7

Integer Filters

Jon D. Pfeffer

When digital filters must operate in a real-time environment, many filter designs become unsatisfactory due to the amount of required computation time. A considerable reduction in computation time is achieved by replacing floating-point coefficients with small integer coefficients in filter equations. This increases the speed at which the resulting filter program executes by replacing floating-point multiplication instructions with integer bit-shift and add instructions. These instructions require fewer clock cycles than floating-point calculations.

Integer filters are a special class of digital filters that have only integer coefficients in their defining equations. This characteristic leads to some design constraints that can often make it difficult to achieve features such as a sharp cutoff frequency. Since integer filters can operate at higher speeds than traditional designs, they are often the best type of filter for high sampling rates or when using a slow microprocessor.

This chapter discusses basic design concepts of integer filters. It also reviews why these filters display certain characteristics, such as linear phase, and how to derive the transfer functions. Next this theory is extended to show how to design low-pass, high-pass, bandpass, and band-reject integer filters by appropriate selection of pole and zero locations. Then a discussion shows how certain aspects of filter performance can be improved by cascading together integer filters. Next a recent design method is summarized which is more complicated but adds flexibility to the design constraints. Finally, a lab provides hands-on experience in the design and use of integer filters.

7.1 BASIC DESIGN CONCEPT

Lynn (1977) presented the best known techniques for integer filter design used today by showing a methodology to design low-pass, high-pass, bandpass, and band-reject filters with integer coefficients. This method is summarized in several steps. First place a number of evenly spaced zeros around the unit circle. These zeros completely attenuate the frequencies corresponding to their locations. Next choose poles that also lie on the unit circle to exactly cancel some of the zeros. When a pole cancels a zero, the frequency corresponding to this location is no longer attenuated. Since each point on the unit circle is representative of a frequency, the locations of the poles and zeros determine the frequency response of the filter. These filters are restricted types of recursive filters.

7.1.1 General form of transfer function

The general form of the filter transfer function used by Lynn for this application is:

$$H_1(z) = \frac{[1 - z^{-m}]^p}{[1 - 2\cos(\theta)z^{-1} + z^{-2}]^t} \tag{7.1a}$$

$$H_2(z) = \frac{[1 + z^{-m}]^p}{[1 - 2\cos(\theta)z^{-1} + z^{-2}]^t} \tag{7.1b}$$

The *m* exponent represents how many evenly spaced zeros to place around the unit circle. The angle θ represents the angular locations of the poles. The powers *p* and *t* represent the order of magnitude of the filter, which has a direct bearing on its gain and on the attenuation of the sidelobes. Raising *p* and *t* by equal integral amounts has the effect of cascading identical filters. If the filter is to be useful and physically realizable, *p* and *t* must both be nonnegative integers. The effects of raising *p* and *t* to values greater than one are further described in section 7.5.

7.1.2 Placement of poles

The denominator of this transfer function comes from the multiplication of a pair of complex-conjugate poles that always lie exactly on the unit circle. Euler's relation, $e^{j\theta} = \cos(\theta) + j\sin(\theta)$, shows that all values of $e^{j\theta}$ lie on the unit circle. Thus we can derive the denominator as follows:

$$\text{Denominator} = (z - e^{j\theta})(z - e^{-j\theta}) \tag{7.2a}$$

Multiplying the two factors, we get

$$\text{Denominator} = z^2 - (e^{j\theta} + e^{-j\theta})z + (e^{j\theta}e^{-j\theta}) \tag{7.2b}$$

Using the identity,

$$\cos(\theta) = \frac{e^{j\theta} + e^{-j\theta}}{2} \tag{7.2c}$$

We arrive at

$$\text{Denominator} = 1 - 2\cos(\theta)z^{-1} + z^{-2} \tag{7.2d}$$

The pair of complex-conjugate poles from the denominator of this transfer function provides integer multiplier and divisor coefficient values only when 2cos(θ) is an integer as shown in Figure 7.1(a). Such integer values result only when θ is equal to 0˚, ± 60˚, ± 90˚, ±120˚, and 180˚ respectively, as shown in Figure 7.1(b). When the coefficients are integers, a multiplication step is saved in the calculations because the multiplication of small integers is done by using bit-shift C-language instructions instead of multiplication instructions.

The poles in these positions are guaranteed to exactly lie on the unit circle. It is impossible to "exactly" cancel a zero on the unit circle unless the pole has integer coefficients. All other locations, say $\theta = 15˚$, have a small region of instability due to round-off error in the floating-point representation of numbers.

These filters have the added benefit of true-linear phase characteristics. This means that all the frequency components in the signal experience the same transmission delay through the filter (Lynn, 1972). This is important when it is desired to preserve the relative timing of the peaks and features of an output waveform. For example, it is crucial for a diagnostic quality ECG waveform to have P waves, T waves, and QRS complexes that remain unchanged in shape or timing. Changes in phase could reshape normal ECG waveforms to appear as possible arrhythmias, which is, of course, unacceptable.

7.1.3 Placement of zeros

We have seen that poles with integer coefficients that lie on the unit circle are restricted to five positions. This places constraints on the placement of zeros and also limits choices for the sampling rate when a designer wants to implement a specific cutoff frequency. To place zeros on the unit circle, one of two simple factors is used in the numerator of the filter's transfer function

$$(1 - z^{-m}) \tag{7.3a}$$

or

$$(1 + z^{-m}) \tag{7.3b}$$

For both equations, m is a positive integer equal to the number of zeros evenly spaced around the unit circle. Equation (7.3a) places a zero at 0˚ with the rest of the zeros evenly displaced by angles of $(360/m)˚$. To prove this statement, set

$$(1 - z^{-m}) = 0$$

Placing the z term on the right-hand side of the equation

$$z^{-m} = 1$$

or equivalently

$$z^{m} = 1$$

You can see that, if z is on the unit circle, it is only equal to 1 at the point $z = (1, 0)$, which occurs every 2π radians. Thus, we can say

$$z^{m} = e^{jn2\pi} = 1$$

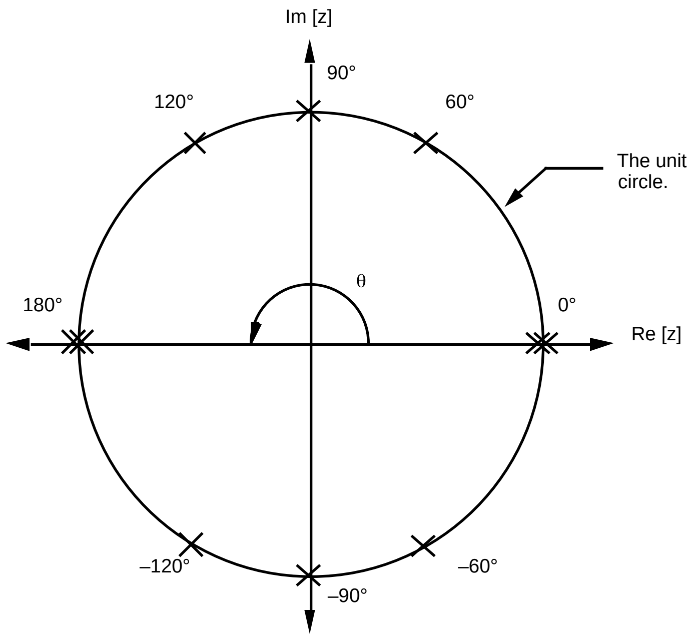
Solving for z, we arrive at

$$z = e^{j(n/m)2\pi}$$

Substituting in various integer values for n and m shows that the zeros are located at regular intervals on the unit circle every $(360/m)^\circ$ beginning with a zero at $z = 1$. You can observe that the same solutions result when n is outside of the range $n = \{0, 1, 2, ... , m - 1\}$.

