excessive data word growth—particularly because sines and cosines are never greater than unity. However, significant data word growth can happen within an FFT butterfly.

(a) In our 8-bit number format scenario, what is the maximum possible decimal value of the real part of the complex output sample *C*?

(b) How many binary bits are needed for a storage register (memory location) to hold that maximum real part of the complex output sample *C*?

4.18 In 2006 the scientists at the Max Planck Institute for Radio Astronomy, in Bonn, Germany, built a hardware spectrum analyzer that performs 16384-point FFTs. This massively parallel analyzer performs 1.744×10^5 such FFTs per second. Assuming that the FFTs use the optimized decimation-in-frequency FFT butterfly structure, shown in Figure P4–18, and that the *A* and *B* samples are complex-valued, how many real-valued multiplies per second are being performed by the spectrum analyzer? Show your work.

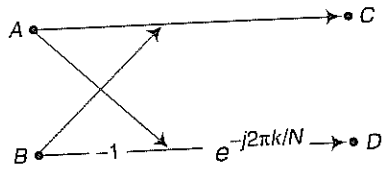


Figure P4–18

REFERENCES

[Pruned FFT 1] Nagai, K. "Pruning the Decimation-in-Time FFT Algorithm with Frequency Shift," *IEEE Trans. on ASSP*, Vol. ASSP-34, August 1986, pp. 1008–1010.

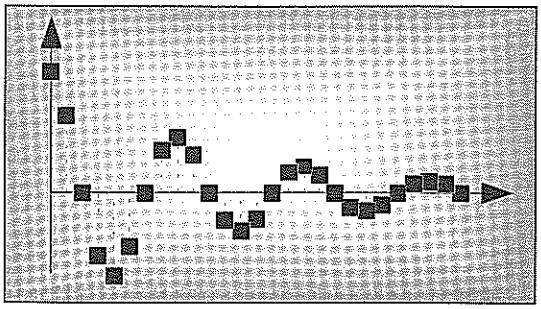
[Pruned FFT 2] Skinner, D. "Pruning the Decimation-in-Time FFT Algorithm," *IEEE Trans. on ASSP*, Vol. ASSP-24, April 1976, pp. 193–194.

[Pruned FFT 3] Markel, J. D. "FFT Pruning," *IEEE Trans. on Audio Electroacoustics*, Vol. AU-19, December 1971, pp. 305–311.

[Pruned FFT 4] Sreenivas, T., and Rao, P. "FFT Algorithm for Both Input and Ouput Pruning," *IEEE Trans. on ASSP*, Vol. ASSP-27, June 1979, pp. 291–292.

CHAPTER FIVE

Finite Impulse Response Filters



The filtering of digitized data, if not the most fundamental, is certainly the oldest discipline in the field of digital signal processing. Digital filtering's origins go back 50 years. The growing availability of digital computers in the early 1950s led to efforts in the smoothing of discrete sampled data and the analysis of discrete data control systems. However, it wasn't until the early to mid-1960s, around the time the Beatles came to America, that the analysis and development of digital equivalents of analog filters began in earnest. That's when digital signal processing experts realized that computers could go beyond the mere analysis of digitized signals into the domain of actually changing signal characteristics through filtering. Today, digital filtering is so widespread that the quantity of literature pertaining to it exceeds that of any other topic in digital signal processing. In this chapter, we introduce the fundamental attributes of digital filters, learn how to quantify their performance, and review the principles associated with the design of finite impulse response digital filters.

So let's get started by illustrating the concept of filtering a time-domain signal as shown in Figure 5–1.

In general, filtering is the processing of a time-domain signal resulting in some change in that signal's original spectral content. The change is usually the reduction, or filtering out, of some unwanted input spectral components; that is, filters allow certain frequencies to pass while attenuating other frequencies. Figure 5–1 shows both analog and digital versions of a filtering process. Where an analog filter operates on a continuous signal, a digital filter processes a sequence of discrete sample values. The digital filter in Figure 5–1(b), of course, can be a software program in a computer, a programmable hardware processor, or a dedicated integrated circuit. Traditional linear digital

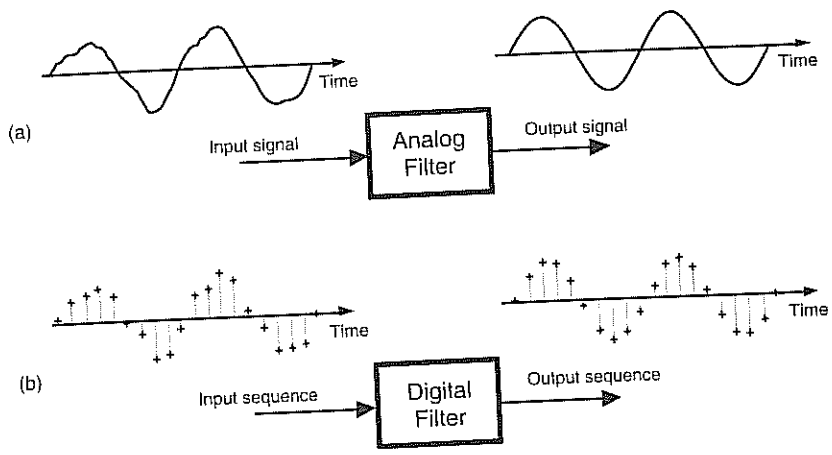


Figure 5-1 Filters: (a) an analog filter with a noisy tone input and a reduced-noise tone output; (b) the digital equivalent of the analog filter.

filters typically come in two flavors: *finite impulse response* (FIR) filters and *infinite impulse response* (IIR) filters. Because FIR filters are the simplest type of digital filter to analyze, we'll examine them in this chapter and cover IIR filters in Chapter 6.

5.1 AN INTRODUCTION TO FINITE IMPULSE RESPONSE (FIR) FILTERS

Given a finite duration of nonzero input values, an FIR filter will always have a finite duration of nonzero output values, and that's how FIR filters got their name. So, if the FIR filter's input suddenly becomes a sequence of all zeros, the filter's output will eventually be all zeros. While not sounding all that unusual, this characteristic is, however, very important, and we'll soon find out why, as we learn more about digital filters.

FIR filters use addition to calculate their outputs in a manner much the same as the process of averaging uses addition. In fact, averaging is a kind of FIR filter that we can illustrate with an example. Let's say we're counting the number of cars that pass over a bridge every minute, and we need to know the average number of cars per minute over five-minute intervals; that is, every minute we'll calculate the average number of cars/minute over the last five minutes. If the results of our car counting for the first ten minutes are those values shown in the center column of Table 5–1, then the average number of cars/minute over the previous five one-minute intervals is listed in the right column of the table. We've added the number of cars for the first five

Table 5-1 Values for the Averaging Example

Minute index	Number of cars/minute over the last minute	Number of cars/minute averaged over the last five minutes
1	10	–
2	22	–
3	24	–
4	42	–
5	37	27
6	77	40.4
7	89	53.8
8	22	53.4
9	63	57.6
10	9	52

one-minute intervals and divided by 5 to get our first five-minute average output value, (10+22+24+42+37)/5 = 27. Next we've averaged the number of cars/minute for the second to the sixth one-minute intervals to get our second five-minute average output of 40.4. Continuing, we average the number of cars/minute for the third to the seventh one-minute intervals to get our third average output of 53.8, and so on. With the number of cars/minute for the one-minute intervals represented by the dashed line in Figure 5–2, we show our five-minute average output as the solid line. (Figure 5–2 shows cars/minute input values beyond the first ten minutes listed in Table 5–1 to illustrate a couple of important ideas to be discussed shortly.)

There's much to learn from this simple averaging example. In Figure 5–2, notice that the sudden changes in our input sequence of cars/minute are flattened out by our averager. The averager output sequence is considerably smoother than the input sequence. Knowing that sudden transitions in a time sequence represent high-frequency components, we can say that our averager is behaving like a lowpass filter and smoothing sudden changes in the input. Is our averager an FIR filter? It sure is—no previous averager output value is used to determine a current output value; only input values are used to calculate output values. In addition, we see that, if the bridge were suddenly closed at the end of the 19th minute, the dashed line immediately goes to zero cars/minute at the end of the 20th minute, and the averager's output in

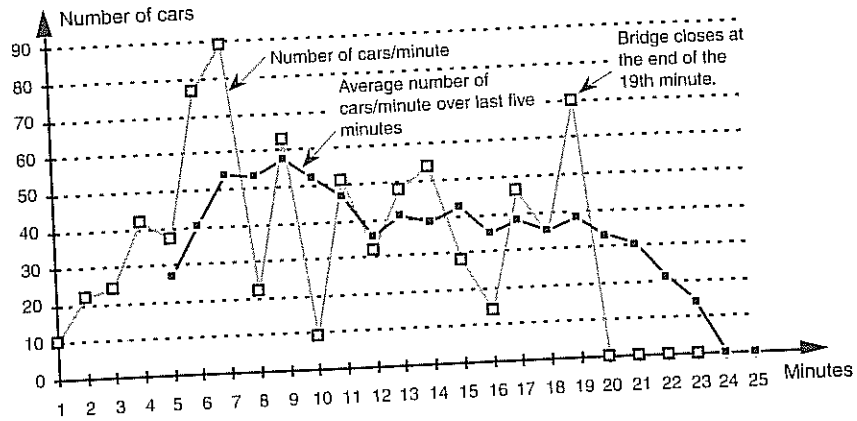


Figure 5-2 Averaging the number of cars/minute. The dashed line shows the individual cars/minute, and the solid line is the number of cars/minute averaged over the last five minutes.

Figure 5–2 approaches and settles to a value of zero by the end of the 24th minute.

Figure 5–2 shows the first averager output sample occurring at the end of the 5th minute because that's when we first have five input samples to calculate a valid average. The 5th output of our averager can be denoted as $y_{ave}(5)$ where

$$y_{ave}(5) = \frac{1}{5}[x(1) + x(2) + x(3) + x(4) + x(5)]. \tag{5-1}$$

In the general case, if the kth input sample is $x(k)$, then the nth output is

$$y_{ave}(n) = \frac{1}{5}[x(n-4) + x(n-3) + x(n-2) + x(n-1) + x(n)] = \frac{1}{5}\sum_{k=n-4}^{n} x(k). \tag{5-2}$$

Look at Eq. (5–2) carefully now. It states that the nth output is the average of the nth input sample and the four previous input samples.

We can formalize the digital filter nature of our averager by creating the block diagram in Figure 5–3 showing how the averager calculates its output samples.

This block diagram, referred to as the filter *structure*, is a physical depiction of how we might calculate our averaging filter outputs with the input sequence of values shifted, in order, from left to right along the top of the filter as new output calculations are performed. This structure, implementing Eqs. (5–1) and (5–2), shows those values used when the first five input sample values are available. The delay elements in Figure 5–3, called *unit delays*,

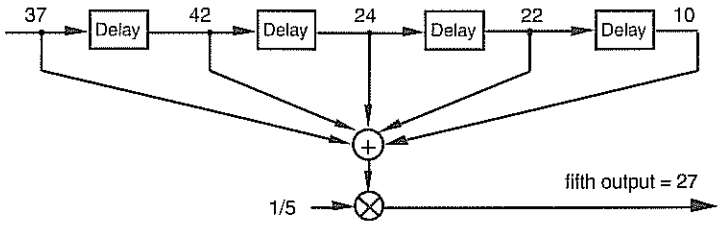


Figure 5–3 Averaging filter block diagram when the fifth input sample value, 37, is applied.

merely indicate a shift register arrangement where input sample values are temporarily stored during an output calculation.

In averaging, we add five numbers and divide the sum by 5 to get our answer. In a conventional FIR filter implementation, we can just as well multiply each of the five input samples by the coefficient 1/5 and then perform the summation as shown in Figure 5–4(a). Of course, the two methods in Figures 5–3 and 5–4(a) are equivalent because Eq. (5–2) describing the structure shown in Figure 5–3 is equivalent to

$$y_{ave}(n) = \frac{1}{5}x(n-4) + \frac{1}{5}x(n-3) + \frac{1}{5}x(n-2) + \frac{1}{5}x(n-1) + \frac{1}{5}x(n)$$

$$= \sum_{k=n-4}^{n} \frac{1}{5}x(k), \tag{5–3}$$

which describes the structure in Figure 5–4(a).†

Let's make sure we understand what's happening in Figure 5–4(a). Each of the first five input values is multiplied by 1/5, and the five products are summed to give the fifth filter output value. The left-to-right sample shifting is illustrated in Figures 5–4(b) and 5–4(c). To calculate the filter's sixth output value, the input sequence is right-shifted, discarding the first input value of 10, and the sixth input value, 77, is accepted on the left. Likewise, to calculate the filter's seventh output value, the input sequence is right-shifted, discarding the second value of 22, and the seventh input value, 89, arrives on the left. So, when a new input sample value is applied, the filter discards the oldest sample value, multiplies the samples by the coefficients of 1/5, and sums the products to get a single new output value. The filter's structure using this bucket brigade shifting process is often called a *transversal filter* due to the

† We've used the venerable distributive law for multiplication and addition of scalars, $a(b+c+d) = ab+ac+ad$, in moving Eq. (5–2)'s factor of 1/5 inside the summation in Eq. (5–3).

