CHAPTER

16

Windowed-Sinc Filters

Windowed-sinc filters are used to separate one band of frequencies from another. They are very stable, produce few surprises, and can be pushed to incredible performance levels. These exceptional frequency domain characteristics are obtained at the expense of poor performance in the time domain, including excessive ripple and overshoot in the step response. When carried out by standard convolution, windowed-sinc filters are easy to program, but slow to execute. Chapter 18 shows how the FFT can be used to dramatically improve the computational speed of these filters.

Strategy of the Windowed-Sinc

Figure 16-1 illustrates the idea behind the windowed-sinc filter. In (a), the frequency response of the *ideal* low-pass filter is shown. All frequencies below the cutoff frequency, f_C, are passed with unity amplitude, while all higher frequencies are blocked. The passband is perfectly flat, the attenuation in the stopband is infinite, and the transition between the two is infinitesimally small.

Taking the Inverse Fourier Transform of this ideal frequency response produces the ideal filter kernel (impulse response) shown in (b). As previously discussed (see Chapter 11, Eq. 11-4), this curve is of the general form: $\sin(x)/x$, called the **sinc function**, given by:

$$h[i] = \frac{\sin(2\pi f_C i)}{i\pi}$$

Convolving an input signal with this filter kernel provides a *perfect* low-pass filter. The problem is, the sinc function continues to both negative and positive infinity without dropping to zero amplitude. While this infinite length is not a problem for *mathematics*, it is a show stopper for *computers*.

To get around this problem, we will make two modifications to the sinc function in (b), resulting in the waveform shown in (c). First, it is truncated to $M+1$ points, symmetrically chosen around the main lobe, where M is an even number. All samples outside these $M+1$ points are set to zero, or simply ignored. Second, the entire sequence is shifted to the right so that it runs from 0 to M. This allows the filter kernel to be represented using only *positive* indexes. While many programming languages allow *negative* indexes, they are a nuisance to use. The sole effect of this $M/2$ shift in the filter kernel is to shift the output signal by the same amount.

Since the modified filter kernel is only an approximation to the ideal filter kernel, it will not have an ideal frequency response. To find the frequency response that is obtained, the Fourier transform can be taken of the signal in (c), resulting in the curve in (d). It's a mess! There is excessive ripple in the passband and poor attenuation in the stopband (recall the Gibbs effect discussed in Chapter 11). These problems result from the abrupt discontinuity at the ends of the truncated sinc function. Increasing the length of the filter kernel does not reduce these problems; the discontinuity is significant no matter how long M is made.

Fortunately, there is a simple method of improving this situation. Figure (e) shows a smoothly tapered curve called a **Blackman window**. Multiplying the truncated-sinc, (c), by the Blackman window, (e), results in the **windowed-sinc** filter kernel shown in (f). The idea is to reduce the abruptness of the truncated ends and thereby improve the frequency response. Figure (g) shows this improvement. The passband is now flat, and the stopband attenuation is so good it cannot be seen in this graph.

Several different windows are available, most of them named after their original developers in the 1950s. Only two are worth using, the **Hamming window** and the **Blackman window** These are given by:

EQUATION 16-1
The Hamming window. These windows run from $i = 0$ to M, for a total of $M+1$ points.

$$w[i] = 0.54 - 0.46\cos(2\pi i/M)$$

EQUATION 16-2
The Blackman window.

$$w[i] = 0.42 - 0.5\cos(2\pi i/M) + 0.08\cos(4\pi i/M)$$

Figure 16-2a shows the shape of these two windows for $M = 50$ (i.e., 51 total points in the curves). Which of these two windows should you use? It's a trade-off between parameters. As shown in Fig. 16-2b, the Hamming window has about a 20% faster *roll-off* than the Blackman. However,

FIGURE 16-1 (facing page)
Derivation of the windowed-sinc filter kernel. The frequency response of the ideal low-pass filter is shown in (a), with the corresponding filter kernel in (b), a sinc function. Since the sinc is infinitely long, it must be truncated to be used in a computer, as shown in (c). However, this truncation results in undesirable changes in the frequency response, (d). The solution is to multiply the truncated-sinc with a smooth window, (e), resulting in the windowed-sinc filter kernel, (f). The frequency response of the windowed-sinc, (g), is smooth and well behaved. These figures are not to scale.

