3.24 What sampling rate and filter characteristics (e.g., cutoff frequency) would you use to sample an ECG signal?

3.25 What type of A/D converter circuit provides the fastest sampling speed?

3.26 What type of A/D converter circuit tends to average out high-frequency noise?

3.27 In an 8-bit successive-approximation A/D converter, what is the initial digital approximation to a signal?

3.28 A 4-bit successive-approximation A/D converter gets a final approximation to a signal of 0110. What approximation did it make just prior to this final result?

3.29 In a D/A converter design, what are the advantages of an *R*-2*R* resistor network over a binary-weighted resistor network?

3.30 For an 8-bit successive approximation analog-to-digital converter, what will be the next approximation made by the converter (in hexadecimal) if the approximation of (a) 0x90 to the input signal is found to be too low, (b) 0x80 to the input signal is found to be too high?

3.31 For an 8-bit successive-approximation analog-to-digital converter, what are the possible results of the next approximation step (in hexadecimal) if the approximation at a certain step is a) 0x10, b) 0x20?

3.32 An 8-bit analog-to-digital converter has a clock that drives the internal successive approximation circuitry at 80 kHz. (a) What is the fastest possible sampling rate that could be achieved by this converter? (b) If 0x80 represents a signal level of 1 V, what is the minimum signal that this converter can resolve (in mV)?

3.33 What circuit is used in a signal conversion system to store analog voltage levels? Draw a schematic of such a circuit and explain how it works.

3.34 The internal IBM PC signal acquisition board described in Appendix A is used to sample an ECG. An amplifier amplifies the ECG so that a 1-mV level uses all 12 bits of the converter. What is the smallest ECG amplitude that can be resolved (in μV)?

3.35 An 8-bit successive-approximation analog-to-digital converter is used to sample an ECG. An amplifier amplifies the ECG so that a 1-mV level uses all 8 bits of the converter. What is the smallest ECG amplitude that can be resolved (in μV)?

3.36 The Computers of Wisconsin (COW) A/D converter chip made with CMOS technology includes an 8-bit successive-approximation converter with a 100-μs sampling period. An on-chip analog multiplexer provides for sampling up to 8 channels. (a) With this COW chip, how fast could you sample a single channel (in samples per s)? (b) How fast could you sample each channel if you wanted to use all eight channels? (c) What is the minimal external clock frequency necessary to drive the successive-approximation circuitry for the maximal sampling rate? (d) List two advantages that this chip has over an equivalent one made with TTL technology.

4

Basics of Digital Filtering

Willis J. Tompkins and Pradeep Tagare

In this chapter we introduce the concept of digital filtering and look at the advantages, disadvantages, and differences between analog and digital filters. Digital filters are the discrete domain counterparts of analog filters. Implementation of different types of digital filters is covered in later chapters. There are many good books that expand on the general topic of digital filtering (Antoniou, 1979; Bogner and Constantinides, 1985; Gold and Rader, 1969; Rabiner and Rader, 1972; Stearns, 1975).

4.1 DIGITAL FILTERS

The function of a digital filter is the same as its analog counterpart, but its implementation is very different. Analog filters are implemented using either active or passive electronic circuits, and they operate on continuous waveforms. Digital filters, on the other hand, are implemented using either a digital logic circuit or a computer program and they operate on a sequence of numbers that are obtained by sampling the continuous waveform. The use of digital filters is widespread today because of the easy availability of computers. A computer program can be written to implement almost any kind of digital filter.

There are several advantages of digital filters over analog filters. A digital filter is highly immune to noise because of the way it is implemented (software/digital circuits). Accuracy is dependent only on round-off error, which is directly determined by the number of bits that the designer chooses for representing the variables in the filter. Also it is generally easy and inexpensive to change a filter's operating characteristics (e.g., cutoff frequency). Unlike an analog filter, performance is not a function of factors such as component aging, temperature variation, and power supply voltage. This characteristic is important in medical applications where most of the signals have low frequencies that might be distorted due to the drift in an analog circuit.

4.2 THE *z* TRANSFORM

As discussed in Chapter 3, the sampling process reduces a continuous signal to a sequence of numbers. Figure 4.1 is a representation of this process which yields the sequence

$$\{a(0), a(T), a(2T), a(3T), \ldots, a(kT)\} \tag{4.1}$$

This set of numbers summarizes the samples of the waveform $a(t)$ at times $0, T, \ldots, kT$, where T is the sampling period.

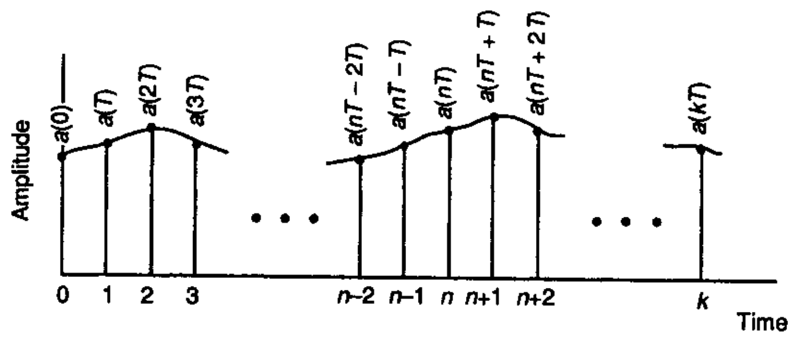


Figure 4.1 Sampling a continuous signal produces a sequence of numbers. Each number is separated from the next in time by the sampling period of T seconds.

We begin our study of digital filters with the z transform. By definition, the z transform of any sequence

$$\{f(0), f(T), f(2T), \ldots, f(kT)\} \tag{4.2}$$

is

$$F(z) = f(0) + f(T)z^{-1} + f(2T)z^{-2} + \ldots + f(kT)z^{-k} \tag{4.3}$$

In general

$$F(z) = \sum_{n=0}^{k} f(nT)z^{-n} \tag{4.4}$$

If we accept this definition, then the z transform of the sequence of Eq. (4.1) is

$$A(z) = a(0) + a(T)z^{-1} + a(2T)z^{-2} + \ldots + a(kT)z^{-k} \tag{4.5}$$

or

$$A(z) = \sum_{n=0}^{k} a(nT)z^{-n} \tag{4.6}$$

Suppose we want to find the z transform of a signal represented by any sequence, for example

$$\{1, 2, 5, 3, 0, 0, \ldots\} \tag{4.7}$$

From Eq. (4.4), we can write the z transform at once as

$$X(z) = 1 + 2z^{-1} + 5z^{-2} + 3z^{-3} \tag{4.8}$$

Since the sequence represents an array of numbers, each separated from the next by the sampling period, we see that the variable z^{-1} in the z transform represents a T-s time separation of one term from the next. The numerical value of the negative exponent of z tells us how many sample periods after the beginning of the sampling process when the sampled data point, which is the multiplier of the z term, occurred. Thus by inspection, we know, for example, that the first sampled data value, which was obtained at $t = 0$, is 1 and that the sample at clock tick 2 (i.e., $t = 2 \times T$ s) is 5.

The z transform is important in digital filtering because it describes the sampling process and plays a role in the digital domain similar to that of the Laplace transform in analog filtering. Figure 4.2 shows two examples of discrete-time signals analogous to common continuous time signals. The unit impulse of Figure 4.2(a), which is analogous to a Dirac delta function, is described by

$$\begin{aligned} f(nT) &= 1 \qquad \text{for } n = 0 \\ f(nT) &= 0 \qquad \text{for } n > 0 \end{aligned} \tag{4.9}$$

This corresponds to a sequence of

$$\{1, 0, 0, 0, 0, 0, \ldots\} \tag{4.10}$$

Therefore, the z transform of the unit impulse function is

$$F(z) = 1 \tag{4.11}$$

This is an important finding since we frequently use the unit impulse as the input function to study the performance of a filter.

For the unit step function in Figure 4.2(b)

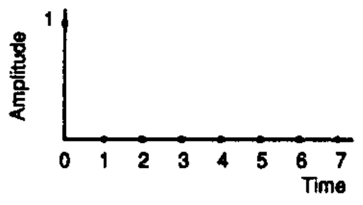
$$f(nT) = 1 \qquad \text{for } n \geq 0$$

giving a sequence of

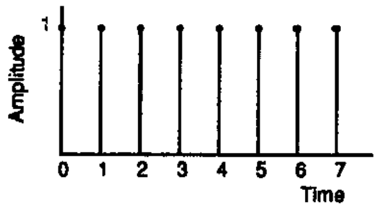
$$\{1, 1, 1, 1, 1, 1, \ldots\} \tag{4.12}$$

Therefore, the z transform of the unit step is

$$F(z) = 1 + z^{-1} + z^{-2} + z^{-3} + \ldots \tag{4.13}$$



(a)



(b)

Figure 4.2 Examples of discrete-time signals. Variable n is an integer. a) Unit impulse. $\delta(nT) = 1$ for $n = 0$; $\delta(nT) = 0$ for $n \neq 0$. b) Unit step. $f(nT) = 1$ for all n.

This transform is an infinite summation of nonzero terms. We can convert this sum to a more convenient ratio of polynomials by using the binomial theorem

$$1 + v + v^2 + v^3 + \ldots = \frac{1}{1 - v} \tag{4.14}$$

If we let $v = z^{-1}$ in the above equation, the z transform of the unit step becomes

$$F(z) = \frac{1}{1 - z^{-1}} \tag{4.15}$$

Figure 4.3 summarizes the z transforms of some common signals.

4.3 ELEMENTS OF A DIGITAL FILTER

We need only three types of operations to implement any digital filter: (1) storage for an interval of time, (2) multiplication by a constant, and (3) addition. Figure 4.4 shows the symbols used to represent these operations. Consider the sequence

$$\{x(0), x(T), x(2T), \ldots, x(nT)\} \tag{4.16}$$

$f(t), t \geq 0$	$f(nT), nT \geq 0$	$F(z)$
1(unit step)	1	$\dfrac{1}{1-z^{-1}}$
t	nT	$\dfrac{Tz^{-1}}{(1-z^{-1})^2}$
e^{-at}	e^{-anT}	$\dfrac{1}{1-e^{-aT}z^{-1}}$
te^{-at}	nTe^{-anT}	$\dfrac{Te^{-aT}z^{-1}}{(1-e^{-aT}z^{-1})^2}$
$\sin\omega_c t$	$\sin n\omega_c T$	$\dfrac{(\sin\omega_c T)z^{-1}}{1-2(\cos\omega_c T)z^{-1}+z^{-2}}$
$\cos\omega_c t$	$\cos n\omega_c T$	$\dfrac{1-(\cos\omega_c T)z^{-1}}{1-2(\cos\omega_c T)z^{-1}+z^{-2}}$

Figure 4.3 Examples of continuous time functions $f(t)$ and analogous discrete-time functions $f(nT)$ together with their z transforms.

Its z transform is

$$X(z) = x(0) + x(T)z^{-1} + x(2T)z^{-2} + \ldots + x(nT)z^{-n} \tag{4.17}$$

If we apply this sequence to the input of the storage element of Figure 4.4(a), we obtain at the output the sequence

$$\{0, x(0), x(T), x(2T), \ldots, x(nT)\} \tag{4.18}$$

This sequence has the z transform

$$Y(z) = 0 + x(0)z^{-1} + x(T)z^{-2} + \ldots + x(nT-T)z^{-n} \tag{4.19}$$

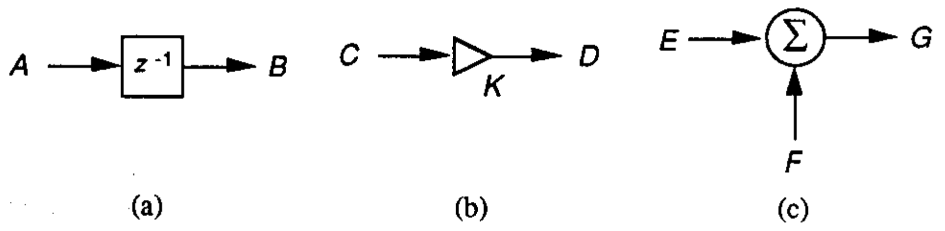


Figure 4.4 Digital filter operators. (a) Storage of a number for one clock period. $B = A$ exactly T seconds after the signal enters A. (b) Multiplication by a constant. $D = K \times C$ instantaneously. (c) Addition of two numbers. $G = E + F$ instantaneously.