θ	$\cos(\theta)$	$2\cos(\theta)$
0°	1	2
$\pm 60^\circ$	+1/2	1
$\pm 90^\circ$	0	0
$\pm 120^\circ$	–1/2	–1
180°	–1	–2

(a)



(b)

Figure 7.1 The possible pole placements given by the denominator of the transfer function in Eq. (7.1). (a) Table of only locations that result in integer coefficients. (b) Pole-zero plot corresponding to the table showing only possible pole locations on the unit circle. Notice that double poles are produced at 0˚ and 180˚.

Equation (7.3b) also places m evenly spaced zeros around the unit circle every $(360/m)^\circ$. Set

$$z^{-m} = -1$$

or equivalently

$$z^{m} = -1$$

We can solve for z in the same manner as described above. On the unit circle, $z = -1$ every $(2n + 1)\pi$ radians. The solution is

$$z = e^{j[(2n+1)/m]\pi}$$

where n and m again are integers.

Comparing the equations, $(1 - z^{-m})$ and $(1 + z^{-m})$, the difference in the placement of the zeros is a rotation of $1/2 \times (360/m)^\circ$. Figure 7.2 shows a comparison of the results from each equation. When m is an odd number, $(1 + z^{-m})$ always places a zero at 180° and appears "flipped over" from the results of $(1 - z^{-m})$.

7.1.4 Stability and operational efficiency of design

For a filter to be stable, certain conditions must be satisfied. It is necessary to have the highest power of poles in the denominator be less than or equal to the number of zeros in the numerator. In other words, you cannot have fewer zeros than poles. This requirement is met for transfer functions of the form used in this chapter since they always have the same number of poles and zeros. Poles that are located at the origin do not show up in the denominator of the general transfer functions given in Eq. (7.1). You can more easily see these poles at the origin by manipulating their transfer functions to have all positive exponents. For example, if a low-pass filter of the type discussed in the next section has five zeros (i.e., $m = 5$), we can show that the denominator has at least five poles by multiplying the transfer function by one (i.e., z^5/z^5):

$$H(z) = \frac{1 - z^{-5}}{1 - z^{-1}} \times \frac{z^5}{z^5} = \frac{z^5 - 1}{z^4(z - 1)}$$

Poles at the origin simply have the effect of a time delay. In frequency domain terms, a time delay of m sampling periods is obtained by placing an mth-order pole at the origin of the z plane (Lynn, 1971).

A finite impulse response filter (FIR) has all its poles located at the origin. These are called trivial poles. Since we force the transfer function of Eq. (7.1) to have all its nontrivial poles exactly canceled by zeros, it is a FIR filter. This transfer function could be expressed without a denominator by not including the zeros and poles that exactly cancel out. For example

$$H(z) = \frac{1 - z^{-m}}{1 - z^{-1}} = 1 + z^1 + z^2 + z^3 + \ldots + z^{m+1} = \frac{Y(z)}{X(z)} \qquad (7.4)$$

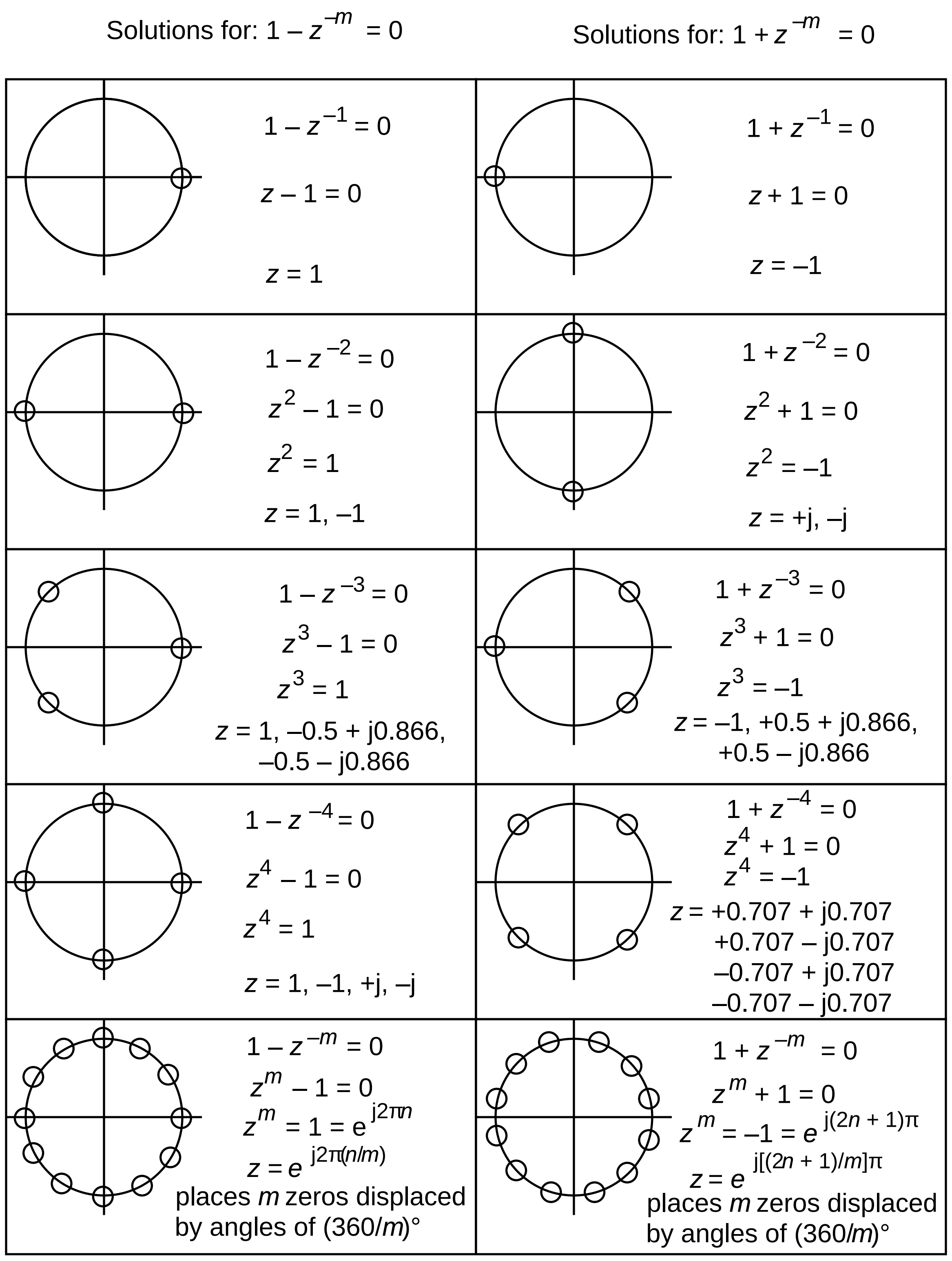


Figure 7.2 A comparison between the factors $(1 - z^{-m})$ and $(1 + z^{-m})$ for different values of m. When m is odd, the zeros appear "flipped over" from each other.