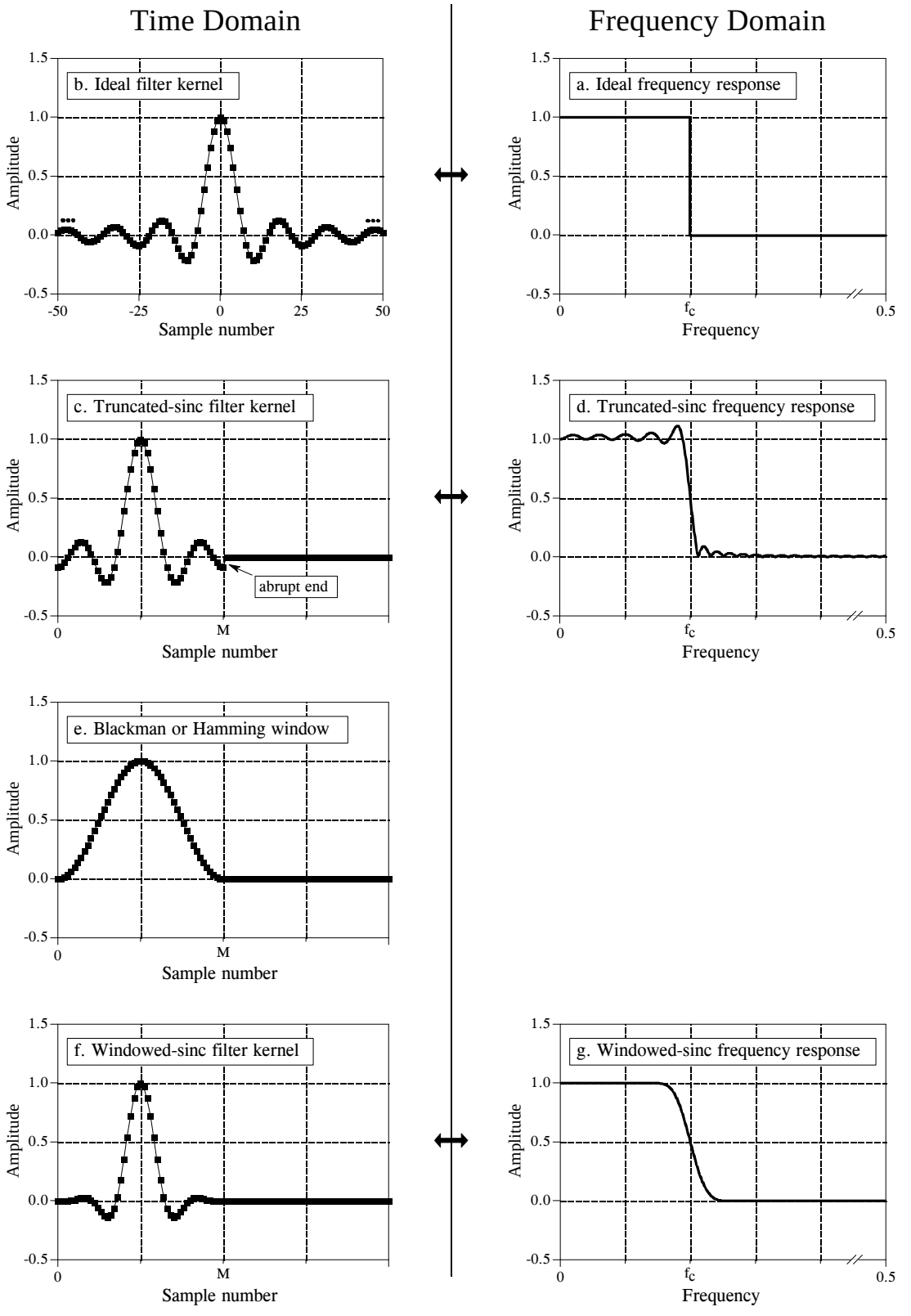
Time Domain
Frequency Domain
b. Ideal filter kernel
a. Ideal frequency response
c. Truncated-sinc filter kernel
d. Truncated-sinc frequency response
abrupt end
e. Blackman or Hamming window
f. Windowed-sinc filter kernel
g. Windowed-sinc frequency response
Amplitude
Sample number
Frequency
1.5
1.0
0.5
0.0
-0.5
-50
-25
0
25
50
M
f_c