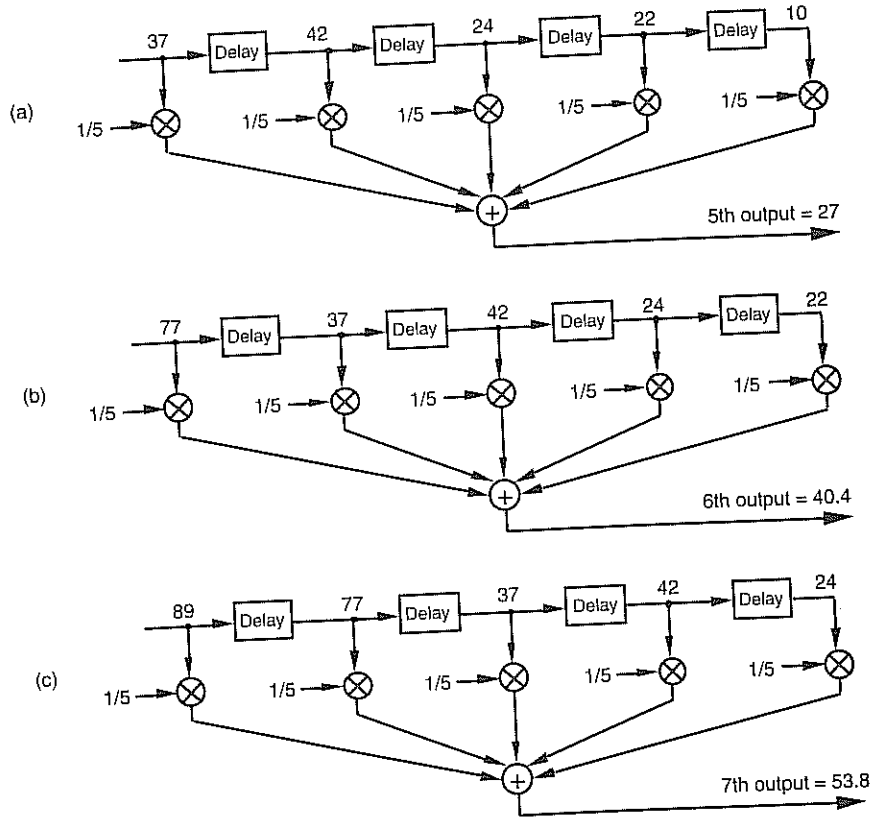


Figure 5-4 Alternate averaging filter structure: (a) input values used for the fifth output value; (b) input values used for the sixth output value; (c) input values used for the seventh output value.

cross-directional flow of the input samples. Because we *tap off* five separate input sample values to calculate an output value, the structure in Figure 5–4 is called a 5-tap tapped-delay line FIR filter, in digital filter vernacular.

One important and, perhaps, most interesting aspect of understanding FIR filters is learning how to predict their behavior when sinusoidal samples of various frequencies are applied to the input, i.e., how to estimate their frequency-domain response. Two factors affect an FIR filter's frequency response: the number of taps and the specific values used for the multiplication coefficients. We'll explore these two factors using our averaging example and, then, see how we can use them to design FIR filters. This brings us to the point where we have to introduce the C word: *convolution*. (Actually, we already slipped a convolution equation in on the reader without saying so. It was Eq. (5–3), and we'll examine it in more detail later.)

5.2 CONVOLUTION IN FIR FILTERS

OK, here's where we get serious about understanding the mathematics behind FIR filters. We can graphically depict Eq. (5–3)'s and Figure 5–4's calculations as shown in Figure 5–5. Also, let's be formal and use the standard notation of digital filters for indexing the input samples and the filter coefficients by

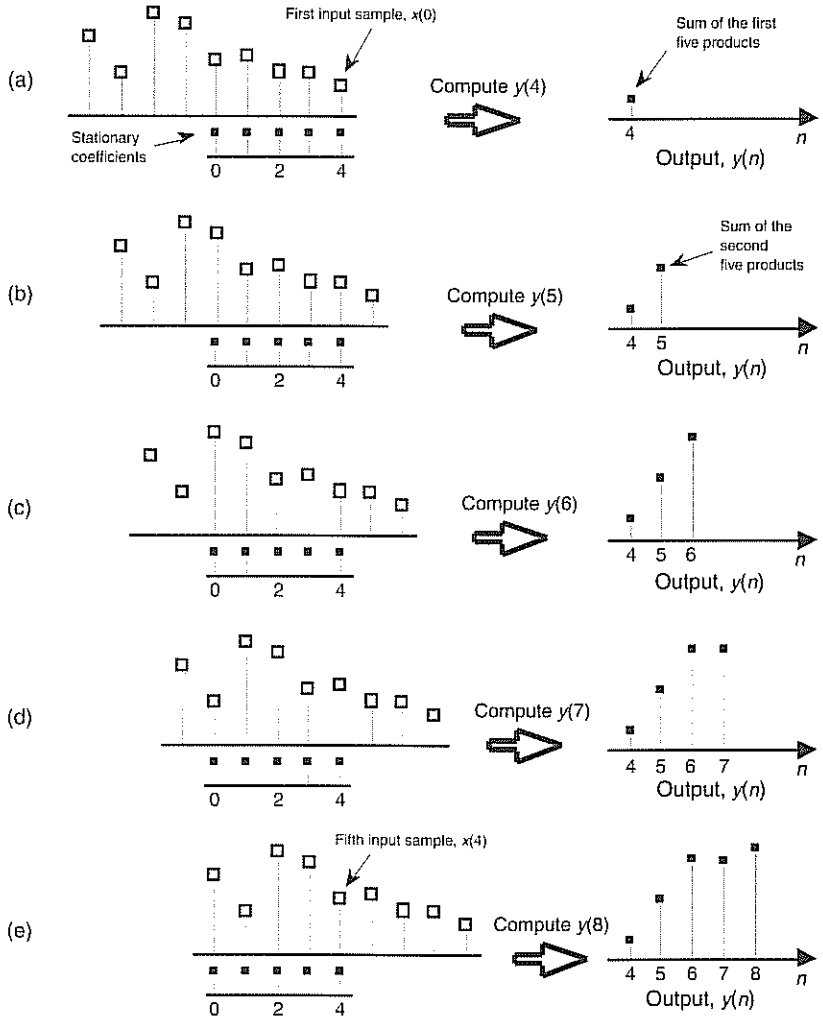


Figure 5–5 Averaging filter convolution: (a) first five input samples aligned with the stationary filter coefficients, index $n = 4$; (b) input samples shift to the right and index $n = 5$; (c) index $n = 6$; (d) index $n = 7$; (e) index $n = 8$.

starting with an initial index value of zero; that is, we'll call the initial input value the 0th sample $x(0)$. The next input sample is represented by the term $x(1)$, the following input sample is called $x(2)$, and so on. Likewise, our five coefficient values will be indexed from zero to four, $h(0)$ through $h(4)$. (This indexing scheme makes the equations describing our example consistent with conventional filter notation found in the literature.)

In Eq. (5–3) we used the factor of 1/5 as the filter coefficients multiplied by our averaging filter's input samples. The left side of Figure 5–5 shows the alignment of those coefficients, black squares, with the filter input sample values represented by the white squares. Notice in Figures 5–5(a) through 5–5(e) that we're marching the input samples to the right, and, at each step, we calculate the filter output sample value using Eq. (5–3). The output samples on the right side of Figure 5–5 match the first five values represented by the black squares in Figure 5–2. The input samples in Figure 5–5 are those values represented by the white squares in Figure 5–2. Notice that the time order of the inputs in Figure 5–5 has been reversed from the input sequence order in Figure 5–2! That is, the input sequence has been flipped in the time domain in Figure 5–5. This time order reversal is what happens to the input data using the filter structure in Figure 5–4.

Repeating the first part of Eq. (5–3) and omitting the subscript on the output term, our original FIR filter's $y(n)$th output is given by

$$y(n) = \frac{1}{5}x(n-4) + \frac{1}{5}x(n-3) + \frac{1}{5}x(n-2) + \frac{1}{5}x(n-1) + \frac{1}{5}x(n). \tag{5–4}$$

Because we'll explore filters whose coefficients are not all the same value, we need to represent the individual filter coefficients by a variable, such as the term $h(k)$, for example. Thus we can rewrite the averaging filter's output from Eq. (5–4) in a more general way as

$$y(n) = h(4)x(n-4) + h(3)x(n-3) + h(2)x(n-2) + h(1)x(n-1) + h(0)x(n)$$

$$= \sum_{k=0}^{4} h(k)\,x(n-k), \tag{5–5}$$

where $h(0)$ through $h(4)$ all equal 1/5. Equation (5–5) is a concise way of describing the filter structure in Figure 5–4 and the process illustrated in Figure 5–5.

Let's take Eq. (5–5) one step further and say, for a general M-tap FIR filter, the nth output is

$$y(n) = \sum_{k=0}^{M-1} h(k)\,x(n-k). \tag{5–6}$$

Well, there it is. Eq. (5–6) is the infamous convolution equation as it applies to digital FIR filters. Beginners in the field of digital signal processing often have trouble understanding the concept of convolution. It need not be that way. Eq. (5–6) is merely a series of multiplications followed by the addition of the products. The process is actually rather simple. We just flip the time order of an input sample sequence and start stepping the flipped sequence across the filter's coefficients as shown in Figure 5–5. For each new filter input sample, we sum a series of products to compute a single filter output value.

Let's pause for a moment and introduce a new term that's important to keep in mind, the *impulse response*. The impulse response of a filter is exactly what its name implies—it's the filter's output time-domain sequence when the input is a single unity-valued sample (impulse) preceded and followed by zero-valued samples. Figure 5–6 illustrates this idea in the same way we determined the filter's output sequence in Figure 5–5. The left side of Figure 5–6 shows the

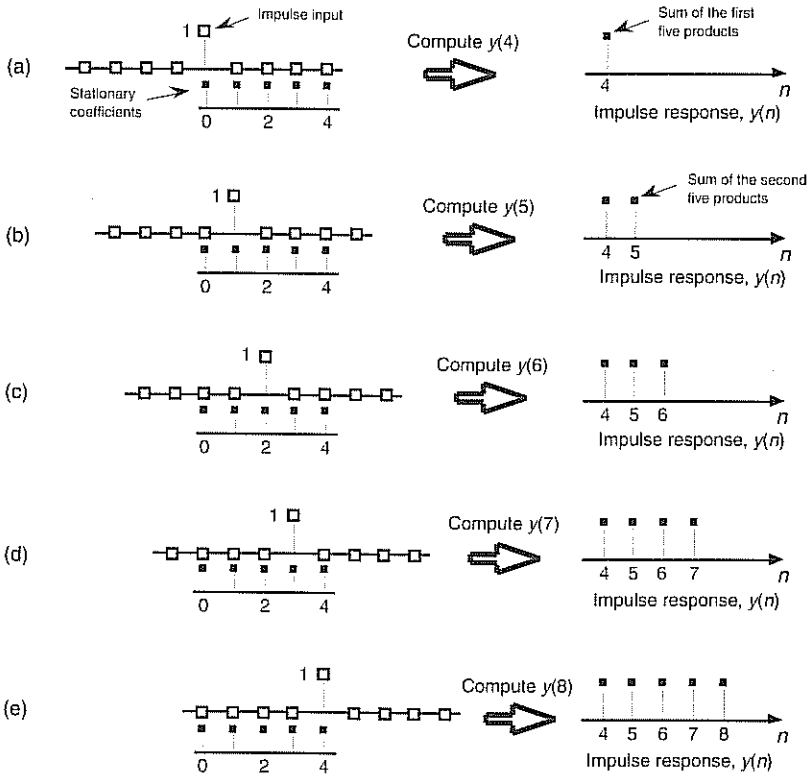


Figure 5–6 Convolution of filter coefficients and an input impulse to obtain the filter's output impulse response: (a) impulse sample aligned with the first filter coefficient, index $n = 4$; (b) impulse sample shifts to the right and index $n = 5$; (c) index $n = 6$; (d) index $n = 7$; (e) index $n = 8$.

alignment of the filter coefficients, black squares, with the filter input impulse sample values represented by the white squares. Again, in Figures 5–6(a) through 5–6(e) we're shifting the input samples to the right, and, at each step, we calculate the filter output sample value using Eq. (5–4). The output samples on the right side of Figure 5–6 are the filter's impulse response. Notice the key point here: the FIR filter's impulse response is identical to the five filter coefficient values. For this reason, the terms *FIR filter coefficients* and *impulse response* are synonymous. Thus, when someone refers to the impulse response of an FIR filter, they're also talking about the coefficients. Because there are a finite number of coefficients, the impulse response will be finite in time duration (finite impulse response, FIR).

