CHAPTER
2

Statistics, Probability and Noise

Lecture 1: P. 11-26 of this chapter and p. 35-38 of chapter 3.

The rest of this chapter after page 26 is recommended but optional

Statistics and probability are used in Digital Signal Processing to characterize signals and the processes that generate them. For example, a primary use of DSP is to reduce interference, noise, and other undesirable components in acquired data. These may be an inherent part of the signal being measured, arise from imperfections in the data acquisition system, or be introduced as an unavoidable byproduct of some DSP operation. Statistics and probability allow these disruptive features to be measured and classified, the first step in developing strategies to remove the offending components. This chapter introduces the most important concepts in statistics and probability, with emphasis on how they apply to acquired signals.

Signal and Graph Terminology

A *signal* is a description of how one parameter is related to another parameter. For example, the most common type of signal in analog electronics is a *voltage* that varies with *time*. Since both parameters can assume a continuous range of values, we will call this a **continuous signal**. In comparison, passing this signal through an analog-to-digital converter forces each of the two parameters to be *quantized*. For instance, imagine the conversion being done with 12 bits at a sampling rate of 1000 samples per second. The voltage is curtailed to 4096 (2^{12}) possible binary levels, and the time is only defined at one millisecond increments. Signals formed from parameters that are quantized in this manner are said to be **discrete signals** or **digitized signals**. For the most part, continuous signals exist in nature, while discrete signals exist inside computers (although you can find exceptions to both cases). It is also possible to have signals where one parameter is continuous and the other is discrete. Since these mixed signals are quite uncommon, they do not have special names given to them, and the nature of the two parameters must be explicitly stated.

Figure 2-1 shows two discrete signals, such as might be acquired with a digital data acquisition system. The **vertical axis** may represent voltage, light

intensity, sound pressure, or an infinite number of other parameters. Since we don't know what it represents in this particular case, we will give it the generic label: **amplitude**. This parameter is also called several other names: the **y-axis**, the **dependent variable**, the **range**, and the **ordinate**.

The **horizontal axis** represents the other parameter of the signal, going by such names as: the **x-axis**, the **independent variable**, the **domain**, and the **abscissa**. *Time* is the most common parameter to appear on the horizontal axis of acquired signals; however, other parameters are used in specific applications. For example, a geophysicist might acquire measurements of rock density at equally spaced *distances* along the surface of the earth. To keep things general, we will simply label the horizontal axis: **sample number**. If this were a continuous signal, another label would have to be used, such as: *time*, *distance*, *x*, etc.

The two parameters that form a signal are generally not interchangeable. The parameter on the y-axis (the dependent variable) is said to be a **function** of the parameter on the x-axis (the independent variable). In other words, the independent variable describes *how* or *when* each sample is taken, while the dependent variable is the actual measurement. Given a specific value on the x-axis, we can always find the corresponding value on the y-axis, but usually not the other way around.

Pay particular attention to the word: *domain*, a very widely used term in DSP. For instance, a signal that uses time as the independent variable (i.e., the parameter on the horizontal axis), is said to be in the **time domain**. Another common signal in DSP uses frequency as the independent variable, resulting in the term, **frequency domain**. Likewise, signals that use distance as the independent parameter are said to be in the **spatial domain** (distance is a measure of space). The type of parameter on the horizontal axis *is* the domain of the signal; it's that simple. What if the x-axis is labeled with something very generic, such as *sample number*? Authors commonly refer to these signals as being in the *time* domain. This is because sampling at equal intervals of time is the most common way of obtaining signals, and they don't have anything more specific to call it.

Although the signals in Fig. 2-1 are discrete, they are displayed in this figure as continuous lines. This is because there are too many samples to be distinguishable if they were displayed as individual markers. In graphs that portray shorter signals, say less than 100 samples, the individual markers are usually shown. Continuous lines may or may not be drawn to connect the markers, depending on how the author wants you to view the data. For instance, a continuous line could imply what is happening *between* samples, or simply be an aid to help the reader's eye follow a trend in noisy data. The point is, examine the labeling of the horizontal axis to find if you are working with a discrete or continuous signal. Don't rely on an illustrator's ability to draw dots.

The variable, N, is widely used in DSP to represent the total number of samples in a signal. For example, $N = 512$ for the signals in Fig. 2-1. To

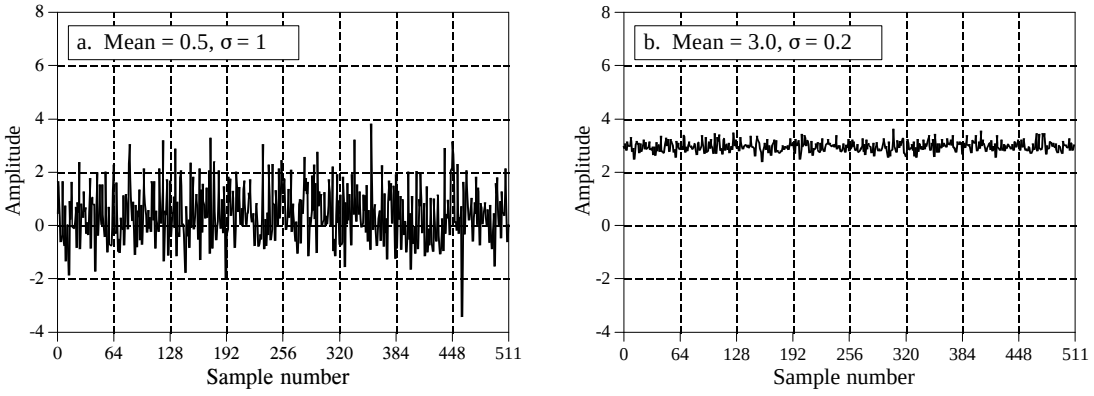

FIGURE 2-1
Examples of two digitized signals with different means and standard deviations.

zero-indexing: C, python, arduino

one-indexing: Matlab, R

keep the data organized, each sample is assigned a **sample number** or **index**. These are the numbers that appear along the horizontal axis. Two notations for assigning sample numbers are commonly used. In the first notation, the sample indexes run from 1 to N (e.g., 1 to 512). In the second notation, the sample indexes run from 0 to $N-1$ (e.g., 0 to 511). Mathematicians often use the first method (1 to N), while those in DSP commonly uses the second (0 to $N-1$). In this book, we will use the second notation. Don't dismiss this as a trivial problem. It *will* confuse you sometime during your career. Look out for it!

Mean and Standard Deviation

The **mean**, indicated by **μ** (a lower case Greek *mu*), is the statistician's jargon for the average value of a signal. It is found just as you would expect: add all of the samples together, and divide by N. It looks like this in mathematical form:

EQUATION 2-1
Calculation of a signal's mean. The signal is contained in x_0 through x_{N-1}, i is an index that runs through these values, and μ is the mean.

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i$$

In words, sum the values in the signal, x_i, by letting the index, i, run from 0 to $N-1$. Then finish the calculation by dividing the sum by N. This is identical to the equation: $\mu = (x_0 + x_1 + x_2 + \cdots + x_{N-1})/N$. If you are not already familiar with Σ (upper case Greek *sigma*) being used to indicate *summation*, study these equations carefully, and compare them with the computer program in Table 2-1. Summations of this type are abundant in DSP, and you need to understand this notation fully.

The "DC component" of a signal is its mean. In the frequency domain, this is the part of the signal with a frequency of 0 Hz.

In electronics, the *mean* is commonly called the **DC** (direct current) value. Likewise, **AC** (alternating current) refers to how the signal fluctuates around the mean value. If the signal is a simple repetitive waveform, such as a sine or square wave, its excursions can be described by its peak-to-peak amplitude. Unfortunately, most acquired signals do not show a well defined peak-to-peak value, but have a random nature, such as the signals in Fig. 2-1. A more generalized method must be used in these cases, called the **standard deviation**, denoted by σ (a lower case Greek *sigma*).

As a starting point, the expression, $|x_i - \mu|$, describes how far the i^{th} sample *deviates* (differs) from the mean. The *average deviation* of a signal is found by summing the deviations of all the individual samples, and then dividing by the number of samples, *N*. Notice that we take the absolute value of each deviation before the summation; otherwise the positive and negative terms would average to zero. The average deviation provides a single number representing the typical distance that the samples are from the mean. While convenient and straightforward, the average deviation is almost never used in statistics. This is because it doesn't fit well with the physics of how signals operate. In most cases, the important parameter is not the *deviation* from the mean, but the *power* represented by the deviation from the mean. For example, when random noise signals combine in an electronic circuit, the resultant noise is equal to the combined *power* of the individual signals, not their combined *amplitude*.

The *standard deviation* is similar to the *average deviation*, except the averaging is done with power instead of amplitude. This is achieved by squaring each of the deviations before taking the average (remember, power $\propto$ voltage^2). To finish, the *square root* is taken to compensate for the initial squaring. In equation form, the standard deviation is calculated:

EQUATION 2-2
Calculation of the standard deviation of a signal. The signal is stored in x_i, μ is the mean found from Eq. 2-1, *N* is the number of samples, and s is the standard deviation.