However, this is not done since expressing filters in recursive form is computationally more efficient. If m is large, the nonrecursive transfer function requires m

additions in place of just one addition and one subtraction for the recursive method. In general, a transfer function can be expressed recursively with far fewer operations than in a nonrecursive form.

7.2 LOW-PASS INTEGER FILTERS

Using the transfer function of Eq. (7.1), we can design a low-pass filter by placing a pole at $z = (1, 0)$. However, the denominator produces two poles at $z = (1, 0)$ when $\cos(\theta) = 0^\circ$. This problem is solved by either adding another zero at $z = (1, 0)$ or by removing a pole. The better solution is removing a pole since this creates a shorter, thus more efficient, transfer function. Also note from Figure 7.2 that the factor $(1 - z^{-m})$ should be used rather than $(1 + z^{-m})$ since it is necessary to have a zero positioned exactly at 0˚ on the unit circle. Thus, the transfer function for a recursive integer low-pass filter is

$$H(z) = \frac{1 - z^{-m}}{1 - z^{-1}} \tag{7.5}$$

This filter has a low-frequency lobe that is larger in magnitude than higher-frequency lobes; thus, it amplifies lower frequencies greater than the higher ones located in the smaller auxiliary sidelobes. These sidelobes result from the poles located at the origin of the z plane. Figure 7.3 shows the amplitude and phase responses for a filter with $m = 10$

$$H(z) = \frac{1 - z^{-10}}{1 - z^{-1}}$$

An intuitive feel for the amplitude response for all of the filters in this chapter is obtained by using the rubber membrane technique described Chapter 4. In short, when an extremely evenly elastic rubber membrane is stretched across the entire z plane, a zero “nails down” the membrane at its location. A pole stretches the membrane up to infinity at its location. Multiple poles at exactly the same location have the effect of stretching the membrane up to infinity, each additional one causing the membrane to stretch more tightly, thus driving it higher up at all locations around the pole. This has the effect of making a tent with a higher roof.

From this picture, we can infer how the amplitude response will look by equating it to the height of the membrane above the unit circle. The effective amplitude response is obtained from plotting the response from $\theta = 0^\circ$ to 180˚. For angles greater than 180˚ the response is a reflection (or foldover) of the first 180˚. The digital filter should never receive frequencies in this range. They should have all been previously removed by an analog low-pass antialias filter.

To achieve lower cutoff frequencies, we must either add more zeros to the unit circle or use a lower sampling frequency. Adding more zeros is usually a better solution since it creates a sharper slope at the cutoff frequency and reduces the danger of information loss from low sampling rates. However, as the number of zeros increases, so does the gain of the filter. This can become a problem if a large output is not desired or if overflow errors occur.

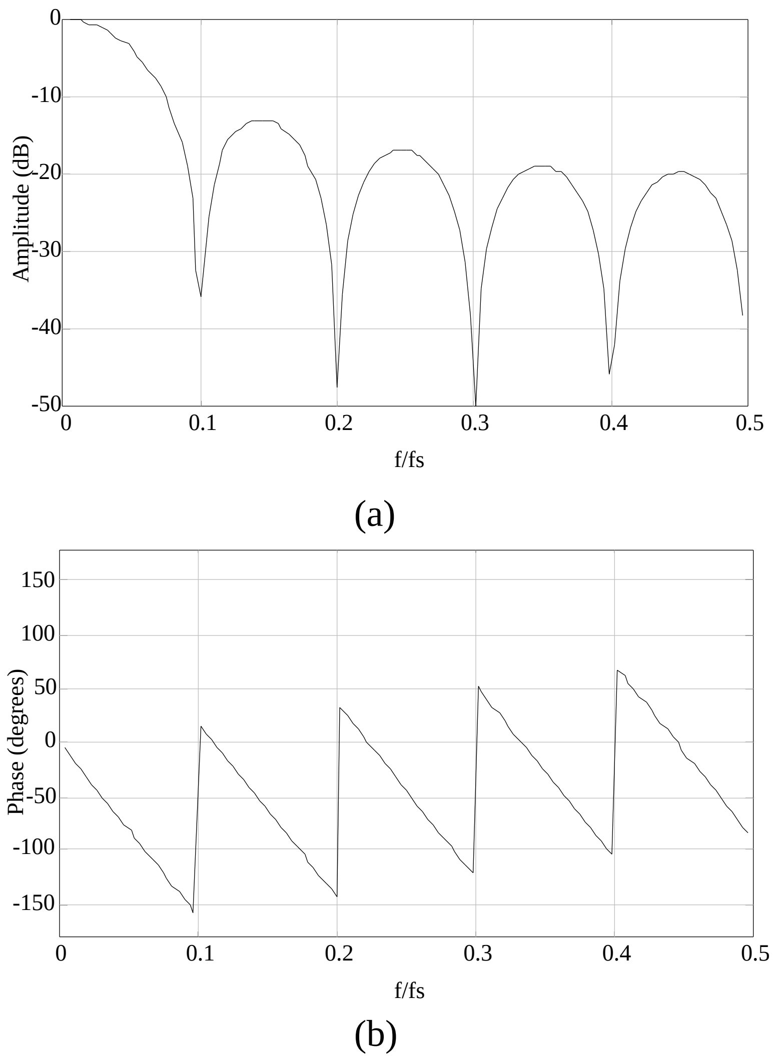


Figure 7.3 Low-pass filter with m = 10. (a) Amplitude response. (b) Phase response.

7.3 HIGH-PASS INTEGER FILTERS

There are several methods for designing a high-pass integer filter. Choice of an appropriate method depends on the desired cutoff frequency of the filter.

7.3.1 Standard high-pass filter design

Using the same design method described in section 7.2, a high-pass filter is constructed by placing a pole at the point $z = (–1, 0)$ corresponding to $\theta = 180^\circ$ in the transfer function shown in section 7.1.1. For this to be possible, the numerator must also have a zero at the point $z = (–1, 0)$. This will always happen if the exponent m is an even number using the factor $(1 – z^{–m})$ in the numerator. A numerator using the factor $(1 + z^{–m})$ requires m to be an odd positive integer. As with the low-pass filter, the denominator of the general equation produces two poles at the location $z = (-1, 0)$, so one of these factors should be removed. The transfer function simplifies to one of two forms:

$$H(z) = \frac{1 - z^{-m}}{1 + z^{-1}} \qquad (m \text{ is even}) \tag{7.6a}$$

or

$$H(z) = \frac{1 + z^{-m}}{1 + z^{-1}} \qquad (m \text{ is odd}) \tag{7.6b}$$

The highest input frequency must be less than half the sampling frequency. This is not a problem since, in a good design, an analog antialias low-pass filter will eliminate all frequencies that are higher than one-half the sampling frequency. A high-pass filter with a cutoff frequency higher than half the sampling frequency is not physically realizable, thus making it useless.

