CHAPTER

11

Fourier Transform Pairs

For every *time domain* waveform there is a corresponding *frequency domain* waveform, and vice versa. For example, a rectangular pulse in the time domain coincides with a sinc function [i.e., sin(x)/x] in the frequency domain. Duality provides that the reverse is also true; a rectangular pulse in the frequency domain matches a sinc function in the time domain. Waveforms that correspond to each other in this manner are called *Fourier transform pairs*. Several common pairs are presented in this chapter.

Delta Function Pairs

For discrete signals, the delta function is a simple waveform, and has an equally simple Fourier transform pair. Figure 11-1a shows a delta function in the time domain, with its frequency spectrum in (b) and (c). The magnitude is a constant value, while the phase is entirely zero. As discussed in the last chapter, this can be understood by using the expansion/compression property. When the time domain is compressed until it becomes an impulse, the frequency domain is expanded until it becomes a constant value.

In (d) and (g), the time domain waveform is shifted four and eight samples to the right, respectively. As expected from the properties in the last chapter, shifting the time domain waveform does not affect the magnitude, but adds a linear component to the phase. The phase signals in this figure have not been *unwrapped*, and thus extend only from $-\pi$ to π. Also notice that the horizontal axes in the frequency domain run from -0.5 to 0.5. That is, they show the *negative* frequencies in the spectrum, as well as the *positive* ones. The negative frequencies are redundant information, but they are often included in DSP graphs and you should become accustomed to seeing them.

Figure 11-2 presents the same information as Fig. 11-1, but with the frequency domain in *rectangular form*. There are two lessons to be learned here. First, compare the polar and rectangular representations of the

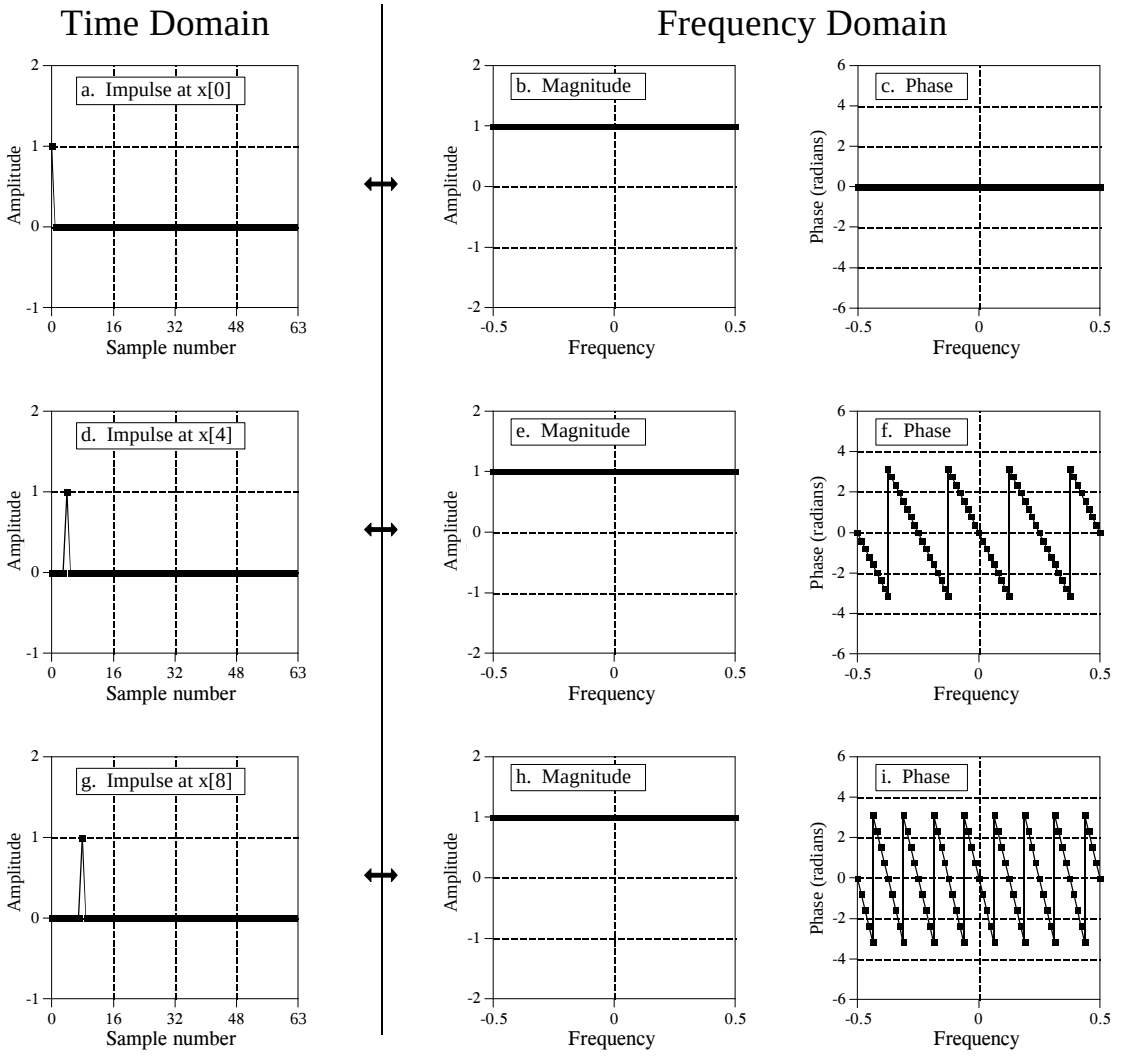

FIGURE 11-1
Delta function pairs in *polar form*. An impulse in the time domain corresponds to a constant magnitude and a linear phase in the frequency domain.

frequency domains. As is usually the case, the polar form is much easier to understand; the magnitude is nothing more than a constant, while the phase is a straight line. In comparison, the real and imaginary parts are sinusoidal oscillations that are difficult to attach a meaning to.