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i - \mu)^2$$

In the alternative notation: $\sigma = \sqrt{(x_0 - \mu)^2 + (x_1 - \mu)^2 + \cdots + (x_{N-1} - \mu)^2 / (N-1)}$. Notice that the average is carried out by dividing by $N - 1$ instead of *N*. This is a subtle feature of the equation that will be discussed in the next section. The term, σ^2, occurs frequently in statistics and is given the name **variance.** The standard deviation is a measure of how far the signal fluctuates from the mean. The variance represents the power of this fluctuation. Another term you should become familiar with is the **rms (root-mean-square)** value, frequently used in electronics. By definition, the standard deviation only measures the AC portion of a signal, while the rms value measures both the AC and DC components. If a signal has no DC component, its rms value is identical to its standard deviation. Figure 2-2 shows the relationship between the standard deviation and the peak-to-peak value of several common waveforms.

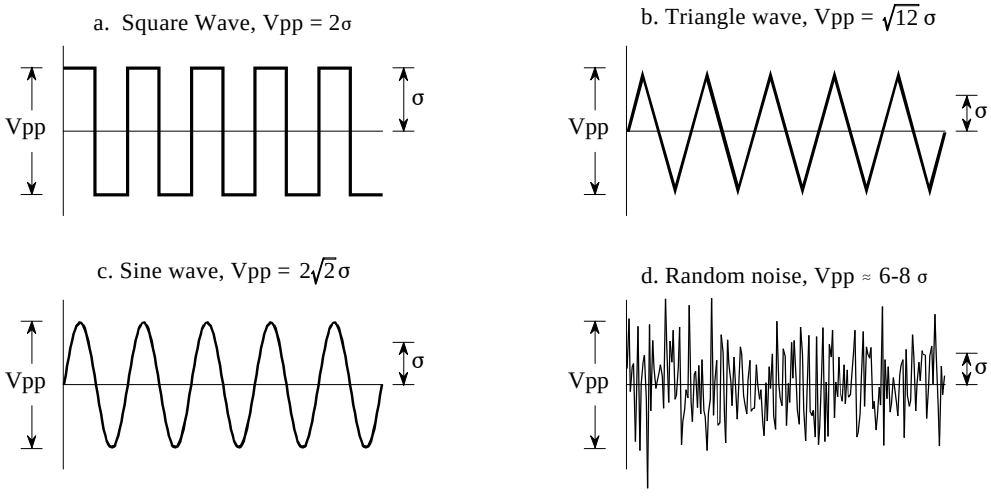

FIGURE 2-2
Ratio of the peak-to-peak amplitude to the standard deviation for several common waveforms. For the square wave, this ratio is 2; for the triangle wave it is $\sqrt{12} = 3.46$; for the sine wave it is $2\sqrt{2} = 2.83$. While random noise has no *exact* peak-to-peak value, it is *approximately* 6 to 8 times the standard deviation.

Table 2-1 lists a computer routine for calculating the mean and standard deviation using Eqs. 2-1 and 2-2. The programs in this book are intended to convey *algorithms* in the most straightforward way; all other factors are treated as secondary. Good programming techniques are disregarded if it makes the program logic more clear. For instance: a simplified version of BASIC is used, line numbers are included, the only control structure allowed is the FOR-NEXT loop, there are no I/O statements, etc. Think of these programs as an alternative way of understanding the equations used

```
100 CALCULATION OF THE MEAN AND STANDARD DEVIATION
110 '
120 DIM X[511]                    'The signal is held in X[0] to X[511]
130 N% = 512                      'N% is the number of points in the signal
140 '
150 GOSUB XXXX                    'Mythical subroutine that loads the signal into X[ ]
160 '
170 MEAN = 0                      'Find the mean via Eq. 2-1
180 FOR I% = 0 TO N%-1
190   MEAN = MEAN + X[I%]
200 NEXT I%
210 MEAN = MEAN/N%
220 '
230 VARIANCE = 0                  'Find the standard deviation via Eq. 2-2
240 FOR I% = 0 TO N%-1
250   VARIANCE = VARIANCE + ( X[I%] - MEAN )^2
260 NEXT I%
270 VARIANCE = VARIANCE/(N%-1)
280 SD = SQR(VARIANCE)
290 '
300 PRINT  MEAN  SD               'Print the calculated mean and standard deviation
310 '
320 END
```

This was ridiculous even in 1997.

TABLE 2-1

in DSP. If you can't grasp one, maybe the other will help. In BASIC, the % character at the end of a variable name indicates it is an integer. All other variables are floating point. Chapter 4 discusses these variable types in detail.

"Numerical precision" is a topic you will encounter. You need to learn about it, but in practice it is rarely relevant if you're using double-precision floats, which you should do most of the time.

This method of calculating the mean and standard deviation is adequate for many applications; however, it has two limitations. First, if the mean is much larger than the standard deviation, Eq. 2-2 involves subtracting two numbers that are very close in value. This can result in excessive round-off error in the calculations, a topic discussed in more detail in Chapter 4. Second, it is often desirable to recalculate the mean and standard deviation as new samples are acquired and added to the signal. We will call this type of calculation: **running statistics.** While the method of Eqs. 2-1 and 2-2 can be used for running statistics, it requires that *all* of the samples be involved in each new calculation. This is a very inefficient use of computational power and memory.

A solution to these problems can be found by manipulating Eqs. 2-1 and 2-2 to provide another equation for calculating the standard deviation:

EQUATION 2-3
Calculation of the standard deviation using running statistics. This equation provides the same result as Eq. 2-2, but with less round-off noise and greater computational efficiency. The signal is expressed in terms of three accumulated parameters: *N*, the total number of samples; *sum*, the sum of these samples; and *sum of squares*, the sum of the squares of the samples. The mean and standard deviation are then calculated from these three accumulated parameters.

$$\sigma^2 = \frac{1}{N-1}\left[\sum_{i=0}^{N-1} x_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} x_i\right)^2\right]$$

or using a simpler notation,

$$\sigma^2 = \frac{1}{N-1}\left[sum\ of\ squares - \frac{sum^2}{N}\right]$$

While moving through the signal, a running tally is kept of three parameters: (1) the number of samples already processed, (2) the sum of these samples, and (3) the sum of the squares of the samples (that is, square the value of each sample and add the result to the accumulated value). After any number of samples have been processed, the mean and standard deviation can be efficiently calculated using only the current value of the three parameters. Table 2-2 shows a program that reports the mean and standard deviation in this manner as each new sample is taken into account. This is the method used in hand calculators to find the statistics of a sequence of numbers. Every time you enter a number and press the Σ (summation) key, the three parameters are updated. The mean and standard deviation can then be found whenever desired, without having to recalculate the entire sequence.

```
100 'MEAN AND STANDARD DEVIATION USING RUNNING STATISTICS
110 '
120 DIM X[511]                    'The signal is held in X[0] to X[511]
130 '
140 GOSUB XXXX                    'Mythical subroutine that loads the signal into X[ ]
150 '
160 N% = 0                        'Zero the three running parameters
170 SUM = 0
180 SUMSQUARES = 0
190 '
200 FOR I% = 0 TO 511             'Loop through each sample in the signal
210   '
220   N% = N%+1                   'Update the three parameters
230   SUM = SUM + X[I%]
240   SUMSQUARES = SUMSQUARES + X[I%]^2
250   '
260   MEAN = SUM/N%               'Calculate mean and standard deviation via Eq. 2-3
270   IF N% = 1 THEN SD = 0: GOTO 300
280   SD = SQR( (SUMSQUARES - SUM^2/N%) / (N%-1) )
290   '
300   PRINT  MEAN   SD            'Print the running mean and standard deviation
310   '
320 NEXT I%
330 '
340 END
```

interesting that old people even write code in all caps

TABLE 2-2

Before ending this discussion on the mean and standard deviation, two other terms need to be mentioned. In some situations, the *mean* describes what is being measured, while the *standard deviation* represents noise and other interference. In these cases, the standard deviation is not important in itself, but only in *comparison* to the mean. This gives rise to the term: **signal-to-noise ratio (SNR)**, which is equal to the mean divided by the standard deviation. Another term is also used, the **coefficient of variation (CV)**. This is defined as the standard deviation divided by the mean, multiplied by 100 percent. For example, a signal (or other group of measure values) with a CV of 2%, has an SNR of 50. Better data means a *higher* value for the SNR and a *lower* value for the CV.

Signal vs. Underlying Process

Statistics is the science of interpreting *numerical data*, such as acquired signals. In comparison, **probability** is used in DSP to understand the *processes* that generate signals. Although they are closely related, the distinction between the **acquired signal** and the **underlying process** is key to many DSP techniques.