To construct such a filter, zeros are placed on the unit circle and a pole cancels the zero at $\theta = 180^\circ$. For a practical high-pass filter, the cutoff frequency must be greater than one-fourth the sampling frequency (i.e., $m \geq 4$). Increasing the number of zeros narrows the bandwidth. To achieve bandwidths greater than one-fourth the sampling frequency, requires the subtraction technique of the next section.

7.3.2 High-pass filter design based on filter subtraction

The frequency response of a composite filter formed by adding the outputs of two (or more) linear phase filters with the same transmission delay is equal to the simple algebraic sum of the individual responses (Ahlstrom and Tompkins, 1985). A high-pass filter $H_{high}(z)$ can also be designed by subtraction of a low-pass filter, $H_{low}(z)$ described in section 7.2, from an all-pass filter. This is shown graphically in Figure 7.4. An all-pass filter is a pure delay network with constant gain. Its transfer function can be represented as $H_a(z) = Az^{-m}$, where A is equal to the gain and m to number of zeros. Ideally, $H_a(z)$ should have the same gain as $H_{low}(z)$ at dc so that the difference is zero. Also the filter can operate more quickly if $A = 2^i$ where i is an integer so that a shift operation is used to scale the high-pass filter. To achieve minimal phase distortion, the number of delay units of the all-pass filter should be equal to the number of delay units needed to design the low-pass filter. Thus, we require that $H_a(z) = H_{low}(z)$. A low-pass filter with many zeros and thus a low cutoff frequency produces a high-pass filter with a low-frequency cutoff.

7.3.3 Special high-pass integer filters

We can also design high-pass filters for the special cases where every zero on the unit circle can potentially be canceled by a pole. This occurs when $m = 2$, 4, or 6 with the factor $(1 - z^{-m})$. This concept can be extended for any even value of m in the factor $(1 - z^{-m})$ if noninteger coefficients are also included. If $m = 2$ and the zero at 180° is canceled, we have designed a high-pass filter with a nonsharp cutoff frequency starting at zero. If $m = 4$, we can either cancel the zeros with conjugate poles at 180° and 90°, or just at 180° if the filter should begin passing frequencies at one-fourth of the sampling rate. Similarly, if $m = 6$, zeros are canceled at 180°, 120°, and 60° to achieve similar results.

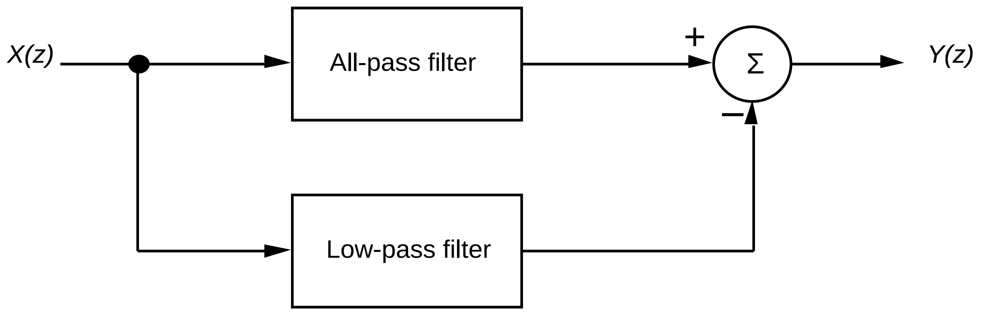


Figure 7.4 A block diagram of the low-pass filter being subtracted from a high-pass filter.

7.4 BANDPASS AND BAND-REJECT INTEGER FILTERS

In the design of an integer bandpass filter, once again one of the transfer functions of Eq. (7.1) is used. Unfortunately, the only possible choices of pole locations yielding integer coefficients are at 60°, 90°, and 120° [see Figure 7.1(a)]. The sampling frequency is chosen so that the passband frequency is at one of these three locations. Next a pair of complex-conjugate poles must be positioned so that one of them exactly cancels the zero in the passband frequency.

The choices for numerator are either $(1 - z^{-m})$ or $(1 + z^{-m})$. The best choice is the one that places a zero where it is needed with a reasonable value of m. The number of zeros chosen depends on the acceptable nominal bandwidth requirements. To get a very narrow bandwidth, we increase the number of zeros by using a higher power m. However, the gain of the filter increases with m, so the filter's output may become greater than the word size used, causing an overflow error. This is a severe error that must be avoided. As the number of zeros increases, (1) the bandwidth decreases, (2) the amplitudes of the neighboring sidelobes decrease, (3) the steepness of the cutoff increases, and (4) the difference equations require a greater history of the input data so that more sampled data values must be stored. Increasing the number of zeros of a bandpass filter increases the Q; however, more ringing occurs in the filter's output. This is similar to the analog-equivalent filter. The effects of different values of Q are illustrated in section 12.5 for an ECG example.

The design of a band-reject (or bandstop) integer filter is achieved in one of two ways. The first solution is to simply place a zero on the unit circle at the frequency to be eliminated. The second method is to use the filter subtraction method to subtract a bandpass filter from an all-pass filter. The same restrictions and principles described in section 7.3.2 apply when using this method.

7.5 THE EFFECT OF FILTER CASCADES

The order of a filter is the number of identical filter stages that are used in series. The output of one filter provides the input to the next filter in the cascade. To increase the order of the general transfer function of Eq. (7.1), simply increase the exponents *p* and *t* by the same integer amount. For a filter of order *n*, the amplitude of the frequency response is expressed by $|H(z)|^n$. As *n* increases, the gain increases so you should be careful to avoid overflow error as mentioned in section 7.4. The gain of a filter can be attenuated by a factor of two by right bit-shifting the output values. However, this introduces a small quantization error as nonzero least-significant bits are shifted away.

All the filters previously discussed in this chapter suffer from substantial sidelobes and a poorly defined cutoff. The sidelobes can be substantially reduced by increasing the order of a filter by an even power. Figure 7.5 shows that, as the order of zeros in a filter increases, for example, 2*nd* order, 4*th* order, etc., the sidelobes become smaller and smaller.

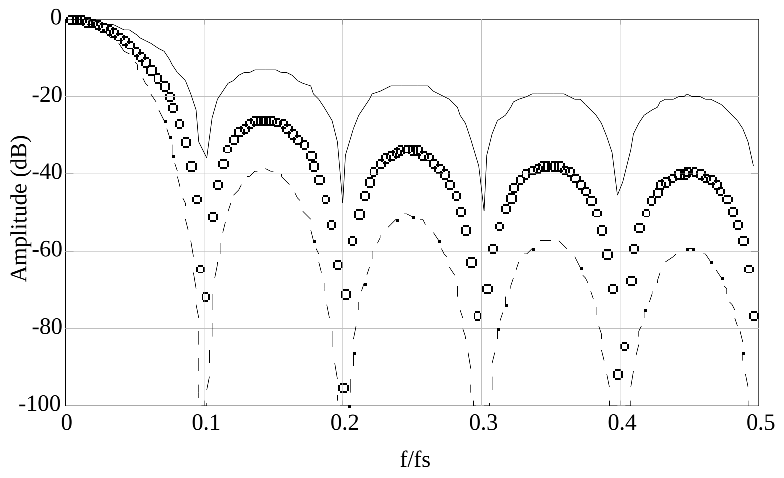


Figure 7.5 The effect of the order of a filter on its gain. This is an example of a high-pass filter with $m = 10$. Solid line: first order. Same as filter of Figure 7.3. Circles: second order. Dashed line: third order.