The second interesting feature in Fig. 11-2 is the *duality* of the DFT. In the conventional view, each sample in the DFT's frequency domain corresponds to a sinusoid in the time domain. However, the reverse of this is also true, each sample in the time domain corresponds to sinusoids in the frequency domain. Including the negative frequencies in these graphs allows the duality property to be more symmetrical. For instance, Figs. (d), (e), and

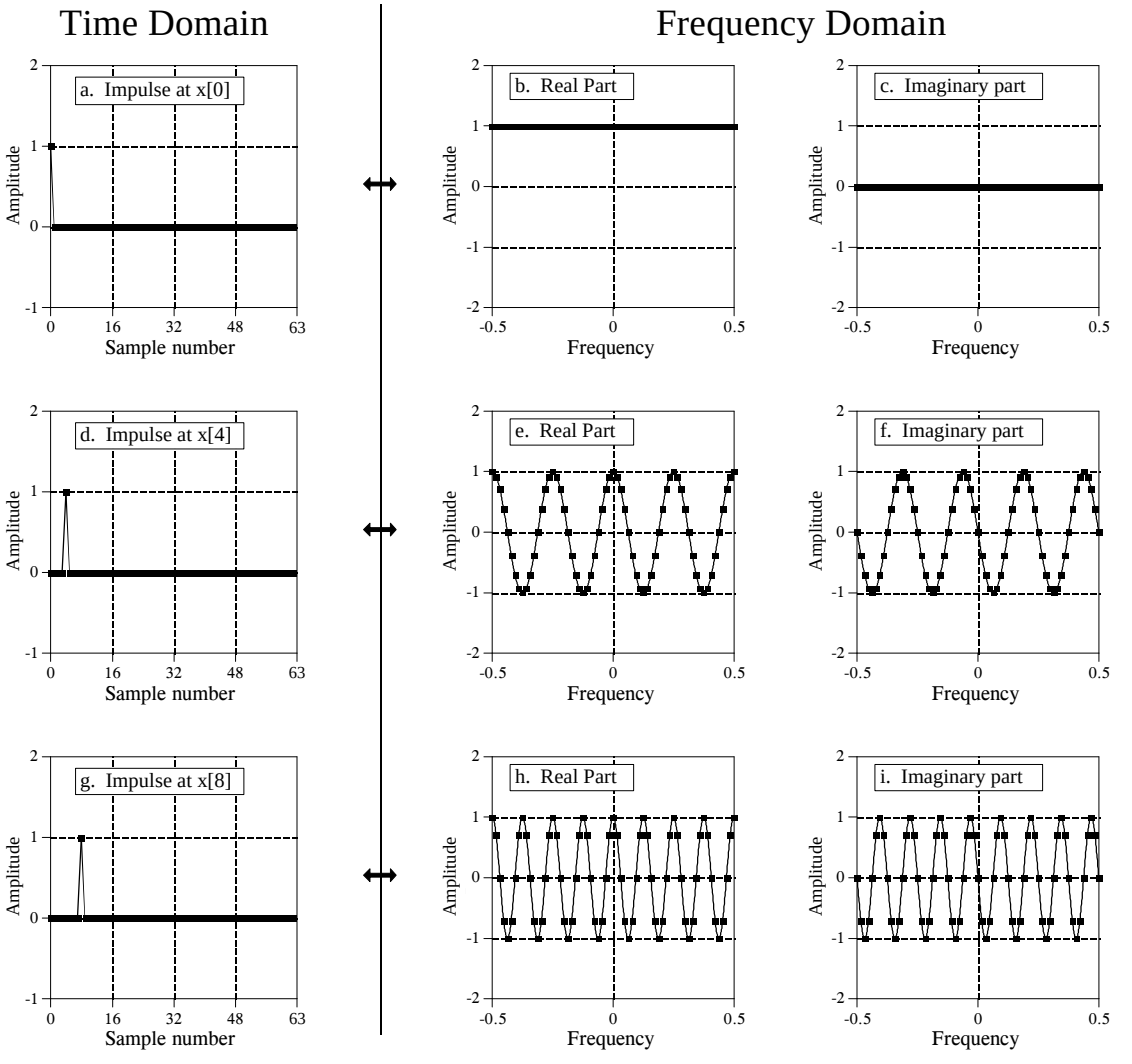

FIGURE 11-2
Delta function pairs in *rectangular form*. Each sample in the time domain results in a cosine wave in the real part, and a negative sine wave in the imaginary part of the frequency domain.

(f) show that an impulse at sample number four in the time domain results in four cycles of a cosine wave in the real part of the frequency spectrum, and four cycles of a negative sine wave in the imaginary part. As you recall, an impulse at sample number four in the real part of the frequency spectrum results in four cycles of a cosine wave in the time domain. Likewise, an impulse at sample number four in the imaginary part of the frequency spectrum results in four cycles of a negative sine wave being added to the time domain wave.

As mentioned in Chapter 8, this can be used as another way to calculate the DFT (besides correlating the time domain with sinusoids). Each sample in the time domain results in a cosine wave being added to the real part of the

frequency domain, and a negative sine wave being added to the imaginary part. The *amplitude* of each sinusoid is given by the *amplitude* of the time domain sample. The *frequency* of each sinusoid is provided by the *sample number* of the time domain point. The algorithm involves: (1) stepping through each time domain sample, (2) calculating the sine and cosine waves that correspond to each sample, and (3) adding up all of the contributing sinusoids. The resulting program is nearly identical to the correlation method (Table 8-2), except that the outer and inner loops are exchanged.

The Sinc Function

Figure 11-4 illustrates a common transform pair: the *rectangular pulse* and the *sinc function* (pronounced "sink"). The sinc function is defined as: $\text{sinc}(a) = \sin(\pi a)/(\pi a)$, however, it is common to see the vague statement: "the sinc function is of the general form: $\sin(x)/x$." In other words, the sinc is a sine wave that decays in amplitude as $1/x$. In (a), the rectangular pulse is symmetrically centered on sample zero, making one-half of the pulse on the right of the graph and the other one-half on the left. This appears to the DFT as a single pulse because of the time domain periodicity. The DFT of this signal is shown in (b) and (c), with the *unwrapped* version in (d) and (e).

First look at the unwrapped spectrum, (d) and (e). The *unwrapped magnitude* is an oscillation that decreases in amplitude with increasing frequency. The phase is composed of all zeros, as you should expect for a time domain signal that is symmetrical around sample number zero. We are using the term *unwrapped magnitude* to indicate that it can have both positive and negative values. By definition, the *magnitude* must always be positive. This is shown in (b) and (c) where the magnitude is made all positive by introducing a phase shift of π at all frequencies where the unwrapped magnitude is negative in (d).

In (f), the signal is shifted so that it appears as one contiguous pulse, but is no longer centered on sample number zero. While this doesn't change the magnitude of the frequency domain, it does add a linear component to the phase, making it a jumbled mess. What does the frequency spectrum look like as real and imaginary parts ? Too confusing to even worry about.