FIGURE 16-1

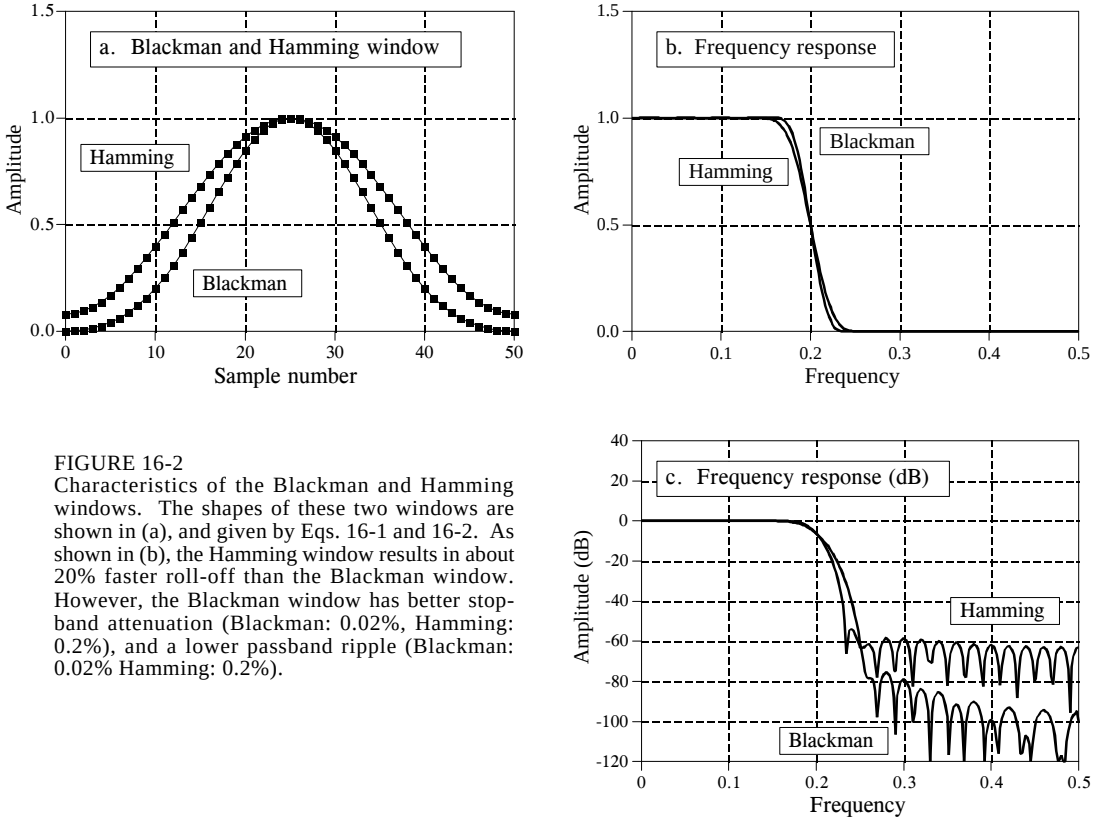

FIGURE 16-2
Characteristics of the Blackman and Hamming windows. The shapes of these two windows are shown in (a), and given by Eqs. 16-1 and 16-2. As shown in (b), the Hamming window results in about 20% faster roll-off than the Blackman window. However, the Blackman window has better stopband attenuation (Blackman: 0.02%, Hamming: 0.2%), and a lower passband ripple (Blackman: 0.02% Hamming: 0.2%).

(c) shows that the Blackman has a better *stopband attenuation*. To be exact, the stopband attenuation for the Blackman is -74dB (~0.02%), while the Hamming is only -53dB (~0.2%). Although it cannot be seen in these graphs, the Blackman has a *passband ripple* of only about 0.02%, while the Hamming is typically 0.2%. In general, the Blackman should be your first choice; a slow roll-off is easier to handle than poor stopband attenuation.

There are other windows you might hear about, although they fall short of the Blackman and Hamming. The **Bartlett window** is a triangle, using straight lines for the taper. The **Hanning window,** also called the **raised cosine window**, is given by: $w[i] = 0.5 - 0.5\cos(2\pi i/M)$. These two windows have about the same roll-off speed as the Hamming, but worse stopband attenuation (Bartlett: -25dB or 5.6%, Hanning -44dB or 0.63%). You might also hear of a **rectangular window**. This is the same as *no* window, just a truncation of the tails (such as in Fig. 16-1c). While the roll-off is ~2.5 times faster than the Blackman, the stopband attenuation is only -21dB (8.9%).

Designing the Filter

To design a windowed-sinc, two parameters must be selected: the cutoff frequency, f_C, and the length of the filter kernel, M. The cutoff frequency

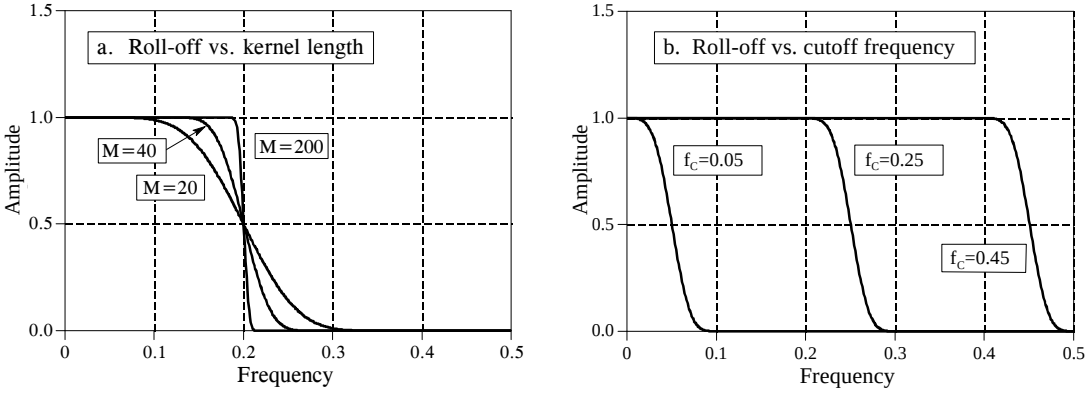

FIGURE 16-3
Filter length vs. roll-off of the windowed-sinc filter. As shown in (a), for M = 20, 40, and 200, the transition bandwidths are BW = 0.2, 0.1, and 0.02 of the sampling rate, respectively. As shown in (b), the shape of the frequency response does not change with different cutoff frequencies. In (b), M = 60.