For example, imagine creating a 1000 point signal by flipping a coin 1000 times. If the coin flip is heads, the corresponding sample is made a value of one. On tails, the sample is set to zero. The *process* that created this signal has a mean of exactly 0.5, determined by the relative probability of each possible outcome: 50% heads, 50% tails. However, it is unlikely that the actual 1000 point signal will have a mean of exactly 0.5. Random chance

will make the number of ones and zeros slightly different each time the signal is generated. The *probabilities* of the underlying process are constant, but the *statistics* of the acquired signal change each time the experiment is repeated. This random irregularity found in actual data is called by such names as: **statistical variation**, **statistical fluctuation**, and **statistical noise.**

This presents a bit of a dilemma. When you see the terms: *mean* and *standard deviation*, how do you know if the author is referring to the statistics of an actual signal, or the probabilities of the underlying process that created the signal? Unfortunately, the only way you can tell is by the context. This is not so for all terms used in statistics and probability. For example, the *histogram* and *probability mass function* (discussed in the next section) are matching concepts that are given separate names.

Now, back to Eq. 2-2, calculation of the standard deviation. As previously mentioned, this equation divides by N-1 in calculating the average of the squared deviations, rather than simply by N. To understand why this is so, imagine that you want to find the mean and standard deviation of some *process* that generates signals. Toward this end, you acquire a signal of N samples from the process, and calculate the mean of the signal via Eq. 2.1. You can then use this as an *estimate* of the mean of the underlying process; however, you know there will be an error due to statistical noise. In particular, for random signals, the typical error between the mean of the N points, and the mean of the underlying process, is given by:

EQUATION 2-4
Typical error in calculating the mean of an underlying process by using a finite number of samples, N. The parameter, s , is the standard deviation.

$$\textit{Typical error} = \frac{\sigma}{N^{1/2}}$$

If N is small, the statistical noise in the calculated mean will be very large. In other words, you do not have access to enough data to properly characterize the process. The larger the value of N, the smaller the expected error will become. A milestone in probability theory, the *Strong Law of Large Numbers*, guarantees that the error becomes zero as N approaches infinity.

In the next step, we would like to calculate the standard deviation of the acquired signal, and use it as an estimate of the standard deviation of the underlying process. Herein lies the problem. Before you can calculate the standard deviation using Eq. 2-2, you need to already know the mean, μ. However, you don't know the mean of the underlying process, only the mean of the N point signal, *which contains an error due to statistical noise*. This error tends to reduce the calculated value of the standard deviation. To compensate for this, N is replaced by N-1. If N is large, the difference doesn't matter. If N is small, this replacement provides a more accurate

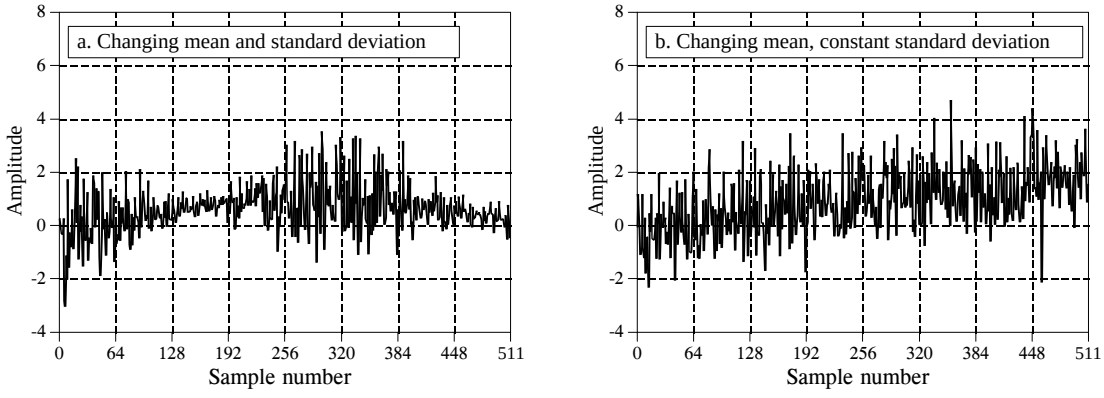

FIGURE 2-3
Examples of signals generated from nonstationary processes. In (a), both the mean and standard deviation change. In (b), the standard deviation remains a constant value of one, while the mean changes from a value of zero to two. It is a common analysis technique to break these signals into short segments, and calculate the statistics of each segment individually.

estimate of the standard deviation of the underlying process. In other words, Eq. 2-2 is an *estimate* of the standard deviation of the *underlying process*. If we divided by *N* in the equation, it would provide the standard deviation of the *acquired signal*.

As an illustration of these ideas, look at the signals in Fig. 2-3, and ask: are the variations in these signals a result of statistical noise, or is the underlying process changing? It probably isn't hard to convince yourself that these changes are too large for random chance, and must be related to the underlying process. Processes that change their characteristics in this manner are called **nonstationary**. In comparison, the signals previously presented in Fig. 2-1 were generated from a stationary process, and the variations result completely from statistical noise. Figure 2-3b illustrates a common problem with nonstationary signals: the slowly changing *mean* interferes with the calculation of the *standard deviation*. In this example, the standard deviation of the signal, over a short interval, is *one*. However, the standard deviation of the entire signal is 1.16. This error can be nearly eliminated by breaking the signal into short sections, and calculating the statistics for each section individually. If needed, the standard deviations for each of the sections can be averaged to produce a single value.

The Histogram, Pmf and Pdf

8 bits: $2^8 = 256$

Suppose we attach an 8 bit analog-to-digital converter to a computer, and acquire 256,000 samples of some signal. As an example, Fig. 2-4a shows 128 samples that might be a part of this data set. The value of each sample will be one of 256 possibilities, 0 through 255. The **histogram** displays the *number of samples* there are in the signal that have each of these *possible values*. Figure (b) shows the histogram for the 128 samples in (a). For

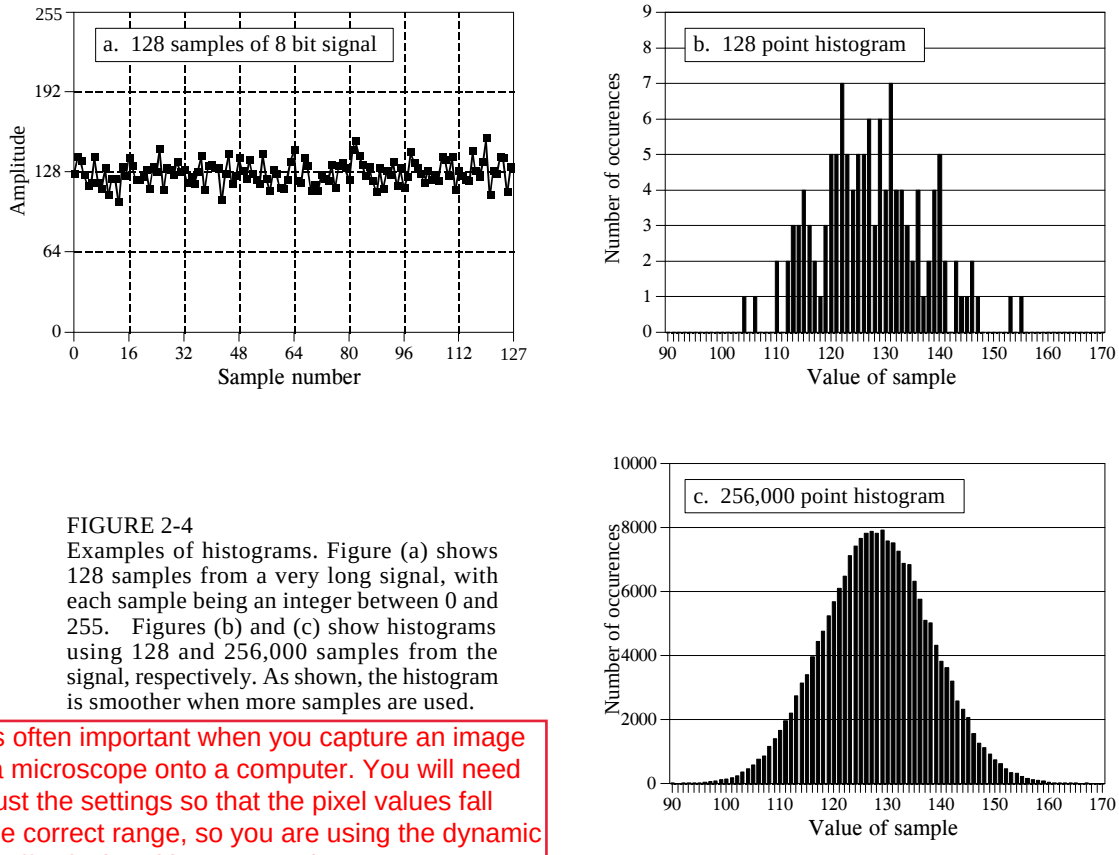

FIGURE 2-4
Examples of histograms. Figure (a) shows 128 samples from a very long signal, with each sample being an integer between 0 and 255. Figures (b) and (c) show histograms using 128 and 256,000 samples from the signal, respectively. As shown, the histogram is smoother when more samples are used.