In this case, $x(0)$ enters the storage at time $t = 0$; simultaneously, the contents of the block, which is always initialized to 0, are forced to the output. At time $t = T$, $x(0)$ is forced out as $x(T)$ enters. Thus at each clock tick of the analog-to-digital converter from which we get the sampled data, a new number enters the storage block and forces out the previous number (which has been stored for T s). Therefore, the output sequence is identical to the input sequence except that the whole sequence has been delayed by T seconds.

By dividing the output given by Eq. (4.19) by the input of Eq. (4.17), we verify that the relation between the output z transform and the input z transform is

$$Y(z) = X(z)\,z^{-1} \tag{4.20}$$

and the transfer function of the delay block is

$$H(z) = \frac{Y(z)}{X(z)} = z^{-1} \tag{4.21}$$

A microcomputer can easily perform the function of a storage block by placing successive data points in memory for later recall at appropriate clock times.

Figure 4.4(b) shows the second function necessary for implementing a digital filter—multiplication by a constant. For each number in the sequence of numbers that appears at the input of the multiplier, the product of that number and the constant appears instantaneously at the output

$$Y(z) = \beta \times X(z) \tag{4.22}$$

where β is a constant. Ideally there is no storage or time delay in the multiplier. Multiplication of the sequence {5, 9, 0, 6} by the constant 5 would produce a new sequence at the output of the multiplier {25, 45, 0, 30} where each number in the output sequence is said to occur exactly at the same point in time as the corresponding number in the input sequence.

The final operation necessary to implement a digital filter is addition as shown in Figure 4.4(c). At a clock tick, the numbers from two different sequences are summed to produce an output number instantaneously. Again we assume zero delay time for the ideal case.

The relation between the output and the input z transforms is

$$Y(z) = X_1(z) + X_2(z) \tag{4.23}$$

Of course, multiplier and adder circuits require some finite length of time to produce their results, but the delays in a digital filter that control the timing are the storage elements that change their outputs at the rate of the sampling process. Thus, as long as all the arithmetic operations in a digital filter occur within T s, the indi vidual multiply and add operations can be thought of as occurring in zero time, and the filter can perform real-time processing. The nonzero delay time of the adders and multipliers is not a big drawback because in a practical system the outputs from the adders and the multipliers are not required immediately. Since their inputs are constant for almost *T* s, they actually have *T* s to generate their outputs.

All general-purpose microprocessors can implement the three operations necessary for digital filtering: storage for a fixed time interval, multiplication, and addition. Therefore they all can implement basic digital filtering. If they are fast enough to do all operations to produce an output value before the next input value appears, they operate in real time. Thus, real-time filters act very much like analog filters in that the filtered signal is produced at the output at the same time as the signal is being applied to the input (with some small delay for processing time).

4.4 TYPES OF DIGITAL FILTERS

The transfer function of a digital filter is the z transform of the output sequence divided by the z transform of the input sequence

$$H(z) = \frac{Y(z)}{X(z)} \tag{4.24}$$

There are two basic types of digital filters—nonrecursive and recursive. For nonrecursive filters, the transfer function contains a finite number of elements and is in the form of a polynomial

$$H(z) = \sum_{i=0}^{n} h_i z^{-i} = h_0 + h_1 z^{-1} + h_2 z^{-2} + \ldots + h_n z^{-n} \tag{4.25}$$

For recursive filters, the transfer function is expressed as the ratio of two such polynomials

$$H(z) = \frac{\sum_{i=0}^{n} a_i z^{-i}}{1 - \sum_{i=1}^{n} b_i z^{-i}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{1 - b_1 z^{-1} - b_2 z^{-2} - \ldots - b_n z^{-n}} \tag{4.26}$$

The values of z for which $H(z)$ equals zero are called the zeros of the transfer function, and the values of z for which $H(z)$ goes to infinity are called the poles. We find the zeros of a filter by equating the numerator to 0 and evaluating for z. To find the poles of a filter, we equate the denominator to 0 and evaluate for z. Thus, we can see that the transfer function (and hence the output) goes to zero at the zeros of the transfer function and becomes indeterminate at the poles of the transfer function.

We can see from the transfer functions of nonrecursive filters that they have poles only at $z = 0$. We will see later in this chapter that the location of the poles in the z plane determines the stability of the filter. Since nonrecursive filters have poles only at $z = 0$, they are always stable.

4.5 TRANSFER FUNCTION OF A DIFFERENCE EQUATION

Once we have the difference equation representing the numerical algorithm for implementing a digital filter, we can quickly determine the transfer equation that totally characterizes the performance of the filter. Consider the difference equation

$$y(nT) = x(nT) + 2x(nT - T) + x(nT - 2T) \tag{4.27}$$

Recognizing that $x(nT)$ and $y(nT)$ are points in the input and output sequences associated with the current sample time, they are analogous to the undelayed z-domain variables, $X(z)$ and $Y(z)$ respectively. Similarly $x(nT - T)$, the input value one sample point in the past, is analogous to the z-domain input variable delayed by one sample point, or $X(z)z^{-1}$. We can then write an equation for output $Y(z)$ as a function of input $X(z)$

$$Y(z) = X(z) + 2X(z)z^{-1} + X(z)z^{-2} \tag{4.28}$$

Thus the transfer function of this difference equation is

$$H(z) = \frac{Y(z)}{X(z)} = 1 + 2z^{-1} + z^{-2} \tag{4.29}$$

From this observation of the relationship between discrete-time variables and z-domain variables, we can quickly write the transfer function if we know the difference equation and vice versa.

4.6 THE z-PLANE POLE-ZERO PLOT

We have looked at the z transform from a *practical* point of view. However the mathematics of the z transform are based on the definition

$$z = e^{sT} \tag{4.30}$$

where the complex frequency is

$$s = \sigma + j\omega \tag{4.31}$$

Therefore

$$z = e^{\sigma T} e^{j\omega T} \tag{4.32}$$

By definition, the magnitude of z is

$$|z| = e^{\sigma T} \tag{4.33}$$

and the phase angle is

$$\angle z = \omega T \tag{4.34}$$

If we set $\sigma = 0$, the magnitude of z is 1 and we have

$$z = e^{j\omega T} = \cos\omega T + j\sin\omega T \tag{4.35}$$

This is the equation of a circle of unity radius called the unit circle in the z plane. Since the z plane is a direct mathematical mapping of the well-known s plane, let us consider the stability of filters by mapping conditions from the s domain to the z domain. We will use our knowledge of the conditions for stability in the s domain to study stability conditions in the z domain.

Mapping the s plane to the z plane shows that the imaginary axis ($j\omega$) in the s plane maps to points on the unit circle in the z plane. Negative values of σ describe the left half of the s plane and map to the interior of the unit circle in the z plane. Positive values of σ correspond to the right half of the s plane and map to points outside the unit circle in the z plane.

For the s plane, poles to the right of the imaginary axis lead to instability. Also any poles on the imaginary axis must be simple. From our knowledge of the mapping between the s and z planes, we can now state the general rule for stability in the z plane. All poles must lie either inside or on the unit circle. If they are on the unit circle, they must be simple. Zeros do not influence stability and can be anywhere in the z plane. Figure 4.5 shows some of the important features of the z plane. Any angle ωT specifies a point on the unit circle. Since $\omega = 2\pi f$ and $T = 1/f_s$, this angle is

$$\omega T = 2\pi \frac{f}{f_s} \tag{4.36}$$

The angular location of any point on the unit circle is then designated by the ratio of a specified frequency f to the sampling frequency f_s. If $f = f_s$, $\omega T = 2\pi$; thus the sampling frequency corresponds to an angular location of 2π radians. For $f = 0$, $\omega T = 0$; hence, dc is located at an angle of 0°.

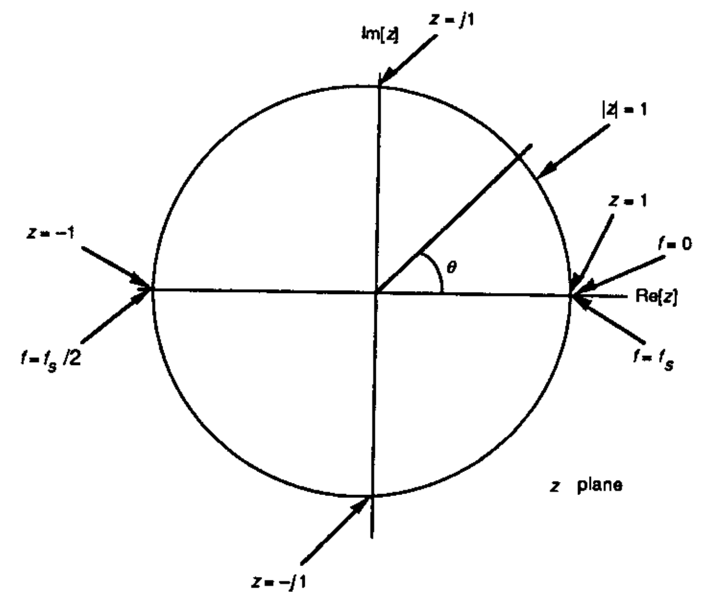


Figure 4.5 Unit circle in the z plane. $\theta = \omega T \approx f \approx f/f_s$, indicating different ways of identifying an angle in the z plane.

Another important frequency is $f = f_s/2 = f_0$ at $\omega T = \pi$. This frequency, called the foldover frequency, equals one-half the sampling rate. Since sampling theory requires a sample rate of twice the highest frequency present in a signal, the foldover frequency represents the maximum frequency that a digital filter can process properly (see Chapter 3). Thus, unlike the frequency axis for the continuous world which extends linearly to infinite frequency, the meaningful frequency axis in the discrete world extends only from 0 to π radians corresponding to a frequency range of dc to $f_s/2$. This is a direct result of the original definition for z in Eq. (4.30) which does a nonlinear mapping of all the points in the s plane into the z plane.

It is important to realize that antialiasing cannot be accomplished with any digital filter except by raising the sampling rate to twice the highest frequency present. This is frequently not practical; therefore, most digital signal processors have an analog front end—the antialias filter.

Figure 4.6 shows that we can refer to the angle designating a point on the unit circle in a number of ways. If we use the ratio f/f_s, it is called the normalized frequency. If we have already established the sampling frequency, we can alternately specify the angular location of a point on the unit circle by frequency f. This illustrates an important feature of a digital filter—the frequency response characteristics are directly related to the sampling frequency. Thus, suppose that the sampling frequency is 200 Hz and a filter has a zero located at 90° on the unit circle (i.e., at 50 Hz). Then if we desire to have the zero of that filter at 25 Hz, a simple way of doing it would be to halve the sampling frequency.

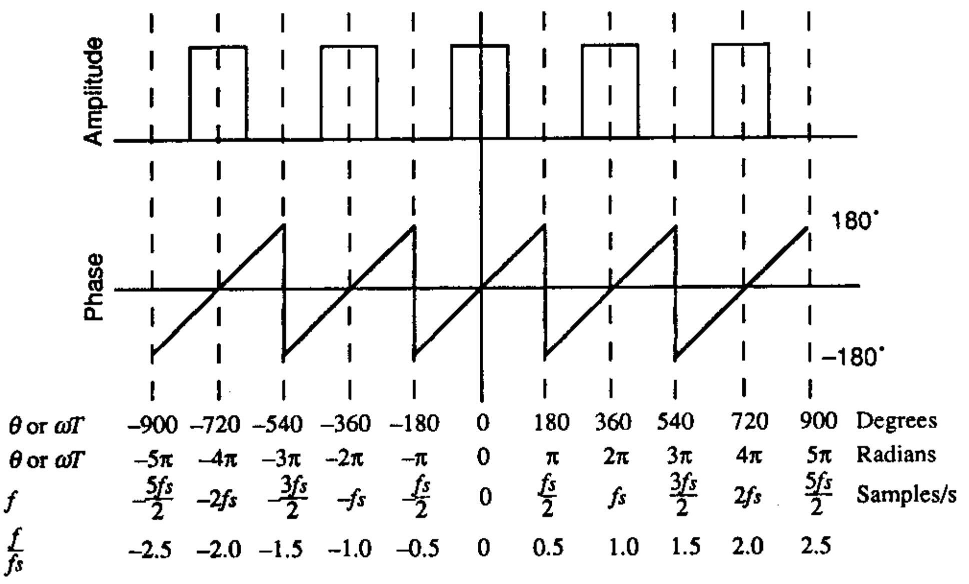


Figure 4.6 The frequency axis of amplitude and phase responses for digital filters can be labeled in several different ways—as angles, fraction of sampling frequency, or ratio of frequency to sampling frequency. An angle of 360° representing one rotation around the unit circle corresponds to the sampling frequency. The only important range of the amplitude and phase response plots is from 0 to 180° since we restrict the input frequencies to half the sampling frequency in order to avoid aliasing.