An N point time domain signal that contains a unity amplitude rectangular pulse M points wide, has a DFT frequency spectrum given by:

EQUATION 11-1
DFT spectrum of a rectangular pulse. In this equation, N is the number of points in the time domain signal, all of which have a value of zero, except M adjacent points that have a value of one. The frequency spectrum is contained in $X[k]$, where k runs from 0 to $N/2$. To avoid the division by zero, use $X[0] = M$. The *sine* function uses radians, not degrees. This equation takes into account that the signal is *aliased*.

$$Mag\ X[k] = \left| \frac{\sin(\pi k M/N)}{\sin(\pi k/N)} \right|$$

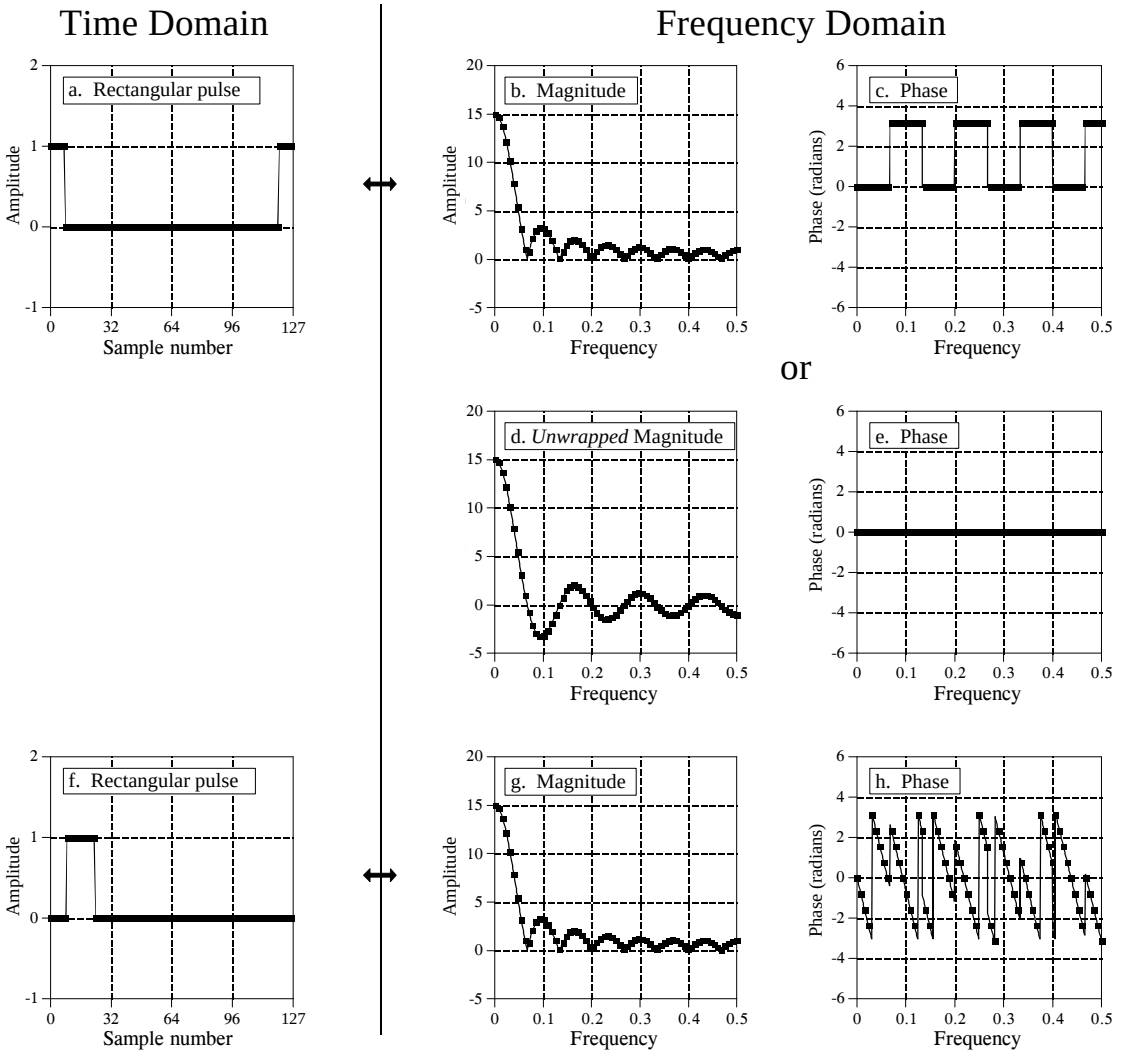

FIGURE 11-3
DFT of a rectangular pulse. A rectangular pulse in one domain corresponds to a sinc function in the other domain.

Alternatively, the DTFT can be used to express the frequency spectrum as a fraction of the sampling rate, *f*:

EQUATION 11-2
Equation 11-1 rewritten in terms of the sampling frequency. The parameter, f, is the fraction of the sampling rate, running continiously from 0 to 0.5. To avoid the division by zero, use $Mag\ X(0) = M$.

$$Mag\ X(f) = \left| \frac{\sin(\pi f M)}{\sin(\pi f)} \right|$$

In other words, Eq. 11-1 provides $N/2 + 1$ *samples* in the frequency spectrum, while Eq. 11-2 provides the *continuous curve* that the samples lie on. These

equations only provide the magnitude. The phase is determined solely by the left-right positioning of the time domain waveform, as discussed in the last chapter.

Notice in Fig. 11-3b that the amplitude of the oscillation does not decay to zero before a frequency of 0.5 is reached. As you should suspect, the waveform continues into the next period where it is *aliased*. This changes the shape of the frequency domain, an effect that is included in Eqs. 11-1 and 11-2.

It is often important to understand what the frequency spectrum looks like when aliasing *isn't* present. This is because *discrete signals* are often used to represent or model *continuous signals*, and continuous signals don't alias. To remove the aliasing in Eqs. 11-1 and 11-2, change the denominators from $sin(\pi kM/N)$ to $\pi kM/N$, and $sin(\pi f)$ to πf, respectively. Figure 11-4 shows the significance of this. The quantity πf can only run from 0 to 1.5708, since f can only run from 0 to 0.5. Over this range there isn't much difference between $sin(\pi f)$ and πf. At zero frequency they have the same value, and at a frequency of 0.5 there is only about a 36% difference. Without aliasing, the curve in Fig. 11-3b would show a slightly lower amplitude near the right side of the graph, and no change near the left side.

When the frequency spectrum of the rectangular pulse is *not* aliased (because the time domain signal is continuous, or because you are ignoring the aliasing), it is of the general form: $sin(x)/x$, i.e., a sinc function. For continuous signals, the *rectangular pulse* and the *sinc function* are Fourier transform pairs. For discrete signals this is only an approximation, with the error being due to aliasing.

The sinc function has an annoying problem at $x = 0$, where $sin(x)/x$ becomes *zero* divided by *zero*. This is not a difficult mathematical problem; as x becomes very small, $sin(x)$ approaches the value of x (see Fig. 11-4).