This is often important when you capture an image from a microscope onto a computer. You will need to adjust the settings so that the pixel values fall into the correct range, so you are using the dynamic range effectively, without saturation.

example, there are 2 samples that have a value of 110, 7 samples that have a value of 131, 0 samples that have a value of 170, etc. We will represent the histogram by H_i, where i is an index that runs from 0 to M-1, and M is the number of possible values that each sample can take on. For instance, H_{50} is the number of samples that have a value of 50. Figure (c) shows the histogram of the signal using the full data set, all 256k points. As can be seen, the larger number of samples results in a much smoother appearance. Just as with the mean, the statistical noise (roughness) of the histogram is inversely proportional to the square root of the number of samples used.

From the way it is defined, the sum of all of the values in the histogram must be equal to the number of points in the signal:

EQUATION 2-5
The sum of all of the values in the histogram is equal to the number of points in the signal. In this equation, H_i is the histogram, N is the number of points in the signal, and M is the number of points in the histogram.

$$N = \sum_{i=0}^{M-1} H_i$$

The histogram can be used to efficiently calculate the mean and standard deviation of very large data sets. This is especially important for *images*, which can contain millions of samples. The histogram groups samples

together that have the same value. This allows the statistics to be calculated by working with a few groups, rather than a large number of individual samples. Using this approach, the mean and standard deviation are calculated from the histogram by the equations:

EQUATION 2-6
Calculation of the mean from the histogram. This can be viewed as combining all samples having the same value into groups, and then using Eq. 2-1 on each group.

$$\mu = \frac{1}{N} \sum_{i=0}^{M-1} i H_i$$

EQUATION 2-7
Calculation of the standard deviation from the histogram. This is the same concept as Eq. 2-2, except that all samples having the same value are operated on at once.

$$\sigma^2 = \frac{1}{N-1} \sum_{i=0}^{M-1} (i - \mu)^2 H_i$$

Table 2-3 contains a program for calculating the histogram, mean, and standard deviation using these equations. Calculation of the histogram is very fast, since it only requires indexing and incrementing. In comparison,

```
100 'CALCULATION OF THE HISTOGRAM, MEAN, AND STANDARD DEVIATION
110 '
120 DIM X%[25000]                 'X%[0] to X%[25000] holds the signal being processed
130 DIM H%[255]                   'H%[0] to H%[255] holds the histogram
140 N% = 25001                    'Set the number of points in the signal
150 '
160 FOR I% = 0 TO 255             'Zero the histogram, so it can be used as an accumulator
170   H%[I%] = 0
180 NEXT I%
190 '
200 GOSUB XXXX                    'Mythical subroutine that loads the signal into X%[ ]
210 '
220 FOR I% = 0 TO 25000 'Calculate the histogram for 25001 points
230   H%[ X%[I%] ] = H%[ X%[I%] ] + 1
240 NEXT I%
250 '
260 MEAN = 0                      'Calculate the mean via Eq. 2-6
270 FOR I% = 0 TO 255
280   MEAN = MEAN + I% * H%[I%]
290 NEXT I%
300 MEAN = MEAN / N%
310 '
320 VARIANCE = 0                  'Calculate the standard deviation via Eq. 2-7
330 FOR I% = 0 TO 255
340   VARIANCE = VARIANCE + H%[I%] * (I%-MEAN)^2
350 NEXT I%
360 VARIANCE = VARIANCE / (N%-1)
370 SD = SQR(VARIANCE)
380 '
390 PRINT  MEAN  SD               'Print the calculated mean and standard deviation.
400 '
410 END
```

TABLE 2-3

calculating the mean and standard deviation requires the time consuming operations of addition and multiplication. The strategy of this algorithm is to use these slow operations only on the few numbers in the histogram, not the many samples in the signal. This makes the algorithm much faster than the previously described methods. Think a factor of ten for very long signals with the calculations being performed on a general purpose computer.

The notion that the acquired signal is a noisy version of the underlying process is very important; so important that some of the concepts are given different names. The *histogram* is what is formed from an acquired signal. The corresponding curve for the underlying process is called the **probability mass function (pmf)**. A histogram is always calculated using a finite number of samples, while the pmf is what *would be* obtained with an infinite number of samples. The pmf can be estimated (inferred) from the histogram, or it may be deduced by some mathematical technique, such as in the coin flipping example.

Figure 2-5 shows an example pmf, and one of the possible histograms that could be associated with it. The key to understanding these concepts rests in the units of the vertical axis. As previously described, the vertical axis of the histogram is the *number of times* that a particular value occurs in the signal. The vertical axis of the pmf contains similar information, except expressed on a *fractional basis*. In other words, each value in the histogram is divided by the total number of samples to approximate the pmf. This means that each value in the pmf must be between zero and one, and that the sum of all of the values in the pmf will be equal to one.

The pmf is important because it describes the *probability* that a certain value will be generated. For example, imagine a signal with the pmf of Fig. 2-5b, such as previously shown in Fig. 2-4a. What is the probability that a sample taken from this signal will have a value of 120? Figure 2-5b provides the answer, 0.03, or about 1 chance in 34. What is the probability that a randomly chosen sample will have a value greater than 150? Adding up the values in the pmf for: 151, 152, 153,···, 255, provides the answer, 0.0122, or about 1 chance in 82. Thus, the signal would be expected to have a value exceeding 150 on an average of every 82 points. What is the probability that any one sample will be between 0 and 255? Summing all of the values in the pmf produces the probability of 1.00, that is, a certainty that this will occur.

The histogram and pmf can only be used with discrete data, such as a digitized signal residing in a computer. A similar concept applies to continuous signals, such as voltages appearing in analog electronics. The **probability density function (pdf)**, also called the **probability distribution function**, is to continuous signals what the probability mass function is to discrete signals. For example, imagine an analog signal passing through an analog-to-digital converter, resulting in the digitized signal of Fig. 2-4a. For simplicity, we will assume that voltages between 0 and 255 millivolts become digitized into digital numbers between 0 and 255. The pmf of this digital

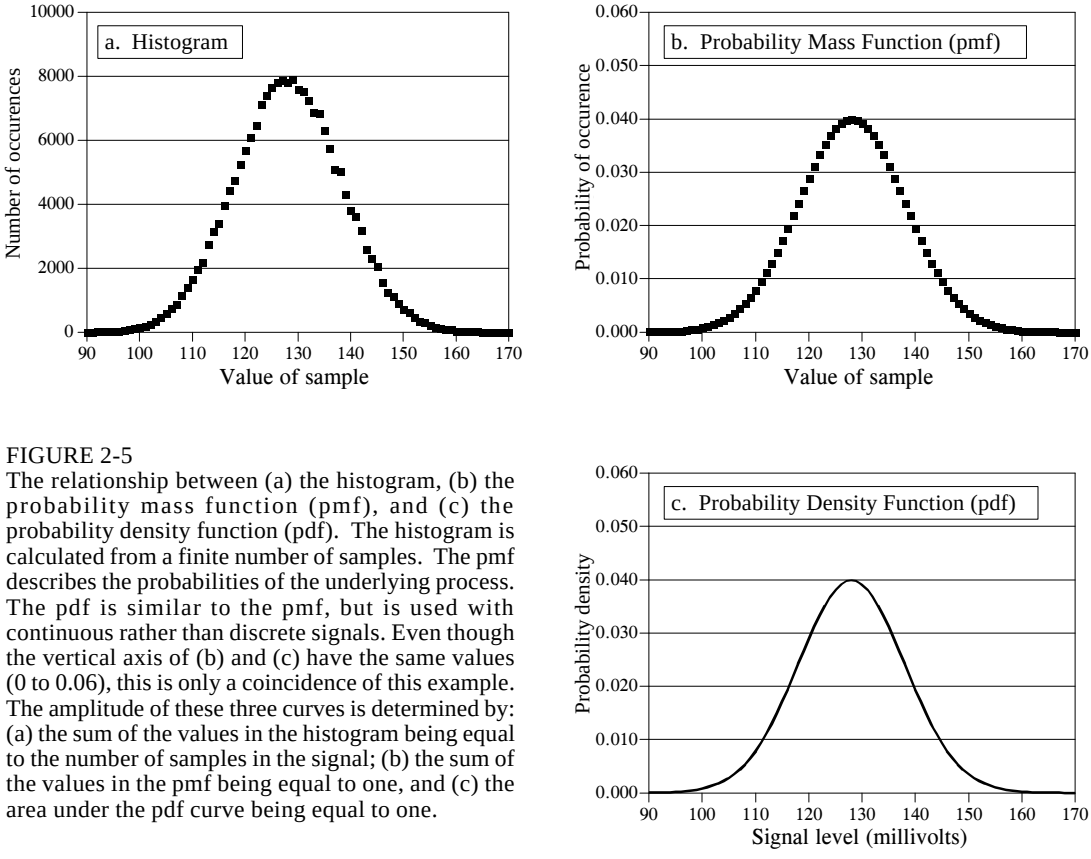

FIGURE 2-5
The relationship between (a) the histogram, (b) the probability mass function (pmf), and (c) the probability density function (pdf). The histogram is calculated from a finite number of samples. The pmf describes the probabilities of the underlying process. The pdf is similar to the pmf, but is used with continuous rather than discrete signals. Even though the vertical axis of (b) and (c) have the same values (0 to 0.06), this is only a coincidence of this example. The amplitude of these three curves is determined by: (a) the sum of the values in the histogram being equal to the number of samples in the signal; (b) the sum of the values in the pmf being equal to one, and (c) the area under the pdf curve being equal to one.

signal is shown by the *markers* in Fig. 2-5b. Similarly, the pdf of the analog signal is shown by the *continuous line* in (c), indicating the signal can take on a continuous range of values, such as the voltage in an electronic circuit.