is expressed as a fraction of the sampling rate, and therefore must be between 0 and 0.5. The value for M sets the *roll-off* according to the approximation:

EQUATION 16-3
Filter length vs. roll-off. The length of the filter kernel, M, determines the transition bandwidth of the filter, BW. This is only an approximation since roll-off depends on the particular window being used.

$$M \approx \frac{4}{BW}$$

where BW is the width of the transition band, measured from where the curve just barely leaves one, to where it almost reaches zero (say, 99% to 1% of the curve). The transition bandwidth is also expressed as a fraction of the sampling frequency, and must between 0 and 0.5. Figure 16-3a shows an example of how this approximation is used. The three curves shown are generated from filter kernels with: M = 20, 40, and 200. From Eq. 16-3, the transition bandwidths are: BW = 0.2, 0.1, and 0.02, respectively. Figure (b) shows that the shape of the frequency response does not depend on the cutoff frequency selected.

Since the time required for a convolution is proportional to the length of the signals, Eq. 16-3 expresses a trade-off between *computation time* (depends on the value of M) and *filter sharpness* (the value of BW). For instance, the 20% slower roll-off of the Blackman window (as compared with the Hamming) can be compensated for by using a filter kernel 20% longer. In other words, it could be said that the Blackman window is 20% slower to execute that an equivalent roll-off Hamming window. This is important because the execution speed of windowed-sinc filters is already terribly slow.

As also shown in Fig. 16-3b, the cutoff frequency of the windowed-sinc filter is measured at the *one-half amplitude* point. Why use 0.5 instead of the

standard 0.707 (-3dB) used in analog electronics and other digital filters? This is because the windowed-sinc's frequency response is *symmetrical* between the passband and the stopband. For instance, the Hamming window results in a passband ripple of 0.2%, and an *identical* stopband attenuation (i.e., ripple in the stopband) of 0.2%. Other filters do not show this symmetry, and therefore have no advantage in using the one-half amplitude point to mark the cutoff frequency. As shown later in this chapter, this symmetry makes the windowed-sinc ideal for *spectral inversion*.

After f_C and M have been selected, the filter kernel is calculated from the relation:

$$h[i] = K \frac{\sin(2\pi f_C (i - M/2))}{i - M/2} \left[0.42 - 0.5\cos\left(\frac{2\pi i}{M}\right) + 0.08\cos\left(\frac{4\pi i}{M}\right) \right]$$

EQUATION 16-4
The windowed-sinc filter kernel. The cutoff frequency, f_C, is expressed as a fraction of the sampling rate, a value between 0 and 0.5. The length of the filter kernel is determined by M, which must be an even integer. The sample number i, is an integer that runs from 0 to M, resulting in $M+1$ total points in the filter kernel. The constant, K, is chosen to provide unity gain at zero frequency. To avoid a divide-by-zero error, for $i = M/2$, use $h[i] = 2\pi f_C K$.

Don't be intimidated by this equation! Based on the previous discussion, you should be able to identify three components: the *sinc function*, the *M/2 shift*, and the *Blackman window*. For the filter to have unity gain at DC, the constant K must be chosen such that the sum of all the samples is equal to one. In practice, ignore K during the calculation of the filter kernel, and then *normalize* all of the samples as needed. The program listed in Table 16-1 shows how this is done. Also notice how the calculation is handled at the center of the sinc, $i = M/2$, which involves a division by zero.

This equation may be long, but it is easy to use; simply type it into your computer program and forget it. Let the computer handle the calculations. If you find yourself trying to evaluate this equation by hand, you are doing something very very wrong.

Let's be specific about where the filter kernel described by Eq. 16-4 is located in your computer array. As an example, M will be chosen to be 100. Remember, M must be an even number. The first point in the filter kernel is in array location 0, while the last point is in array location 100. This means that the entire signal is 101 points long. The center of symmetry is at point 50, i.e., $M/2$. The 50 points to the left of point 50 are symmetrical with the 50 points to the right. Point 0 is the same value as point 100, and point 49 is the same as point 51. If you must have a specific number of samples in the filter kernel, such as to use the FFT, simply add zeros to one end or the other. For example, with $M = 100$, you could make samples 101 through 127 equal to zero, resulting in a filter kernel 128 points long.