Returning to our averaging filter, recall that coefficients (or impulse response) $h(0)$ through $h(4)$ were all equal to 1/5. As it turns out, our filter's performance can be improved by using coefficients whose values are not all the same. By "performance" we mean how well the filter passes desired signals and attenuates unwanted signals. We judge that performance by determining the shape of the filter's frequency-domain response that we obtain by the convolution property of linear systems. To describe this concept, let's repeat Eq. (5–6) using the abbreviated notation of

$$y(n) = h(k) * x(n) \tag{5–7}$$

where the * symbol means convolution. (Equation 5–7 is read as "y of n equals the convolution of h of k and x of n.") The process of convolution, as it applies to FIR filters, is as follows: the discrete Fourier transform (DFT) of the convolution of a filter's impulse response (coefficients) and an input sequence is equal to the product of the spectrum of the input sequence and the DFT of the impulse response. The idea we're trying to convey here is that if two time-domain sequences $h(k)$ and $x(n)$ have DFTs of $H(m)$ and $X(m)$, respectively, then the DFT of $y(n) = h(k) * x(n)$ is $H(m) \cdot X(m)$. Making this point in a more compact way, we state this relationship with the expression

$$y(n) = h(k) * x(n) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) \cdot X(m). \tag{5–8}$$

With IDFT indicating the inverse DFT, Eq. (5–8) indicates that two sequences resulting from $h(k)*x(n)$ and $H(m)\cdot X(m)$ are Fourier transform pairs. So taking the DFT of $h(k)*x(n)$ gives us the product $H(m)\cdot X(m)$ that is the spectrum of our filter output $Y(m)$. Likewise, we can determine $h(k)*x(n)$ by taking the inverse DFT of $H(m)\cdot X(m)$. The very important conclusion to learn from Eq. (5–8) is that convolution in the time domain is equivalent to multiplication in the frequency domain. To help us appreciate this principle, Figure 5–7 sketches the relationship between convolution in the time domain and multiplication in the frequency domain. The process of convolution with regard to linear systems is discussed in more detail in Section 5.9. The beginner is en

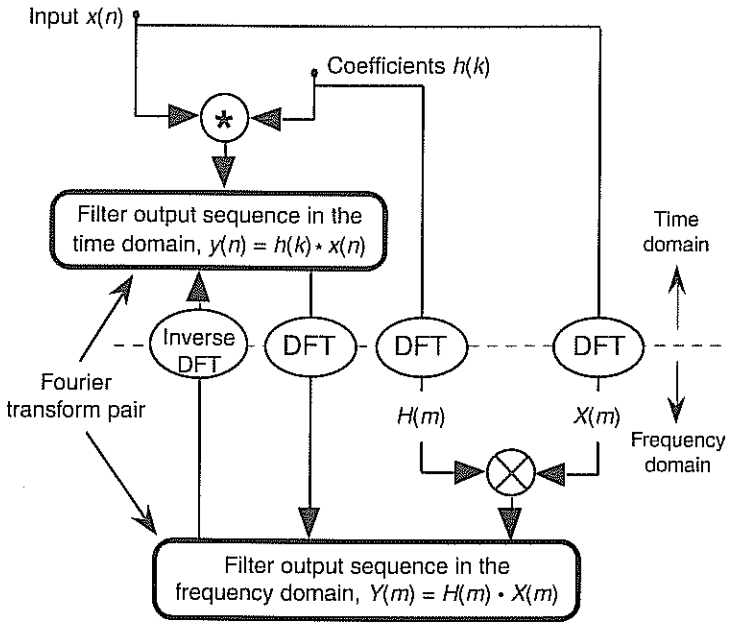


Figure 5-7 Relationships of convolution as applied to FIR digital filters.

couraged to review that material to get a general idea of why and when the convolution process can be used to analyze digital filters.

Equation (5–8) and the relationships in Figure 5–7 tell us what we need to do to determine the frequency response of an FIR filter. The product $X(m) \cdot H(m)$ is the DFT of the filter output. Because $X(m)$ is the DFT of the filter's input sequence, the frequency response of the filter is then defined as $H(m)$, the DFT of the filter's impulse response $h(k)$.[†] Getting back to our original problem, we can determine our averaging filter's frequency-domain response by taking the DFT of the individual filter coefficients (impulse response) in Eq. (5–4). If we take the five $h(k)$ coefficient values of 1/5 and append 59 zeros, we have the sequence depicted in Figure 5–8(a). Performing a 64-point DFT on that sequence, and normalizing the DFT magnitudes, gives us the filter's frequency magnitude response $|H(m)|$ in Figure 5–8(b) and phase response shown in Figure 5–8(c).[††] $H(m)$ is our old friend, the $\sin(x)/x$ function from Section 3.13.

[†] We use the term *impulse response* here, instead of *coefficients*, because this concept also applies to IIR filters. IIR filter frequency responses are also equal to the DFT of their impulse responses.

[††] There's nothing sacred about using a 64-point DFT here. We could just as well have appended only enough zeros to take a 16- or 32-point FFT. We chose 64 points to get a frequency resolution that would make the shape of the response in Figure 5–8(b) reasonably smooth. Remember, the more points in the FFT, the finer the frequency granularity—right?

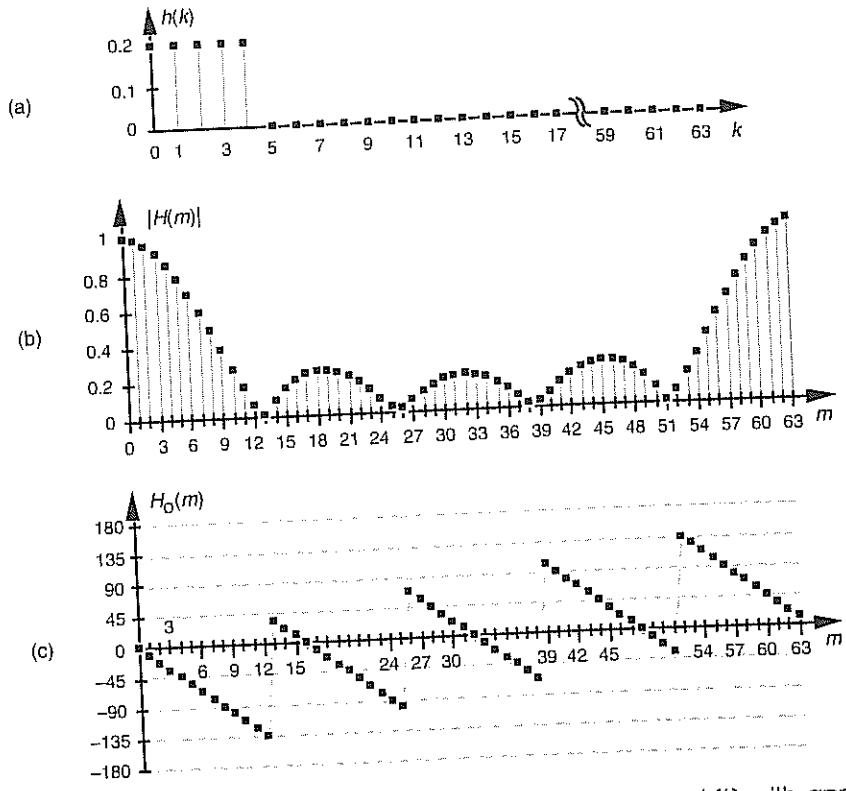


Figure 5–8 Averaging FIR filter: (a) filter coefficient sequence $h(k)$ with appended zeros; (b) normalized discrete frequency magnitude response $|H(m)|$ of the $h(k)$ filter coefficients; (c) phase-angle response of $H(m)$ in degrees.

Let's relate the discrete frequency response samples in Figures 5–8(b) and 5–8(c) to the physical dimension of the sample frequency f_s. We know, from Section 3.5 and our experience with the DFT, that the $m = N/2$ discrete frequency sample, $m = 32$ in this case, is equal to the folding frequency, or half the sample rate, $f_s/2$. Keeping this in mind, we can convert the discrete frequency axis in Figure 5–8 to that shown in Figure 5–9. In Figure 5–9(a), notice that the filter's magnitude response is, of course, periodic in the frequency domain with a period of the equivalent sample rate f_s. Because we're primarily interested in the filter's response between 0 and half the sample rate, Figure 5–9(c) shows that frequency band in greater detail, affirming the notion that averaging behaves like a lowpass filter. It's a relatively poor lowpass filter compared to an arbitrary, *ideal* lowpass filter indicated by the dashed lines in Figure 5–9(c), but our averaging filter will attenuate higher-frequency inputs relative to its response to low-frequency input signals.

We can demonstrate this by way of example. Suppose we applied a low-frequency sinewave to a 5-point averaging FIR filter as shown by the white

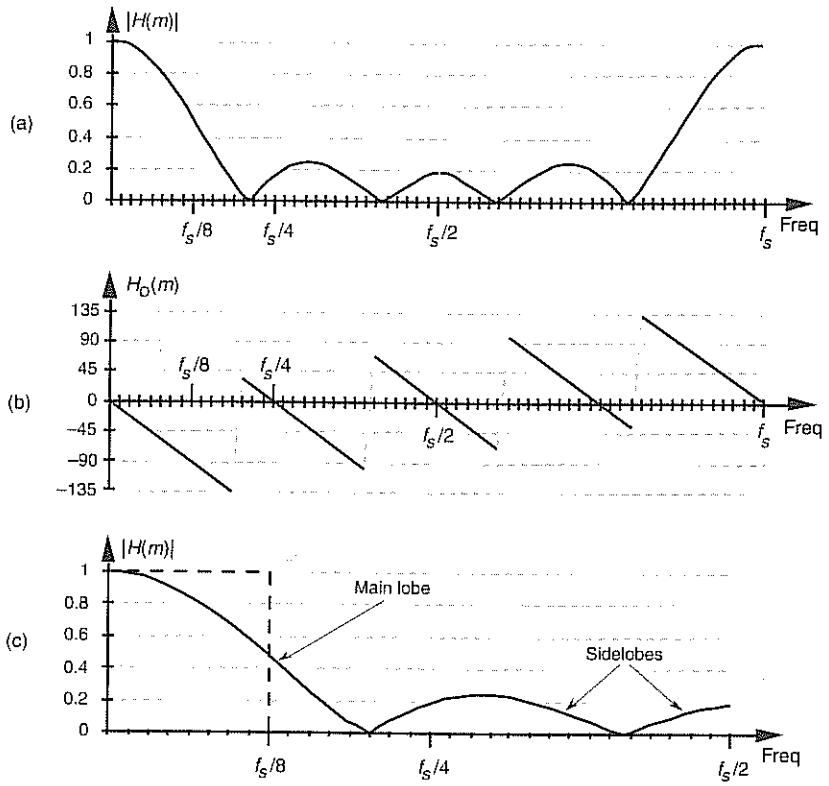


Figure 5-9 Averaging FIR filter frequency response shown as continuous curves: (a) normalized frequency magnitude response, $|H(m)|$; (b) phase-angle response of $H(m)$ in degrees; (c) the filter's magnitude response between zero Hz and half the sample rate, $f_s/2$ Hz.

squares in Figure 5–10(a). The input sinewave's frequency is $f_s/32$ Hz and its peak amplitude is unity. The filter's output sequence is shown by the black squares.

Figure 5–10(a) is rich in information! First, the filter's output is a sinewave of the same frequency as the input. This is a characteristic of a linear system. We apply a single sinewave input, and the output will be a single sinewave (shifted in phase and perhaps reduced in amplitude) of the same frequency as the input. Second, notice that the initial four output samples are not exactly sinusoidal. Those output samples are the *transient response* of the filter. With tapped-delay line FIR filters, the sample length of that transient response is equal to the number of filter unit-delay elements D, after which the filter's output begins its steady-state time response.

The above transient response property is important. It means that tapped-delay line FIR filter outputs are not valid until D+1 input samples have been applied to the filter. That is, the output samples are not valid until

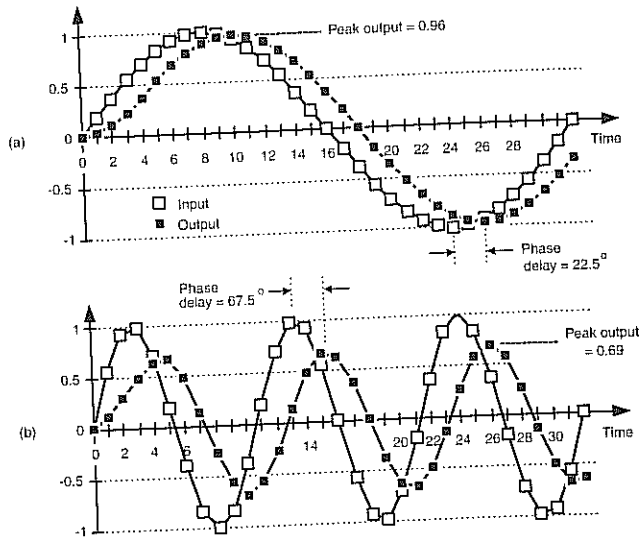


Figure 5-10 Averaging FIR filter input and output responses: (a) with an input sinewave of frequency $f_s/32$; (b) with an input sinewave of frequency $3f_s/32$.

the filter's delay line is filled with input data. So, for an FIR filter having D = 70 unit-delay elements the first 70 output samples are not valid and would be ignored in practice. WARNING: There are tapped-delay line FIR filters, used in practice, that have more unit-delay elements than nonzero-valued tap coefficients. The transient response length for those filters, measured in samples, is equal to the number of unit-delay elements, D (and is unrelated to the number of nonzero-valued tap coefficients).

The filter's output sinewave peak amplitude is reduced to a value of 0.96 and the output sinewave is delayed from the input by a phase angle of 22.5 degrees. Notice that the time delay between the input and output sinewaves, in Figure 5–10(a), is two samples in duration. (Although we discuss this time delay topic in more detail later, for now we'll just say that, because the filter's coefficients are symmetrical, the input/output delay measured in samples is equal to half the number of unit-delay elements in the filter's tapped-delay line.)

Next, if we applied a higher-frequency sinewave of $3f_s/32$ Hz to our 5-tap FIR filter as shown in Figure 5–10(b), the filter output is a sinewave of frequency $3f_s/32$ Hz and its peak amplitude is even further reduced to a value of 0.69. That's the nature of lowpass filters—they attenuate higher-frequency inputs more than they attenuate low-frequency inputs. As in Figure 5–10(a), the time delay between the input and output sinewaves, in Figure 5–10(b), is two samples in duration (corresponding to a phase-angle delay of 67.5 degrees). That property, where the input/output delay does not depend on frequency,

is a very beneficial property of FIR filters having symmetrical coefficients. We'll discuss this important issue again later in this chapter. In Figure 5–10(b) we see that the nonsinusoidal filter output transient response is even more obvious than it was in Figure 5–10(a).