The vertical axis of the pdf is in units of **probability density**, rather than just probability. For example, a pdf of 0.03 at 120.5 *does not* mean that the a voltage of 120.5 millivolts will occur 3% of the time. In fact, the probability of the continuous signal being exactly 120.5 millivolts is infinitesimally small. This is because there are an infinite number of possible values that the signal needs to divide its time between: 120.49997, 120.49998, 120.49999, etc. The chance that the signal happens to be exactly 120.50000··· is very remote indeed!

To calculate a *probability*, the *probability density* is multiplied by a *range* of values. For example, the probability that the signal, at any given instant, will be between the values of 120 and 121 is: (121−120) × 0.03 = 0.03. The probability that the signal will be between 120.4 and 120.5 is: (120.5−120.4) × 0.03 = 0.003, etc. If the pdf is not constant over the range of interest, the multiplication becomes the integral of the pdf over that range. In other words, the area under the pdf bounded by the specified values. Since the value of the signal must always be *something*, the total area under the pdf

curve, the integral from $-\infty$ to $+\infty$, will always be equal to one. This is analogous to the sum of all of the pmf values being equal to one, and the sum of all of the histogram values being equal to *N*.

The histogram, pmf, and pdf are very similar concepts. Mathematicians always keep them straight, but you will frequently find them used interchangeably (and therefore, incorrectly) by many scientists and

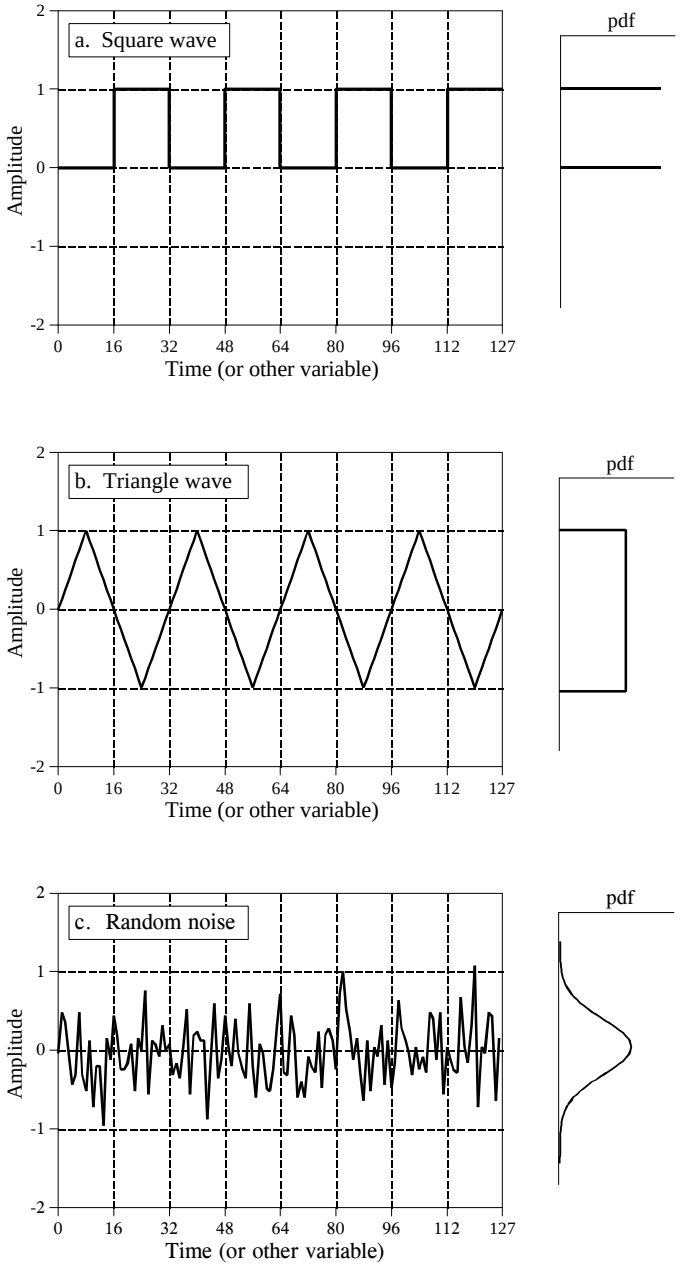

FIGURE 2-6
Three common waveforms and their probability density functions. As in these examples, the pdf graph is often rotated one-quarter turn and placed at the side of the signal it describes. The pdf of a square wave, shown in (a), consists of two infinitesimally narrow spikes, corresponding to the signal only having two possible values. The pdf of the triangle wave, (b), has a constant value over a range, and is often called a *uniform* distribution. The pdf of random noise, as in (c), is the most interesting of all, a bell shaped curve known as a *Gaussian*.

engineers. Figure 2-6 shows three *continuous* waveforms and their *pdfs*. If these were *discrete signals*, signified by changing the horizontal axis labeling to "sample number," *pmfs* would be used.

Data is almost always acquired as integers (ints) or unsigned ints (uints), but calculations are usually done using floating point variables (floats or doubles).

A relevant consequence of this is that floating point variables often require much more memory to represent the same data (we will discuss that in greater depth later on).

A problem occurs in calculating the histogram when the number of levels each sample can take on is much larger than the number of samples in the signal. This is always true for signals represented in *floating point* notation, where each sample is stored as a fractional value. For example, integer representation might require the sample value to be 3 or 4, while floating point allows millions of possible fractional values *between* 3 and 4. The previously described approach for calculating the histogram involves counting the number of samples that have each of the possible quantization levels. This is not possible with floating point data because there are *billions* of possible levels that would have to be taken into account. Even worse, nearly all of these possible levels would have no samples that correspond to them. For example, imagine a 10,000 sample signal, with each sample having one billion possible values. The conventional histogram would consist of one billion data points, with all but about 10,000 of them having a value of zero.

The solution to these problems is a technique called **binning**. This is done by arbitrarily selecting the length of the histogram to be some convenient number, such as 1000 points, often called **bins**. The value of each bin represents the total number of samples in the signal that have a value within a *certain range*. For example, imagine a floating point signal that contains values between 0.0 and 10.0, and a histogram with 1000 bins. Bin 0 in the histogram is the number of samples in the signal with a value between 0 and 0.01, bin 1 is the number of samples with a value between 0.01 and 0.02, and so forth, up to bin 999 containing the number of samples with a value between 9.99 and 10.0. Table 2-4 presents a program for calculating a binned histogram in this manner.

```
100 'CALCULATION OF BINNED HISTOGRAM
110 '
120 DIM X[25000]                    'X[0] to X[25000] holds the floating point signal,
130 '                               'with each sample having a value between 0.0 and 10.0.
140 DIM H%[999]                     'H%[0] to H%[999] holds the binned histogram
150 '
160 FOR I% = 0 TO 999               'Zero the binned histogram for use as an accumulator
170   H%[I%] = 0
180 NEXT I%
190 '
200 GOSUB XXXX                      'Mythical subroutine that loads the signal into X%[ ]
210 '
220 FOR I% = 0 TO 25000 '           'Calculate the binned histogram for 25001 points
230   BINNUM% = INT( X[I%] * 100 )
240   H%[ BINNUM%] = H%[ BINNUM%] + 1
250 NEXT I%
260 '
270 END
```

