high bandwidth efficiency require a smaller value of m for a given $(SNR)_a$.

3.5 Summary

The purpose of this chapter was to cover a number of topics related to sampling and the representation of sample values in communication system simulations. Two fundamental sampling theorems were considered the lowpass sampling theorem and the bandpass sampling theorem. Since bandpass signals are usually represented by lowpass signals in system simulations, the lowpass sampling theorem is the most

[8]The reference is $f = 0$ rather than $f = f_c$, in which f_c is a nonzero carrier frequency, since (3.55) represents a lowpass model of a bandpass process.

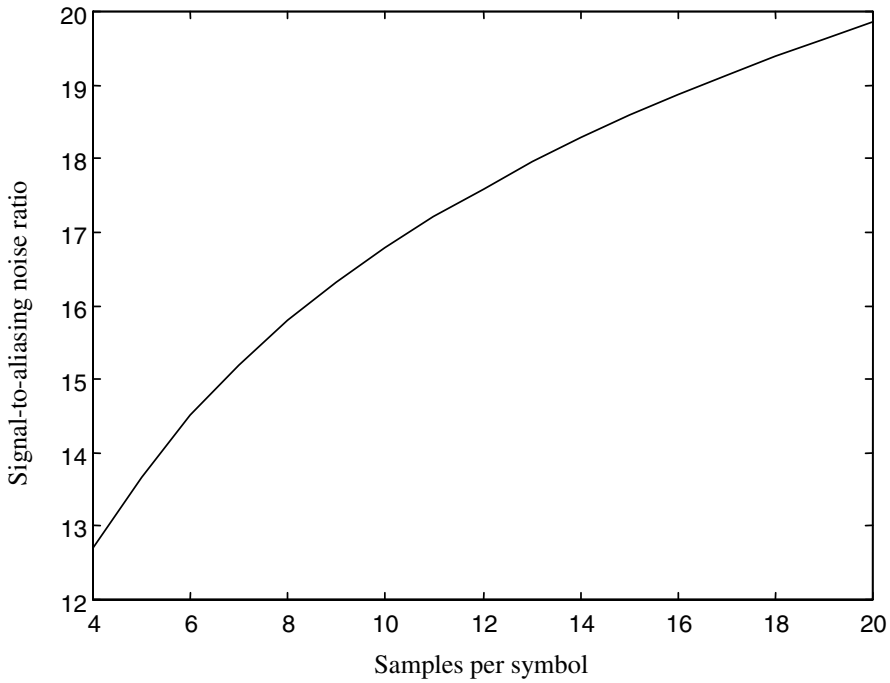


Figure 3.19 Signal-to-aliasing-noise ratio for the rectangular pulse shape.

important of these to theorems for our application. We saw that a bandlimited lowpass signal may be sampled and that the underlying bandpass signal may be reconstructed from the sample values if the sampling frequency exceeds twice the bandwidth of the bandlimited lowpass signal. The bandpass sampling theorem, although less useful in the simulation context than the lowpass sampling theorem, gave a somewhat similar result. Bandpass signals could be sampled and reconstructed if the sampling frequency is between $2B$ and $4B$ where B is the bandwidth of the bandpass signal being sampled.

Next quantizing was considered. Quantizing errors are present in all simulations, since sample values must be represented by digital words of finite length. Two types of quantizing errors were considered; errors resulting from fixed-point number representations and errors resulting from floating-point number representations. When fixed-point number representations are used, the signal-to-quantizing-noise ratio increases 6 dB for each bit added to the word length. When simulations are preformed on general-purpose computers, which use floating-point number representations, the noise resulting from quantizing errors is usually negligible. This noise, however, is never zero and there are situations in which errors can accumulate and significantly degrade the accuracy of the simulation result. The simulation user must therefore be aware of this potential error source.