Although the output amplitudes and phase delays in Figure 5–10 were measured values from actually performing a 5-tap FIR filter process on the input sinewaves' samples, we could have obtained those amplitude and phase delay values directly from Figures 5–8(b) and 5–8(c). The point is, we don't have to implement an FIR filter and apply various sinewave inputs to discover what its frequency response will be. We need merely take the DFT of the FIR filter's coefficients (impulse response) to determine the filter's frequency response as we did for Figure 5–8.

Figure 5–11 is another depiction of how well our 5-tap averaging FIR filter performs, where the dashed line is the filter's magnitude response $|H(m)|$, and the shaded line is the $|X(m)|$ magnitude spectrum of the filter's input values (the white squares in Figure 5–2). The solid line is the magnitude spectrum of the filter's output sequence, which is shown by the black squares in Figure 5–2. So in Figure 5–11, the solid output spectrum is the product of the dashed filter response curve and the shaded input spectrum, or $|X(m) \cdot H(m)|$. Again, we see that our averager does indeed attenuate the higher-frequency portion of the input spectrum.

Let's pause for a moment to let all of this soak in a little. So far we've gone through the averaging filter example to establish that

- FIR filters perform time-domain convolution by summing the products of the shifted input samples and a sequence of filter coefficients,

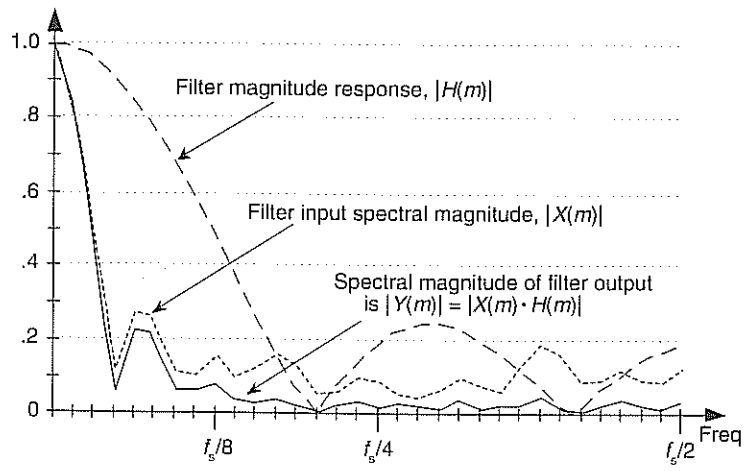


Figure 5–11 Averaging FIR filter input magnitude spectrum, frequency magnitude response, and output magnitude spectrum.

- an FIR filter's output sequence is equal to the convolution of the input sequence and a filter's impulse response (coefficients),
- an FIR filter's frequency response is the DFT of the filter's impulse response,
- an FIR filter's output spectrum is the product of the input spectrum and the filter's frequency response, and
- convolution in the time domain and multiplication in the frequency domain are Fourier transform pairs.

OK, here's where FIR filters start to get really interesting. Let's change the values of the five filter coefficients to modify the frequency response of our 5-tap lowpass filter. In fact, Figure 5–12(a) shows our original five filter coefficients and two other arbitrary sets of 5-tap coefficients. Figure 5–12(b) compares the frequency magnitude responses of those three sets of coefficients. Again, the frequency responses are obtained by taking the DFT of the three individual sets of coefficients and plotting the magnitude of the transforms, as we did for Figure 5–9(c). So we see three important characteristics in Figure 5–12. First, as we expected, different sets of coefficients give us different frequency magnitude responses. Second, a sudden change in the values of the coefficient sequence, such as the 0.2 to 0 transition in the first coefficient set, causes ripples, or sidelobes, in the frequency response. Third, if we minimize the suddenness of the changes in the coefficient values, such as the third set of coefficients in Figure 5–12(a), we reduce the sidelobe ripples in the frequency response. However, reducing the sidelobes results in increasing the main lobe width of our lowpass filter. (As we'll see, this is exactly the same effect encountered in the discussion of window functions used with the DFT in Section 3.9.)

To reiterate the function of the filter coefficients, Figure 5–13 shows the 5-tap FIR filter structure using the third set of coefficients from Figure 5–12. The implementation of constant-coefficient transversal FIR filters does not get any more complicated than that shown in Figure 5–13. It's that simple. We can have a filter with more than 5 taps, but the input signal sample shifting, the multiplications by the constant coefficients, and the summation are all there is to it. (By constant coefficients, we don't mean coefficients whose values are all the same; we mean coefficients whose values remain unchanged, or time invariant. There is a class of digital filters, called adaptive filters, whose coefficient values are periodically changed to adapt to changing input signal parameters. While we won't discuss these adaptive filters in this introductory text, their descriptions are available in the literature[1–5].)

So far, our description of an FIR filter implementation has been presented from a hardware perspective. In Figure 5–13, to calculate a single filter output sample, five multiplications and five additions must take place before the arrival of the next input sample value. In a software implementation of a 5-tap FIR filter, however, all of the input data samples would be

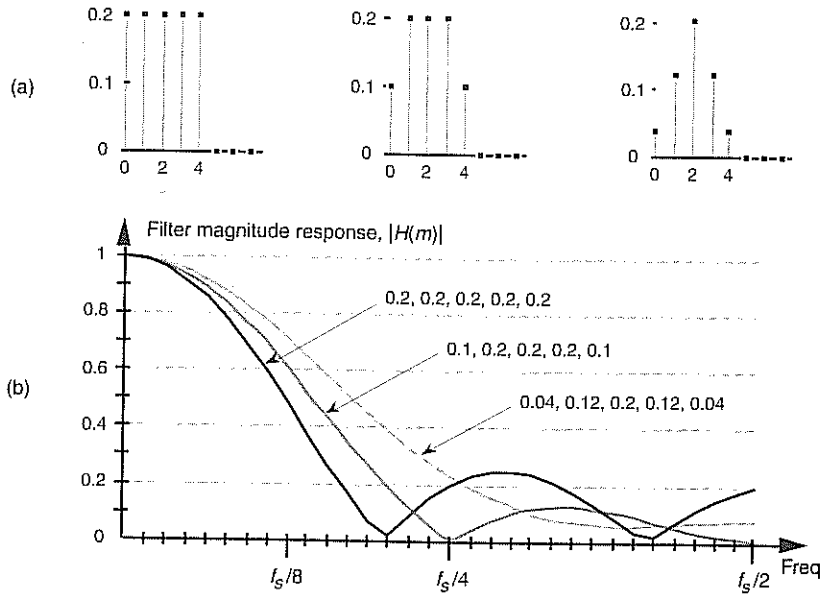


Figure 5-12 Three sets of 5-tap lowpass filter coefficients: (a) sets of coefficients: 0.2, 0.2, 0.2, 0.2, 0.2; 0.1, 0.2, 0.2, 0.2, 0.1; and 0.04, 0.12, 0.2, 0.12, 0.04; (b) frequency magnitude response of three lowpass FIR filters using those sets of coefficients.

previously stored in memory. The software filter routine's job, then, is to access different five-sample segments of the $x(n)$ input data space, perform the calculations shown in Figure 5–13, and store the resulting filter $y(n)$ output sequence in an array of memory locations.†

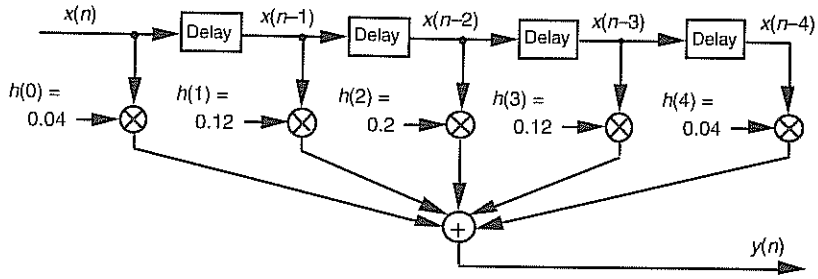


Figure 5-13 Five-tap lowpass FIR filter implementation using the coefficients 0.04, 0.12, 0.2, 0.12, and 0.04.

† In reviewing the literature of FIR filters, the reader will often find the term z^{-1} replacing the delay function in Figure 5–13. This equivalence is explained in the next chapter when we study IIR filters.

Now that we have a basic understanding of what a digital FIR filter is, let's see what effect is had by using more than 5 filter taps by learning to design FIR filters.

5.3 LOWPASS FIR FILTER DESIGN

OK, instead of just accepting a given set of FIR filter coefficients and analyzing their frequency response, let's reverse the process and design our own lowpass FIR filter. The design procedure starts with the determination of a *desired* frequency response followed by calculating the filter coefficients that will give us that response. There are two predominant techniques used to design FIR filters: the window method and the so-called optimum method. Let's discuss them in that order.

5.3.1 Window Design Method

The window method of FIR filter design (also called the Fourier series method) begins with our deciding what frequency response we want for our lowpass filter. We can start by considering a continuous lowpass filter, and simulating that filter with a digital filter. We'll define the continuous frequency response $H(f)$ to be ideal, i.e., a lowpass filter with unity gain at low frequencies and zero gain (infinite attenuation) beyond some *cutoff frequency*, as shown in Figure 5–14(a). Representing this $H(f)$ response by a discrete frequency response is straightforward enough because the idea of a discrete frequency response is essentially the same as a continuous frequency response—with one important difference. As described in Sections 2.2 and 3.13, discrete frequency-domain representations are always periodic with the

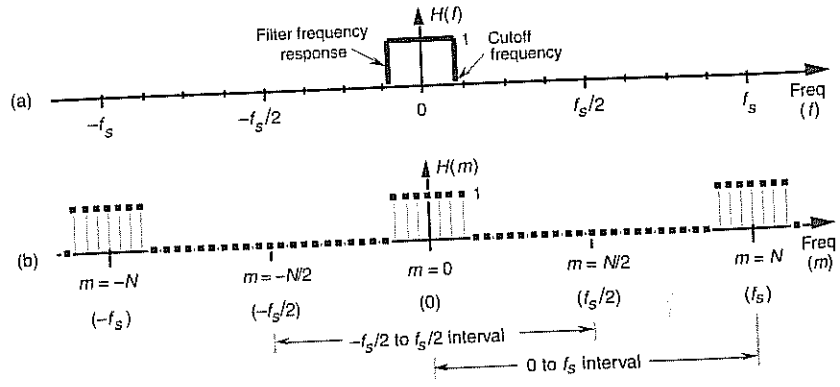


Figure 5–14 Lowpass filter frequency responses: (a) continuous frequency response $H(f)$; (b) periodic, discrete frequency response $H(m)$.

period being the sample rate f_s. The discrete representation of our ideal, continuous lowpass filter $H(f)$ is the periodic response $H(m)$ depicted by the frequency-domain samples in Figure 5–14(b).

We have two ways to determine our lowpass filter's time-domain coefficients. The first way is algebraic:

1. Develop an expression for the discrete frequency response $H(m)$.
2. Apply that expression to the inverse DFT equation to get the time domain $h(k)$.
3. Evaluate that $h(k)$ expression as a function of time index k.

The second method is to define the individual frequency-domain samples representing $H(m)$ and then have a software routine perform the inverse DFT of those samples, giving us the FIR filter coefficients. In either method, we need only define the periodic $H(m)$ over a single period of f_s Hz. As it turns out, defining $H(m)$ in Figure 5–14(b) over the frequency span $-f_s/2$ to $f_s/2$ is the easiest form to analyze algebraically, and defining $H(m)$ over the frequency span 0 to f_s is the best representation if we use the inverse DFT to obtain our filter's coefficients. Let's try both methods to determine the filter's time-domain coefficients.

In the algebraic method, we can define an arbitrary discrete frequency response $H(m)$ using N samples to cover the $-f_s/2$ to $f_s/2$ frequency range and establish K unity-valued samples for the passband of our lowpass filter as shown in Figure 5–15. To determine $h(k)$ algebraically we need to take the inverse DFT of $H(m)$ in the form of

$$h(k) = \frac{1}{N} \sum_{m=-(N/2)+1}^{N/2} H(m) e^{j2\pi mk/N}, \tag{5–9}$$

where our time-domain index is k. The solution to Eq. (5–9), derived in Section 3.13 as Eq. (3–59), is repeated here as

$$h(k) = \frac{1}{N} \cdot \frac{\sin(\pi kK/N)}{\sin(\pi k/N)}. \tag{5–10}$$

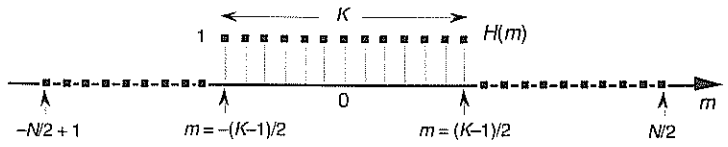


Figure 5–15 Arbitrary, discrete lowpass FIR filter frequency response defined over N frequency-domain samples covering the frequency range of f_s Hz.