TABLE 2-4

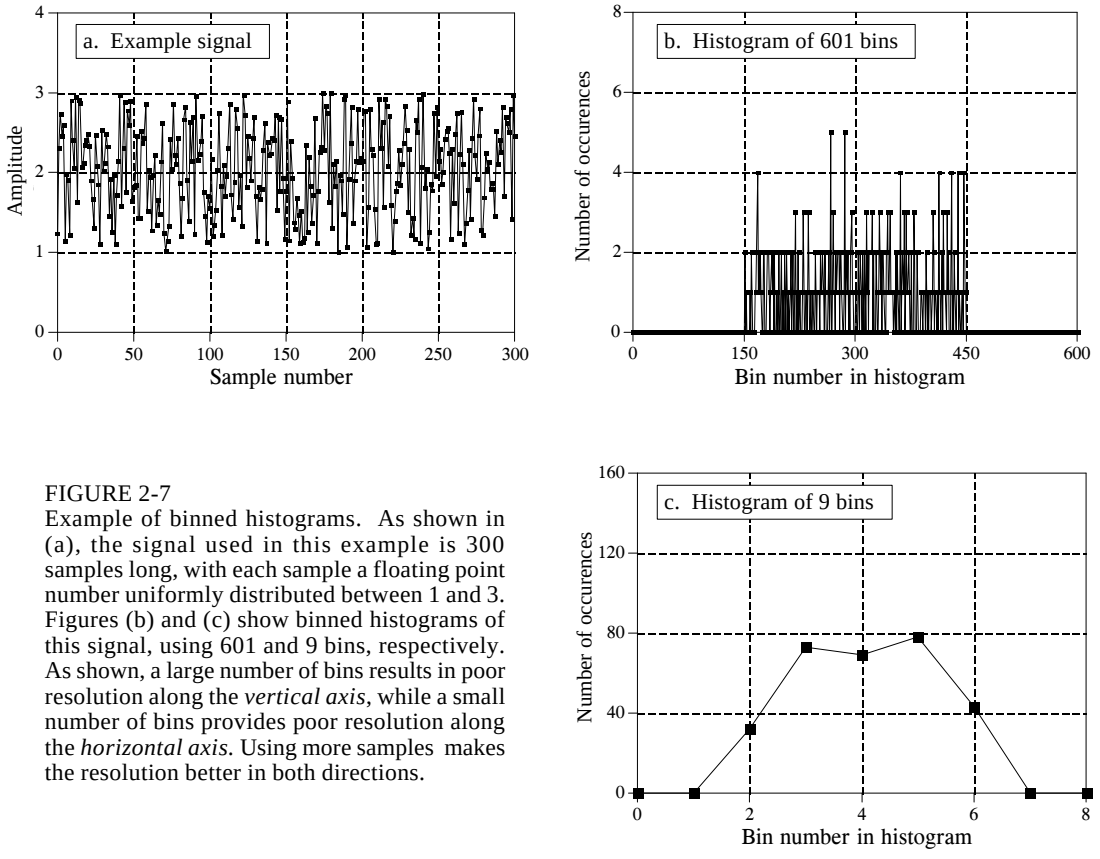

FIGURE 2-7
Example of binned histograms. As shown in (a), the signal used in this example is 300 samples long, with each sample a floating point number uniformly distributed between 1 and 3. Figures (b) and (c) show binned histograms of this signal, using 601 and 9 bins, respectively. As shown, a large number of bins results in poor resolution along the *vertical axis*, while a small number of bins provides poor resolution along the *horizontal axis*. Using more samples makes the resolution better in both directions.

How many bins should be used? This is a compromise between two problems. As shown in Fig. 2-7, too many bins makes it difficult to estimate the *amplitude* of the underlying pmf. This is because only a few samples fall into each bin, making the statistical noise very high. At the other extreme, too few of bins makes it difficult to estimate the underlying pmf in the *horizontal* direction. In other words, the number of bins controls a tradeoff between resolution along the y-axis, and resolution along the x-axis.

The Normal Distribution

Signals formed from random processes usually have a bell shaped pdf. This is called a **normal distribution**, a **Gauss distribution**, or a **Gaussian**, after the great German mathematician, Karl Friedrich Gauss (1777-1855). The reason why this curve occurs so frequently in nature will be discussed shortly in conjunction with *digital noise generation*. The basic shape of the curve is generated from a *negative squared exponent*:

$$y(x) = e^{-x^2}$$

This raw curve can be converted into the complete Gaussian by adding an adjustable mean, μ, and standard deviation, σ. In addition, the equation must be normalized so that the total area under the curve is equal to *one*, a requirement of all probability distribution functions. This results in the general form of the normal distribution, one of the most important relations in statistics and probability:

EQUATION 2-8
Equation for the *normal distribution*, also called the *Gauss distribution*, or simply a *Gaussian*. In this relation, $P(x)$ is the probability distribution function, μ is the mean, and s is the standard deviation.

$$P(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-(x-\mu)^2/2\sigma^2}$$

Figure 2-8 shows several examples of Gaussian curves with various means and standard deviations. The *mean* centers the curve over a particular value, while the *standard deviation* controls the width of the bell shape.

An interesting characteristic of the Gaussian is that the *tails* drop toward zero very rapidly, much faster than with other common functions such as decaying exponentials or $1/x$. For example, at two, four, and six standard

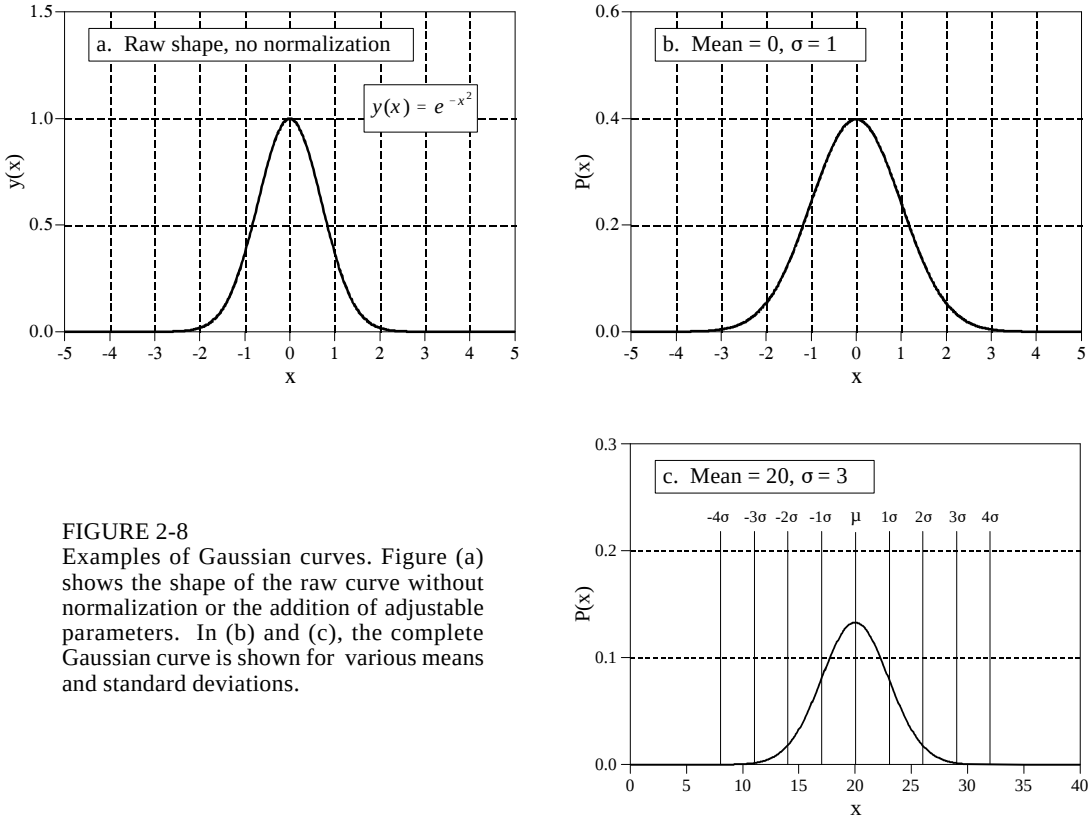

FIGURE 2-8
Examples of Gaussian curves. Figure (a) shows the shape of the raw curve without normalization or the addition of adjustable parameters. In (b) and (c), the complete Gaussian curve is shown for various means and standard deviations.

deviations from the mean, the value of the Gaussian curve has dropped to about 1/19, 1/7563, and 1/166,666,666, respectively. This is why normally distributed signals, such as illustrated in Fig. 2-6c, *appear* to have an approximate peak-to-peak value. In principle, signals of this type can experience excursions of unlimited amplitude. In practice, the sharp drop of the Gaussian pdf dictates that these extremes almost never occur. This results in the waveform having a relatively bounded appearance with an apparent peak-to-peak amplitude of about 6-8σ.

As previously shown, the integral of the pdf is used to find the probability that a signal will be within a certain range of values. This makes the integral of the pdf important enough that it is given its own name, the **cumulative distribution function (cdf)**. An especially obnoxious problem with the Gaussian is that it cannot be integrated using elementary methods. To get around this, the integral of the Gaussian can be calculated by *numerical integration*. This involves sampling the continuous Gaussian curve very finely, say, a few million points between -10σ and +10σ. The samples in this discrete signal are then *added* to simulate *integration*. The discrete curve resulting from this simulated integration is then stored in a table for use in calculating probabilities.