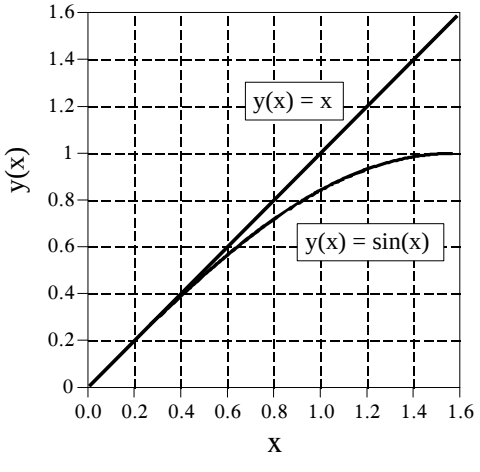

FIGURE 11-4
Comparing x and $sin(x)$. The functions: $y(x) = x$, and $y(x) = sin(x)$ are similar for small values of x, and only differ by about 36% at 1.57 ($\pi/2$). This describes how aliasing distorts the frequency spectrum of the rectangular pulse from a pure sinc function.

This turns the sinc function into x/x, which has a value of *one*. In other words, as x becomes smaller and smaller, the value of $sinc(x)$ approaches *one*, which includes $sinc(0) = 1$. Now try to tell your computer this! All it sees is a division by zero, causing it to complain and stop your program. The important point to remember is that your program must include special handling at $x = 0$ when calculating the sinc function.

A key trait of the sinc function is the location of the **zero crossings**. These occur at frequencies where an integer number of the sinusoid's cycles fit evenly into the rectangular pulse. For example, if the rectangular pulse is 20 points wide, the first zero in the frequency domain is at the frequency that makes one complete cycle in 20 points. The second zero is at the frequency that makes two complete cycles in 20 points, etc. This can be understood by remembering how the DFT is calculated by correlation. The amplitude of a frequency component is found by multiplying the time domain signal by a sinusoid and adding up the resulting samples. If the time domain waveform is a rectangular pulse of unity amplitude, this is the same as *adding* the sinusoid's samples that are within the rectangular pulse. If this summation occurs over an integral number of the sinusoid's cycles, the result will be zero.

The sinc function is widely used in DSP because it is the Fourier transform pair of a very simple waveform, the rectangular pulse. For example, the sinc function is used in *spectral analysis*, as discussed in Chapter 9. Consider the analysis of an infinitely long discrete signal. Since the DFT can only work with *finite* length signals, N samples are selected to represent the longer signal. The key here is that "selecting N samples from a longer signal" is the same as multiplying the longer signal by a rectangular pulse. The *ones* in the rectangular pulse retain the corresponding samples, while the *zeros* eliminate them. How does this affect the frequency spectrum of the signal? Multiplying the time domain by a rectangular pulse results in the frequency domain being *convolved* with a *sinc function*. This reduces the frequency spectrum's resolution, as previously shown in Fig. 9-5a.

Other Transform Pairs

Figure 11-5 (a) and (b) show the duality of the above: a rectangular pulse in the frequency domain corresponds to a sinc function (plus aliasing) in the time domain. Including the effects of aliasing, the time domain signal is given by:

EQUATION 11-3
Inverse DFT of the rectangular pulse. In the frequency domain, the pulse has an amplitude of one, and runs from sample number 0 through sample number M-1. The parameter N is the length of the DFT, and $x[i]$ is the time domain signal with i running from 0 to N-1. To avoid the division by zero, use $x[0] = (2M-1)/N$.

$$x[i] = \frac{1}{N} \frac{\sin(2\pi i (M - 1/2)/N)}{\sin(\pi i / N)}$$

To eliminate the effects of aliasing from this equation, imagine that the frequency domain is so finely sampled that it turns into a continuous curve. This makes the time domain infinitely long with no periodicity. The DTFT is the Fourier transform to use here, resulting in the time domain signal being given by the relation:

EQUATION 11-4
Inverse DTFT of the rectangular pulse. In the frequency domain, the pulse has an amplitude of one, and runs from zero frequency to the cutoff frequency, f_c, a value between 0 and 0.5. The time domain signal is held in $x[i]$ with i running from 0 to N-1. To avoid the division by zero, use $x[0] = 2f_c$.

$$x[i] = \frac{\sin(2\pi f_c i)}{i\pi}$$

This equation is very important in DSP, because the rectangular pulse in the frequency domain is the perfect *low-pass filter*. Therefore, the sinc function described by this equation is the filter kernel for the perfect low-pass filter. This is the basis for a very useful class of digital filters called the *windowed-sinc filters*, described in Chapter 15.

Figures (c) & (d) show that a triangular pulse in the time domain coincides with a sinc function *squared* (plus aliasing) in the frequency domain. This transform pair isn't as important as the *reason* it is true. A $2M - 1$ point triangle in the time domain can be formed by convolving an M point rectangular pulse with itself. Since convolution in the time domain results in multiplication in the frequency domain, convolving a waveform with itself will *square* the frequency spectrum.

Is there a waveform that is its own Fourier Transform? The answer is yes, and there is *only* one: the Gaussian. Figure (e) shows a Gaussian curve, and (f) shows the corresponding frequency spectrum, also a Gaussian curve. This relationship is only true if you ignore aliasing. The relationship between the standard deviation of the time domain and frequency domain is given by: $2\pi\sigma_f = 1/\sigma_t$. While only one side of a Gaussian is shown in (f), the negative frequencies in the spectrum complete the full curve, with the center of symmetry at zero frequency.

Figure (g) shows what can be called a **Gaussian burst**. It is formed by multiplying a sine wave by a Gaussian. For example, (g) is a sine wave multiplied by the same Gaussian shown in (e). The corresponding frequency domain is a Gaussian centered somewhere other than zero frequency. As before, this transform pair is not as important as the *reason* it is true. Since the time domain signal is the multiplication of two signals, the frequency domain will be the convolution of the two frequency spectra. The frequency spectrum of the sine wave is a delta function centered at the frequency of the sine wave. The frequency spectrum of a Gaussian is a Gaussian centered at zero frequency. Convolving the two produces a Gaussian centered at the frequency of the sine wave. This should look familiar; it is identical to the procedure of *amplitude modulation* described in the last chapter.