Increasing the filter order quickly reaches a point of diminishing returns as the filter recurrence formulas become more complex and the sidelobes are decreased by smaller increments.

The order of a filter can also be raised to an odd power; however, the phase response usually has better characteristics when the order is even. For the general transform of Eq. (7.1), the behavior of the phase response of filters of this type can be summarized as follows:

1. Even-order filters exhibit true-linear phase, thus eliminating phase distortion.
2. Odd-order filters have a piecewise-linear phase response. The discontinuities jump by 180° wherever the amplitude response is forced to zero by locating a zero on the unit circle.

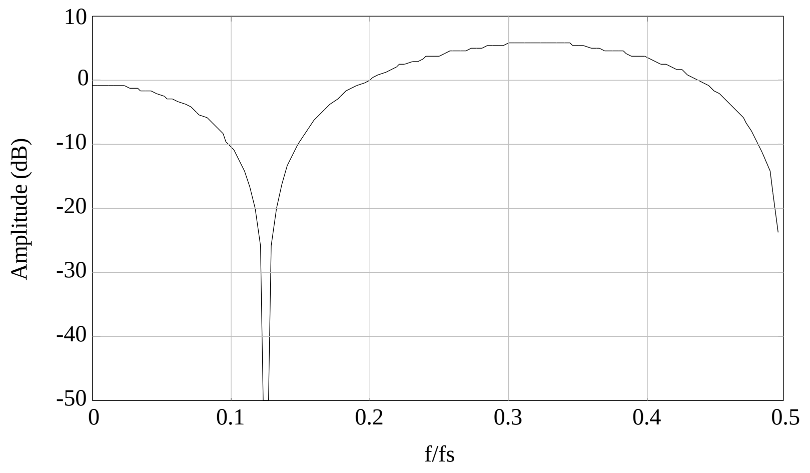
A piecewise-linear phase response is acceptable if the filter displays significant attenuation in the regions where the phase response has changed to a new value. If the stopband is not well attenuated, phase distortions will occur whenever the signals that are being filtered have significant energy in those regions. The concept of phase distortions caused by linear phase FIR filters is thoroughly explained in a paper by Kohn (1987).

It is also possible to combine nonidentical filters together. Often complicated filters are crafted from simple subfilters. Examples of these include the high-pass and band-reject filters described in previous sections that were derived from filter subtraction. Lynn (1983) also makes use of this principle when he expands the design method described in this chapter to include resonator configurations. Lynn uses these resonator configurations to expand the design format to include 11 pole locations which add to design flexibility; however, this concept is not covered in this chapter.

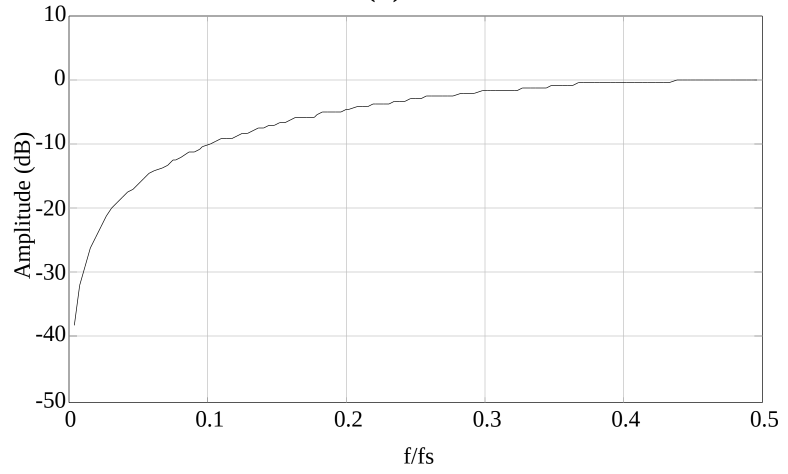
7.6 OTHER FAST-OPERATING DESIGN TECHNIQUES

Principe and Smith (1986) used a method slightly different from Lynn's for designing digital filters to operate on electroencephalographic (EEG) data. Their method is a dual to the frequency sampling technique described in section 5.6. They construct the filter's transfer function in two steps. First zeros are placed on the unit circle creating several passbands, one of which corresponds to the desired passband. Next zeros are placed on the unit circle in the unwanted passbands to squelch gain and to produce stopbands.

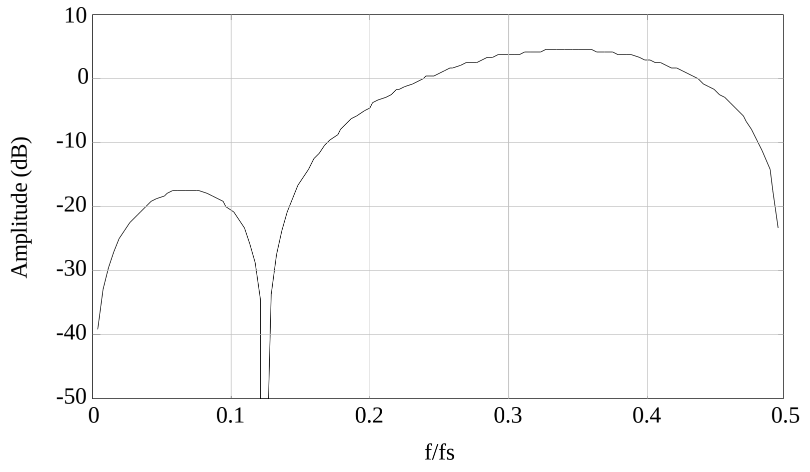
An example of this is shown in Figure 7.6. Since all the poles are located at the origin, these FIR filters are guaranteed to be stable. Placing zeros to attenuate gain is more flexible than placing poles to cancel zeros on the unit circle. Zeros are placed in conjugate pairs by using the factor $1 - 2\cos(\theta)\, z^{-1} + z^{-2}$ in the numerator of the transfer function.



(a)



(b)



(c)

Figure 7.6 An example using only zeros to create a high-pass filter. The outputs of magnitude (a) plus magnitude (b) are summed to create magnitude (c).

If a zero is needed at a location that involves multiplication to calculate the coefficient, it is acceptable to place it at a nearby location that only requires a few binary-shift and add instructions so as to reduce computation time. A 3-bit approximation for 2cos(θ) which uses at most two shifts and two additions can place a zero at every location that the factors $(1 - z^{-m})$ or $(1 + z^{-m})$ do with an error of less than 6.5 percent if *m* is less than 8. This is a common technique to save multiplications. The zero can be slightly moved to an easy-to-calculate location at the cost of slightly decreasing the stopband attenuation.