The cdf of the normal distribution is shown in Fig. 2-9, with its numeric values listed in Table 2-5. Since this curve is used so frequently in probability, it is given its own symbol: $\Phi(x)$ (upper case Greek *phi*). For example, $\Phi(-2)$ has a value of 0.0228. This indicates that there is a 2.28% probability that the value of the signal will be between -∞ and two standard deviations below the mean, at any randomly chosen time. Likewise, the value: $\Phi(1) = 0.8413$, means there is an 84.13% chance that the value of the signal, at a randomly selected instant, will be between -∞ and one standard deviation above the mean. To calculate the probability that the signal will be will be *between* two values, it is necessary to subtract the appropriate numbers found in the $\Phi(x)$ table. For example, the probability that the value of the signal, at some randomly chosen time, will be between two standard deviations below the mean and one standard deviation above the mean, is given by: $\Phi(1) - \Phi(-2) = 0.8185$, or 81.85%

Using this method, samples taken from a normally distributed signal will be within ±1σ of the mean about 68% of the time. They will be within ±2σ about 95% of the time, and within ±3σ about 99.75% of the time. The probability of the signal being more than 10 standard deviations from the mean is so minuscule, it would be expected to occur for only a few *microseconds* since the beginning of the universe, about 10 billion years!

Equation 2-8 can also be used to express the probability mass function of normally distributed *discrete* signals. In this case, x is restricted to be one of the quantized levels that the signal can take on, such as one of the 4096 binary values exiting a 12 bit analog-to-digital converter. Ignore the $1/\sqrt{2\pi}\sigma$ term, it is only used to make the total area under the pdf curve equal to *one*. Instead, you must include whatever term is needed to make the sum of all the values in the pmf equal to *one*. In most cases, this is done by

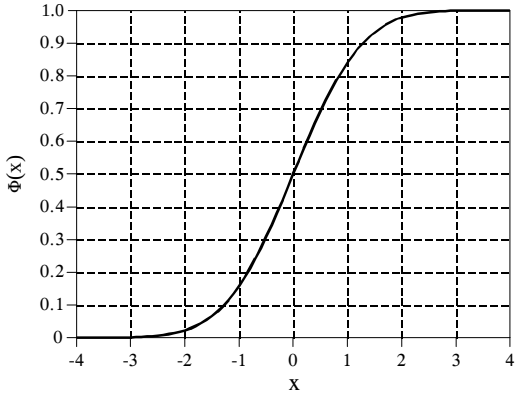

FIGURE 2-9 & TABLE 2-5
$\Phi(x)$, the cumulative distribution function of the normal distribution (mean = 0, standard deviation = 1). These values are calculated by numerically integrating the normal distribution shown in Fig. 2-8b. In words, $\Phi(x)$ is the probability that the value of a normally distributed signal, at some randomly chosen time, will be less than x. In this table, the value of x is expressed in units of standard deviations referenced to the mean.

x	Φ(x)	x	Φ(x)
-3.4	.0003	0.0	.5000
-3.3	.0005	0.1	.5398
-3.2	.0007	0.2	.5793
-3.1	.0010	0.3	.6179
-3.0	.0013	0.4	.6554
-2.9	.0019	0.5	.6915
-2.8	.0026	0.6	.7257
-2.7	.0035	0.7	.7580
-2.6	.0047	0.8	.7881
-2.5	.0062	0.9	.8159
-2.4	.0082	1.0	.8413
-2.3	.0107	1.1	.8643
-2.2	.0139	1.2	.8849
-2.1	.0179	1.3	.9032
-2.0	.0228	1.4	.9192
-1.9	.0287	1.5	.9332
-1.8	.0359	1.6	.9452
-1.7	.0446	1.7	.9554
-1.6	.0548	1.8	.9641
-1.5	.0668	1.9	.9713
-1.4	.0808	2.0	.9772
-1.3	.0968	2.1	.9821
-1.2	.1151	2.2	.9861
-1.1	.1357	2.3	.9893
-1.0	.1587	2.4	.9918
-0.9	.1841	2.5	.9938
-0.8	.2119	2.6	.9953
-0.7	.2420	2.7	.9965
-0.6	.2743	2.8	.9974
-0.5	.3085	2.9	.9981
-0.4	.3446	3.0	.9987
-0.3	.3821	3.1	.9990
-0.2	.4207	3.2	.9993
-0.1	.4602	3.3	.9995
0.0	.5000	3.4	.9997

generating the curve without worrying about normalization, summing all of the unnormalized values, and then dividing all of the values by the sum.

Digital Noise Generation

Random noise is an important topic in both electronics and DSP. For example, it limits how small of a signal an instrument can measure, the distance a radio system can communicate, and how much radiation is required to produce an x-ray image. A common need in DSP is to generate signals that resemble various types of random noise. This is required to test the performance of algorithms that must *work* in the presence of noise.

The heart of digital noise generation is the **random number generator**. Most programming languages have this as a standard function. The *BASIC* statement: *X = RND*, loads the variable, *X*, with a new random number each time the command is encountered. Each random number has a value between zero and one, with an equal probability of being anywhere between these two extremes. Figure 2-10a shows a signal formed by taking 128 samples from this type of random number generator. The mean of the underlying process that generated this signal is 0.5, the standard deviation is $1/\sqrt{12} = 0.29$, and the distribution is uniform between zero and one.

Algorithms need to be tested using the same kind of data they will encounter in actual operation. This creates the need to generate digital noise with a *Gaussian* pdf. There are two methods for generating such signals using a random number generator. Figure 2-10 illustrates the first method. Figure (b) shows a signal obtained by adding two random numbers to form each sample, i.e., $X = RND+RND$. Since each of the random numbers can run from zero to one, the sum can run from zero to two. The mean is now *one*, and the standard deviation is $1/\sqrt{6}$ (remember, when independent random signals are added, the variances also add). As shown, the pdf has changed from a *uniform* distribution to a *triangular* distribution. That is, the signal spends more of its time around a value of *one*, with less time spent near *zero* or *two*.

Figure (c) takes this idea a step further by adding twelve random numbers to produce each sample. The mean is now *six*, and the standard deviation is *one*. What is most important, the pdf has virtually become a *Gaussian*. This procedure can be used to create a normally distributed noise signal with an arbitrary mean and standard deviation. For each sample in the signal: (1) add twelve random numbers, (2) subtract six to make the mean equal to zero, (3) multiply by the standard deviation desired, and (4) add the desired mean.

The mathematical basis for this algorithm is contained in the **Central Limit Theorem**, one of the most important concepts in probability. In its simplest form, the Central Limit Theorem states that a *sum* of random numbers becomes normally distributed as more and more of the random numbers are added together. The Central Limit Theorem *does not* require the individual random numbers be from any particular distribution, or even that the random numbers be from the *same* distribution. The Central Limit Theorem provides the reason why normally distributed signals are seen so widely in nature. Whenever many different random forces are interacting, the resulting pdf becomes a Gaussian.

In the second method for generating normally distributed random numbers, the random number generator is invoked *twice*, to obtain R_1 and R_2. A normally distributed random number, X, can then be found:

EQUATION 2-9
Generation of normally distributed random numbers. R_1 and R_2 are random numbers with a uniform distribution between zero and one. This results in X being normally distributed with a mean of zero, and a standard deviation of one. The log is base *e*, and the cosine is in radians.

$$X = (-2 \log R_1)^{1/2} \cos(2\pi R_2)$$

Just as before, this approach can generate normally distributed random signals with an arbitrary mean and standard deviation. Take each number generated by this equation, multiply it by the desired standard deviation, and add the desired mean.