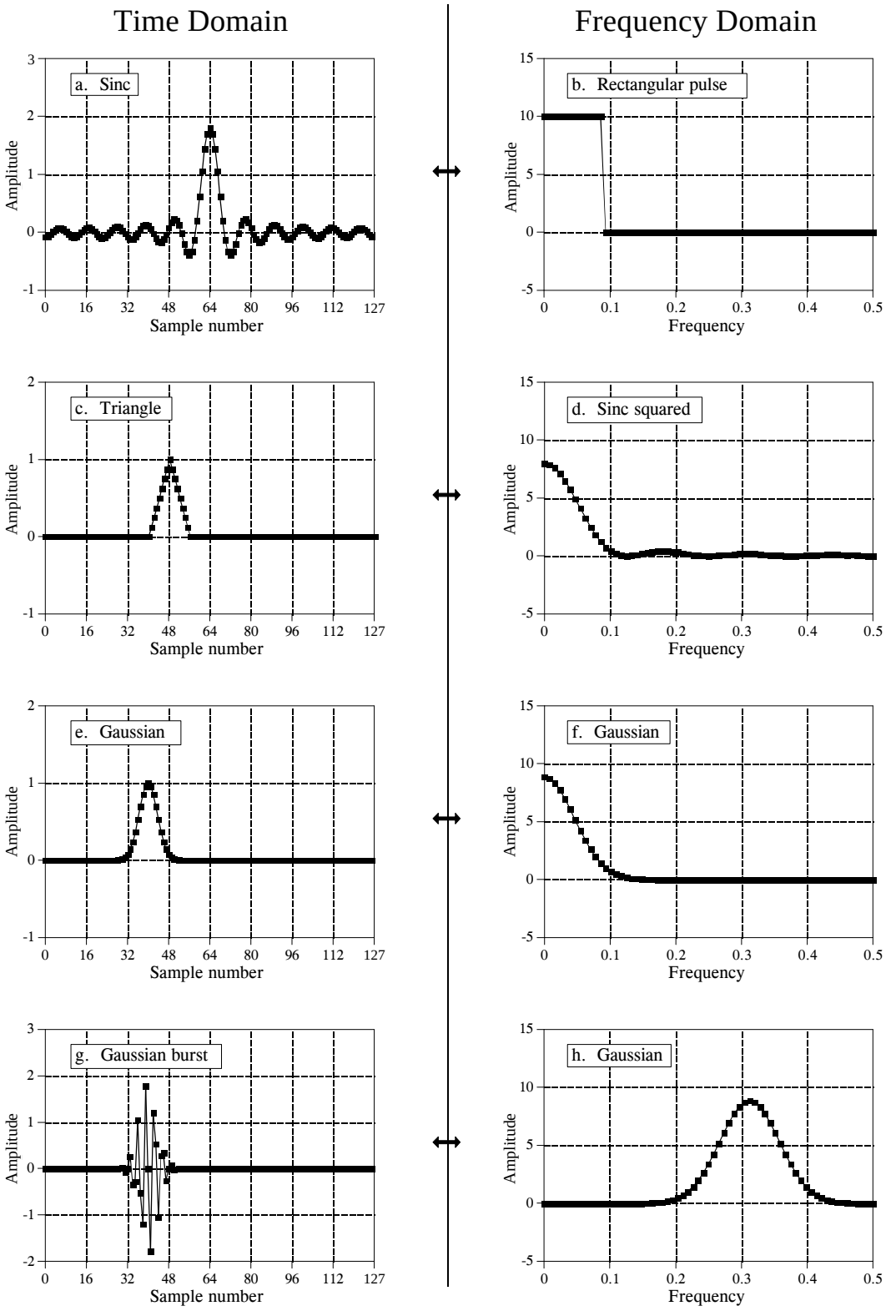

FIGURE 11-5
Common transform pairs.

Gibbs Effect

Figure 11-6 shows a time domain signal being synthesized from sinusoids. The signal being reconstructed is shown in the last graph, (h). Since this signal is 1024 points long, there will be 513 individual frequencies needed for a complete reconstruction. Figures (a) through (g) show what the reconstructed signal looks like if only *some* of these frequencies are used. For example, (f) shows a reconstructed signal using frequencies 0 through 100. This signal was created by taking the DFT of the signal in (h), setting frequencies 101 through 512 to a value of zero, and then using the Inverse DFT to find the resulting time domain signal.

As more frequencies are added to the reconstruction, the signal becomes closer to the final solution. The interesting thing is *how* the final solution is approached at the *edges* in the signal. There are three sharp edges in (h). Two are the edges of the rectangular pulse. The third is between sample numbers 1023 and 0, since the DFT views the time domain as periodic. When only some of the frequencies are used in the reconstruction, each edge shows *overshoot* and *ringing* (decaying oscillations). This overshoot and ringing is known as the **Gibbs effect,** after the mathematical physicist Josiah Gibbs, who explained the phenomenon in 1899.

Look closely at the overshoot in (e), (f), and (g). As more sinusoids are added, the *width* of the overshoot decreases; however, the *amplitude* of the overshoot remains about the same, roughly 9 percent. With discrete signals this is not a problem; the overshoot is eliminated when the last frequency is added. However, the reconstruction of continuous signals cannot be explained so easily. An infinite number of sinusoids must be added to synthesize a continuous signal. The problem is, the amplitude of the overshoot does not decrease as the number of sinusoids approaches infinity, it stays about the same 9%. Given this situation (and other arguments), it is reasonable to question if a summation of continuous sinusoids *can* reconstruct an edge. Remember the squabble between Lagrange and Fourier?

The critical factor in resolving this puzzle is that the *width* of the overshoot becomes smaller as more sinusoids are included. The overshoot is still present with an infinite number of sinusoids, but it has *zero* width. Exactly at the discontinuity the value of the reconstructed signal converges to the midpoint of the step. As shown by Gibbs, the summation converges to the signal in the sense that the *error* between the two has zero energy.

Problems related to the Gibbs effect are frequently encountered in DSP. For example, a low-pass filter is a *truncation* of the higher frequencies, resulting in overshoot and ringing at the edges in the *time domain*. Another common procedure is to truncate the ends of a time domain signal to prevent them from extending into neighboring periods. By duality, this distorts the edges in the *frequency domain*. These issues will resurface in future chapters on filter design.