As an example, let us consider the following transfer function:

$$H(z) = \frac{1}{3}(1 + z^{-1} + z^{-2}) \tag{4.37}$$

In order to find the locations of the poles and zeros in the z plane, we first multiply by z^2/z^2 in order to make all the exponents of variable z positive.

$$H(z) = \frac{1}{3}(1 + z^{-1} + z^{-2}) \times \frac{z^2}{z^2} = \frac{1}{3}\frac{(z^2 + z + 1)}{z^2} \tag{4.38}$$

Solving for the zeros by setting the numerator equal to zero

$$z^2 + z + 1 = 0 \tag{4.39a}$$

We find that there are two complex conjugate zeros located at

$$z = -0.5 \pm j0.866 \tag{4.39b}$$

The two zeros are located on the unit circle at $\omega T = \pm 2\pi/3$ ($\pm 120°$). If the sampling frequency is 180 Hz, the zero at +120° will completely eliminate any signal at 60 Hz. Solving for the poles by setting the denominator equal to zero

$$z^2 = 0 \tag{4.39c}$$

We find that there are two poles, both located at the origin of the z plane

$$z = 0 \tag{4.39d}$$

These poles are equally distant from all points on the unit circle, so they influence the amplitude response by an equal amount at all frequencies. Because of this, we frequently do not bother to show them on pole-zero plots.

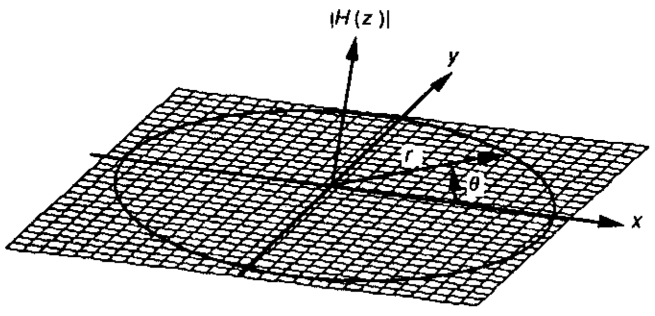
4.7 THE RUBBER MEMBRANE CONCEPT

To get a practical feeling of the pole-zero concept, imagine the z plane to be an infinitely large, flat rubber membrane "nailed" down around its edges at infinity but unconstrained elsewhere. The pole-zero pattern of a filter determines the points at which the membrane is pushed up to infinity (i.e., poles) or is "nailed" to the ground (i.e., zeros). Figure 4.7(a) shows this z-plane rubber membrane with the unit circle superimposed. The magnitude of the transfer function $|H(z)|$ is plotted orthogonal to the x and y axes.

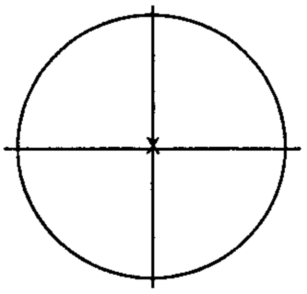
Consider the following transfer function, which represents a pole located in the z plane at the origin, $z = 0$

$$H(z) = z^{-1} = \frac{1}{z} \tag{4.40}$$

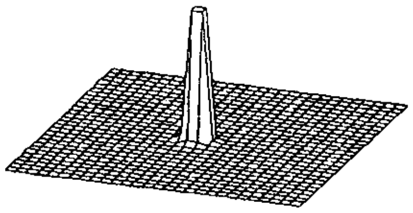
Imagine making a tent by placing an infinitely long tent pole under the rubber membrane located at $z = 0$. Figure 4.7(b) illustrates how this pole appears in a pole-zero plot. Figure 4.7(c) is a view from high above the z plane that shows how the pole stretches the membrane. In Figure 4.7(d) we move our observation location from a distant point high above the rubber membrane to a point just above the membrane since we are principally interested in the influence of the pole directly on the unit circle. Notice how the membrane is distorted symmetrically in all directions from the location of the pole at $z = 0$.



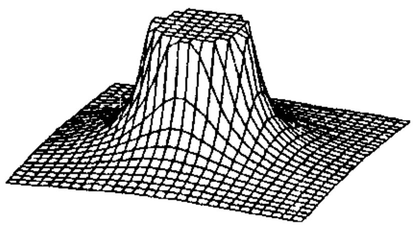
(a)



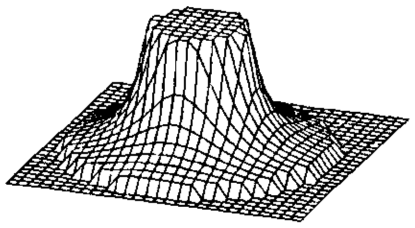
(b)



(c)



(d)



(e)

Figure 4.7 Rubber membrane analogy for the z plane. (a) Region of complex z plane showing unit circle, x and y axes, and $|H(z)|$ axis. (b) Pole-zero plot showing single pole at $z = 0$. (c) View of distortion caused by the pole from high above the membrane. (d) View of distortion from viewpoint close to membrane. (e) View with $|H(z)|$ set equal to zero outside the unit circle to visualize how the membrane is stretched on the unit circle itself.

Thus the unit circle is lifted an equal distance from the surface all the way around its periphery. Since the unit circle represents the frequency axis, the amount of stretch of the membrane directly over the unit circle represents $|H(z)|$, the magnitude of the amplitude response.

In order to characterize the performance of a filter, we are principally interested in observations of the amount of membrane stretch directly above the unit circle, and the changes inside and outside the circle are not of particular importance. Therefore in Figure 7(e), we constrain the magnitude of the transfer function to be zero everywhere outside the unit circle in order to be able to better visualize what happens on the unit circle itself. Now we can easily see that the magnitude of the transfer function is the same all the way around the unit circle. We evaluate the magnitude of the transfer function by substituting $z = e^{j\omega T}$ into the function of Eq. (4.40), giving for the complex frequency response

$$H(z) = e^{-j\omega T} = \cos(\omega T) - j\sin(\omega T) \tag{4.41}$$

The magnitude of this function is

$$|H(\omega T)| = \sqrt{\cos^2(\omega T) + \sin^2(\omega T)} = 1 \tag{4.42}$$

and the phase response is

$$\angle H(\omega T) = -\omega T \tag{4.43}$$

Thus the height of the membrane all the way around the unit circle is unity. The magnitude of this function on the unit circle between the angles of 0° and 180°, corresponding to the frequency range of dc to $f_s/2$ respectively, represents the amplitude response of the single pole at $z = 0$. Thus Eq. (4.42) indicates that a signal of any legal frequency entering the input of this unity-gain filter passes through the filter without modification to its amplitude.

The phase response in Eq. (4.43) tells us that an input signal has a phase delay at the output that is linearly proportional to its frequency. Thus this filter is an all-pass filter, since it passes all frequencies equally well, and it has linear phase delay.

In order to see how multiple poles and zeros distort the rubber membrane, let us consider the following transfer function, which has two zeros and two poles.

$$H(z) = \frac{1 - z^{-2}}{1 - 1.0605z^{-1} + 0.5625z^{-2}} \tag{4.44}$$

To find the locations of the poles and zeros in the z plane, we first multiply by z^2/z^2 in order to make all the exponents of variable z positive.

$$H(z) = \frac{1 - z^{-2}}{1 - 1.0605z^{-1} + 0.5625z^{-2}} \times \frac{z^2}{z^2} = \frac{z^2 - 1}{z^2 - 1.0605z + 0.5625} \tag{4.45}$$

In order to find the zeros, we set the numerator to zero. This gives

$$z^2 - 1 = 0 \tag{4.46}$$

or

$$(z + 1)(z - 1) = 0 \tag{4.47}$$

Therefore, there are two zeros of the function located at

$$z = \pm 1 \tag{4.48}$$

To find the locations of the poles, we set the denominator equal to zero.

$$z^2 - 1.0605z + 0.5625 = 0 \tag{4.49}$$

Solving for z, we obtain a complex-conjugate pair of poles located at

$$z = 0.53 \pm j0.53 \tag{4.50}$$

Figure 4.8(a) is the pole-zero plot showing the locations of the two zeros and two poles for this transfer function $H(z)$. The zeros "nail" down the membrane at the two locations, $z = \pm 1$ (i.e., radius r and angle θ of $1\angle 0°$ and $-1\angle 180°$ in polar notation). The poles stretch the membrane to infinity at the two points, $z = 0.53 \pm j0.53$ (i.e., $0.75\angle \pm 45°$ in polar notation).

Figure 4.8(b) shows the distortion of the unit circle (i.e., the frequency axis). In this view, the plane of Figure 4.8(a) is rotated clockwise around the vertical axis by an azimuth angle of 20° and is tilted at an angle of elevation angle of 40° where the top view in Figure 4.8(a) is at an angle of 90°. The membrane distortion between the angles of 0 to 180° represents the amplitude response of the filter.

The zero at an angle of 0° completely attenuates a dc input. The zero at an angle of 180° (i.e., $f_s/2$) eliminates the highest possible input frequency that can legally be applied to the filter since sampling theory restricts us from putting any frequency into the filter greater than half the sampling frequency. You can see that the unit circle is "nailed" down at these two points. The poles distort the unit circle so as to provide a passband between dc and the highest input frequency at a frequency of $f_s/8$. Thus, this filter acts as a bandpass filter.

The positive frequency axis, which is the top half of the unit circle (see Figure 4.5), is actually hidden from our view in this presentation since it is in the background as we view the rubber membrane with the current viewing angle. Of course, it is symmetrical with the negative frequency axis between angles 0 to −180°, which we see in the foreground.

In order to better visualize the amplitude response, we can "walk" around the membrane and look at the hidden positive frequency side. In Figures 4.8(c) to 4.8(f), we rotate clockwise to azimuth angles of 60°, 100°, 140°, and finally 180° respectively. Thus in Figure 4.8(f), the positive frequency axis runs from dc at the left of the image to $f_s/2$ at the right.

We next tilt the image in Figures 4.8(g) and 4.8(h) to elevation angles of 20° and 0° respectively. Thus Figure 4.8(h) provides a direct side view with the horizontal axis representing the edge of the z plane. This view shows us the amplitude response of this filter with a peak output at a frequency corresponding to the location of the pole at an angle of 45° corresponding to a frequency of $f_s/8$.