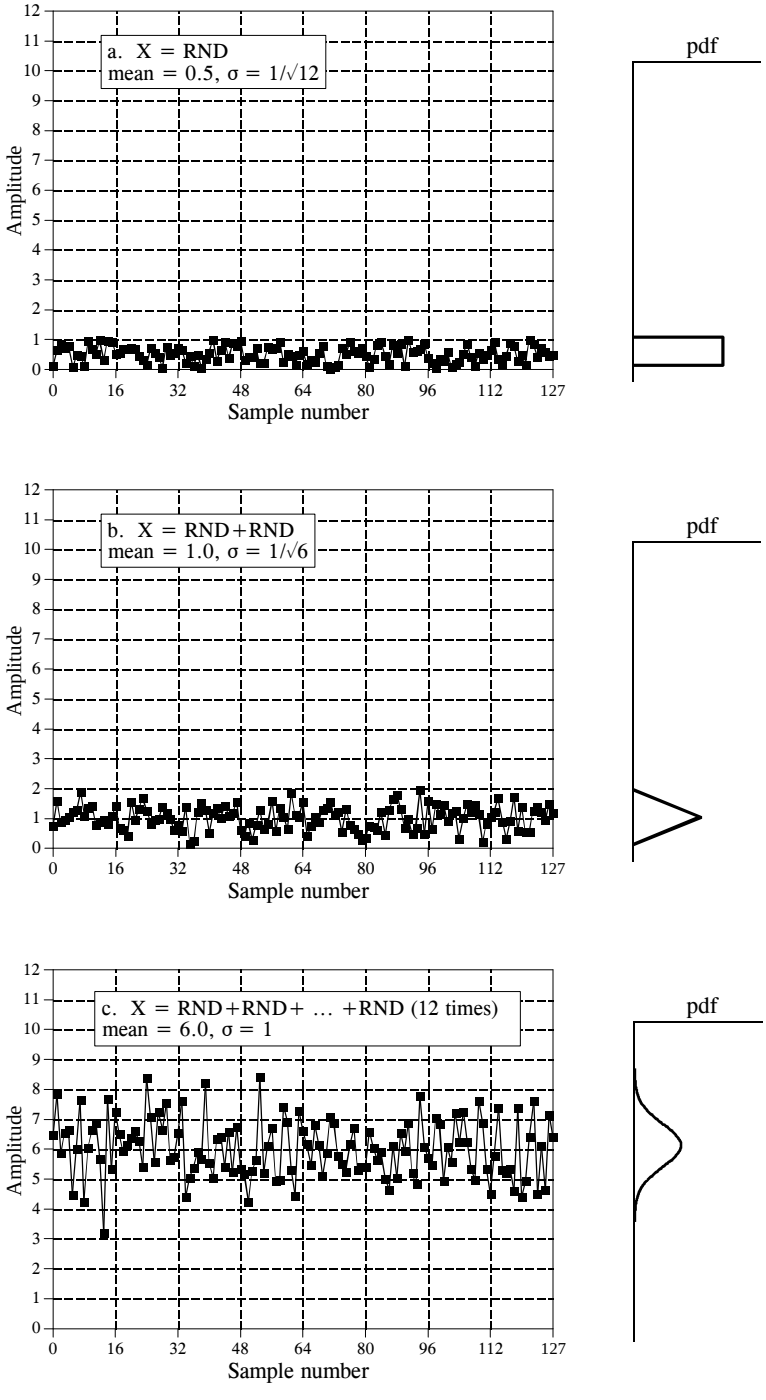

FIGURE 2-10
Converting a uniform distribution to a Gaussian distribution. Figure (a) shows a signal where each sample is generated by a random number generator. As indicated by the pdf, the value of each sample is uniformly distributed between zero and one. Each sample in (b) is formed by adding *two* values from the random number generator. In (c), each sample is created by adding *twelve* values from the random number generator. The pdf of (c) is very nearly Gaussian, with a mean of *six*, and a standard deviation of *one*.

Random number generators operate by starting with a **seed**, a number between zero and one. When the random number generator is invoked, the seed is passed through a fixed algorithm, resulting in a new number between zero and one. This new number is reported as the *random number*, and is then internally stored to be used as the seed the next time the random number generator is called. The algorithm that transforms the seed into the new random number is often of the form:

EQUATION 2-10
Common algorithm for generating uniformly distributed random numbers between zero and one. In this method, S is the seed, R is the new random number, and a,b,& c are appropriately chosen constants. In words, the quantity $aS+b$ is divided by c, and the remainder is taken as R.

$$R = (aS + b) \text{ modulo } c$$

In this manner, a continuous sequence of random numbers can be generated, all starting from the same seed. This allows a program to be run multiple times using exactly the same random number sequences. If you want the random number sequence to change, most languages have a provision for **reseeding** the random number generator, allowing you to choose the number first used as the seed. A common technique is to use the *time* (as indicated by the system's clock) as the seed, thus providing a new sequence each time the program is run.

From a pure mathematical view, the numbers generated in this way cannot be absolutely random since each number is fully determined by the *previous* number. The term **pseudo-random** is often used to describe this situation. However, this is not something you should be concerned with. The sequences generated by random number generators are statistically random to an exceedingly high degree. It is very unlikely that you will encounter a situation where they are not adequate.

Precision and Accuracy

Precision and accuracy are terms used to describe systems and methods that *measure*, *estimate*, or *predict*. In all these cases, there is some parameter you wish to know the value of. This is called the **true value**, or simply, **truth**. The method provides a **measured value**, that you want to be as close to the true value as possible. *Precision* and *accuracy* are ways of describing the error that can exist between these two values.

Unfortunately, precision and accuracy are used interchangeably in non-technical settings. In fact, dictionaries define them by referring to each other! In spite of this, science and engineering have very specific definitions for each. You should make a point of using the terms correctly, and quietly tolerate others when they use them incorrectly.

As an example, consider an oceanographer measuring water depth using a *sonar* system. Short bursts of sound are transmitted from the ship, reflected from the ocean floor, and received at the surface as an echo. Sound waves travel at a relatively constant velocity in water, allowing the depth to be found from the elapsed time between the transmitted and received pulses. As with all empirical measurements, a certain amount of error exists between the measured and true values. This particular measurement could be affected by many factors: random noise in the electronics, waves on the ocean surface, plant growth on the ocean floor, variations in the water temperature causing the sound velocity to change, etc.

To investigate these effects, the oceanographer takes many successive readings at a location known to be *exactly* 1000 meters deep (the true value). These measurements are then arranged as the histogram shown in Fig. 2-11. As would be expected from the Central Limit Theorem, the acquired data are normally distributed. The *mean* occurs at the center of the distribution, and represents the best estimate of the depth based on all of the measured data. The *standard deviation* defines the width of the distribution, describing how much variation occurs between successive measurements.

This situation results in two general types of error that the system can experience. First, the mean may be shifted from the true value. The amount of this shift is called the **accuracy** of the measurement. Second, individual measurements may not agree well with each other, as indicated by the width of the distribution. This is called the **precision** of the measurement, and is expressed by quoting the standard deviation, the signal-to-noise ratio, or the CV.

Consider a measurement that has good accuracy, but poor precision; the histogram is centered over the true value, but is very broad. Although the measurements are correct *as a group*, each individual reading is a poor measure of the true value. This situation is said to have poor *repeatability;* measurements taken in succession don't agree well. Poor precision results from **random errors**. This is the name given to errors that change each

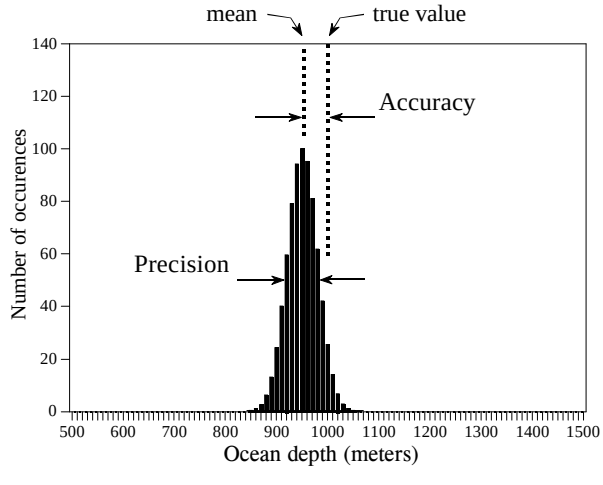

FIGURE 2-11
Definitions of accuracy and precision. Accuracy is the difference between the true value and the mean of the under-lying process that generates the data. Precision is the spread of the values, specified by the standard deviation, the signal-to-noise ratio, or the CV.

time the measurement is repeated. Averaging several measurements will *always* improve the precision. In short, *precision is a measure of random noise.*

Now, imagine a measurement that is very precise, but has poor accuracy. This makes the histogram very slender, but not centered over the true value. Successive readings are close in value; however, they *all* have a large error. Poor accuracy results from **systematic errors.** These are errors that become repeated in exactly the same manner each time the measurement is conducted. Accuracy is usually dependent on how you *calibrate* the system. For example, in the ocean depth measurement, the parameter directly measured is elapsed time. This is converted into depth by a calibration procedure that relates *milliseconds* to *meters*. This may be as simple as multiplying by a fixed velocity, or as complicated as dozens of second order corrections. Averaging individual measurements does nothing to improve the accuracy. In short, *accuracy is a measure of calibration.*

In actual practice there are many ways that precision and accuracy can become intertwined. For example, imagine building an electronic amplifier from 1% resistors. This tolerance indicates that the value of each resistor will be within 1% of the stated value over a wide range of conditions, such as temperature, humidity, age, etc. This error in the resistance will produce a corresponding error in the gain of the amplifier. Is this error a problem of accuracy or precision?

The answer depends on how you take the measurements. For example, suppose you build *one* amplifier and test it several times over a few minutes. The error in gain remains constant with each test, and you conclude the problem is *accuracy*. In comparison, suppose you build *one thousand* of the amplifiers. The gain from device to device will fluctuate randomly, and the problem appears to be one of *precision*. Likewise, any one of these amplifiers will show gain fluctuations in response to temperature and other environmental changes. Again, the problem would be called *precision*.

When deciding which name to call the problem, ask yourself two questions. First: Will averaging successive readings provide a better measurement? If yes, call the error precision; if no, call it accuracy. Second: Will calibration correct the error? If yes, call it accuracy; if no, call it precision. This may require some thought, especially related to how the device will be calibrated, and how often it will be done.