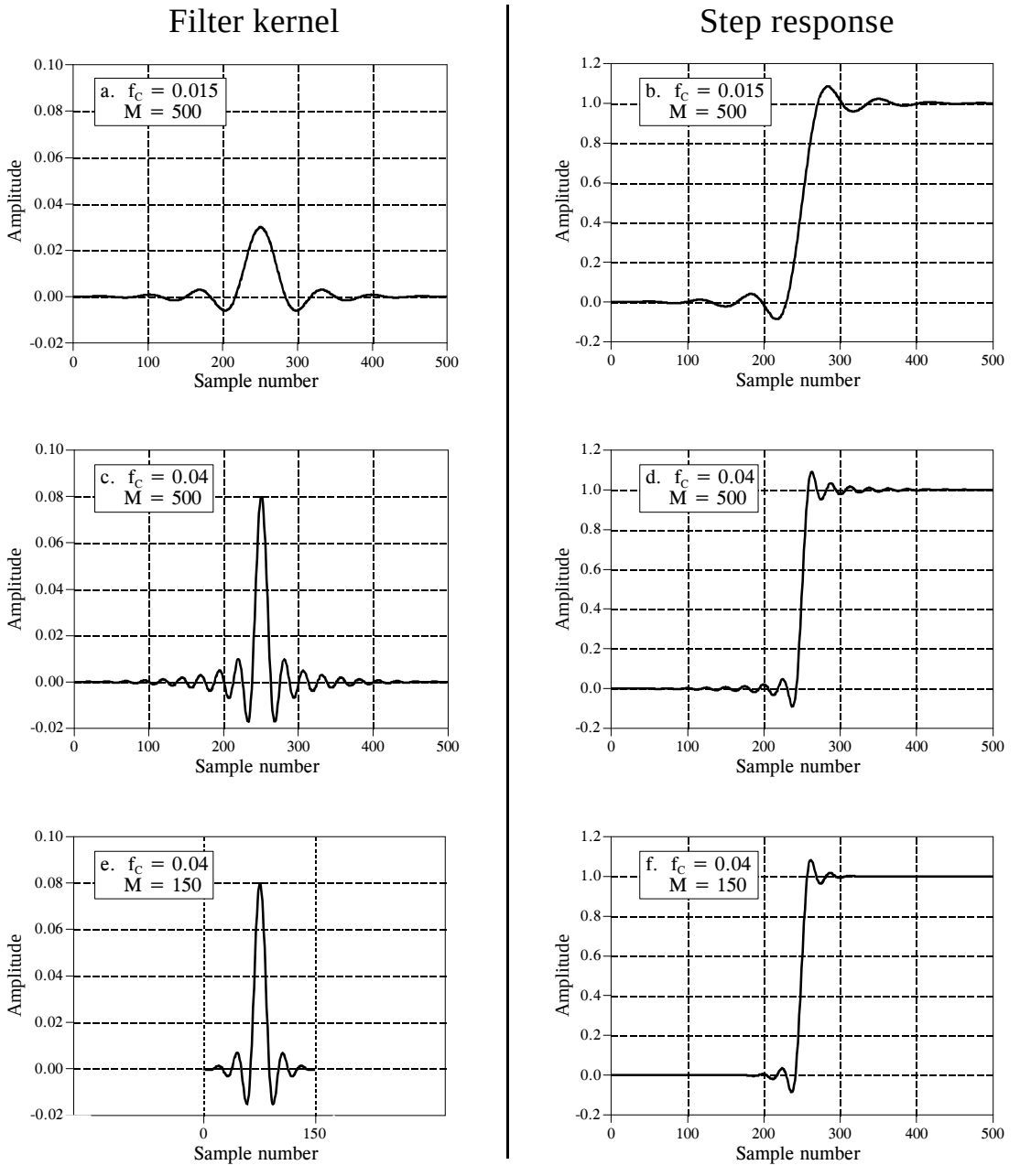

FIGURE 16-4
Example filter kernels and the corresponding step responses. The frequency of the sinusoidal oscillation is approximately equal to the cutoff frequency, f_C, while M determines the kernel length.

Figure 16-4 shows examples of windowed-sinc filter kernels, and their corresponding step responses. The samples at the beginning and end of the filter kernels are so small that they can't even be seen in the graphs. Don't make the mistake of thinking they are unimportant! These samples may be small in value; however, they collectively have a large effect on the

performance of the filter. This is also why floating point representation is typically used to implement windowed-sinc filters. Integers usually don't have enough dynamic range to capture the large variation of values contained in the filter kernel. How does the windowed-sinc filter perform in the time domain? Terrible! The step response has overshoot and ringing; this is *not* a filter for signals with information encoded in the time domain.

Examples of Windowed-Sinc Filters

An electroencephalogram, or EEG, is a measurement of the electrical activity of the brain. It can be detected as millivolt level signals appearing on electrodes attached to the surface of the head. Each nerve cell in the brain generates small electrical pulses. The EEG is the combined result of an enormous number of these electrical pulses being generated in a (hopefully) coordinated manner. Although the relationship between thought and this electrical coordination is very poorly understood, different frequencies in the EEG can be identified with specific mental states. If you close your eyes and relax, the predominant EEG pattern will be a slow oscillation between about 7 and 12 hertz. This waveform is called the *alpha rhythm*, and is associated with contentment and a decreased level of attention. Opening your eyes and looking around causes the EEG to change to the *beta rhythm*, occurring between about 17 and 20 hertz. Other frequencies and waveforms are seen in children, different depths of sleep, and various brain disorders such as epilepsy.