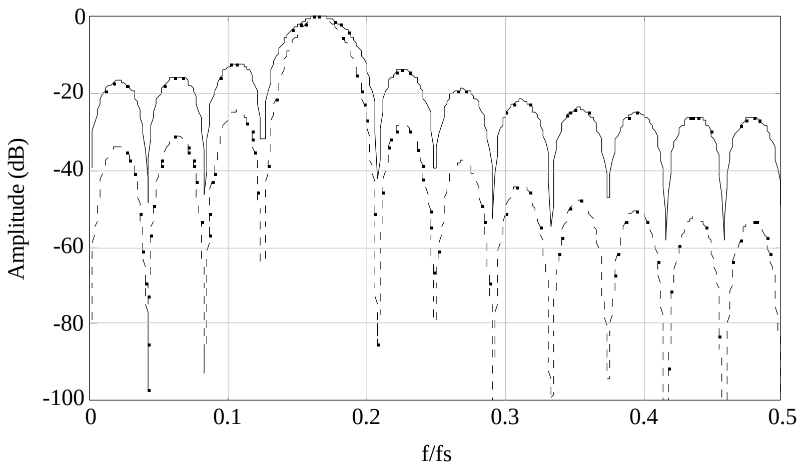
This technique of squelching the stopband by adding more zeros could also be used with Lynn's method described in all of the previous sections. Adding zeros at specific locations can make it easier to achieve desired nominal bandwidths for bandpass filters, remove problem frequencies in the stopband (such as 60-Hz noise), or eliminate sidelobes without increasing the filter order.

All the previously discussed filters displayed a true-linear or piecewise-linear phase response. Sometimes situations demand a filter to have a sharp cutoff frequency, but phase distortion is irrelevant. For these cases, placing interior poles near zeros on the unit circle has the effect of amplifying the passband frequencies and attenuating the stopband frequencies. Whenever nontrivial poles remain uncanceled by zeros, an IIR rather than an FIR filter results. IIR filters have sharper cutoff slopes at the price of a nonlinear phase response. However, we do not wish to express the pole's coefficients, which have values between 0 and 1, using floating-point representation.

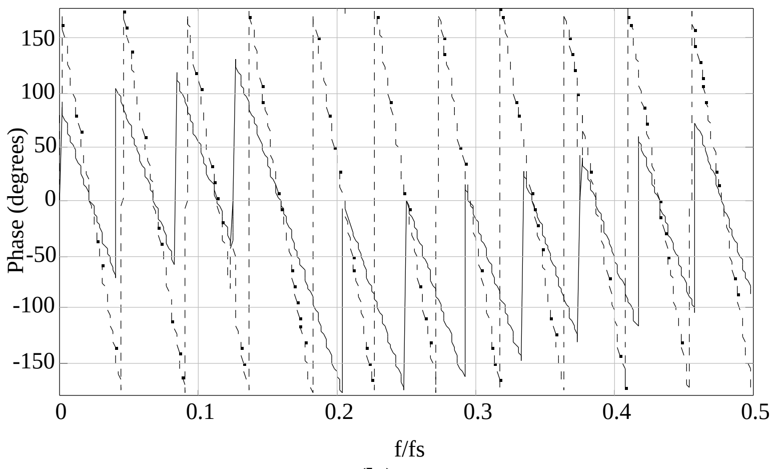
Thakor and Moreau (1987) solved this problem in a paper about the use of "quantized coefficients." To place poles inside the unit circle, they use a less restrictive method that retains some of the advantages of integer coefficients. They allow coefficients of the poles to have values of $x/2^y$ where *x* is an integer between 1 and $(2^y - 1)$ and *y* is a nonnegative integer called the quantization value. Quantization *y* is the designer's choice, but is limited by the microprocessor's word length (e.g., $y = 8$ for an 8-bit microprocessor). Using these values, fractional coefficients, such as 1/8, can be implemented with right shift instructions. A coefficient, such as 17/32, will not show much speed improvement over a multiplication instruction since many shifts and adds are required for its calculation. Thakor and Moreau give an excellent description of the use of these coefficients in filters and analyze the possible quantization, truncation, roundoff, filter-coefficient representation, and overflow errors that can occur.

7.7 DESIGN EXAMPLES AND TOOLS

This chapter includes enough material to design a wide variety of fast-operating digital filters. It would require many example problems to provide a feeling for all of the considerations needed to design an "optimal" filter for a certain application. However, the following design problems adequately demonstrate several of the methods used to theoretically analyze some of the characteristics of integer filters. We also demonstrate how a filter's difference equation is converted into C language so that it can be quickly implemented.



(a)



(b)

Figure 7.7 Piecewise-linear FIR bandpass filters. (a) Magnitude responses. Solid line: first-order filter with transfer function of $H(z) = (1 - z^{-24})/(1 - z^{-1} + z^{-2})$. Dashed line: second-order filter with transfer function of $H(z) = (1 - z^{-24})^2/(1 - z^{-1} + z^{-2})^2$. (b) Phase responses. Solid line: piecewise-linear phase response of first-order filter. Dashed line: true-linear phase response of second-order filter.

7.7.1 First-order and second-order bandpass filters

This section demonstrates the design of a bandpass filter that has 24 zeros evenly spaced around the unit circle beginning at 0˚. The peak of the passband is located at 60˚. A theoretical calculation of the amplitude and phase response is provided.

The transfer equation for this filter is:

$$H(z) = \frac{1 - z^{-24}}{1 - z^{-1} + z^{-2}} \tag{7.7}$$

Substitute $e^{j\omega T}$ for z and rearrange to produce positive and negative exponents of equal magnitude:

$$H(\omega T) = \frac{1 - e^{-j24\omega T}}{1 - e^{-j\omega T} + e^{-j2\omega T}} = \frac{e^{-j12\omega T}\ (e^{j12\omega T} - e^{-j12\omega T})}{e^{-j\omega T}\ (e^{j\omega T} - 1 + e^{-j\omega T})} \tag{7.8}$$

Substituting inside the parentheses the relation

$$e^{j\omega T} = \cos(\omega T) + j\sin(\omega T) \tag{7.9}$$

gives

$$H(\omega T) = \frac{e^{-j12\omega T}\,[\cos(12\omega T) + j\sin(12\omega T) - \cos(12\omega T) + j\sin(12\omega T)]}{e^{-j\omega T}\,[\cos(\omega T) + j\sin(\omega T) - 1 + \cos(\omega T) - j\sin(\omega T)]} \tag{7.10}$$

Combining terms

$$H(\omega T) = \frac{j2\sin(12\omega T)}{2\cos(\omega T) - 1} \times e^{-j11\omega T} \tag{7.11}$$

Substituting $j = e^{j\pi/2}$ gives

$$H(\omega T) = \frac{2\sin(12\omega T)}{2\cos(\omega T) - 1} \times e^{j(\pi/2 - 11\omega T)} \tag{7.12}$$

Thus, the magnitude response shown in Figure 7.7(a) as a solid line is

$$|H(\omega T)| = \left|\frac{2\sin(12\omega T)}{2\cos(\omega T) - 1}\right| \tag{7.13}$$

and the phase response shown in Figure 7.7(b) as a solid line is

$$H(\omega T) = \pi/2 - 11\omega T \tag{7.14}$$

Next we cascade the filter with itself and calculate the amplitude and phase responses. This permits a comparison between the first-order and second-order filters.