The third section of this chapter treated reconstruction and interpolation. We saw that if a lowpass bandlimited signal is sampled with a sampling frequency exceeding twice the signal bandwidth, the underlying continuous-time signal can be reconstructed without error by weighting each sample with a $\sin(x)/x$ waveform, which is equivalent to passing the samples through an ideal lowpass filter. The result is a waveform defined for all values of time and, by extracting "new" samples between the original samples, interpolated samples can be generated. This operation, known as upsampling, increases the effective sampling frequency. The inverse operation, downsampling, can be accomplished by extracting every M^{th} sample from the original set of samples. Using the operations of upsampling and downsampling, one can develop a simulation in which multiple sampling frequencies are present. This is useful when the system being simulated contains signals having widely differing bandwidths. A spread-spectrum communications system is an example of such a system.

The final topic treated in this chapter was the important problem of relating the sampling frequency to the pulse shape used for waveform transmission. The pulse shape was assumed time limited and therefore cannot be bandlimited. Therefore aliasing errors occur. The criterion used for selecting the sampling frequency was to determine the required signal-to-noise ratio, where aliasing error constituted the noise source. A general method was developed for determining the PSD of the modulated signal and numerical integration of this PSD determined the signal-to-aliasing-noise ratio.

3.6 Further Reading

Most textbooks on basic communication theory consider several of the topics presented in this chapter. Included are the sampling theorem and models for transmitted signals using various pulse shape functions. Examples are:

R. E. Ziemer and W. H. Tranter, *Principles of Communications: Systems, Modulation and Noise*, 5th ed., New York: Wiley, 2001.

R. E. Ziemer and R. L. Peterson, *Introduction to Digital Communication*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2001.

The topics of quantizing, interpolation, and decimation are typically covered in textbooks on digital signal processing. Although a wide variety of books are available in this category, the following is recommended:

A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Upper Saddle River, NJ: Prentice Hall, 1989.

The following textbook is an excellent reference on multirate signal processing and sampling rate conversion:

R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Upper Saddle River, NJ: Prentice Hall, 1983.

Simulation applications of the topics presented in this chapter can be found in:

M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems*, 2nd ed., New York: Kluwer Academic/Plenum Publishers, 2000.

3.7 References

1. A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Upper Saddle River, NJ: Prentice-Hall, 1989.

2. R. E. Ziemer and W. H. Tranter, *Principles of Communications: Systems, Modulation and Noise*, 5th ed., New York: Wiley, 2002.

3. The ANSI/IEEE Standard 754-1985 (available from IEEE).

3.8 Problems

3.1 A signal $x(t)$, given by

$$x(t) = 5\cos(6\pi t) + 3\sin(8\pi t)$$

is sampled using a sampling frequency f_s of 10 samples per second. Plot $X(f)$ and $X_s(f)$. Plot the output of the reconstruction filter assuming that the reconstruction filter is an ideal lowpass filter with a bandwidth of $f_s/2$. The passband gain of the reconstruction filter is $T_s = 1/f_s$.

3.2 Repeat the preceding problem using a sampling frequency of 7 samples per second.

3.3 Develop a MATLAB program that produces, and therefore verifies, Figure 3.5.

3.4 A bandpass signal has a center frequency of 15 MHz and a bandwidth of 750 kHz (375 kHz each side of the carrier).

(a) Using the bandpass sampling theorem determine the minimum sampling frequency at which the bandpass signal can be sampled and reconstructed without error.

(b) By drawing the spectrum of the sampled signal and defining all frequencies of interest, show that the signal can be reconstructed without error if the sampling frequency found in (a) is used.

(c) Starting with the sampling frequency found in (a) consider the effect of increasing the sampling frequency. By how much can the sampling frequency be increased without incurring aliasing errors?

3.5 Assume that a signal defined by $5\sin(10\pi t)$ is sampled and quantized using a fixed-point number representation.

(a) Determine the dynamic range of the signal.

(b) Determine the crest factor of the signal.

(c) Determine the signal-to-noise ratio $(SNR)_a$ for $b = 4, 8, 16$, and 32 bits.