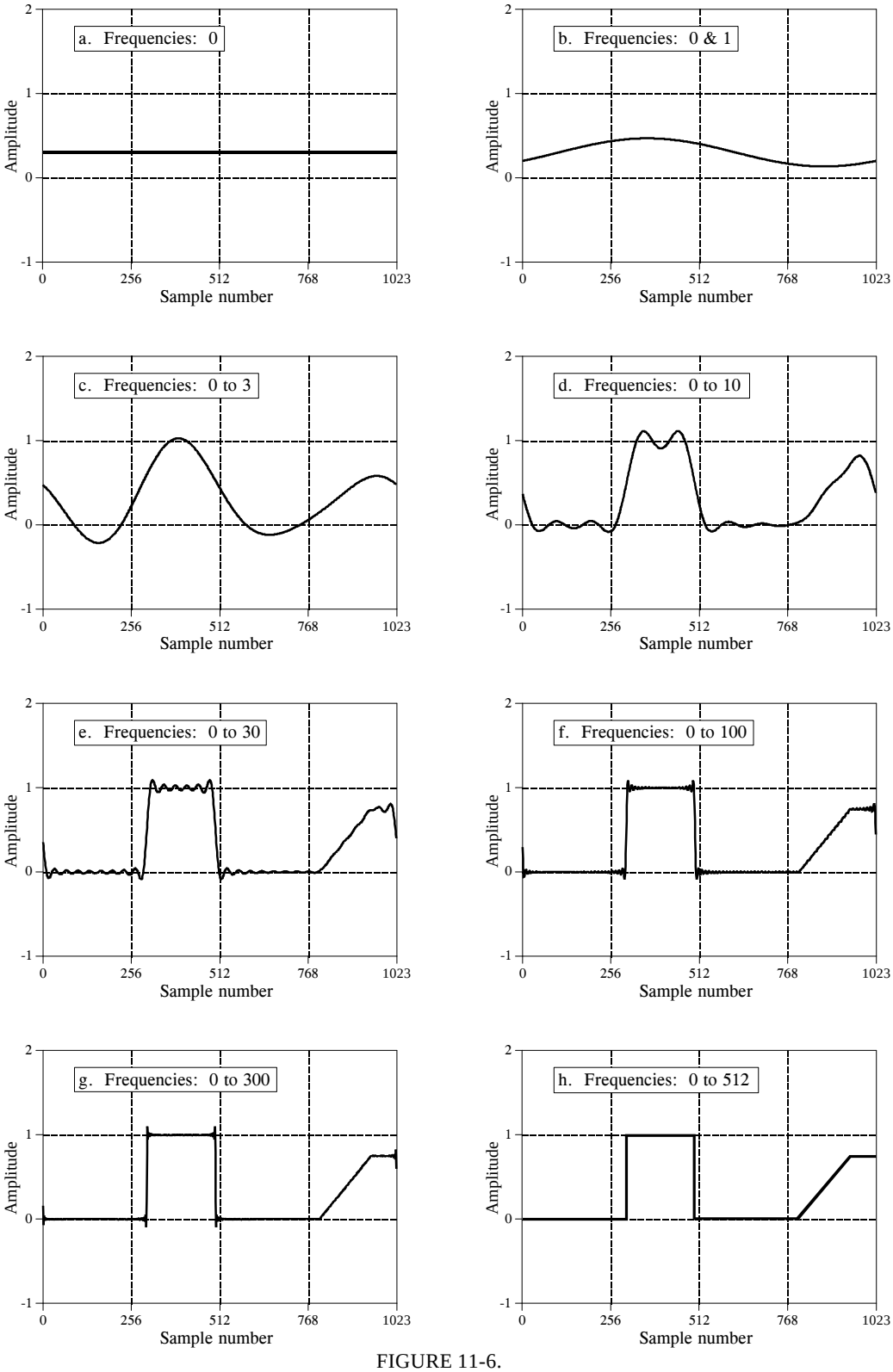

FIGURE 11-6.
The Gibbs effect.

Harmonics

If a signal is periodic with frequency *f*, the only frequencies composing the signal are integer multiples of *f*, i.e., *f*, 2*f*, 3*f*, 4*f*, etc. These frequencies are called **harmonics.** The **first harmonic** is *f*, the **second harmonic** is 2*f*, the **third harmonic** is 3*f*, and so forth. The first harmonic (i.e., *f*) is also given a special name, the **fundamental frequency**. Figure 11-7 shows an