The transfer equation now becomes

$$H(z) = \frac{(1 - z^{-24})^2}{(1 - z^{-1} + z^{-2})^2} \tag{7.15}$$

Again we substitute $e^{j\omega T}$ for z. The steps are similar to those for the first-order filter until we arrive at the squared form of Eq. (7.12). The transfer equation of the second-order example is

$$H(\omega T) = \left(\frac{2\sin(12\omega T)}{2\cos(\omega T) - 1} \times e^{j(\pi/2 - 11\omega T)}\right)^2$$

$$= \left(\frac{2\sin(12\omega T)}{2\cos(\omega T) - 1} \right)^2 \times e^{j(\pi - 22\omega T)} \tag{7.16}$$

The magnitude response of this second-order bandpass filter shown in Figure 7.7(a) as a dashed line is

$$|H(\omega T)| = \left| \left(\frac{2\sin(12\omega T)}{2\cos(\omega T) - 1} \right)^2 \right| \tag{7.17}$$

and the phase response shown in Figure 7.7(b) as a dashed line is

$$H(\omega T) = \pi - 22\omega T \tag{7.18}$$

Comparing the phase responses of the two filters, we see that the first-order filter is piecewise linear, whereas the second-order filter has a true-linear phase response. A true-linear phase response is recognized on these plots when every phase line has the same slope and travels from 360˚ to –360˚.

To calculate the gain of the filter, substitute into the magnitude equation the critical frequency. For this bandpass filter, this frequency is located at an angle of 60˚. Substituting this value into the magnitude response equation gives an indeterminate result

$$|H(\omega T)| = \left| \left(\frac{2\sin(12\omega T)}{2\cos(\omega T) - 1} \right)^2 \right| = \left| \left(\frac{2\sin(60^\circ)}{2\cos(60^\circ) - 1} \right)^2 \right|_{\omega T = 60^\circ} = \frac{3}{0} \tag{7.17}$$

Thus, to find the gain, L'Hôpital's Rule must be used. This method requires differentiation of the numerator and differentiation of the denominator prior to evaluation at 60˚.

This procedure yields

$$\frac{d(|H(\omega T)|)}{d(\omega T)} = \left| \left(\frac{24\cos(12\omega T)}{-2\sin(\omega T)} \right)^2 \right|_{\omega T = 60^\circ} = \left| \left(\frac{24}{-\sqrt{3}} \right)^2 \right| = 192 \tag{7.19}$$

Thus, the gain for the second-order filter is 192 compared to 13.9 for the first-order filter. Increasing the order of a filter also increases the filter's gain. However, Figure 7.7(a) shows that the sidelobes are more attenuated than the passband lobe for the second-order filter.

We use the filter's difference equation to write a C-language function that implements this filter:

$$\begin{aligned} y(nT) = {} & 2y(nT - T) - 3y(nT - 2T) + 2y(nT - 3T) - y(nT - 4T) \\ & + x(nT) - 2x(nT - 24T) + x(nT - 48T) \end{aligned} \tag{7.20}$$

Figure 7.8 shows the C-language program for a real-time implementation of the second-order bandpass filter. This code segment is straightforward; however, it is not the most time-efficient method for implementation. Each `for()` loop shifts all the values in each array one period in time.

```
/* Bandpass filter implementation of
 *
 *  H(z) = [( 1 - z^-24)^2] / [(1 - z^-1 + z^-2)^2]
 *
 * Notes: Static variables are automatically initialized to
zero.
 * Their scope is local to the function. They retain their
values
 * when the function is exited and reentered.
 *
 * Long integers are used for y, the output array. Since this
 * filter has a gain of 192, values for y can easily exceed the
 * values that a 16-bit integer can represent.
*/

long bandpass(int x_current)
{
register int i;
static int x[49];    /* history of input data points */
static long y[5];     /* history of difference equation outputs
*/

 for (i=4; i>0; i--)         /* shift past outputs */
    y[i] = y[i–1];

 for (i=48; i>0; i--)        /* shift past inputs */
    x[i] = x[i-1];

 x[0] = x_current;   /* update input array with new data point
*/

/* Implement difference equation */
 y[0] = 2*y[1] - 3*y[2] + 2*y[3] - y[4] + x[0] - 2*x[24] +
x[48];

 return y[0];
}
```

Figure 7.8 C-language implementation of the second-order bandpass filter.

7.7.2 First-order low-pass filter

Here is a short C-language function for a six-zero low-pass filter. This function is passed a sample from the A/D converter. The data is filtered and returned. A FIFO circular buffer implements the unit delays by holding previous samples. This function consists of only one addition and one subtraction. It updates the pointer to the buffer rather than shifting all the data as in the previous example. This becomes more important as the number of zeros increases in the filter's difference equation. The difference equation for a six-zero low-pass filter is

$$y(nT) = y(nT - T) + x(nT) - x(nT - 6T) \tag{7.21}$$

Figure 7.9 shows how to implement this equation in efficient C-language code.

```
/* Low-pass filter implementation of
 *
 *  H(z) = ( 1 - z^-6) / (1 - z^-1)
 *
 * Note 1: Static variables are initialized only once. Their
scope
 * is local to the function. Unless set otherwise, they are
 * initialized to zero. They retain their values when the
function
 * is exited and reentered.
 *
 * Note 2: This line increments pointer x_6delay along the x
array
 * and wraps it to the first element when its location is at the
 * last element. The ++ must be a prefix to x_6delay; it is
 * equivalent to x_delay + 1.
*/

int lowpass(int x_current)
{
static x[6],                /* FIFO buffer of past samples */
       y,                   /* serves as both y(nT) and y(nT-T)
*/
       *x_6delay = &x[0};   /* pointer to x(nT-6T); see Note 1
*/

y += (x_current - *x_6delay);   /* y(nT)=y(nT-T)+x(nT)-x(nT-6T)
*/
*x_6delay = x_current;     /* x_current becomes x(nT-T) in FIFO
*/
x_6delay = (x_6delay == &x[5]) ? &x[0] : ++x_6delay;
                                                  /* See Note 2
*/
 return(y);
}
```

Figure 7.9 Efficient C-language implementation of a first-order low-pass filter. Increments a pointer instead of shifting all the data.

7.8 LAB: INTEGER FILTERS FOR ECG ANALYSIS

Equipment designed for ECG analysis often must operate in real time. This means that every signal data point received by the instrument must be processed to produce an output before the next input data point is received. In the design process, cost constraints often make it desirable to use smaller, low-performance microprocessors to control a device. If the driver for an instrument is a PC, many times it is not equipped with a math coprocessor or a high-performance microprocessor. For this lab, you will design and implement several of the filters previously discussed to compare their performance in several situations.

Execute **`(F)ilters`**, **`(D)esign`**, then **`i(N)teger`** to enter the integer filter design shell.

1. Use the **`(G)enwave`** function to generate a normal ECG from template 1 and an abnormal ECG from template 5, both with a sampling rate of 100 sps and an amplitude resolution of eight bits.

(a) Design a bandpass filter for processing these signals with six zeros and a center frequency of 16.7 Hz. This type of filter is sometimes used in cardiotachometers to find the QRS complex determine heart rate.

(b) Filter the two ECGs with these filters. Since this filter has gain, you will probably need to adjust the amplitudes with the **`(Y) Sens`** function. Sketch the responses.