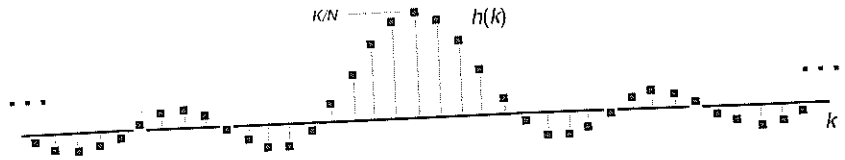


Figure 5-16 Time-domain $h(k)$ coefficients obtained by evaluating Eq. (5–10).

If we evaluate Eq. (5–10) as a function of k, we get the sequence shown in Figure 5–16, taking the form of the classic $\sin(x)/x$ function. By reviewing the material in Section 3.13, it's easy to see the great deal of algebraic manipulation required to arrive at Eq. (5–10) from Eq. (5–9). So much algebra, in fact, with its many opportunities for making errors, that digital filter designers like to avoid evaluating Eq. (5–9) algebraically. They prefer to use software routines to perform inverse DFTs (in the form of an inverse FFT) to determine $h_l(k)$, and so will we.

We can demonstrate the software inverse DFT method of FIR filter design with an example. Let's say we need to design a lowpass FIR filter simulating the continuous frequency response shown in Figure 5–17(a). The discrete representation of the filter's frequency response $H(m)$ is shown in Figure 5–17(b), where we've used $N = 32$ points to represent the frequency-domain variable $H(f)$. Because it's equivalent to Figure 5–17(b) but avoids the negative values of the frequency index m, we represent the discrete fre-

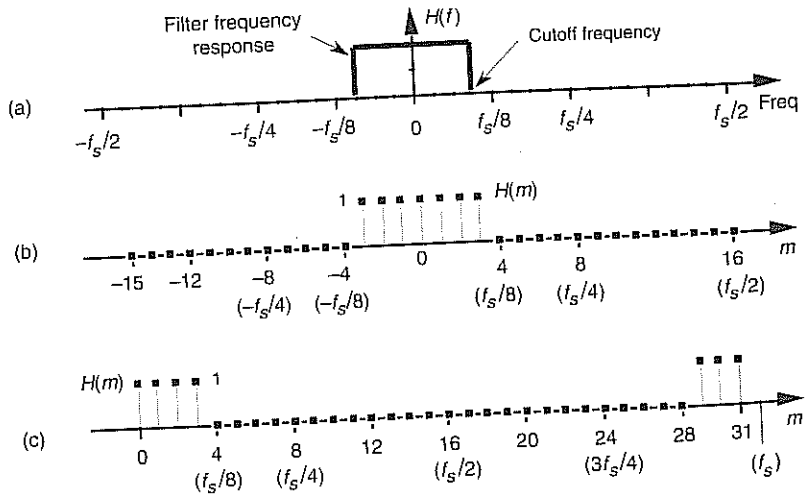


Figure 5-17 An ideal lowpass filter: (a) continuous frequency response $H(f)$; (b) discrete response $H(m)$ over the range $-f_s/2$ to $f_s/2$ Hz; (c) discrete response $H(m)$ over the range 0 to f_s Hz.

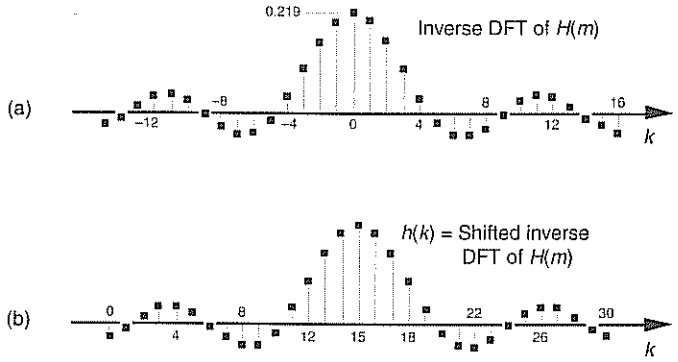


Figure 5-18 Inverse DFT of the discrete response in Figure 5-17(c): (a) normal inverse DFT indexing for k; (b) symmetrical coefficients used for a 31-tap lowpass FIR filter.

quency samples over the range 0 to f_s in Figure 5–17(c), as opposed to the $-f_s/2$ to $+f_s/2$ range in Figure 5–17(b). OK, we're almost there. Using a 32-point inverse FFT to implement a 32-point inverse DFT of the $H(m)$ sequence in Figure 5–17(c), we get the 32 $h(k)$ values depicted by the dots from $k = -15$ to $k = 16$ in Figure 5–18(a).† We have one more step to perform. Because we want our final 31-tap $h(k)$ filter coefficients to be symmetrical with their peak value in the center of the coefficient sample set, we drop the $k = 16$ sample and shift the k index to the left from Figure 5–18(a), giving us the desired $\sin(x)/x$ form of $h(k)$ as shown in Figure 5–18(b). This shift of the index k will not change the frequency magnitude response of our FIR filter. (Remember from our discussion of the DFT shifting theorem in Section 3.6 that a shift in the time domain manifests itself only as a linear phase shift in the frequency domain with no change in the frequency-domain magnitude.) The sequence in Figure 5–18(b), then, is now the coefficients we use in the convolution process of Figure 5–5 to implement a lowpass FIR filter.

It's important to demonstrate that the more $h(k)$ terms we use as filter coefficients, the closer we'll approximate our ideal lowpass filter response. Let's be conservative, just use the center nine $h(k)$ coefficients, and see what our filter response looks like. Again, our filter's magnitude response in this case will be the DFT of those nine coefficients as shown on the right side of Figure 5–19(a). The ideal filter's frequency response is also shown for reference as the dashed curve. (To show the details of its shape, we've used a continuous curve for $|H(m)|$ in Figure 5–19(a), but we have to remember that $|H(m)|$ is really a

† If you want to use this FIR design method but only have a forward FFT software routine available, Section 13.6 shows a slick way to perform an inverse FFT with the forward FFT algorithm.

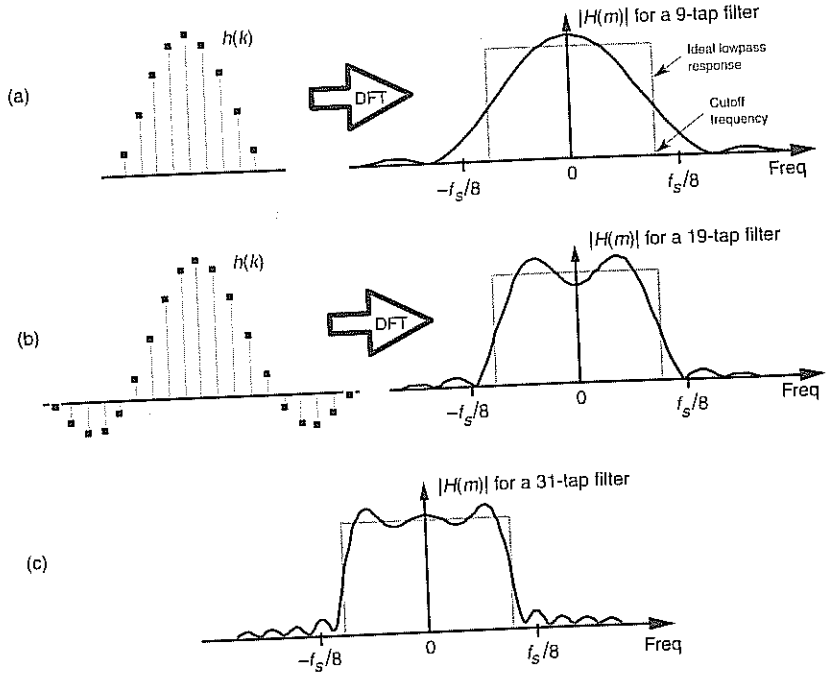


Figure 5–19 Coefficients and frequency responses of three lowpass filters: (a) 9-tap FIR filter; (b) 19-tap FIR filter; (c) frequency response of the full 31-tap FIR filter.

sequence of discrete values.) Notice that using nine coefficients gives us a low-pass filter, but it's certainly far from ideal. Using more coefficients to improve our situation, Figure 5–19(b) shows 19 coefficients and their corresponding frequency magnitude response that is beginning to look more like our desired rectangular response. Notice that magnitude fluctuations, or ripples, are evident in the passband of our $H(m)$ filter response. Continuing, using all 31 of the $h(k)$ values for our filter coefficients results in the frequency response in Figure 5–19(c). Our filter's response is getting better (approaching the ideal), but those conspicuous passband magnitude ripples are still present.

It's important that we understand why those passband ripples are in the lowpass FIR filter response in Figure 5–19. Recall the above discussion of convolving the 5-tap averaging filter coefficients, or impulse response, with an input data sequence to obtain the averager's output. We established that convolution in the time domain is equivalent to multiplication in the frequency domain, which we symbolized with Eq. (5–8) and repeat here as

$$h(k) * x(n) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) \cdot X(m). \tag{5–11}$$

This association between convolution in the time domain and multiplication in the frequency domain, sketched in Figure 5–7, indicates that if two time-domain sequences $h(k)$ and $x(n)$ have DFTs of $H(m)$ and $X(m)$, respectively, then the DFT of $h(k) * x(n)$ is $H(m) \cdot X(m)$. No restrictions whatsoever need be placed on what the time-domain sequences $h(k)$ and $x(n)$ in Eq. (5–11) actually represent. As detailed later in Section 5.9, convolution in one domain is equivalent to multiplication in the other domain, allowing us to state that multiplication in the time domain is equivalent to convolution in the frequency domain, or

$$h(k) \cdot x(n) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H(m) * X(m). \tag{5–12}$$

Now we're ready to understand why the magnitude ripples are present in Figure 5–19.

Rewriting Eq. (5–12) and replacing the $h(k)$ and $x(n)$ expressions with $h^{\infty}(k)$ and $w(k)$, respectively,

$$h^{\infty}(k) \cdot w(k) \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} H^{\infty}(m) * W(m). \tag{5–13}$$

Let's say that $h^{\infty}(k)$ represents an infinitely long $\sin(x)/x$ sequence of ideal lowpass FIR filter coefficients and that $w(k)$ represents a window sequence that we use to truncate the $\sin(x)/x$ terms as shown in Figure 5–20. Thus, the $w(k)$ sequence is a finite-length set of unity values and its DFT is $W(m)$. The length of $w(k)$ is merely the number of coefficients, or taps, we intend to use to implement our lowpass FIR filter. With $h^{\infty}(k)$ defined as such, the product

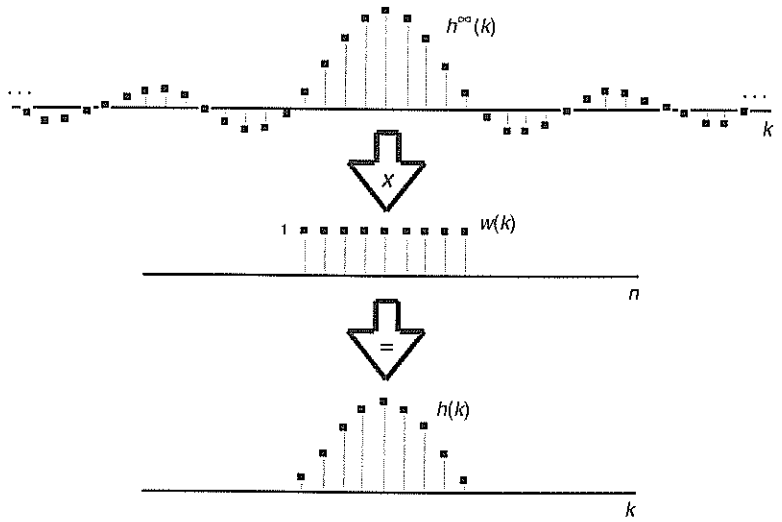


Figure 5-20 Infinite $h^{\infty}(k)$ sequence windowed by $w(k)$ to define the final filter coefficients $h(k)$.

$h^{\infty}(k) \cdot w(k)$ represents the truncated set of filter coefficients $h(k)$ in Figures 5–19(a) and 5–19(b). So, from Eq. (5–13), the FIR filter's true frequency response $H(m)$ is the convolution

$$H(m) = H^{\infty}(m) * W(m). \tag{5–14}$$

We depict this convolution in Figure 5–21 where, to keep the figure from being so busy, we show $H^{\infty}(m)$ (the DFT of the $h^{\infty}(k)$ coefficients) as the gray rectangle. Keep in mind that it's really a sequence of constant-amplitude sample values.