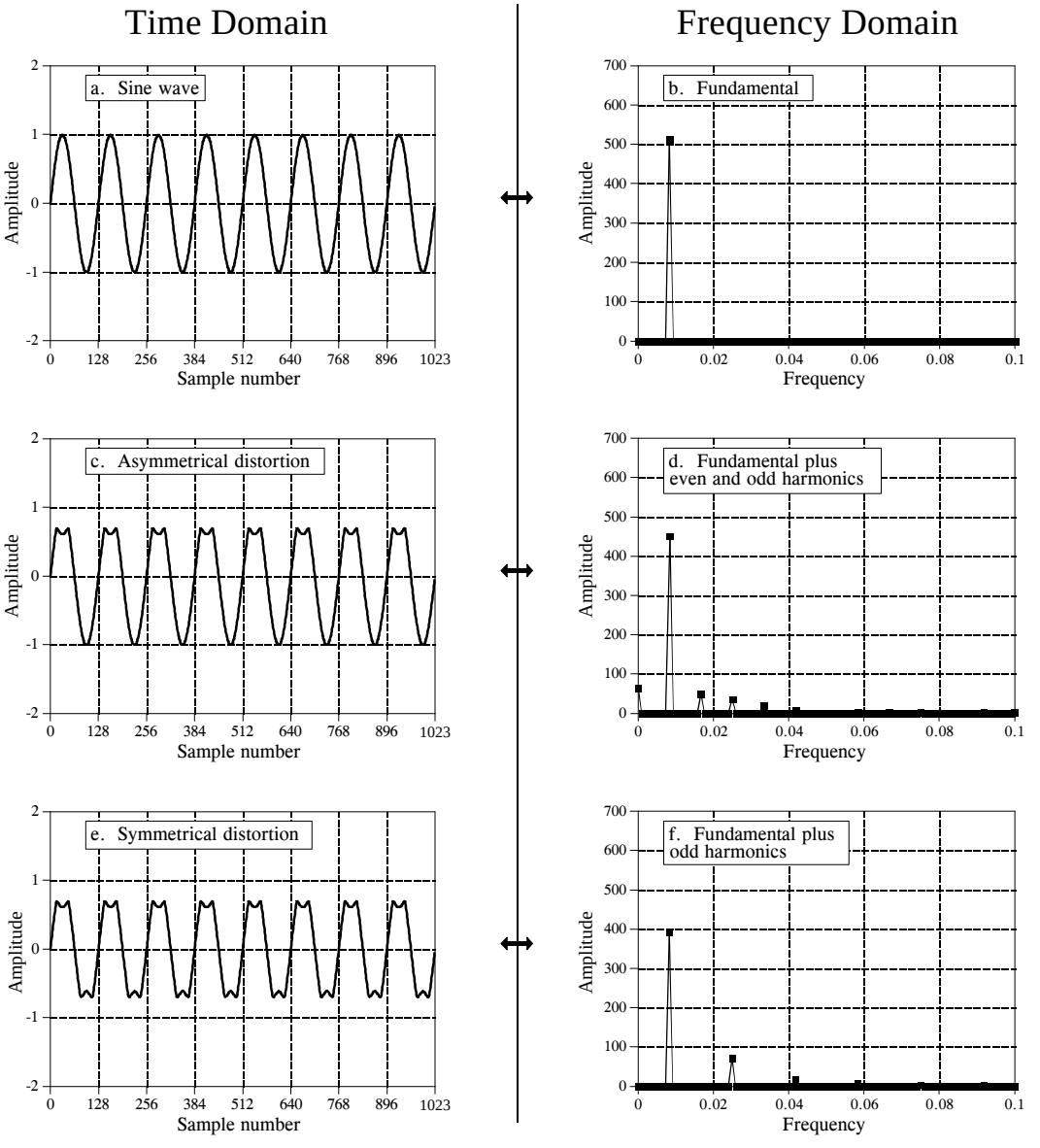

FIGURE 11-7
Example of harmonics. Asymmetrical distortion, shown in (c), results in even and odd harmonics, (d), while symmetrical distortion, shown in (e), produces only even harmonics, (f).

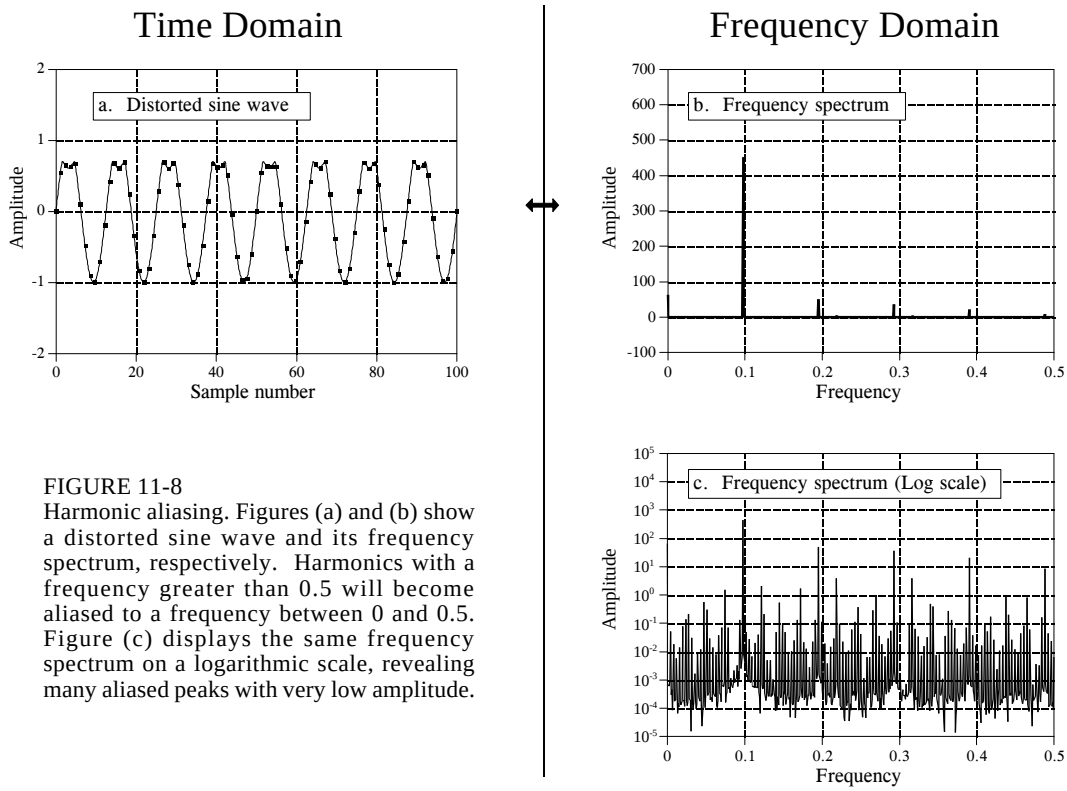

FIGURE 11-8
Harmonic aliasing. Figures (a) and (b) show a distorted sine wave and its frequency spectrum, respectively. Harmonics with a frequency greater than 0.5 will become aliased to a frequency between 0 and 0.5. Figure (c) displays the same frequency spectrum on a logarithmic scale, revealing many aliased peaks with very low amplitude.

example. Figure (a) is a pure sine wave, and (b) is its DFT, a single peak. In (c), the sine wave has been distorted by poking in the tops of the peaks. Figure (d) shows the result of this distortion in the frequency domain. Because the distorted signal is periodic with the same frequency as the original sine wave, the frequency domain is composed of the original peak plus harmonics. Harmonics can be of any amplitude; however, they usually become smaller as they increase in frequency. As with any signal, *sharp edges* result in *higher frequencies*. For example, consider a common TTL logic gate generating a 1 kHz square wave. The edges rise in a few nanoseconds, resulting in harmonics being generated to nearly 100 MHz, the *ten-thousandth* harmonic!

Figure (e) demonstrates a subtlety of harmonic analysis. If the signal is symmetrical around a horizontal axis, i.e., the top lobes are mirror images of the bottom lobes, all of the even harmonics will have a value of zero. As shown in (f), the only frequencies contained in the signal are the fundamental, the third harmonic, the fifth harmonic, etc.

All *continuous* periodic signals can be represented as a summation of harmonics, just as described. *Discrete* periodic signals have a problem that disrupts this simple relation. As you might have guessed, the problem is *aliasing*. Figure 11-8a shows a sine wave distorted in the same manner as before, by poking in the tops of the peaks. This waveform looks much less

regular and smooth than in the previous example because the sine wave is at a much higher frequency, resulting in fewer samples per cycle. Figure (b) shows the frequency spectrum of this signal. As you would expect, you can identify the fundamental and harmonics. This example shows that harmonics can extend to frequencies greater than 0.5 of the sampling frequency, and will be *aliased* to frequencies somewhere between 0 and 0.5. You don't notice them in (b) because their amplitudes are too low. Figure (c) shows the frequency spectrum plotted on a logarithmic scale to reveal these low amplitude aliased peaks. At first glance, this spectrum looks like random noise. It isn't; this is a result of the many harmonics overlapping as they are aliased.

It is important to understand that this example involves distorting a signal *after* it has been digitally represented. If this distortion occurred in an analog signal, you would remove the offending harmonics with an antialias filter *before* digitization. Harmonic aliasing is only a problem when nonlinear operations are performed directly on a discrete signal. Even then, the amplitude of these aliased harmonics is often low enough that they can be ignored.