(c) Read the unit impulse signal file **`ups.dat`** from the **`STDLIB`** directory, and filter it. Sketch the response. Take the power spectrum of the impulse response using **`(P)wr Spect`**. The result is the amplitude response of the filter, which is the same as the response that you obtained when you designed the filter except that the frequency axis is double that of your design because the unit impulse signal was sampled at 200 sps instead of the 100 sps for which you designed your filter. Use the cursors of the **`(M)easure`** function to locate the 3-dB points, and find the bandwidth. Note that the actual bandwidth is half the measurement because of doubling of the frequency axis. Calculate the Q of the filter.

(d) Design two bandpass filters similar to the one in part (a) except with 18 and 48 zeros. Repeat parts (b) and (c) using these filters. As the number of zeros increases, the gain of these filters also increases, so you may have an overflow problem with some of the responses, particularly for the unit impulse. If this occurs, the output will appear to have discontinuities, indicating that the 16-bit representation of data in DigiScope has overflowed. In this case, attenuate the amplitude of the unit impulse signal prior to filtering it. Which of these three filter designs is most appropriate for detecting the QRS complexes? Why?

2. Design a low-pass filter with 10 zeros. Filter the normal ECG from part 1, and sketch the output. Which waves are attenuated and which are amplified? What is the 3-dB passband?

3. Generate a normal ECG with a sampling rate of 180 sps. Include 10% 60-Hz noise and 10% random noise in the signal. Design a single low-pass filter that (a) has a low-pass bandwidth of about 15 Hz to attenuate random noise, and (b) completely eliminates 60-Hz noise. Measure the actual 3-dB bandwidth. Comment on the performance of your design.

7.9 REFERENCES

Ahlstrom, M. L., and Tompkins, W. J. 1985. Digital filters for real-time ECG signal processing using microprocessors. *IEEE Trans. Biomed. Eng.*, **BME-32**(9): 708–13.

Kohn, A. F. 1987. Phase distortion in biological signal analysis caused by linear phase FIR filters. *Med. & Biol. Eng. & Comput.* **25:** 231–38.

Lynn, P. A. 1977. Online digital filters for biological signals: some fast designs for a small computer. *Med. & Biol. Eng. & Comput.* **15:** 534–40.

Lynn, P. A. 1971. Recursive digital filters for biological signals. *Med. & Biol. Eng. & Comput.* **9:** 37–44.

Lynn, P. A. 1972. Recursive digital filters with linear-phase characteristics. *Comput. J.* **15:** 337–42.

Lynn, P. A. 1983. Transversal resonator digital filters: fast and flexible online processors for biological signals. *Med. & Biol. Eng. & Comput.* **21:** 718–30.

Principe, J. C., and Smith, J. R. 1986. Design and implementation of linear phase FIR filters for biological signal processing. *IEEE Trans. Biomed. Eng.* **BME-33**(6):550–59.

Thakor, N. V., and Moreau, D. 1987. Design and analysis of quantised coefficient digital filters: application to biomedical signal processing with microprocessors. *Med. & Biol. & Comput.* **25:** 18–25.

7.10 STUDY QUESTIONS

7.1 Why is it advantageous to use integer coefficients in a digital filter's difference equation?

7.2 Explain why it is more difficult to design a digital filter when all coefficients are restricted to having integer values.

7.3 Show how the denominator of Eq. (7.1) will always place two poles on the unit circle for all values of θ. What values of θ produce integer coefficients? Why should values of θ that yield floating-point numbers be avoided?

7.4 Show mathematically why the numerators $(1 + z^{-m})$ and $(1 - z^{-m})$ place zeros on the unit circle. Both numerators produce evenly spaced zeros. When would it be advantageous to use $(1 - z^{-m})$ instead of $(1 + z^{-m})$?

7.5 When does Eq. (7.1) behave as a FIR filter? How can this equation become unstable? What are the advantages of expressing a FIR filter in recursive form?

7.6 When does Eq. (7.1) behave as a low-pass filter? Discuss what characteristics of filter behavior a designer must consider when a filter is to have a low cutoff frequency.

7.7 What is the difference between a true-linear phase response and a piecewise-linear phase response? When is a linear phase response essential? Can a filter with a piecewise-linear phase response behave as one with a true-linear phase response?

7.8 Name four ways in which an integer digital filter's magnitude and phase response change when the filter is cascaded with itself. Why or why not are these changes helpful?

7.9 If poles and zeros are placed at noninteger locations, how can a digital filter still remain computationally efficient? Describe two methods that use this principle.

7.10 Calculate expressions for the amplitude and phase response of a filter with the z transform

$$H(z) = 1 - z^{-6}$$

7.11 The numerator of a transfer function is $(1 - z^{-10})$. Where are its zeros located?

7.12 A filter has 12 zeros located on the unit circle starting at dc and equally spaced at 30˚ increments (i.e., $1 - z^{-12}$). There are three poles located at $z = +0.9$, and $z = \pm j$. The sampling frequency is 360 samples/s. (a) At what frequency is the output at its maximal amplitude? (b) What is the gain at this frequency?

7.13 A digital filter has the following transfer function. (a) What traditional filter type best describes this filter? (b) What is its gain at dc?

$$H(z) = \frac{1 - z^{-6}}{(1 - z^{-1})(1 - z^{-1} + z^{-2})}$$

7.14 For a filter with the following transfer function, what is the (a) amplitude response, (b) phase response, (c) difference equation?

$$H(z) = \frac{1 - z^{-8}}{1 + z^{-2}}$$

7.15 A digital filter has the following transfer function. (a) What traditional filter type best describes this filter? (b) Draw its pole-zero plot. (c) Calculate its amplitude response. (d) What is its difference equation?

$$H(z) = \frac{(1 - z^{-8})^2}{(1 + z^{-2})^2}$$

7.16 What is the gain of a filter with the transfer function

$$H(z) = \frac{1 - z^{-6}}{1 - z^{-1}}$$

7.17 What traditional filter type best describes a filter with the transfer function

$$H(z) = \frac{1 - z^{-256}}{1 - z^{-128}}$$

7.18 What traditional filter type best describes a filter with the transfer function

$$H(z) = \frac{1 - z^{-200}}{1 - z^{-2}}$$

7.19 A digital filter has four zeros located at $z = \pm 1$ and $z = \pm j$ and four poles located at $z = 0$, $z = 0$, and $z = \pm j$. The sampling frequency is 800 samples/s. The maximal output amplitude occurs at what frequency?

7.20 For a sampling rate of 100 samples/s, a digital filter with the following transfer function has its maximal gain at approximately what frequency (in Hz)?

$$H(z) = \frac{1 - z^{-36}}{1 - z^{-1} + z^{-2}}$$

7.21 The z transform of a filter is:

$$H(z) = 1 - z^{-360}$$

The following sine wave is applied at the input: $x(t) = 100 \sin(2\pi 10 t)$. The sampling rate is 720 samples/s. (a) What is the peak-to-peak output of the filter? (b) If a *unit step* input is applied, what will the output amplitude be after 361 samples? (c) Where could poles be placed to convert this to a bandpass filter with integer coefficients?

7.22 What is the phase delay (in milliseconds) through the following filter which operates at 200 samples/sec?

$$H(z) = \frac{1 - z^{-100}}{1 - z^{-2}}$$

7.23 A filter has 8 zeros located on the unit circle starting at dc and equally spaced at 45° increments. There are two poles located at $z = \pm j$. The sampling frequency is 360 samples/s. What is the gain of the filter?