Let's look at Figure 5–21(a) very carefully to see why all three $|H(m)|$s exhibit passband ripple in Figure 5–19. We can view a particular sample value

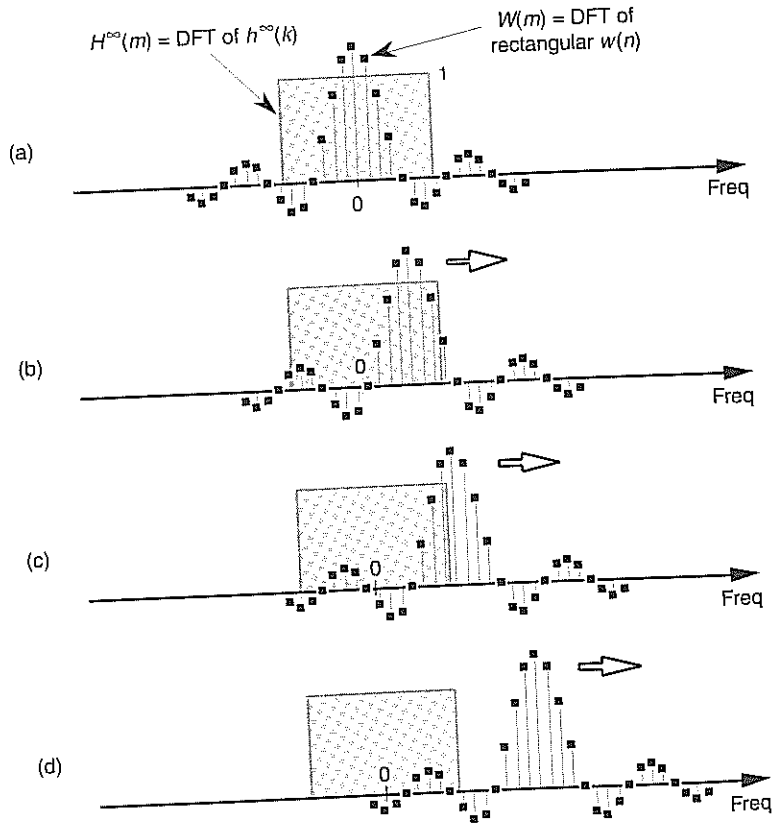


Figure 5-21 Convolution $W(m) * H^{\infty}(m)$: (a) unshifted $W(m)$ and $H^{\infty}(m)$; (b) shift of $W(m)$ leading to ripples within $H(m)$'s positive-frequency passband; (c) shift of $W(m)$ causing response roll-off near $H(m)$'s positive cutoff frequency; (d) shift of $W(m)$ causing ripples beyond $H(m)$'s positive cutoff frequency.

of the $H(m) = H^{\infty}(m) * W(m)$ convolution as being the sum of the products of $H^{\infty}(m)$ and $W(m)$ for a particular frequency shift of $W(m)$. $H^{\infty}(m)$ and the unshifted $W(m)$ are shown in Figure 5–21(a.) With an assumed value of unity for all of $H^{\infty}(m)$, a particular $H(m)$ value is now merely the sum of the $W(m)$ samples that overlap the $H^{\infty}(m)$ rectangle. So, with a $W(m)$ frequency shift of 0 Hz, the sum of the $W(m)$ samples that overlap the $H^{\infty}(m)$ rectangle in Figure 5–21(a) is the value of $H(m)$ at 0 Hz. As $W(m)$ is shifted to the right to give us additional positive-frequency $H(m)$ values, we can see that the sum of the positive and negative values of $W(m)$ under the rectangle oscillates during the shifting of $W(m)$. As the convolution shift proceeds, Figure 5–21(b) shows why there are ripples in the passband of $H(m)$—again, the sum of the positive and negative $W(m)$ samples under the $H^{\infty}(m)$ rectangle continues to vary as the $W(m)$ function is shifted. The $W(m)$ frequency shift, indicated in Figure 5–21(c), where the peak of $W(m)$'s main lobe is now outside the $H^{\infty}(m)$ rectangle, corresponds to the frequency where $H(m)$'s passband begins to roll off. Figure 5–21(d) shows that, as the $W(m)$ shift continues, there will be ripples in $H(m)$ beyond the positive cutoff frequency.[†] The point of all of this is that the ripples in $H(m)$ are caused by the sidelobes of $W(m)$.

Figure 5–22 helps us answer the question "How many $\sin(x)/x$ coefficients do we have to use (or how wide must $w(k)$ be) to get nice sharp falling edges and no ripples in our $H(m)$ passband?" The answer is that we can't get there from here. It doesn't matter how many $\sin(x)/x$ coefficients (filter taps) we use; there will always be filter passband ripple. As long as $w(k)$ is a finite number of unity values (i.e., a rectangular window of finite width), there will be sidelobe ripples in $W(m)$, and this will induce passband ripples in the final $H(m)$ frequency response. To illustrate that increasing the number of $\sin(x)/x$ coefficients doesn't reduce passband ripple, we repeat the 31-tap lowpass filter response in Figure 5–22(a). The frequency response, using 63 coefficients, is shown in Figure 5–22(b), and the passband ripple remains. We can make the filter's transition region narrower using additional $h(k)$ filter coefficients, but we cannot eliminate the passband ripple. That ripple, known as Gibbs's phenomenon, manifests itself anytime a function ($w(k)$ in this case) with an instantaneous discontinuity is represented by a Fourier series[6–8]. No finite set of sinusoids will be able to change fast enough to be exactly equal to an instantaneous discontinuity. Another way to state this Gibbs's dilemma is that, no matter how wide our $w(k)$ window is, $W(m)$ will always have sidelobe ripples. As shown in Figure 5–22(b), we can use more coefficients by extending the width of the rectangular $w(k)$ to narrow the filter transition region, but a wider $w(k)$ does not eliminate the filter passband ripple, nor does it even

[†] In Figure 5–21(b), had we started to shift $W(m)$ to the left in order to determine the negative-frequency portion of $H(m)$, we would have obtained the mirror image of the positive-frequency portion of $H(m)$.

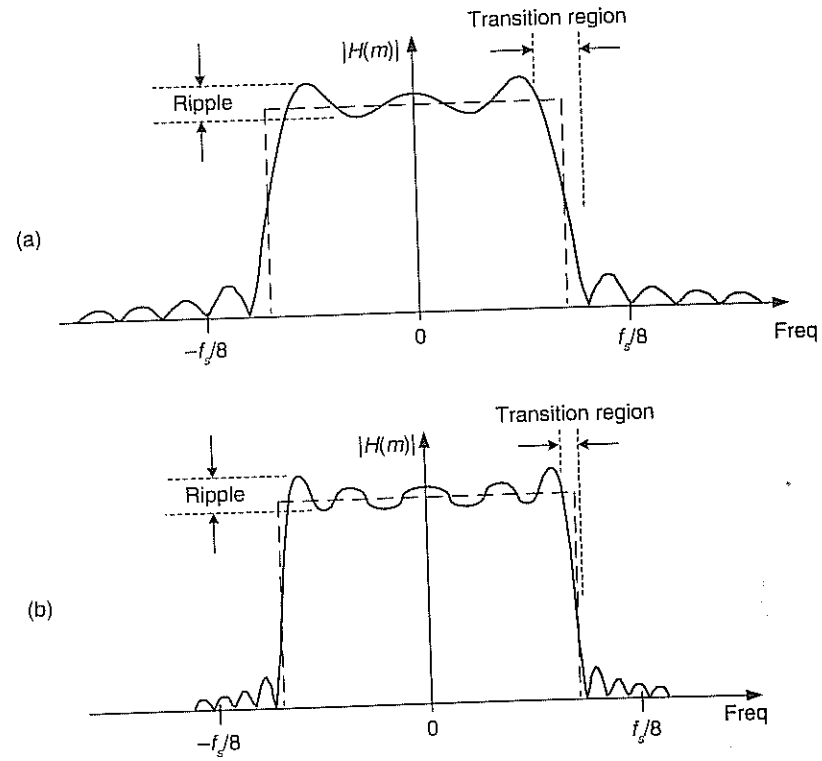


Figure 5-22 Passband ripple and transition regions: (a) for a 31-tap lowpass filter; (b) for a 63-tap lowpass filter.

reduce their peak-to-peak ripple magnitudes, as long as $w(k)$ has sudden discontinuities.

5.3.2 Windows Used in FIR Filter Design

OK. The good news is that we can minimize FIR passband ripple with window functions the same way we minimized DFT leakage in Section 3.9. Here's how. Looking back at Figure 5–20, by truncating the infinitely long $h^{\infty}(k)$ sequence through multiplication by the rectangular $w(k)$, our final $h(k)$ exhibited ripples in the frequency-domain passband. Figure 5–21 shows us that the passband ripples were caused by $W(m)$'s sidelobes that, in turn, were caused by the sudden discontinuities from zero to one and one to zero in $w(k)$. If we think of $w(k)$ in Figure 5–20 as a rectangular window, then it is $w(k)$'s abrupt amplitude changes that are the source of our filter passband ripple. The window FIR design method is the technique of reducing $w(k)$'s discontinuities by using window functions other than the rectangular window.

Consider Figure 5–23 to see how a nonrectangular window function can be used to design low-ripple FIR digital filters. Imagine if we replaced Figure

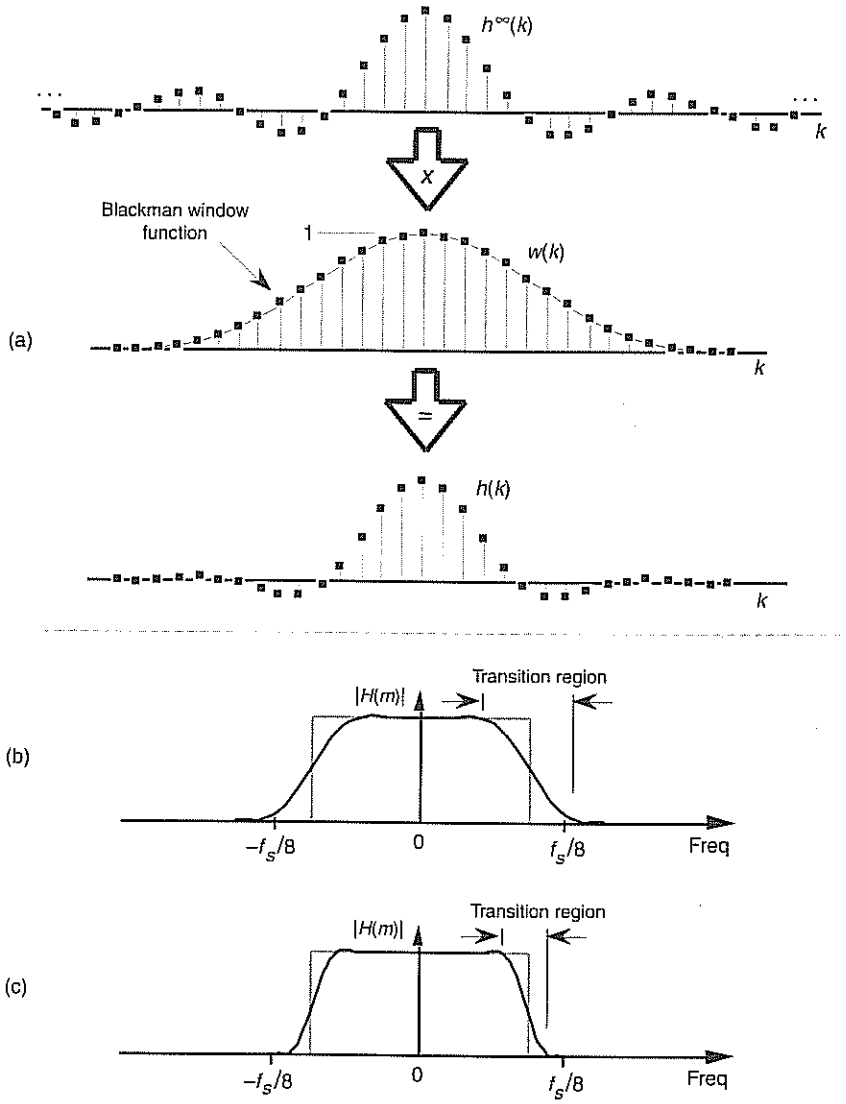


Figure 5-23 Coefficients and frequency response of a 31-tap Blackman-windowed FIR filter: (a) defining the windowed filter coefficients $h(k)$; (b) low-ripple 31-tap frequency response; (c) low-ripple 63-tap frequency response.

5–20's rectangular $w(k)$ with the Blackman window function whose discrete values are defined as

$$w(k) = 0.42 - 0.5\cos\left(\frac{2\pi k}{N-1}\right) + 0.08\cos\left(\frac{4\pi k}{N-1}\right), \text{for } k = 0,1,2,\ldots,N-1. \quad (5\text{–}15)$$

This situation is depicted for $N = 31$ in Figure 5–23(a), where Eq. (5–15)'s $w(k)$ looks very much like the Hanning window function in Figure 3–17(a). This Blackman window function results in the 31 smoothly tapered $h(k)$ coefficients at the bottom of Figure 5–23(a). Notice two things about the resulting $H(m)$ in Figure 5–23(b). First, the good news. The passband ripples are greatly reduced from those evident in Figure 5–22(a)—so our Blackman window function did its job. Second, the price we paid for reduced passband ripple is a wider $H(m)$ transition region. We can get a steeper filter response roll-off by increasing the number of taps in our FIR filter. Figure 5–23(c) shows the improved frequency response had we used a 63-coefficient Blackman window function for a 63-tap FIR filter. So using a nonrectangular window function reduces passband ripple at the expense of slower passband to stopband roll-off.