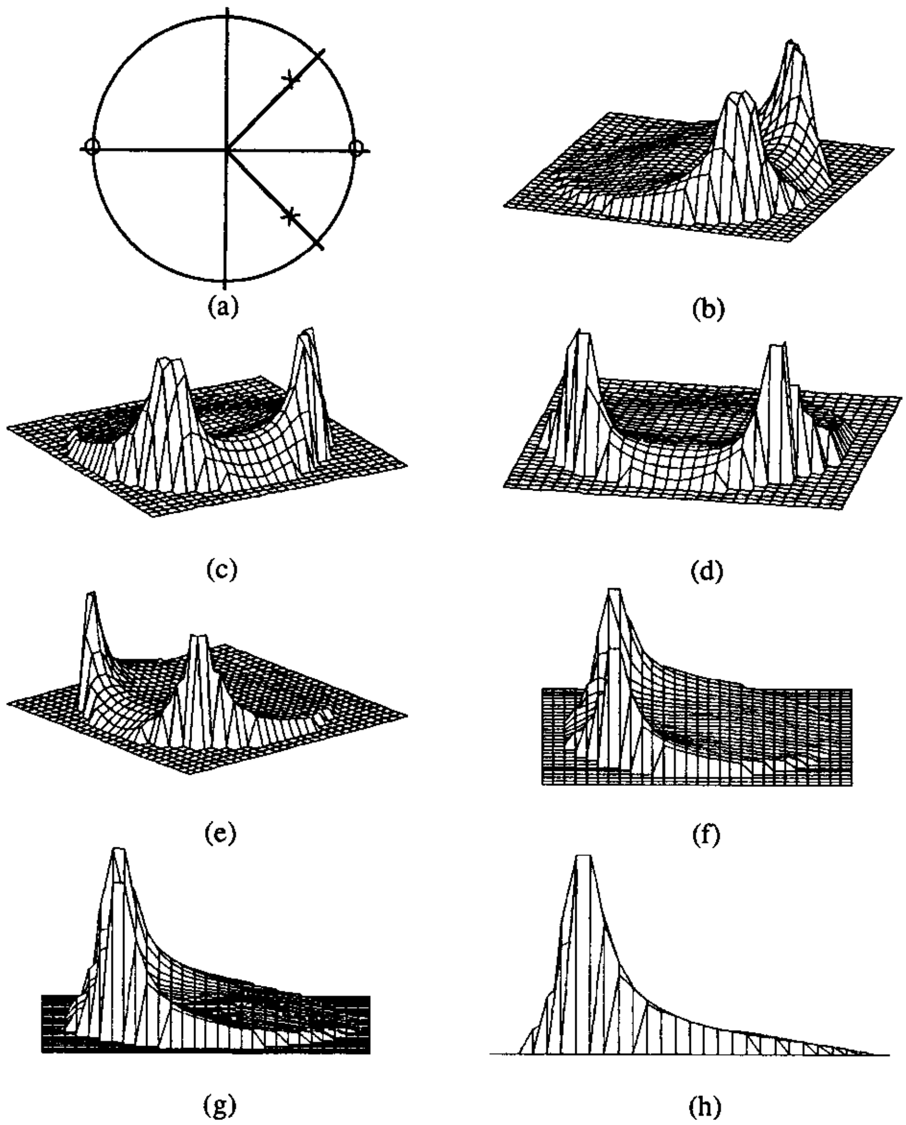


Figure 4.8 A filter with two zeros and two poles. Zeros are located at $z = \pm 1$ and poles at $z = 0.53 \pm j0.53$ (i.e., $r = 0.75$, $\theta = 45°$). Zeros nail the membrane down where they are located. Poles stretch it to infinite heights at the points where they located. (a) Pole-zero plot. Azimuth (AZ) is 0°; elevation (EL) is 90°. (b) Rubber membrane view. AZ = 20°; EL = 40°. (c) AZ = 60°, EL = 40°. (d) AZ = 100°, EL = 40°. (e) AZ = 140°, EL = 40°. (f) AZ = 180°, EL = 40°. (g) AZ = 180°, EL = 20°. (h) AZ = 180°, EL = 0°.

The amplitude reponse clearly goes to zero at the left and right sides of the plot corresponding to the locations of zeros at dc and $f_s/2$. This plot is actually a projection of the response from the circular unit circle axis using a linear amplitude scale.

In order to see how the frequency response looks on a traditional response plot, we calculate the amplitude response by substituting into the transfer function of Eq. (4.44) the equation $z = e^{j\omega T}$

$$H(\omega T) = \frac{1 - e^{-j2\omega T}}{1 - 1.0605e^{-j\omega T} + 0.5625e^{-j2\omega T}} \qquad (4.51)$$

We now substitute into this function the relationship

$$e^{j\omega T} = \cos(\omega T) + j\sin(\omega T) \qquad (4.52)$$

giving

$$H(\omega T) = \qquad (4.53)$$
$$\frac{1 - \cos(2\omega T) + j\sin(2\omega T)}{1 - 1.0605\cos(\omega T) + j1.0605\sin(\omega T) + 0.5625\cos(2\omega T) - j0.5625\sin(2\omega T)}$$

Collecting real and imaginary terms gives

$$H(\omega T) = \qquad (4.54)$$
$$\frac{[1 - \cos(2\omega T)] + j[\sin(2\omega T)]}{[1 - 1.0605\cos(\omega T) + 0.5625\cos(2\omega T)] + j[1.0605\sin(\omega T) - 0.5625\sin(2\omega T)]}$$

This equation has the form

$$H(\omega T) = \frac{A + jB}{C + jD} \qquad (4.55)$$

In order to find the amplitude and phase responses of this filter, we first multiply the numerator and denominator by the complex conjugate of the denominator

$$H(\omega T) = \frac{A + jB}{C + jD} \times \frac{C - jD}{C - jD} = \frac{(AC + BD) + j(BC - AD)}{C^2 + D^2} \qquad (4.56)$$

The amplitude response is then

$$|H(\omega T)| = \frac{\sqrt{((AC + BD)^2 + (BC - AD)^2)}}{C^2 + D^2} \qquad (4.57)$$

This response curve is plotted in Figure 4.9(a). Note that the abscissa is f/f_s, so that the range of the axis goes from 0 (dc) to 0.5 ($f_s/2$).

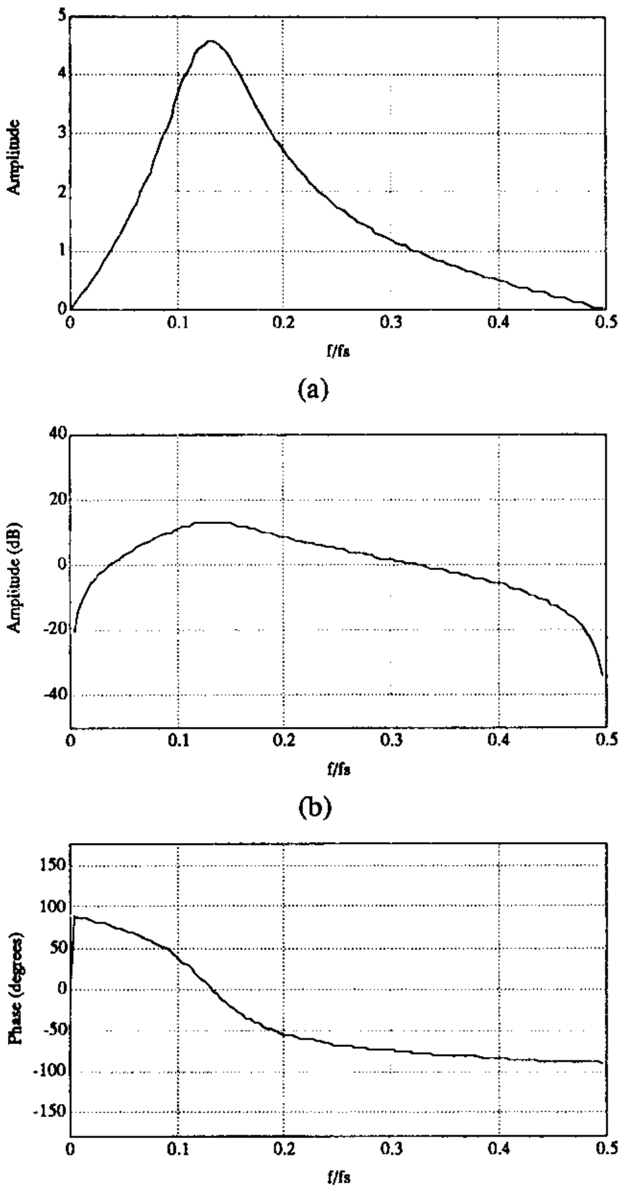


Figure 4.9 Frequency response for the filter of Figure 4.8. (a) Amplitude response with linear amplitude scale. (b) Amplitude response with decibel amplitude scale. (c) Phase response.

Compare this response to Figure 4.8(h) which shows the same information as a projection of the response from the circular unit circle axis. Figure 4.9(b) shows the same amplitude response plotted as a more familiar decibel (dB) plot.

Figure 4.10 shows how the radial locations of the poles influence the distortion of the rubber membrane. Note how moving the poles toward the unit circle makes the peaks appear sharper. The closer the poles get to the unit circle, the sharper the rolloff of the filter.

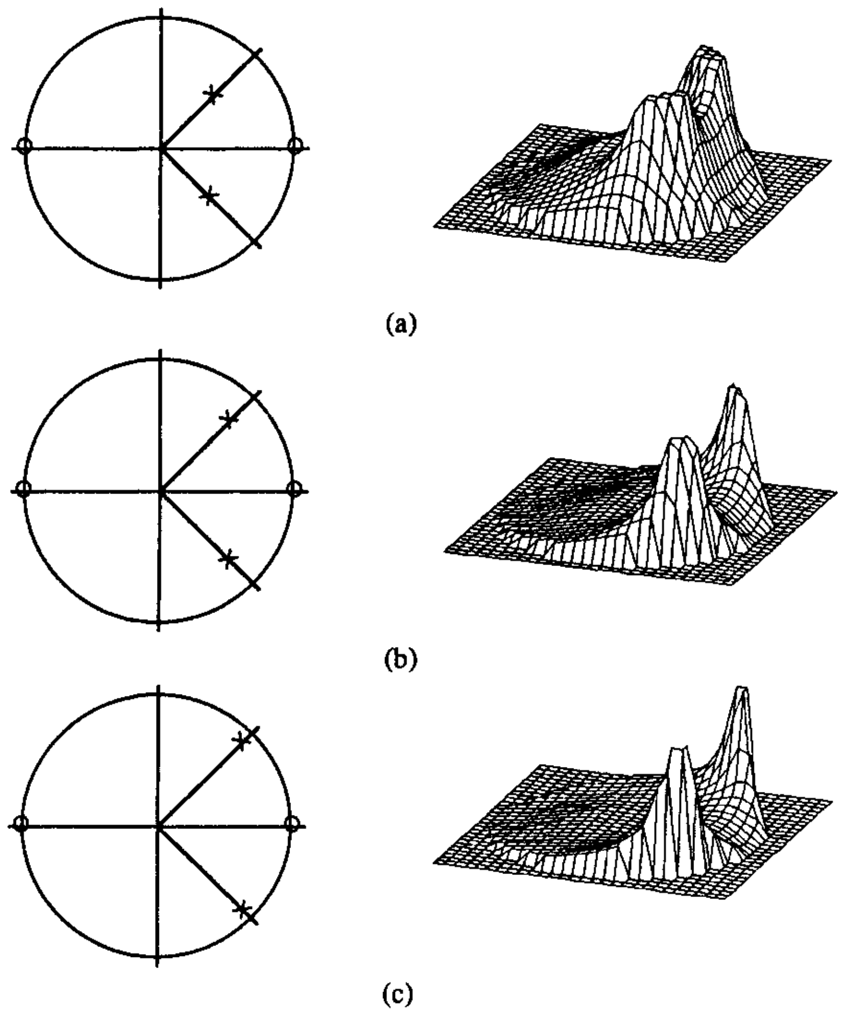
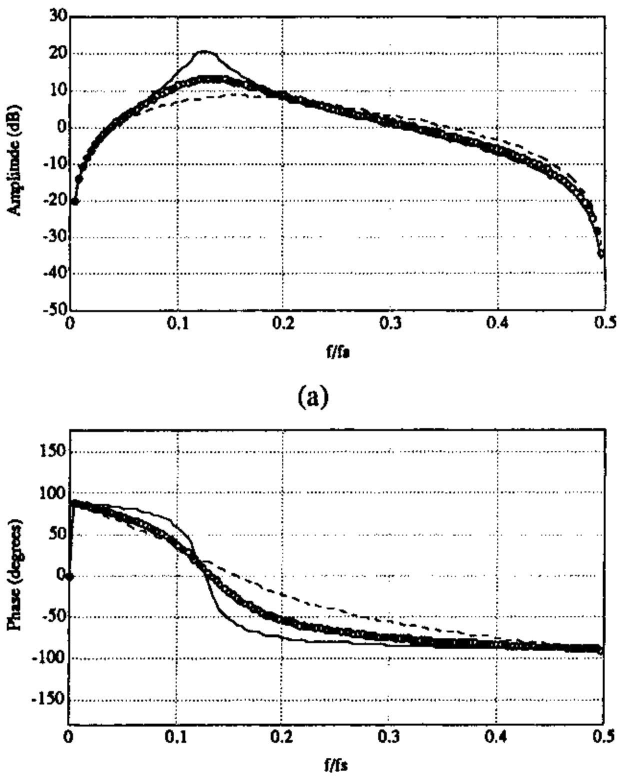


Figure 4.10 Pole-zero and rubber membrane plots of bandpass filter. Angular frequency of pole locations is $f_s/8$. (a) r = 0.5. (b) r = 0.75. (c) r = 0.9.

The phase response for this filter, which is plotted in Figure 4.9(c), is

$$\angle H(\omega T) = \tan^{-1}\left(\frac{BC - AD}{AC + BD}\right) \tag{4.58}$$

Figure 4.11(a) gives a superposition of the amplitude responses for each of these three pole placements. Note how the filter with the pole closest to the unit circle has the sharpest rolloff (i.e., the highest Q) of the three filters, thereby providing the best rejection of frequencies outside its 3-dB passband.



(a)

(b)

Figure 4.11 Amplitude and phase response plots for the bandpass filters of Figure 4.10, all with poles at angles of ±45° (i.e., $f/f_s = 0.125$). Solid line: Poles at $r = 0.9$. Circles: $r = 0.75$. Dashed line: $r = 0.5$.