3.6 Repeat the preceding problem for the signal illustrated Figure 3.20.

3.7 In evaluating the effect of a fixed-point quantizing process, the assumption was made that the error induced by the quantizing process can be represented by a uniformly distributed random value. In this problem we investigate the validity of this assumption.

(a) Use $\sin(6t)$ as a signal. Using a sampling frequency of 20 Hz, generate, using MATLAB, a vector of 10,000 samples of this waveform. Note that the signal frequency and the sampling frequency are not harmonically related. Why was this done?

(b) Develop a MATLAB model for a fixed-point quantizer that contains 16 quantizing levels ($b = 4$). Using this model quantize the sample values generated in (a). Generate a vector representing the 10,000 values of quantizing error.

(c) Compute the values of $E\{e[k]\}$ and $E\{e^2[k]\}$. Compare with the theoretical values and explain the results.

(d) Using the MATLAB function `hist`, generate a histogram of the quantizing errors. What do you conclude?

3.8 The value of `realmax` is the largest number that can be represented on a computer that adheres to the ANSI/IEEE standard for representing floating-point numbers. Anything larger results in an overflow, which in MATLAB is represented by `Inf`. Using a computer that adheres to the ANSI/IEEE standard make the following computations and answer the accompanying questions:

(a) Compute `realmax + 1`. Note that no overflow occurs. Explain this apparent contradiction.

(b) Compute `realmax + 1.0e291` and `realmax + 1.0e292`. Explain the results.

3.9 Using MATLAB, compute

$$A = 1 - 0.5 - 0.25 - 0.125 - 0.125$$

Compare the result of this calculation with the result of Example 3.2. Explain the difference.

3.10 Fill in the steps to derive (3.56) and (3.57).

3.11 Data is transmitted as modeled by (3.55) in which the pulse shape $p(t)$ is the triangular pulse illustrated in Figure 3.16(a). Develop a MATLAB program to plot the signal to aliasing noise ratio $(SNR)_a$ as the number of samples per symbol varies from 4 to 20. Compare the result with that of the rectangular pulse shape by plotting both on the same set of axes. Explain the results.

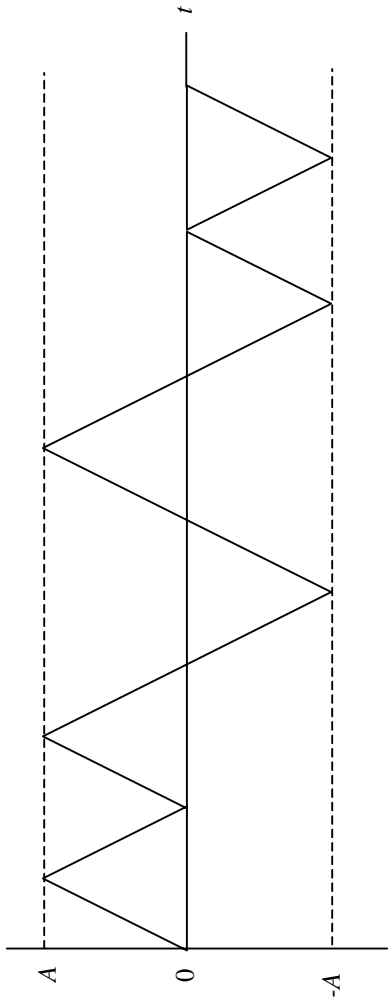


Figure 3.20 Figure for Problem 3.6.

3.12 The energy spectral density of an MSK (minimum shift keyed) signal is defined by

$$G_{MSK}(f) = \frac{16T_b \cos^2(2\pi T_b f)}{\pi^2 \left[1 - (4T_b f)^2\right]^2}$$

where T_b is the bit time [2]. Develop a MATLAB program to plot the signal to aliasing noise ratio $(SNR)_a$ as the number of samples per symbol varies from 4 to 20. Compare the result with that of the rectangular pulse shape by plotting both on the same set of axes. Explain the results.

3.13 Repeat the preceding problem for the QPSK signal for which

$$G(f) = 2T_b \operatorname{sinc}^2(2T_b f)$$