In this example, we will assume that the EEG signal has been amplified by analog electronics, and then digitized at a sampling rate of 100 samples per second. Acquiring data for 50 seconds produces a signal of 5,000 points. Our goal is to separate the alpha from the beta rhythms. To do this, we will design a digital low-pass filter with a cutoff frequency of 14 hertz, or 0.14

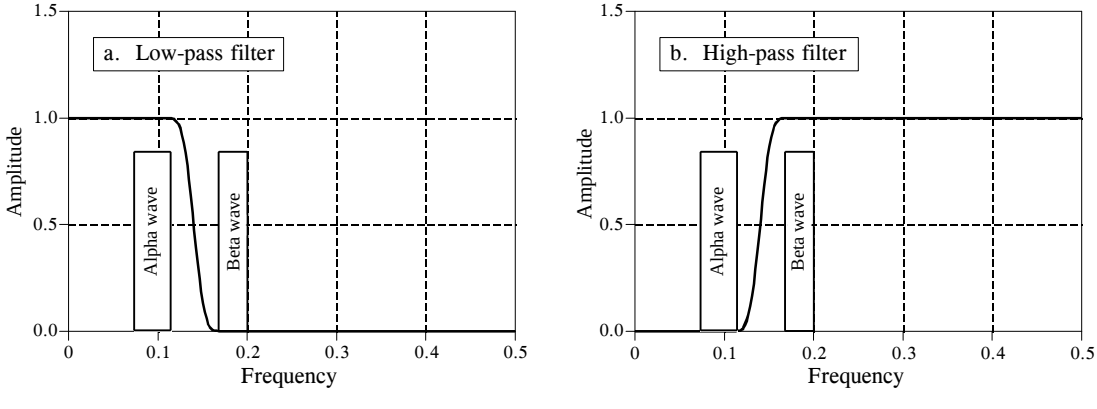

FIGURE 16-5
Example of windowed-sinc filters. The alpha and beta rhythms in an EEG are separated by low-pass and high-pass filters with $M = 100$. The program to implement the low-pass filter is shown in Table 16-1. The program for the high-pass filter is identical, except for a *spectral inversion* of the low-pass filter kernel.

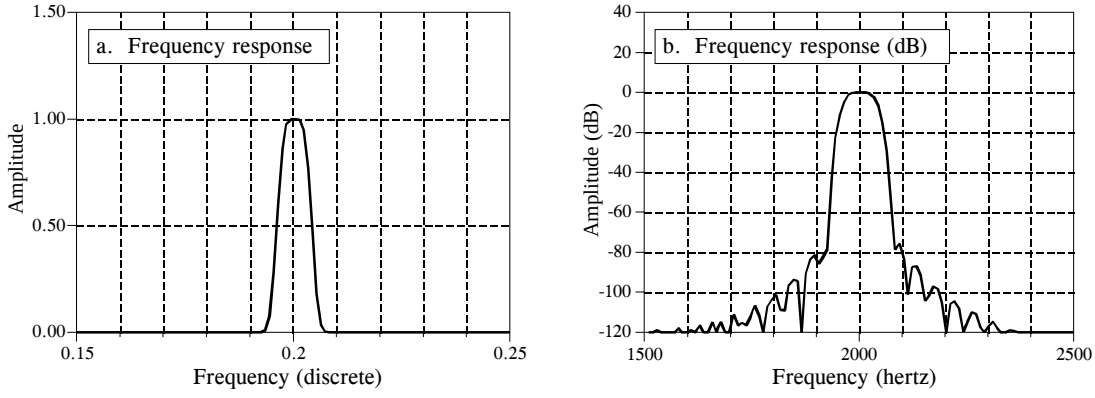

FIGURE 16-6
Example of a windowed-sinc band-pass filter. This filter was designed for a sampling rate of 10 kHz. When referenced to the analog signal, the center frequency of the passband is at 2 kHz, the passband is 80 hertz, and the transition bands are 50 hertz. The windowed-sinc uses 801 points in the filter kernel to achieve this roll-off, and a Blackman window for good stopband attenuation. Figure (a) shows the resulting frequency response on a linear scale, while (b) shows it in decibels. The frequency axis in (a) is expressed as a fraction of the sampling frequency, while (b) is expressed in terms of the analog signal before digitization.

of the sampling rate. The transition bandwidth will be set at 4 hertz, or 0.04 of the sampling rate. From Eq. 16-3, the filter kernel needs to be about 101 points long, and we will arbitrarily choose to use a Hamming window. The program in Table 16-1 shows how the filter is carried out. The frequency response of the filter, obtained by taking the Fourier Transform of the filter kernel, is shown in Fig. 16-5.