As the pole moves toward the center of the unit circle, the rolloff becomes decreasingly sharp. Figure 4.11(b) shows the phase responses of these three filters. Note how the phase response becomes more and more linear as the poles are moved toward the center of the unit circle. We will see in the next chapter that the phase response becomes completely piecewise linear when all the poles are at the center of the unit circle (i.e., at $r = 0$).

Thus, the rubber membrane concept helps us to visualize the behavior of any digital filter and can be used as an important tool in the design of digital filters.

4.8 REFERENCES

Antoniou, A. 1979. *Digital Filters: Analysis and Design,* New York: McGraw-Hill.

Bogner, R. E. and Constantinides, A. G. 1985. *Introduction to Digital Filtering,* New York: John Wiley and Sons.

Gold, B. and Rader, C. 1969. *Digital Processing of Signals,* New York: Lincoln Laboratory Publications, McGraw-Hill.

Rabiner, L. R. and Rader, C. M. 1972. *Digital Signal Processing* New York: IEEE Press.

Stearns, S. D. 1975. *Digital Signal Analysis.* Rochelle Park, NJ: Hayden.

4.9 STUDY QUESTIONS

4.1 What are the differences between an analog filter and a digital filter?

4.2 If the output sequence of a digital filter is {1, 0, 0, 2, 0, 1} in response to a unit impulse, what is the transfer function of this filter?

4.3 Draw the pole-zero plot of the filter described by the following transfer function:

$$H(z) = \frac{1}{4} + \frac{1}{2}z^{-1} + \frac{1}{4}z^{-2}$$

4.4 Suppose you are given a filter with a zero at 30° on the unit circle. You are asked to use this filter as a notch filter to remove 60-Hz noise. How will you do this? Can you use the same filter as a notch filter, rejecting different frequencies?

4.5 What is the z transform of a step function having an amplitude of five {i.e., 5, 5, 5, 5, ...}?

4.6 A function e^{-at} is to be applied to the input of a filter. Derive the z transform of the discrete version of this function.

4.7 Application of a *unit impulse* to the input of a filter whose performance is unknown produces the output sequence {1, –2, 0, 0, ...}. What would the output sequence be if a *unit step* were applied?

4.8 A digital filter has the transfer function: $H(z) = z^{-1} + 6z^{-4} - 2z^{-7}$. What is the difference equation for the output, $y(nT)$?

4.9 A digital filter has the output sequence {1, 2, –3, 0, 0, 0, ...} when its input is the *unit impulse* {1, 0, 0, 0, 0, ...}. If its input is a *unit step*, what is its output sequence?

4.10 A *unit impulse* applied to a digital filter results in the output sequence: {3, 2, 3, 0, 0, 0, ...}. A *unit step* function applied to the input of the same filter would produce what output sequence?

4.11 The z transform of a filter is: $H(z) = 2 - 2z^{-4}$. What is its (a) amplitude response, (b) phase response, (c) difference equation?

4.12 The transfer function of a filter designed for a sampling rate of 800 samples/s is:

$$H(z) = (1 - 0.5z^{-1})(1 + 0.5z^{-1})$$

A 200-Hz sine wave with a peak amplitude of 4 is applied to the input. What is the peak value of the output signal?

4.13 A *unit impulse* applied to a digital filter results in the following output sequence: {1, 2, 3, 4, 0, 0, ...}. A *unit step* function applied to the input of the same filter would produce what output sequence?

4.14 The transfer function of a filter designed for a sampling rate of 600 samples/s is:

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

A sinusoidal signal is applied to the input: $10 \sin(628t)$. What is the peak value of the output signal?

5

Finite Impulse Response Filters

Jesse D. Olson

A finite impulse response (FIR) filter has a unit impulse response that has a limited number of terms, as opposed to an infinite impulse response (IIR) filter which produces an infinite number of output terms when a unit impulse is applied to its input. FIR filters are generally realized nonrecursively, which means that there is no feedback involved in computation of the output data. The output of the filter depends only on the present and past inputs. This quality has several important implications for digital filter design and applications. This chapter discusses several FIR filters typically used for real-time ECG processing, and also gives an overview of some general FIR design techniques.

5.1 CHARACTERISTICS OF FIR FILTERS

5.1.1 Finite impulse response

Finite impulse response implies that the effect of transients or initial conditions on the filter output will eventually die away. Figure 5.1 shows a signal-flow graph (SFG) of a FIR filter realized nonrecursively. The filter is merely a set of "tap weights" of the delay stages. The unit impulse response is equal to the tap weights, so the filter has a difference equation given by Eq. (5.1), and a transfer function equation given by Eq. (5.2).

$$y(nT) = \sum_{k=0}^{N} b_k x(nT - kT) \tag{5.1}$$

$$H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_N z^{-N} \tag{5.2}$$

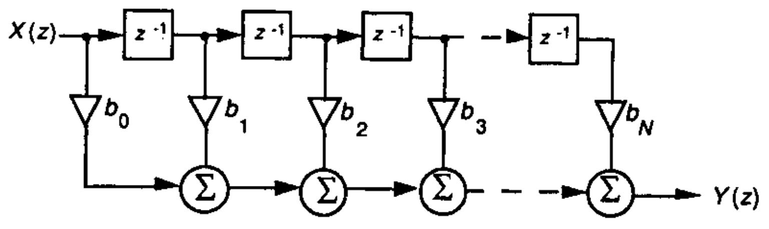


Figure 5.1 The output of a FIR filter of order N is the weighted sum of the values in the storage registers of the delay line.

5.1.2 Linear phase

In many biomedical signal processing applications, it is important to preserve certain characteristics of a signal throughout the filtering operation, such as the height and duration of the QRS pulse. A filter with linear phase has a pure time delay as its phase response, so phase distortion is minimized. A filter has linear phase if its frequency response $H(e^{j\theta})$ can be expressed as

$$H(e^{j\theta}) = H_1(\theta)\, e^{-j(\alpha\theta + \beta)} \tag{5.3}$$

where $H_1(\theta)$ is a real and even function, since the phase of $H(e^{j\theta})$ is

$$\angle H(e^{j\theta}) = \begin{cases} -\alpha\theta - \beta & ; H_1(\theta) > 0 \\ -\alpha\theta - \beta - \pi & ; H_1(\theta) < 0 \end{cases} \tag{5.4}$$

FIR filters can easily be designed to have a linear phase characteristic. Linear phase can be obtained in four ways, as combinations of even or odd symmetry (defined as follows) with even or odd length.

$$\left.\begin{aligned} h(N-1-k) &= h(k), \textit{ even symmetry} \\ h(N-1-k) &= -h(k), \textit{ odd symmetry} \end{aligned}\right\} \quad \text{for} \quad 0 \le k \le N \tag{5.5}$$

5.1.3 Stability

Since a nonrecursive filter does not use feedback, it has no poles except those that are located at $z = 0$. Thus there is no possibility for a pole to exist outside the unit circle. This means that it is inherently stable. As long as the input to the filter is bounded, the output of the filter will also be bounded. This contributes to ease of design, and makes FIR filters especially useful for adaptive filtering where filter coefficients change as a function of the input data. Adaptive filters are discussed in Chapter 8.

5.1.4 Desirable finite-length register effects

When data are converted from analog form to digital form, some information is lost due to the finite number of storage bits. Likewise, when coefficient values for a filter are calculated, digital implementation can only approximate the desired values. The limitations introduced by digital storage are termed finite-length register effects. Although we will not treat this subject in detail in this book, finite-length register effects can have significant negative impact on a filter design. These effects include quantization error, roundoff noise, limit cycles, conditional stability, and coefficient sensitivity. In FIR filters, these effects are much less significant and easier to analyze than in IIR filters since the errors are not fed back into the filter. See Appendix F for more details about finite-length register effects.

5.1.5 Ease of design

All of the above properties contribute to the ease in designing FIR filters. There are many straightforward techniques for designing FIR filters to meet arbitrary frequency and phase response specifications, such as window design or frequency sampling. Many software packages exist that automate the FIR design process, often computing a filter realization that is in some sense optimal.

5.1.6 Realizations

There are three methods of realizing an FIR filter (Bogner and Constantinides, 1985). The most common method is direct convolution, in which the filter's unit impulse sequence is convolved with the present input and past inputs to compute each new output value. FIR filters attenuate the signal very gradually outside the passband (i.e., they have slow rolloff characteristics). Since they have significantly slower rolloff than IIR filters of the same length, for applications that require sharp rolloffs, the order of the FIR filter may be quite large. For higher-order filters the direct convolution method becomes computationally inefficient.

For FIR filters of length greater than about 30, the "fast convolution" realization offers a computational savings. This technique takes advantage of the fact that time-domain multiplication, the frequency-domain dual of convolution, is computationally less intensive. Fast convolution involves taking the FFT of a block of data, multiplying the result by the FFT of the unit impulse sequence, and finally taking the inverse FFT. The process is repeated for subsequent blocks of data. This method is discussed in detail in section 11.3.2.

The third method of realizing FIR filters is an advanced, recursive technique involving a comb filter and a bank of parallel digital resonators (Rabiner and Rader, 1972). This method is advantageous for frequency sampling designs if a large number of the coefficients in the desired frequency response are zero, and can be used for filters with integer-valued coefficients, as discussed in Chapter 7. For the remainder of this chapter, only the direct convolution method will be considered.

5.2 SMOOTHING FILTERS

One of the most common signal processing tasks is smoothing of the data to reduce high-frequency noise. Some sources of high-frequency noise include 60-Hz, movement artifacts, and quantization error. One simple method of reducing high-frequency noise is to simply average several data points together. Such a filter is referred to as a moving average filter.

5.2.1 Hanning filter

One of the simplest smoothing filters is the Hanning moving average filter. Figure 5.2 summarizes the details of this filter. As illustrated by its difference equation, the Hanning filter computes a weighted moving average, since the central data point has twice the weight of the other two:

$$y(nT) = \frac{1}{4}\left[x(nT) + 2x(nT-T) + x(nT-2T)\right] \tag{5.6}$$

As we saw in section 4.5, once we have the difference equation representing the numerical algorithm for implementing a digital filter, we can quickly determine the transfer equation that totally characterizes the performance of the filter by using the analogy between discrete-time variables and z-domain variables.

Recognizing that $x(nT)$ and $y(nT)$ are points in the input and output sequences associated with the current sample time, they are analogous to the undelayed z-domain variables, $X(z)$ and $Y(z)$ respectively. Similarly $x(nT-T)$, the input value one sample point in the past, is analogous to the z-domain input variable delayed by one sample point, or $X(z)z^{-1}$. We can then write an equation for output $Y(z)$ as a function of input $X(z)$:

$$Y(z) = \frac{1}{4}\left[X(z) + 2X(z)z^{-1} + X(z)z^{-2}\right] \tag{5.7}$$

The block diagram of Figure 5.2(a) is drawn using functional blocks to directly implement the terms in this equation. Two delay blocks are required as designated by the -2 exponent of z. Two multipliers are necessary to multiply by the factors 2 and 1/4, two summers are needed to combine the terms. The transfer function of this equation is

$$H(z) = \frac{1}{4}\left[1 + 2z^{-1} + z^{-2}\right] \tag{5.8}$$

This filter has two zeros, both located at $z = -1$, and two poles, both located at $z = 0$ (see section 4.6 to review how to find pole and zero locations). Figure 5.2(b) shows the pole-zero plot. Note the poles are implicit; they are not drawn since they influence all frequencies in the amplitude response equally.

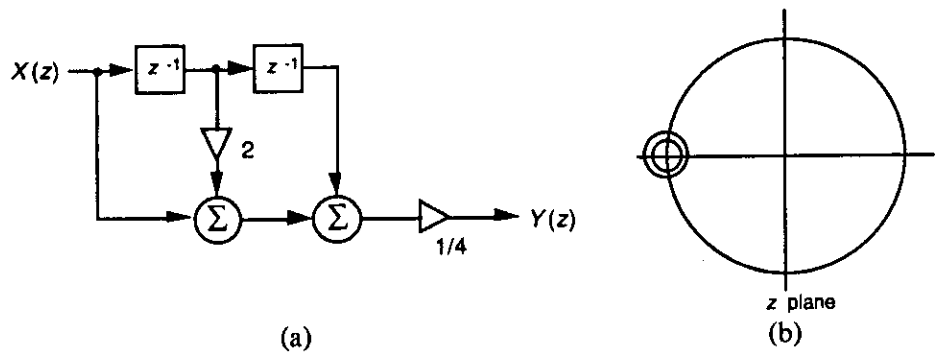


Figure 5.2 Hanning filter. (a) Signal-flow graph. (b) Pole-zero diagram.