A graphical comparison of the frequency responses for the rectangular and Blackman windows is provided in Figure 5–24. (The curves in Figure 5–24 were obtained for the window functions defined by 16 discrete samples,

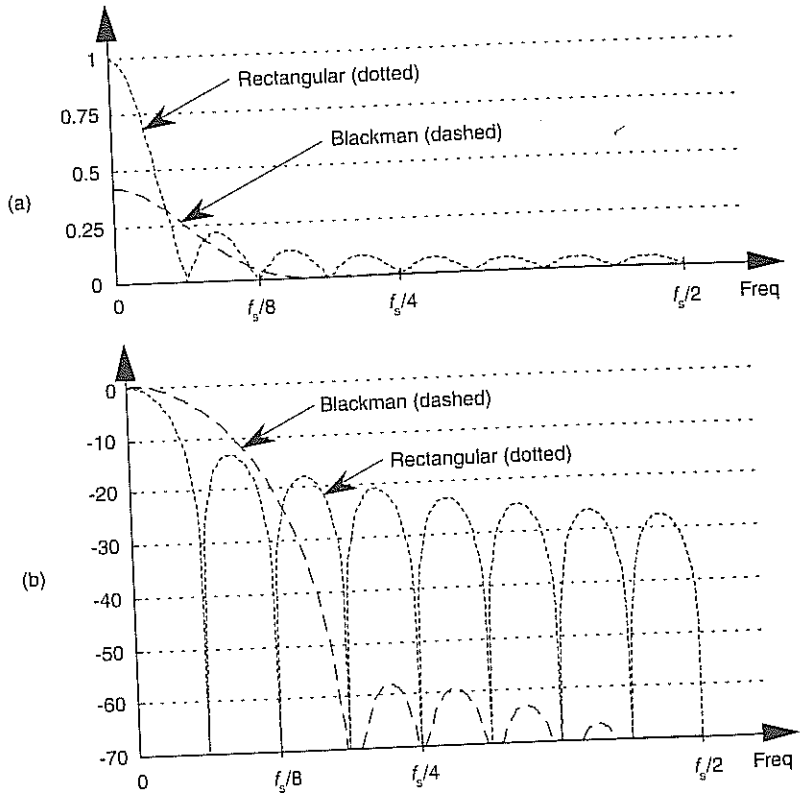


Figure 5-24 Rectangular versus Blackman window frequency magnitude responses: (a) $|W(m)|$ on a linear scale; (b) normalized logarithmic scale of $W_{dB}(m)$.

to which 496 zeros were appended, applied to a 512-point DFT.) The sidelobe magnitudes of the Blackman window's $|W(m)|$ are too small to see on a linear scale. We can see those sidelobe details by plotting the two windows' frequency responses on a logarithmic scale and normalizing each plot so that their main lobe peak values are both zero dB. For a given window function, we can get the log magnitude response of $W_{dB}(m)$ by using the expression

$$W_{dB}(m) = 20 \cdot \log_{10}\left(\frac{|W(m)|}{|W(0)|}\right). \tag{5-16}$$

(The $|W(0)|$ term in Eq. (5–16) is the magnitude of $W(m)$ at the peak of the main lobe when $m = 0$.) Figure 5–24(b) shows us the greatly reduced sidelobe levels of the Blackman window and how that window's main lobe is almost three times as wide as the rectangular window's main lobe.

Of course, we could have used any of the other window functions, discussed in Section 3.9, for our lowpass FIR filter. That's why this FIR filter design technique is called the window design method. We pick a window function and multiply it by the $\sin(x)/x$ values from $H^{\infty}(m)$ in Figure 5–23(a) to get our final $h(k)$ filter coefficients. It's that simple. Before we leave the window method of FIR filter design, let's introduce two other interesting window functions.

Although the Blackman window and those windows discussed in Section 3.9 are useful in FIR filter design, we have little control over their frequency responses; that is, our only option is to select some window function and accept its corresponding frequency response. Wouldn't it be nice to have more flexibility in trading off, or striking a compromise between, a window's main lobe width and sidelobe levels? Fortunately, there are two popular window functions that give us this opportunity. Called the Chebyshev (or Dolph-Chebyshev) and the Kaiser window functions, they're defined by the following formidable expressions:

Chebyshev window:→ (also called the Dolph–Chebyshev and the Tchebyschev window)

$$w(k) = \text{the } N\text{-point inverse DFT of } \frac{\cos\left[N \cdot \cos^{-1}\left[\alpha \cdot \cos\left(\pi \frac{m}{N}\right)\right]\right]}{\cosh[N \cdot \cosh^{-1}(\alpha)]}, \tag{5-17}$$

$$\text{where } \alpha = \cosh\left(\frac{1}{N}\cosh^{-1}(10^{\gamma})\right) \text{ and } m = 0, 1, 2, \ldots, N$$

Kaiser window:→ (also called the Kaiser–Bessel window)

$$w(k) = \frac{I_o\left[\beta\sqrt{1-\left(\frac{k-p}{p}\right)^2}\right]}{I_o(\beta)}, \tag{5-18}$$

$$\text{for } k = 0, 1, 2, \ldots, N\text{–}1, \text{ and } p = (N\text{–}1)/2.$$

Two typical Chebyshev and Kaiser window functions and their frequency magnitude responses are shown in Figure 5–25. For comparison, the rectangular and Blackman window functions are also shown in that figure. (Again, the curves in Figure 5–25(b) were obtained for window functions defined by 32 discrete time samples, with 480 zeros appended, applied to a 512-point DFT.)

Equation (5–17) was originally based on the analysis of antenna arrays using the mathematics of Chebyshev polynomials[9–11]. Equation (5–18) evolved from Kaiser's approximation of prolate spheroid functions using zeroth-order Bessel functions[12–13]. For each sample of the N-length sequence inside the brackets of the numerator of Eq. (5–18), as well as for the

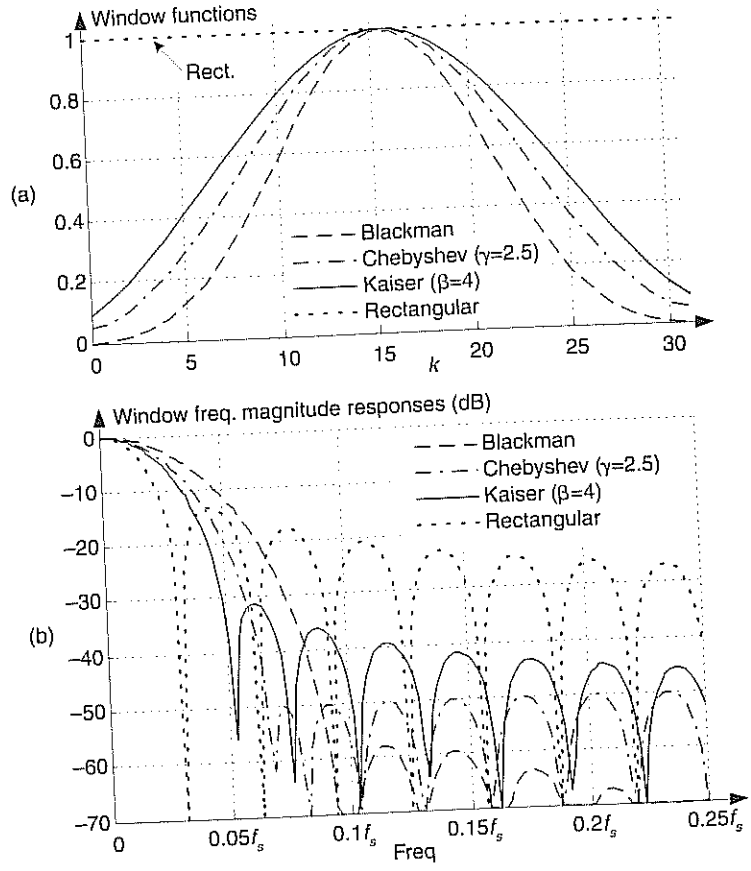


Figure 5–25 Typical window functions used with digital filters: (a) window coefficients in the time domain; (b) frequency-domain magnitude responses in dB.

β term in the denominator, the $I_0(x)$ zeroth-order Bessel function values can be approximated using

$$I_0(x) = \sum_{q=0}^{24} \frac{x^{2q}}{4^q \cdot (q!)^2}. \quad (5\text{–}18')$$

In theory the upper limit of the summation in Eq. (5–18′) should be infinity but, fortunately, 25 summations give us sufficient accuracy when evaluating $I_0(x)$.

Don't be intimidated by the complexity of Eqs. (5–17) and (5–18)—at this point, we need not be concerned with the mathematical details of their development. We just need to realize that the γ and β control parameters give us control over the Chebyshev and Kaiser windows' main lobe widths and the sidelobe levels.

Let's see how this works for Chebyshev window functions, having four separate values of γ, and their frequency responses shown in Figure 5–26. FIR filter designers applying the window method typically use predefined software routines to obtain their Chebyshev window coefficients. Commercial digital signal processing software packages allow the user to specify three things: the window function (Chebyshev in this case), the desired number of coefficients (the number of taps in the FIR filter), and the value of γ. Selecting different values for γ enables us to adjust the sidelobe levels and see what effect those values have on main lobe width, a capability that we didn't have with the Blackman window or the window functions discussed in Section 3.9. The Chebyshev window function's stopband attenuation, in dB, is equal to

$$Atten_{\text{Cheb}} = -20\gamma. \quad (5\text{–}19)$$

So, for example, if we needed our sidelobe levels to be no greater than –60 dB below the main lobe, we use Eq. (5–19) to establish a γ value of 3.0 and let the software generate the Chebyshev window coefficients.†

The same process applies to the Kaiser window, as shown in Figure 5–27. Commercial software packages allow us to specify β in Eq. (5–18) and provide us with the associated window coefficients. The curves in Figure 5–27(b), obtained for Kaiser window functions defined by 32 discrete samples, show that we can select the desired sidelobe levels and see what effect this has on the main lobe width.

Chebyshev or Kaiser, which is the best window to use? It depends on the application. Returning to Figure 5–25(b), notice that, unlike the constant

† By the way, some digital signal processing software packages require that we specify $Atten_{\text{Cheb}}$ in decibels instead of γ. That way, we don't have to bother using Eq. (5–19) at all.

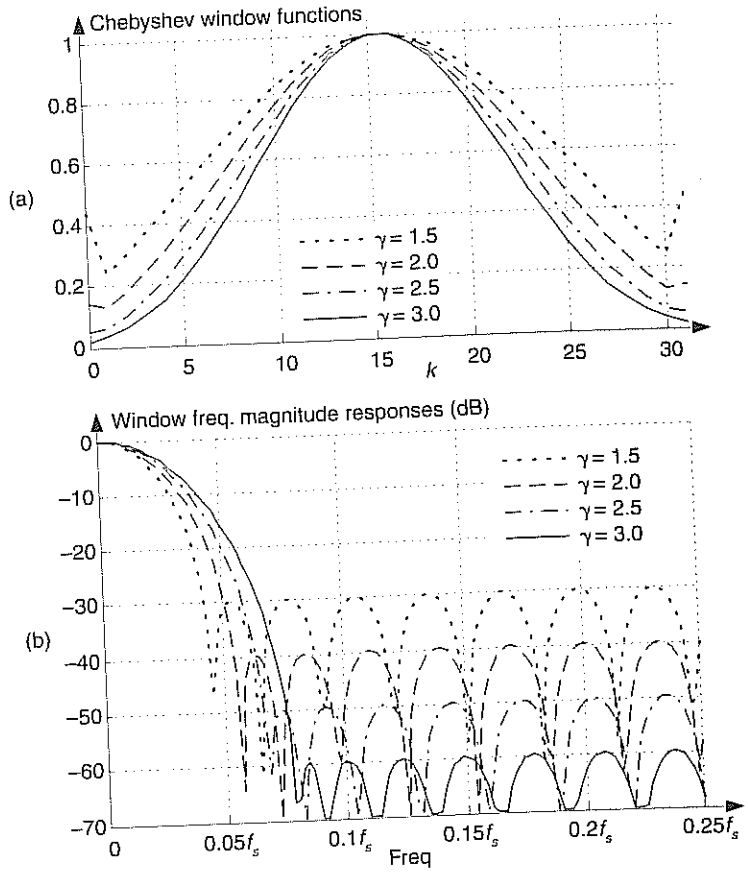


Figure 5-26 Chebyshev window functions for various γ values: (a) window coefficients in the time domain; (b) frequency-domain magnitude responses in dB.

sidelobe peak levels of the Chebyshev window, the Kaiser window's sidelobes decrease with increased frequency. However, the Kaiser sidelobes are higher than the Chebyshev window's sidelobes near the main lobe. Our primary trade-off here is trying to reduce the sidelobe levels without broadening the main lobe too much. Digital filter designers typically experiment with various values of γ and β for the Chebyshev and Kaiser windows to get the optimum $W_{dB}(m)$ for a particular application. (For that matter, the Blackman window's very low sidelobe levels outweigh its wide main lobe in many applications.) For some reason, algorithms for computing Chebyshev window functions are not readily available in the literature of DSP. To remedy that situation, Appendix I presents a straightforward procedure for computing N-sample Chebyshev window sequences.

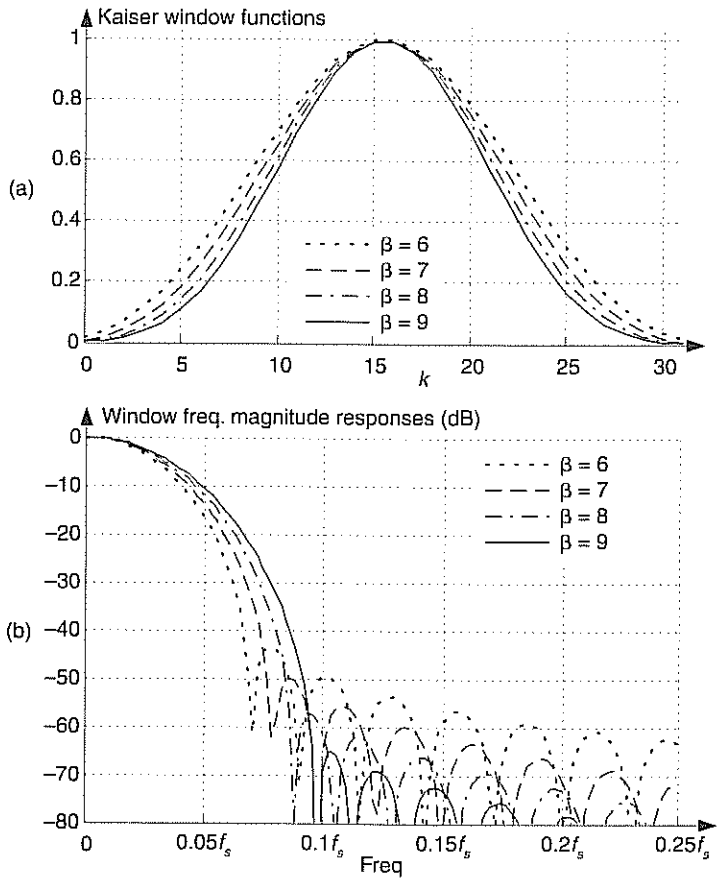


Figure 5-27 Kaiser window functions for various β values: (a) window coefficients in the time domain; (b) frequency-domain magnitude responses in dB.