In a second example, we will design a *band-pass filter* to isolate a *signaling tone* in an audio signal, such as when a button on a telephone is pressed. We will assume that the signal has been digitized at 10 kHz, and the goal is to isolate an 80 hertz band of frequencies centered on 2 kHz. In terms of the sampling rate, we want to block all frequencies below 0.196 and above 0.204 (corresponding to 1960 hertz and 2040 hertz, respectively). To achieve a transition bandwidth of 50 hertz (0.005 of the sampling rate), we will make the filter kernel 801 points long, and use a Blackman window. Table 16-2 contains a program for calculating the filter kernel, while Fig. 16-6 shows the frequency response. The design involves several steps. First, *two* low-pass filters are designed, one with a cutoff at 0.196, and the other with a cutoff at 0.204. This second filter is then *spectrally inverted*, making it a high-pass filter (see Chapter 14, Fig. 14-6). Next, the two filter kernels are added, resulting in a band-reject filter (see Fig. 14-8). Finally, another *spectral inversion* makes this into the desired band-pass filter.

Pushing it to the Limit

The windowed-sinc filter can be pushed to incredible performance levels without nasty surprises. For instance, suppose you need to isolate a 1 *millivolt* signal riding on a 120 *volt* power line. The low-pass filter will need

```
100 'LOW-PASS WINDOWED-SINC FILTER
110 'This program filters 5000 samples with a 101 point windowed-sinc filter,
120 'resulting in 4900 samples of filtered data.
130 '
140 DIM X[4999]                    'X[ ] holds the input signal
150 DIM Y[4999]                    'Y[ ] holds the output signal
160 DIM H[100]                     'H[ ] holds the filter kernel
170 '
180 PI = 3.14159265
190 FC = .14                       'Set the cutoff frequency (between 0 and 0.5)
200 M% = 100                       'Set filter length (101 points)
210 '
220 GOSUB XXXX                     'Mythical subroutine to load X[ ]
230 '
240 '                              'Calculate the low-pass filter kernel via Eq. 16-4
250 FOR I% = 0 TO 100
260   IF (I%-M%/2) = 0 THEN H[I%] = 2*PI*FC
270   IF (I%-M%/2) <> 0  THEN H[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
280   H[I%] = H[I%] * (0.54 - 0.46*COS(2*PI*I%/M%) )
290 NEXT I%
300 '
310 SUM = 0                        'Normalize the low-pass filter kernel for
320 FOR I% = 0 TO 100              'unity gain at DC
330   SUM = SUM + H[I%]
340 NEXT I%
350 '
360 FOR I% = 0 TO 100
370   H[I%] = H[I%] / SUM
380 NEXT I%
390 '
400 FOR J% = 100 TO 4999           'Convolve the input signal & filter kernel
410   Y[J%] = 0
420   FOR I% = 0 TO 100
430     Y[J%] = Y[J%] + X[J%-I%] * H[I%]
440   NEXT I%
450 NEXT J%
460 '
470 END
```

TABLE 16-1

a stopband attenuation of at least -120dB (one part in one-million for those that refuse to learn decibels). As previously shown, the Blackman window only provides -74dB (one part in five-thousand). Fortunately, greater stopband attenuation is easy to obtain. The input signal can be filtered using a conventional windowed-sinc filter kernel, providing an intermediate signal. The intermediate signal can then be passed through the filter a second time, further increasing the stopband attenuation to -148dB (1 part in 30 million, wow!). It is also possible to combine the two stages into a single filter. The kernel of the combined filter is equal to the *convolution* of the filter kernels of the two stages. This also means that convolving any filter kernel *with itself* results in a filter kernel with a much improved stopband attenuation. The price you pay is a longer filter kernel and a slower roll-off. Figure 16-7a shows the frequency response of a 201 point low-pass filter, formed by convolving a 101 point Blackman windowed-sinc with itself. Amazing performance! (If you really need more than -100dB of stopband attenuation, you should use double precision. Single precision

```
100 'BAND-PASS WINDOWED-SINC FILTER
110 'This program calculates an 801 point band-pass filter kernel
120 '
130 DIM A[800]                      'A[ ]  workspace for the lower cutoff
140 DIM B[800]                      'B[ ]  workspace for the upper cutoff
150 DIM H[800]                      'H[ ]  holds the final filter kernel
160 '
170 PI = 3.1415926
180 M% = 800                        'Set filter kernel length (801 points)
190 '
200 '                               'Calculate the first low-pass filter kernel via Eq. 16-4,
210 FC = 0.196                      'with a cutoff frequency of 0.196, store in A[ ]
220 FOR I% = 0 TO 800
230   IF (I%-M%/2) = 0 THEN A[I%] = 2*PI*FC
240   IF (I%-M%/2) <> 0  THEN A[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
250   A[I%] = A[I%] * (0.42 - 0.5*COS(2*PI*I%/M%) + 0.08*COS(4*PI*I%/M%))
260 NEXT I%
270 '
280 SUM = 0                         'Normalize the first low-pass filter kernel for
290 FOR I% = 0 TO 800               'unity gain at DC
300   SUM = SUM + A[I%]
310 NEXT I%
320 '
330 FOR I% = 0 TO 800
340   A[I%] = A[I%] / SUM
350 NEXT I%
360 '                               'Calculate the second low-pass filter kernel via Eq. 16-4,
370 FC = 0.204                      'with a cutoff frequency of 0.204, store in B[ ]
380 FOR I% = 0 TO 800
390   IF (I%-M%/2) = 0 THEN B[I%] = 2*PI*FC
400   IF (I%-M%/2) <> 0  THEN B[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
410   B[I%] = B[I%] * (0.42 - 0.5*COS(2*PI*I%/M%) + 0.08*COS(4*PI*I%/M%))
420 NEXT I%
430 '
440 SUM = 0                         'Normalize the second low-pass filter kernel for
450 FOR I% = 0 TO 800               'unity gain at DC
460   SUM = SUM + B[I%]
470 NEXT I%
480 '
490 FOR I% = 0 TO 800
500   B[I%] = B[I%] / SUM
510 NEXT I%
520 '
530 FOR I% = 0 TO 800               'Change the low-pass filter kernel in B[ ] into a high-pass
540  B[I%] = - B[I%]                'filter kernel using spectral inversion (as in Fig. 14-5)
550 NEXT I%
560 B[400] =  B[400] + 1
570 '
580 '
590 FOR I% = 0 TO 800               'Add the low-pass filter kernel in A[ ], to the high-pass
600   H[I%] = A[I%] + B[I%]         'filter kernel in B[ ], to form a band-reject filter kernel
610 NEXT I%                         'stored in H[ ] (as in Fig. 14-8)
620 '
630 FOR I% = 0 TO 800               'Change the band-reject filter kernel into a band-pass
640   H[I%] = -H[I%]                'filter kernel by using spectral inversion
650 NEXT I%
660 H[400] =  H[400] + 1
670 '                               'The band-pass filter kernel now resides in H[ ]
680 END
```