The filter's amplitude and phase responses are found by substituting $e^{j\omega T}$ for z in Eq. (5.8):

$$H(\omega T) = \frac{1}{4}\left[1 + 2e^{-j\omega T} + e^{-j2\omega T}\right] \tag{5.9}$$

We could now directly substitute into this function the trigonometric relationship

$$e^{j\omega T} = \cos(\omega T) + j\sin(\omega T) \tag{5.10}$$

However, a common trick prior to this substitution that leads to quick simplification of expressions such as this one is to extract a power of e as a multiplier such that the final result has two similar exponential terms with equal exponents of opposite sign

$$H(\omega T) = \frac{1}{4}\left[e^{-j\omega T}\left(e^{j\omega T} + 2 + e^{-j\omega T}\right)\right] \tag{5.11}$$

Now substituting Eq. (5.10) for the terms in parentheses yields

$$H(\omega T) = \frac{1}{4}\left\{e^{-j\omega T}\left[\cos(\omega T) + j\sin(\omega T) + 2 + \cos(\omega T) - j\sin(\omega T)\right]\right\} \tag{5.12}$$

The sin(ωT) terms cancel leaving

$$H(\omega T) = \frac{1}{4}\left[(2 + 2\cos(\omega T))e^{-j\omega T}\right] \tag{5.13}$$

This is of the form $Re^{j\theta}$ where R is the real part and θ is the phase angle. Thus, the magnitude response of the Hanning filter is $|R|$, or

$$|H(\omega T)| = \left|\frac{1}{2}\left[1 + \cos(\omega T)\right]\right| \tag{5.14}$$

Figure 5.3(a) shows this cosine-wave amplitude response plotted with a linear ordinate scale while Figure 5.3(b) shows the same response using the more familiar decibel plot, which we will use throughout this book. The relatively slow rolloff of the Hanning filter can be sharpened by passing its output into the input of another identical filter. This process of connecting multiple filters together is called cascading filters. The linear phase response shown in Figure 5.3(c) is equal to angle θ, or

$$\angle H(\omega T) = -\omega T \tag{5.15}$$

Implementation of the Hanning filter is accomplished by writing a computer program. Figure 5.4 illustrates a C-language program for an off-line (i.e., not real-time) application where data has previously been sampled by an A/D converter and left in an array. This program directly computes the filter's difference equation [Eq. (5.6)]. Within the `for()` loop, a value for $x(nT)$ (called `xnt` in the program) is obtained from the array `idb[]`. The difference equation is computed to find the output value $y(nT)$ (or `ynt` in the program). This value is saved into the data array, replacing the value of $x(nT)$. Then the input data variables are shifted through the delay blocks. Prior to the next input, the data point that was one point in the past $x(nT - T)$ (called `xm1` in the program) moves two points in the past and becomes $x(nT - 2T)$ (or `xm2`). The most recent input $x(nT)$ (called `xnt`) moves one point back in time, replacing $x(nT - T)$ (or `xm1`). In the next iteration of the `for()` loop, a new value of $x(nT)$ is retrieved, and the process repeats until all 256 array values are processed. The filtered output waveform is left in array `idb[]`.

The Hanning filter is particularly efficient for use in real-time applications since all of its coefficients are integers, and binary shifts can be used instead of multiplications. Figure 5.5 is a real-time Hanning filter program. In this program, the computation of the output value $y(nT)$ must be accomplished during one sample interval T. That is, every new input data point acquired by the A/D converter must produce an output value before the next A/D input. Otherwise the filter would not keep up with the sampling rate, and it would not be operating in real time.

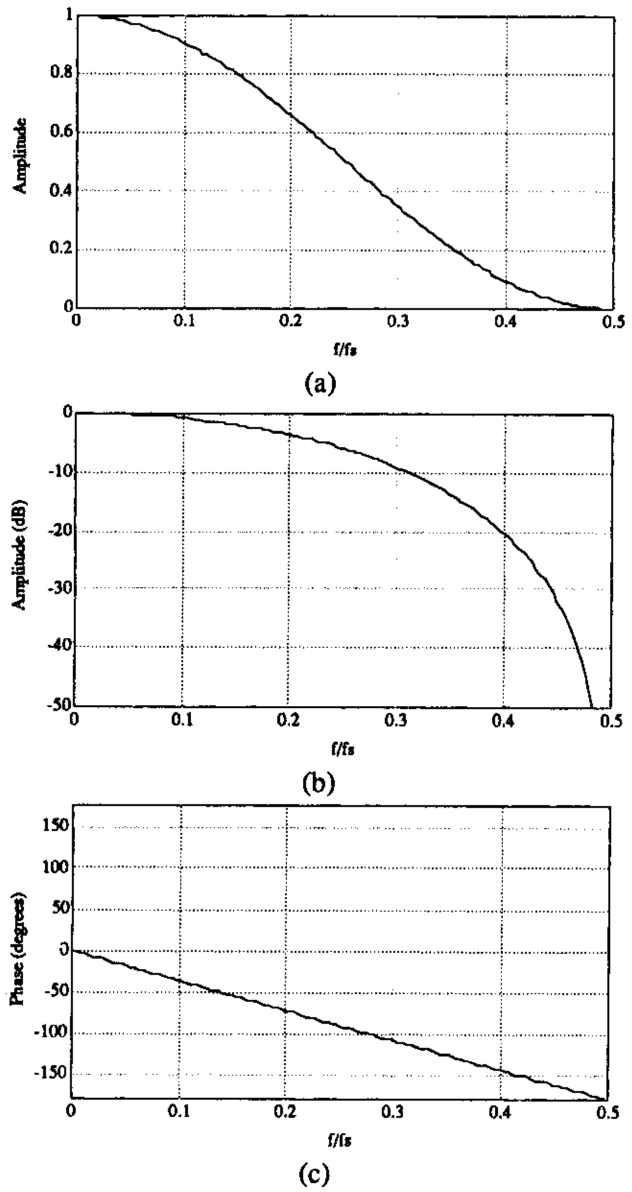


Figure 5.3 Hanning filter. (a) Frequency response (linear magnitude axis). (b) Frequency response (dB magnitude axis). (c)Phase response.

In this program, sampling from the A/D converter, computation of the results, and sending the filtered data to a D/A converter are all accomplished within a `for()` loop. The `wait()` function is designed to wait for an interrupt caused by an A/D clock tick. Once the interrupt occurs, a data point is sampled with the `adget()` function and set equal to `xnt`. The Hanning filter's difference equation is then computed using C-language shift operators to do the multiplications efficiently. Expression `<<1` is a binary shift to the left by one bit position corresponding to multiplication by a factor of two, and `>>2` is a binary shift right of two bit positions representing division by four.

```
/*  Hanning filter
    Difference equation:
        y(nT) = (x(nT) + 2*x(nT - T) + x(nT - 2T))/4
    C language implementation equation:
        ynt = (xnt + 2*xm1 + xm2)/4;
*/

main()
{
      int i, xnt, xm1, xm2, ynt, idb[256];

      xm2 = 0;
      xm1 = 0;

      for(i = 0; i <= 255; i++)
          {
          xnt = idb[i];
          ynt = (xnt + 2*xm1 + xm2)/4;
          idb[i] = ynt;
          xm2 = xm1;
          xm1 = xnt;
          }
}
```

Figure 5.4 C-language code to implement the Hanning filter. Data is presampled by an ADC and stored in array `idb[]`. The filtered signal is left in `idb[]`.

The computed output value is sent to a D/A converter with function `daput()`. Then the input data variables are shifted through the delay blocks as in the previous program. For the next input, the data point that was one point in the past $x(nT-T)$ (called `xm1` in the program) moves two points in the past and becomes $x(nT-2T)$ (or `xm2`). The most recent input $x(nT)$ (called `xnt`) moves one point back in time, replacing $x(nT-T)$ (or `xm1`). Then the `for()` loop repeats with the `wait()` function waiting until the next interrupt signals that a new sampled data point is available to be acquired by `adget()` as the new value for $x(nT)$.

```
/*  Real-time Hanning filter
    Difference equation:
        y(nT) = (x(nT) + 2*x(nT - T) + x(nT - 2T))/4
    C language implementation equation:
        ynt = (xnt + xm1<<1 + xm2)>>2;
*/

#define  AD  31;
#define  DA  32;

main()
{
      int i, xnt, xm1, xm2, ynt;

      xm2 = 0;
      xm1 = 0;

      tmic 2000;        /* Start ADC clock ticking at 2000 µs  */
                        /* intervals (2 ms period for 500 sps) */

      for( ; ; )
          {
          wait();               /* Wait for ADC clock to tick  */
          xnt = adget(AD);
          ynt = (xnt + xm1<<1 + xm2)>>2;
          daput(ynt, DA);
          xm2 = xm1;
          xm1 = xnt;
          }
}
```

Figure 5.5 C-language code to implement the real-time Hanning filter.

5.2.2 Least-squares polynomial smoothing

This family of filters fits a parabola to an odd number $(2L + 1)$ of input data points in a least-squares sense. Figure 5.6(a) shows that the output of the filter is the midpoint of the parabola. Writing the equation for a parabola at each input point, we obtain

$$p(nT + kT) = a(nT) + b(nT)k + c(nT)k^2 \tag{5.16}$$

where k ranges from $-L$ to L. The fit is found by selecting $a(nT)$, $b(nT)$ and $c(nT)$ to minimize the squared error between the parabola and the input data. Setting the partial derivatives of the error with respect to $a(nT)$, $b(nT)$, and $c(nT)$ equal to zero results in a set of simultaneous equations in $a(nT)$, $b(nT)$, $c(nT)$, k, and $p(nT - kT)$. Solving to obtain an expression for $a(nT)$, the value of the parabola at $k = 0$, yields an expression that is a function of the input values. The coefficients of this expression are the tap weights for the least-squares polynomial filter as shown in the signal-flow graph of Figure 5.6(b) for a five-point filter. The difference equation for the five-point parabolic filter is

$$y(nT) = \frac{1}{35}[(-3x(nT) + 12x(nT-T) + 17x(nT-2T) + 12x(nT-3T) - 3x(nT-4T)] \tag{5.17}$$

Its transfer function is

$$H(z) = \frac{1}{35}[-3 + 12z^{-1} + 17z^{-2} + 12z^{-3} - 3z^{-4}] \tag{5.18}$$

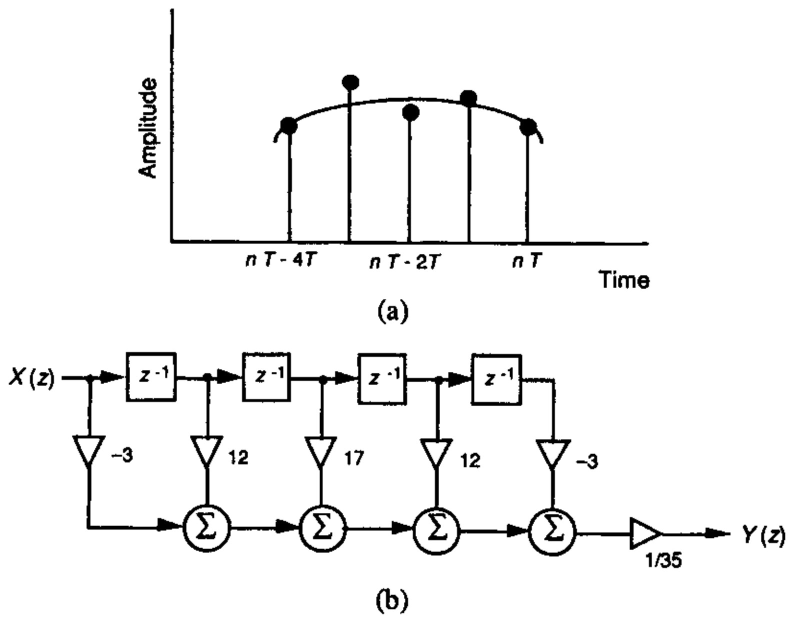


Figure 5.6 Polynomial smoothing filter with $L = 2$. (a) Parabolic fitting of groups of 5 sampled data points. (b) Signal-flow graph.

Figure 5.7 shows the tap weights for filters with L equal to 2, 3, 4, and 5, and Figure 5.8 illustrates their responses. The order of the filter can be chosen to meet the desired rolloff.