To conclude this section, remember that different window functions have their own individual advantages and disadvantages for FIR filter design. Regardless of the non-rectangular window function used, they always decrease an FIR filter's passband ripple over that of the rectangular window. For the enthusiastic reader, a thorough discussion of many window functions can be found in reference [14].

5.4 BANDPASS FIR FILTER DESIGN

The window method of lowpass FIR filter design can be used as the first step in designing a bandpass FIR filter. Let's say we want a 31-tap FIR filter with the frequency response shown in Figure 5–22(a), but instead of being centered

about zero Hz, we want the filter's passband to be centered about $f_s/4$ Hz. If we define a lowpass FIR filter's coefficients as $h_{lp}(k)$, our problem is to find the $h_{bp}(k)$ coefficients of a bandpass FIR filter. As shown in Figure 5–28, we can shift $H_{lp}(m)$'s frequency response by multiplying the filter's $h_{lp}(k)$ lowpass coefficients by a sinusoid of $f_s/4$ Hz. That sinusoid is represented by the $s_{shift}(k)$ sequence in Figure 5–28(a), whose values are a sinewave sampled at a rate of four samples per cycle. Our final 31-tap $h_{bp}(k)$ FIR bandpass filter coefficients are

$$h_{bp}(k) = h_{lp}(k) \cdot s_{shift}(k)\,, \tag{5–20}$$

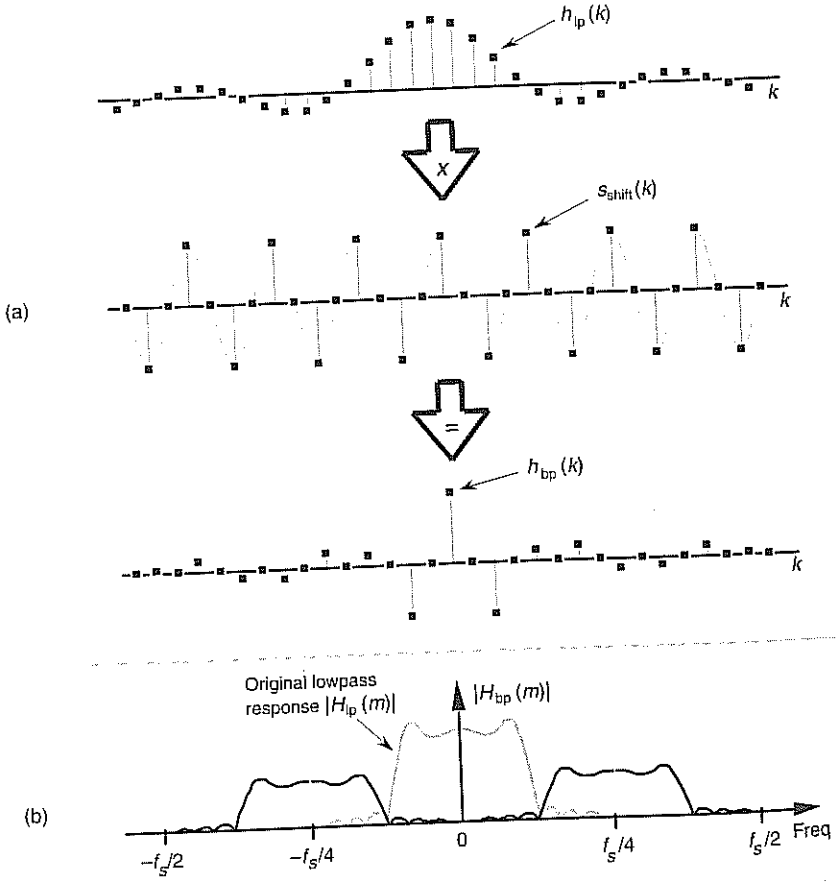


Figure 5–28 Bandpass filter with frequency response centered at $f_s/4$: (a) generating 31-tap filter coefficients $h_{bp}(k)$; (b) frequency magnitude response $|H_{bp}(m)|$.

whose frequency magnitude response $|H_{bp}(m)|$ is shown as the solid curves in Figure 5–28(b). The actual magnitude of $|H_{bp}(m)|$ is half that of the original $|H_{lp}(m)|$ because half the values in $h_{bp}(k)$ are zero when $s_{shift}(k)$ corresponds exactly to $f_s/4$. This effect has an important practical implication. It means that, when we design an N-tap bandpass FIR filter centered at a frequency of $f_s/4$ Hz, we only need to perform approximately $N/2$ multiplications for each filter output sample. (There's no reason to multiply an input sample value, $x(n-k)$, by zero before we sum all the products from Eq. (5–6) and Figure 5–13, right? We just don't bother to perform the unnecessary multiplications at all.) Of course, when the bandpass FIR filter's center frequency is other than $f_s/4$, we're forced to perform the full number of N multiplications for each FIR filter output sample.

Notice, here, that the $h_{lp}(k)$ lowpass coefficients in Figure 5–28(a) have not been multiplied by any window function. In practice, we'd use an $h_{lp}(k)$ that has been windowed prior to implementing Eq. (5–20) to reduce the passband ripple. If we wanted to center the bandpass filter's response at some frequency other than $f_s/4$, we merely need to modify $s_{shift}(k)$ to represent sampled values of a sinusoid whose frequency is equal to the desired bandpass center frequency. That new $s_{shift}(k)$ sequence would then be used in Eq. (5–20) to get the new $h_{bp}(k)$.

5.5 HIGHPASS FIR FILTER DESIGN

Going one step further, we can use the bandpass FIR filter design technique to design a highpass FIR filter. To obtain the coefficients for a highpass filter, we need only modify the shifting sequence $s_{shift}(k)$ to make it represent a sampled sinusoid whose frequency is $f_s/2$. This process is shown in Figure 5–29. Our final 31-tap highpass FIR filter's $h_{hp}(k)$ coefficients are

$$\begin{aligned} h_{hp}(k) &= h_{lp}(k) \cdot s_{shift}(k) \\ &= h_{lp}(k) \cdot (1, -1, 1, -1, 1, -1, \text{etc.}), \end{aligned} \tag{5–21}$$

whose $|H_{hp}(m)|$ frequency response is the solid curve in Figure 5–29(b). Because $s_{shift}(k)$ in Figure 5–29(a) has alternating plus and minus ones, we can see that $h_{hp}(k)$ is merely $h_{lp}(k)$ with the sign changed for every other coefficient. Unlike $|H_{bp}(m)|$ in Figure 5–28(b), the $|H_{hp}(m)|$ response in Figure 5–29(b) has the same amplitude as the original $|H_{lp}(m)|$.

Again, notice that the $h_{lp}(k)$ lowpass coefficients in Figure 5–29(a) have not been modified by any window function. In practice, we'd use a windowed $h_{lp}(k)$ to reduce the passband ripple before implementing Eq. (5–21).

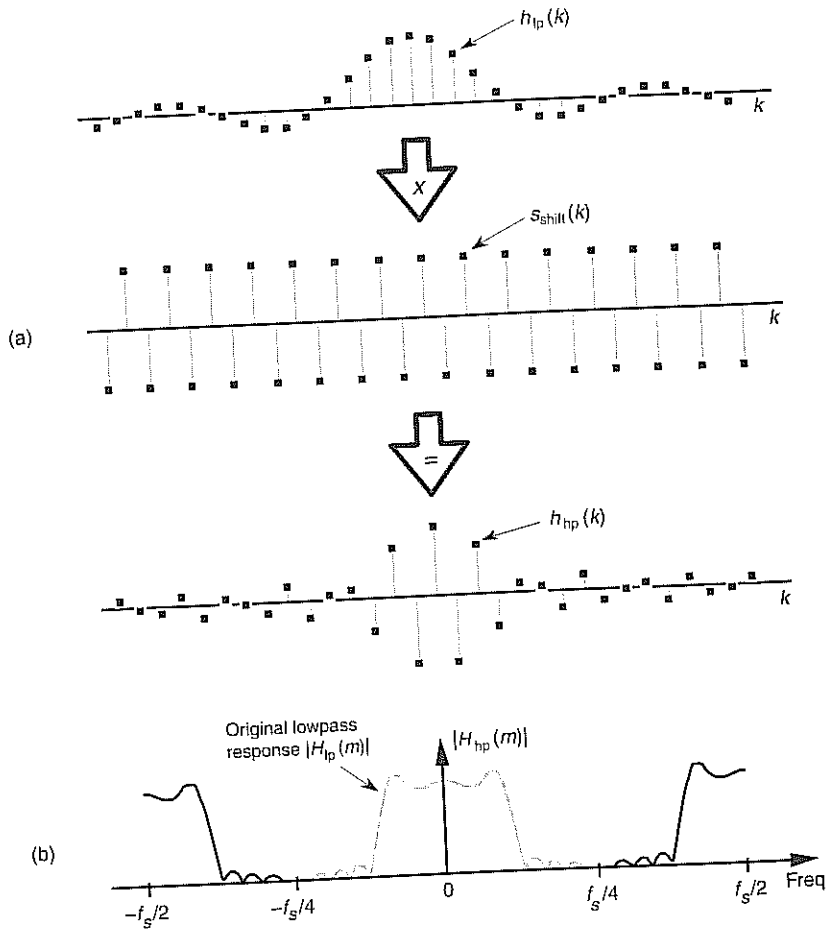


Figure 5-29 Highpass filter with frequency response centered at $f_s/2$: (a) generating 31-tap filter coefficients $h_{hp}(k)$; (b) frequency magnitude response $|H_{hp}(m)|$.

5.6 PARKS-MCCLELLAN EXCHANGE FIR FILTER DESIGN METHOD

Let's introduce one last FIR filter design technique that has found wide acceptance in practice. The Parks-McClellan FIR filter design method (also called the Remez Exchange, or Optimal method[†]) is a popular technique used to design high-performance FIR filters. To use this design method, we have to visualize a desired frequency response $H_d(m)$ like that shown in Figure 5–30.

[†] Remez is pronounced re-'mā.

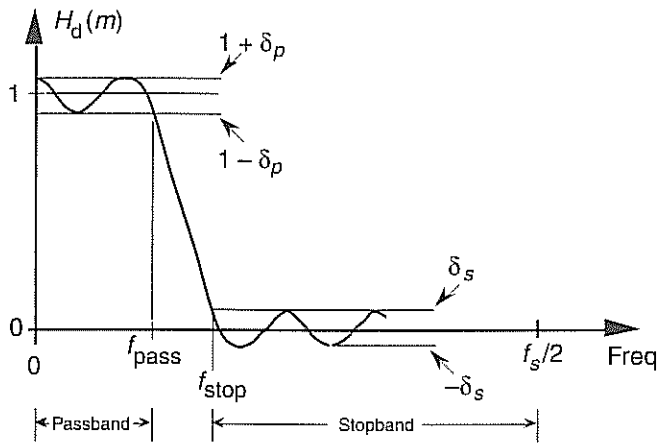


Figure 5-30 Desired frequency response definition of a lowpass FIR filter using the Parks-McClellan Exchange design method.

We have to establish a desired passband cutoff frequency f_{pass} and the frequency where the attenuated stopband begins, f_{stop}. In addition, we must establish the variables δ_p and δ_s that define our desired passband and stopband ripple. Passband and stopband ripples, in decibels, are related to δ_p and δ_s by[15]

$$\text{Passband ripple} = 20 \cdot \log_{10}(1 + \delta_p) \tag{5–22}$$

and

$$\text{Stopband ripple} = -20 \cdot \log_{10}(\delta_s). \tag{5–22′}$$

(Some of the early journal papers describing the Parks-McClellan design method used the equally valid expression $-20 \cdot \log_{10}(\delta_p)$ to define the passband ripple in dB. However, Eq. (5–22) is the most common form used today.) Next, we apply these parameters to a computer software routine that generates the filter's N time-domain $h(k)$ coefficients where N is the minimum number of filter taps to achieve the desired filter response.

On the other hand, some software Parks-McClellan routines assume that we want δ_p and δ_s to be as small as possible and require us only to define the desired values of the $H_d(m)$ response as shown by the solid black dots in Figure 5–31. The software then adjusts the values of the undefined (shaded dots) values of $H_d(m)$ to minimize the error between our desired and actual frequency response while minimizing δ_p and δ_s. The filter designer has the option to define some of the $H_d(m)$ values in the transition band, and the software calculates the remaining undefined $H_d(m)$ transition band values. With this version of the Parks-McClellan algorithm, the issue of most