TABLE 16-2

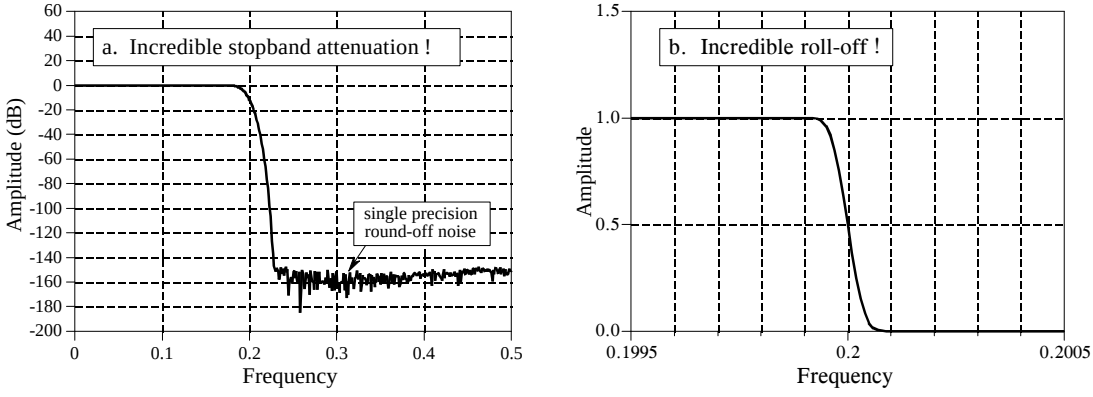

FIGURE 16-7
The incredible performance of the windowed-sinc filter. Figure (a) shows the frequency response of a windowed-sinc filter with increased stopband attenuation. This is achieved by convolving a windowed-sinc filter kernel with itself. Figure (b) shows the very rapid roll-off a 32,001 point windowed-sinc filter.

round-off noise on signals in the *passband* can erratically appear in the *stopband* with amplitudes in the -100dB to -120dB range).

Figure 16-7b shows another example of the windowed-sinc's incredible performance: a low-pass filter with 32,001 points in the kernel. The frequency response appears as expected, with a roll-off of 0.000125 of the sampling rate. How good is this filter? Try building an analog electronic filter that passes signals from DC to 1000 hertz with less than a 0.02% variation, and blocks all frequencies above 1001 hertz with less than 0.02% residue. Now that's a filter! If you really want to be impressed, remember that both the filters in Fig. 16-7 use *single precision*. Using *double precision* allows these performance levels to be extended by a *million* times.

The strongest limitation of the windowed-sinc filter is the *execution time*; it can be unacceptably long if there are many points in the filter kernel and standard convolution is used. A high-speed algorithm for this filter (FFT convolution) is presented in Chapter 18. Recursive filters (Chapter 19) also provide good frequency separation and are a reasonable alternative to the windowed-sinc filter.

Is the windowed-sinc the optimal filter kernel for separating frequencies? No, filter kernels resulting from more sophisticated techniques can be better. But beware! Before you jump into this very mathematical field, you should consider exactly what you hope to gain. The windowed-sinc will provide *any* level of performance that you could possibly need. What the advanced filter design methods may provide is a slightly shorter filter kernel for a given level of performance. This, in turn, may mean a slightly faster execution speed. Be warned that you may get little return for the effort expended.