L	Tap weights
2	$\frac{1}{35}(-3, 12, 17, 12, -3)$
3	$\frac{1}{21}(-2, 3, 6, 7, 6, 3, -2)$
4	$\frac{1}{231}(-21, 14, 39, 54, 59, 54, 39, 14, -21)$
5	$\frac{1}{429}(-36, 9, 44, 69, 84, 89, 84, 69, 44, 9, -36)$

Figure 5.7 Tap weights of polynomial smoothing filters (Hamming, 1977).

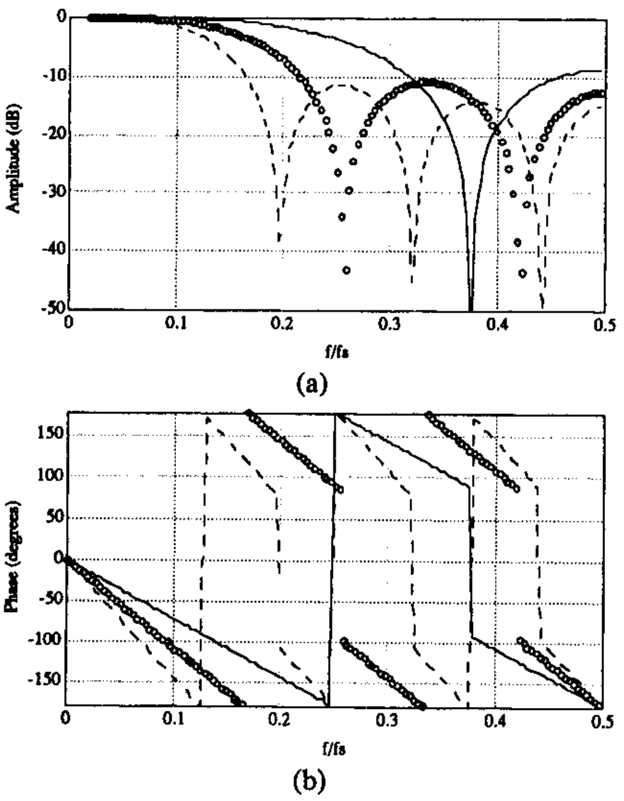


Figure 5.8 Polynomial smoothing filters (a) Amplitude responses. (b) Phase responses. Solid line: $L = 2$. Circles: $L = 3$. Dashed line: $L = 4$.

5.3 NOTCH FILTERS

A common biomedical signal processing problem involves the removal of noise of a particular frequency or frequency range (such as 60 Hz) from a signal while passing higher and/or lower frequencies without attenuation. A filter that performs this task is referred to as a notch, bandstop, or band-reject filter.

One simple method of completely removing noise of a specific frequency from the signal is to place a zero on the unit circle at the location corresponding to that frequency. For example, if a sampling rate of 180 samples per second is used, a zero at $2\pi/3$ removes 60-Hz line frequency noise from the signal. The difference equation is

$$y(nT) = \frac{1}{3}\left[x(nT) + x(nT - T) + x(nT - 2T)\right] \tag{5.19}$$

The filter has zeros at

$$z = -0.5 \pm j\,0.866 \tag{5.20}$$

and its amplitude and phase response are given by

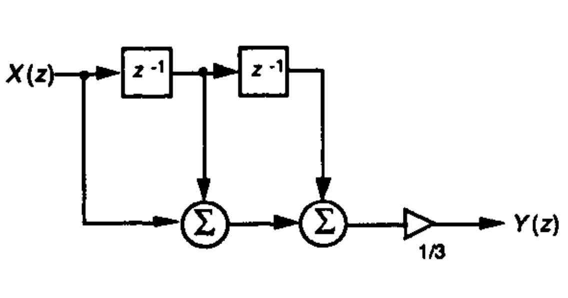
$$|H(\omega T)| = \left|\frac{1}{3}\left[1 + 2\cos(\omega T)\right]\right| \tag{5.21}$$

$$\angle H(\omega T) = -\omega T \tag{5.22}$$

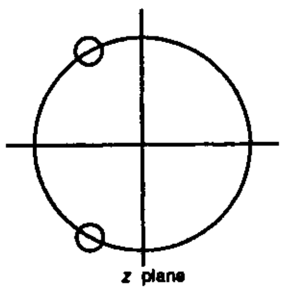
Figure 5.9 shows the details of the design and performance of this filter. The relatively slow rolloff of this filter causes significant attenuation of frequencies other than 60 Hz as well.

5.4 DERIVATIVES

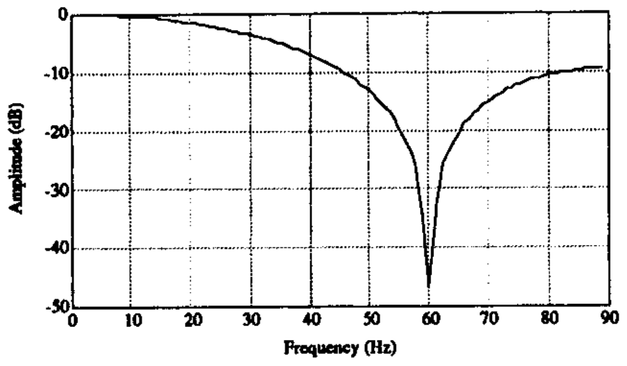
The response of the true derivative function increases linearly with frequency. However, for digital differentiation such a response is not possible since the frequency response is periodic. The methods discussed in this section offer trade-offs between complexity of calculation, approximation to the true derivative, and elimination of high-frequency noise. Figure 5.10 shows the signal-flow graphs and pole-zero plots for three different differentiation algorithms: two-point difference, three-point central difference, and least-squares polynomial fit.



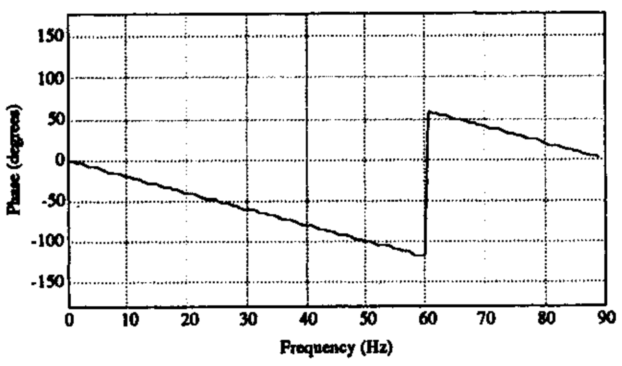
(a)



(b)

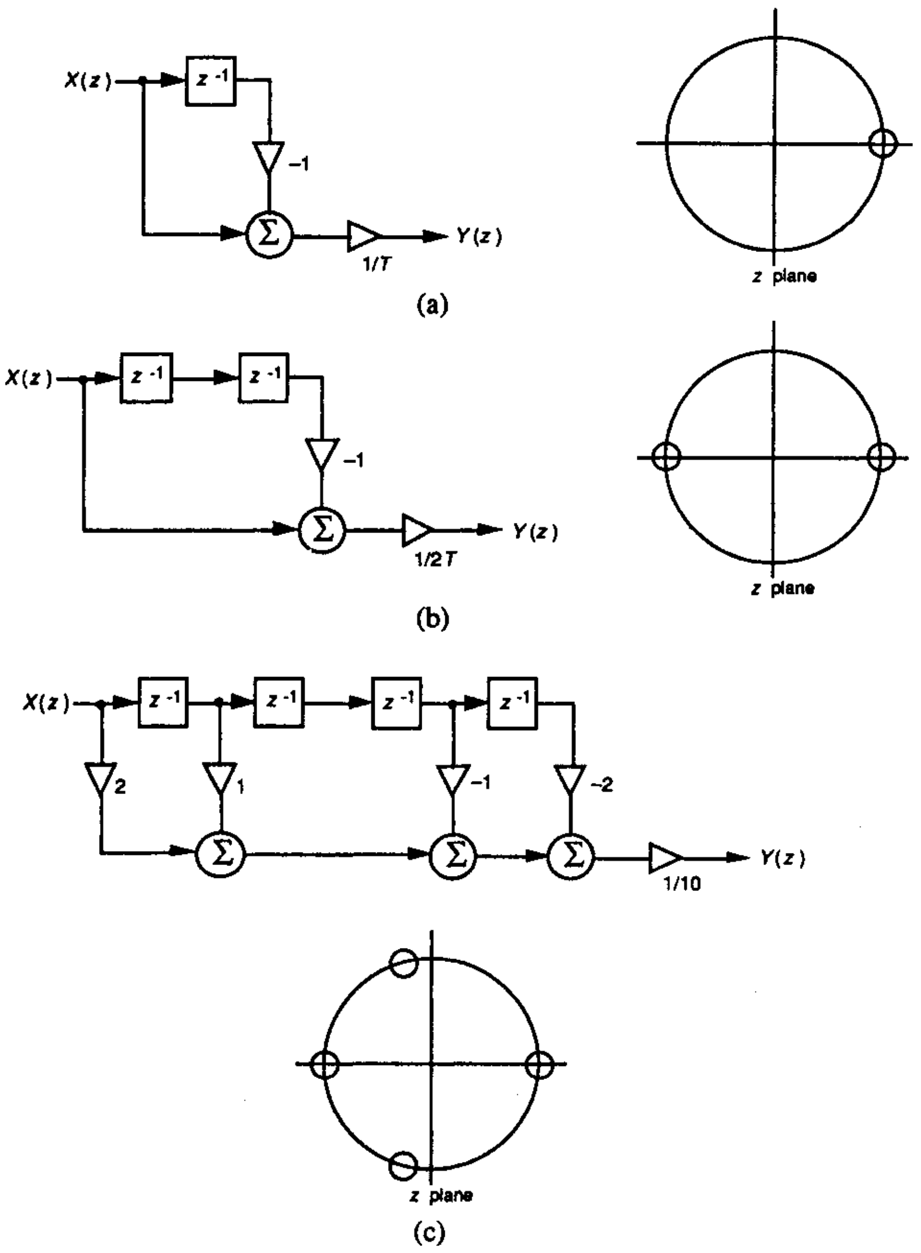


(c)



(d)

Figure 5.9 The 60-Hz notch filter. (a) Signal-flow graph. (b) Pole-zero plot. (c) Frequency response. (d) Phase response.



igure 5.10 Signal flow graphs and pole-zero diagrams for derivative algorithms. (a) Two-point.) Three-point central difference. (c) 5-point least squares polynomial.

5.4.1 Two-point difference

The two-point difference algorithm, the simplest of these derivative algorithms, places a zero at $z = 1$ on the unit circle. Its amplitude response shown in Figure 5.11(a) closely approximates the ideal response, but since it does not go to zero at $f_s/2$, it greatly amplifies high-frequency noise. It is often followed by a low-pass filter. Its difference equation is

$$y(nT) = \frac{1}{T}\left[x(nT) - x(nT - T)\right] \tag{5.23}$$

Its transfer function is

$$H(z) = \frac{1}{T}\,(1 - z^{-1}) \tag{5.24}$$

5.4.2 Three-point central difference

The three-point central difference algorithm places zeros at $z = 1$ and $z = -1$, so the approximation to the derivative is poor above $f_s/10$ seen in Figure 5.11(a). However, since the response goes to zero at $f_s/2$, the filter has some built-in smoothing. Its difference equation is

$$y(nT) = \frac{1}{2T}\left[x(nT) - x(nT - 2T)\right] \tag{5.25}$$

Its transfer function is

$$H(z) = \frac{1}{2T}\,(1 - z^{-2}) \tag{5.26}$$

5.4.3 Least-squares polynomial derivative approximation

This filter is similar to the parabolic smoothing filter described earlier, except that the slope of the polynomial is taken at the center of the parabola as the value of the derivative. The coefficients of the transfer equations for filters with $L = 2, 3, 4$, and 5 are illustrated in Figure 5.12. Figure 5.10(c) shows the signal-flow graph and pole-zero diagram for this filter with $L = 2$. Note that the filter has zeros at $z = \pm 1$, as did the three-point central difference, with additional zeros at $z = -0.25 \pm j0.968$. The difference equation for the five-point parabolic filter is

$$y(nT) = \frac{1}{10T}\left[2x(nT) + x(nT - T) - x(nT - 3T) - 2x(nT - 4T)\right] \tag{5.27}$$

Its transfer function is

$$H(z) = \frac{1}{10T}\left[2 + z^{-1} - z^{-3} - 2z^{-4}\right] \tag{5.28}$$

As Figure 5.11(a) shows, the response only approximates the true derivative at low frequencies, since the smoothing nature of the parabolic fit attenuates high frequencies significantly.