The concept of harmonics is also useful for another reason: it explains why the DFT views the time and frequency domains as *periodic*. In the frequency domain, an N point DFT consists of $N/2+1$ equally spaced frequencies. You can view the frequencies *between* these samples as (1) having a value of zero, or (2) not existing. Either way they don't contribute to the synthesis of the time domain signal. In other words, a *discrete* frequency spectrum consists of *harmonics*, rather than a continuous range of frequencies. This requires the time domain to be periodic with a frequency equal to the lowest sinusoid in the frequency domain, i.e., the fundamental frequency. Neglecting the DC value, the lowest frequency represented in the frequency domain makes one complete cycle every N samples, resulting in the time domain being periodic with a period of N. In other words, if one domain is *discrete*, the other domain must be *periodic*, and vice versa. This holds for all four members of the Fourier transform family. Since the DFT views both domains as discrete, it must also view both domains as periodic. The samples in each domain represent harmonics of the periodicity of the opposite domain.

Chirp Signals

Chirp signals are an ingenious way of handling a practical problem in echo location systems, such as radar and sonar. Figure 11-9 shows the frequency response of the chirp system. The magnitude has a constant value of one, while the phase is a parabola:

EQUATION 11-7
Phase of the chirp system.

$$Phase\ X[k] = \alpha k + \beta k^2$$

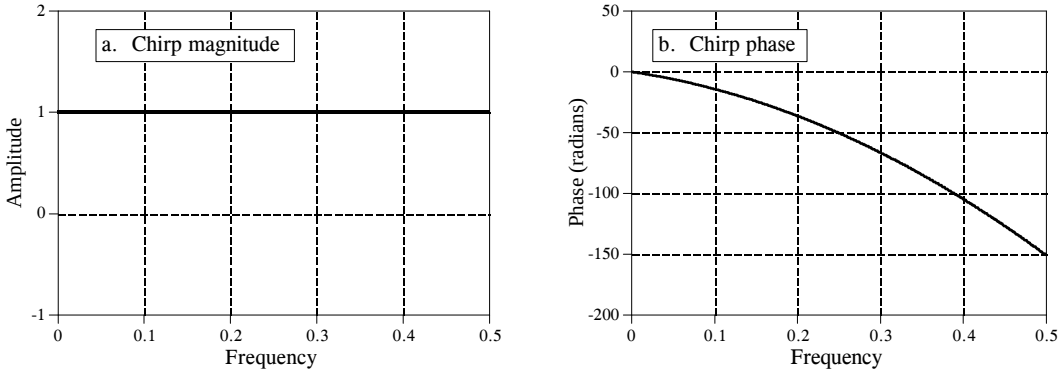

FIGURE 11-9
Frequency response of the chirp system. The magnitude is a constant, while the phase is a parabola.

The parameter α introduces a linear slope in the phase, that is, it simply shifts the impulse response left or right as desired. The parameter β controls the *curvature* of the phase. These two parameters must be chosen such that the phase at frequency 0.5 (i.e. $k = N/2$) is a multiple of 2π. Remember, whenever the phase is directly manipulated, frequency 0 and 0.5 must both have a phase of zero (or a multiple of 2π, which is the same thing).

Figure 11-10 shows an impulse entering a chirp system, and the impulse response exiting the system. The impulse response is an oscillatory burst that starts at a low frequency and changes to a high frequency as time progresses. This is called a *chirp* signal for a very simple reason: it sounds like the chirp of a bird when played through a speaker.

The key feature of the chirp system is that it is completely *reversible*. If you run the chirp signal through an *antichirp* system, the signal is again made into an impulse. This requires the antichirp system to have a magnitude of one, and the *opposite* phase of the chirp system. As discussed in the last

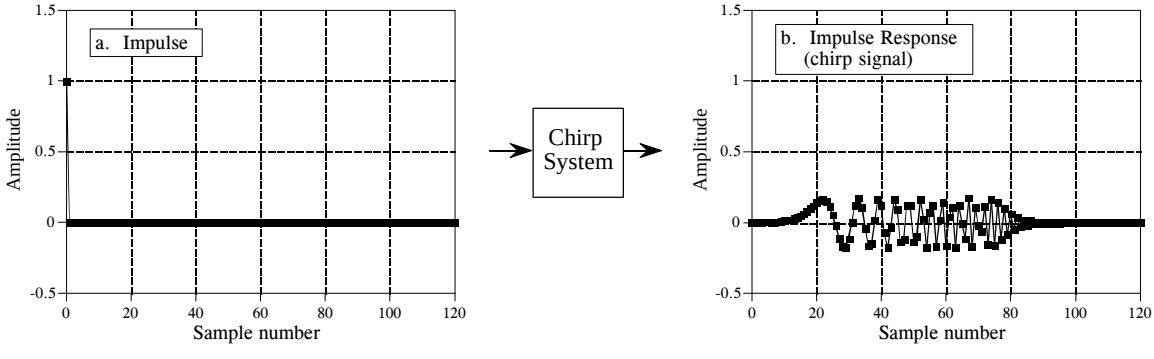

FIGURE 11-10
The chirp system. The impulse response of a chirp system is a chirp signal.

chapter, this means that the impulse response of the antichirp system is found by preforming a left-for-right flip of the chirp system's impulse response. Interesting, but what is it good for?

Consider how a radar system operates. A short burst of radio frequency energy is emitted from a directional antenna. Aircraft and other objects reflect some of this energy back to a radio receiver located next to the transmitter. Since radio waves travel at a constant rate, the elapsed time between the transmitted and received signals provides the distance to the target. This brings up the first requirement for the pulse: it needs to be as short as possible. For example, a 1 microsecond pulse provides a radio burst about 300 meters long. This means that the distance information we obtain with the system will have a resolution of about this same length. If we want better distance resolution, we need a shorter pulse.

The second requirement is obvious: if we want to detect objects farther away, you need more energy in your pulse. Unfortunately, *more energy* and *shorter pulse* are conflicting requirements. The electrical power needed to provide a pulse is equal to the energy of the pulse divided by the pulse length. Requiring both *more energy* and a *shorter pulse* makes electrical power handling a limiting factor in the system. The output stage of a radio transmitter can only handle so much power without destroying itself.

Chirp signals provide a way of breaking this limitation. Before the impulse reaches the final stage of the radio transmitter, it is passed through a chirp system. Instead of bouncing an impulse off the target aircraft, a chirp signal is used. After the chirp echo is received, the signal is passed through an antichirp system, restoring the signal to an impulse. This allows the portions of the system that measure distance to see short pulses, while the power handling circuits see long duration signals. This type of waveshaping is a fundamental part of modern radar systems.