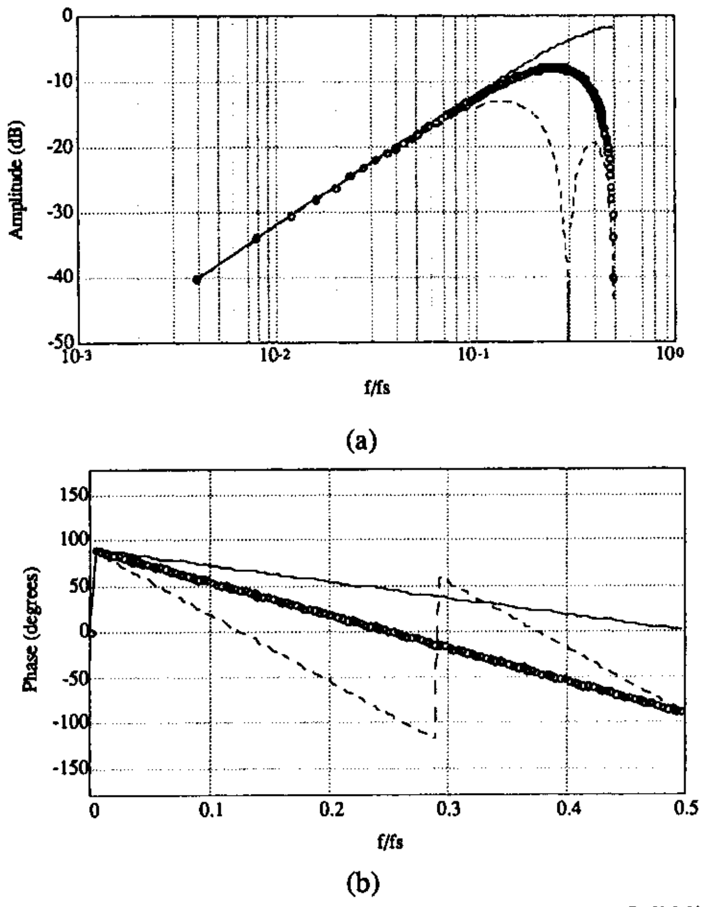


Figure 5.11 Derivatives. (a) Amplitude response. (b) Phase response. Solid line: Two-point. Circles: Three-point central difference. Dashed line: Least-squares parabolic approximation for $L = 2$.

L	Tap weights
2	$\frac{1}{10T}$ (2, 1, 0, –1, –2)
3	$\frac{1}{28T}$ (3, 2, 1, 0, –1, –2, –3)
4	$\frac{1}{60T}$ (4, 3, 2, 1, 0, –1, –2, –3, –4)
5	$\frac{1}{110T}$ (5, 4, 3, 2, 1, 0, –1, –2, –3, –4, –5)

Figure 5.12 Least-squares derivative approximation coefficients for $L = 2, 3, 4$, and 5.

5.4.4 Second derivative

Figure 5.13 shows a simple filter for approximating the second derivative (Friesen et al., 1990), which has the difference equation

$$y(nT) = x(nT) - 2x(nT - 2T)) + x(nT - 4T) \qquad (5.29)$$

This filter was derived by cascading two stages of the three-point central difference derivative of Eq. (5.26) and setting the amplitude multiplier to unity to obtain the transfer function

$$H(z) = (1 - z^{-2}) \times (1 - z^{-2}) = (1 - 2z^{-2} - z^{-4}) \qquad (5.30)$$

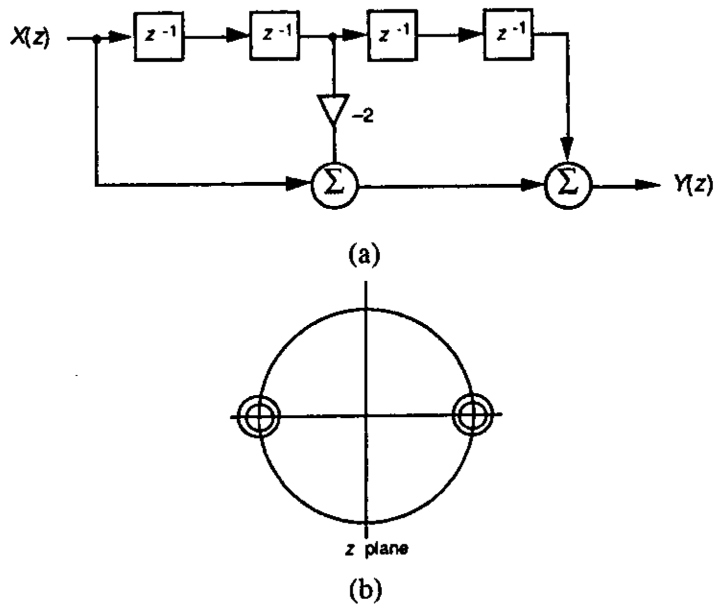


Figure 5.13 Second derivative. (a) Signal-flow graph. (b) Unit-circle diagram.

5.5 WINDOW DESIGN

A desired frequency response $H_d(\theta)$ (a continuous function) has as its inverse discrete-time Fourier transform (IDTFT), which is the desired unit pulse sequence, $h_d(k)$ (a discrete function). This sequence will have an infinite number of terms, so it is not physically realizable. The objective of window design is to choose an actual $h(k)$ with a finite number of terms such that the frequency response $H(e^{j\theta})$ will be in some sense close to $H_d(\theta)$. If the objective is to minimize the mean-squared error between the actual frequency response and the desired frequency response, then it can be shown by Parseval's theorem that the error is minimized by directly truncating $h_d(k)$. In other words, the pulse response $h(k)$ is chosen to be the first N terms of $h_d(k)$. Unfortunately, such a truncation results in large overshoots at sharp transitions in the frequency response, referred to as *Gibb's phenomenon*, illustrated in Figure 5.14.

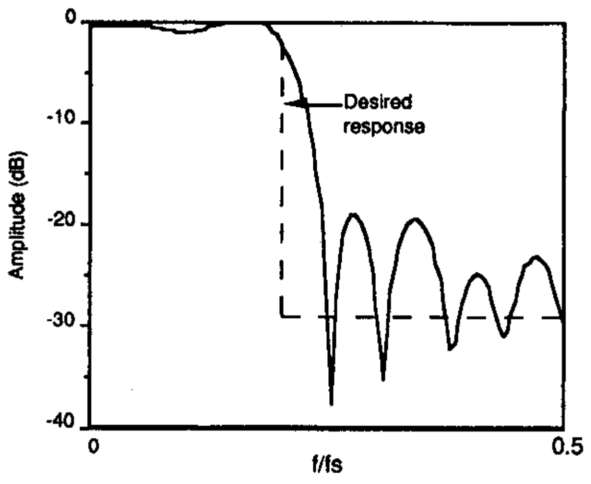


Figure 5.14 The overshoot that occurs at sharp transitions in the desired frequency response due to truncation of the pulse response is referred to as Gibb's phenomenon.

To understand this effect, consider that direct truncation of $h_d(k)$ is a multiplication of the desired unit pulse sequence by a rectangular window $w_R(k)$. Since multiplication in the time domain corresponds to convolution in the frequency domain, the frequency response of the resulting $h(k)$ is the convolution of the desired frequency response with the frequency response of the window function.

For example, consider that the frequency response of a rectangular window of infinite length is simply a unit pulse. The convolution of the desired frequency response with the unit pulse simply returns the desired response. However, as the width of the window decreases, its frequency response becomes less like an impulse function and sidelobes become more evident. The convolution of these sidelobes with the desired frequency response results in the overshoots at the transitions.

For a rectangular window, the response function $W_R(e^{j\theta})$ is given by

$$W_R(e^{j\theta}) = \frac{\sin\left(\frac{N\theta}{2}\right)}{\sin\left(\frac{\theta}{2}\right)} \tag{5.31}$$

where N is the length of the rectangular window. There are two important results of the convolution of the window with the desired frequency response. First the window function *smears* the desired response at transitions. This smearing of the transition bands increases the width of the transition from passband to stopband. This has the effect of increasing the width of the *main lobe* of the filter. Second, windowing causes undesirable ripple called *window leakage* or *sideband ripple* in the stopbands of the filter.

By using window functions other than a rectangular window, stopband ripple can be reduced at the expense of increasing main lobe width. Many types of windows have been studied in detail, including triangular (Bartlett), Hamming, Hanning, Blackman, Kaiser, Chebyshev, and Gaussian windows. Hanning and Hamming windows are of the form

$$w_H(k) = w_R(k)\left[\alpha + (1-\alpha)\cos\left(\frac{2\pi}{N}k\right)\right] \qquad \text{for } 0 < \alpha < 1 \tag{5.32}$$

where $\alpha = 0.54$ for the Hamming window and $\alpha = 0.50$ for the Hanning window.

The Kaiser window is of the form

$$w_K(k) = w_R(k)\, I_0 \frac{\left(\alpha \sqrt{1-\left(\frac{k}{M}\right)^2}\right)}{I_0(\alpha)} \tag{5.33}$$

where α allows the designer to choose main lobe widths from the extreme minimum of the rectangular window to the width of the Blackman window, trading off sidelobe amplitude (Antoniou, 1979). See Roberts and Mullis (1987) for a discussion of the Chebyshev and Gaussian windows. All of these window functions taper to zero at the edges of the window. Figure 5.15 compares the performance of several of these windows.

Window Type	Sideband Ripple (dB)	Main Lobe Width
Rectangular	-13	$4\pi/N$
Triangular	-25	$8\pi/N$
Hanning	-31	$8\pi/N$
Hamming	-41	$8\pi/N$
Blackman	-38	$12\pi/N$

Figure 5.15 Responses of various windows.

Window design is usually performed iteratively, since it is difficult to predict the extent to which the transition band of the window will smear the frequency response. To design an FIR filter to meet a specific frequency response, one computes the unit pulse sequence of the desired response by taking the IDTFT, applies the window, and then computes the actual response by taking the DFT. The band edges or filter order are adjusted as necessary, and the process is repeated. Many algorithms have been developed to perform this process by computer. The technique can easily be constrained to generate filters with linear phase responses. Window designs do not generally yield the lowest possible order filter to meet the specifications. Windowing is also discussed in Chapter 11.

5.6 FREQUENCY SAMPLING

The frequency sampling method of design is more straightforward than the window design method since it circumvents the transformations from the time domain to the frequency domain. The terms in the filter response $H(e^{j\theta})$ are directly specified to match $H_d(\theta)$ at N uniformly spaced frequencies around the unit circle. As in the window design method, large overshoots will occur at sharp transitions in the response. This overshoot can be minimized by allowing some unconstrained terms in the transition band. Figure 5.16 illustrates a frequency sampling design with the same desired response and number of terms as in Figure 5.14, but with one unconstrained value chosen to minimize stopband ripple. The unconstrained values can be chosen to minimize some measure of error between $H(e^{j\theta})$ and $H_d(\theta)$. The frequency sampling method generally yields more efficient filters than the window method.

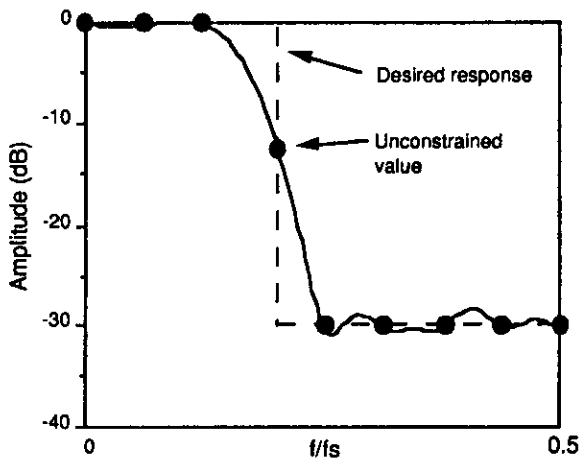


Figure 5.16 Frequency sampling design for $N = 16$ with one unconstrained term.

5.7 MINIMAX DESIGN

The window design method generates large errors at sharp transitions in the fre quency response. Rather than minimizing the energy in the error

$$\varepsilon^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H_d(\theta) - H_1(e^{j\theta}) \right|^2 d\theta \tag{5.34}$$

As in the window design method, it may be more desirable in certain application to minimize the maximum error

$$\max_{\theta} \left| H_d(\theta) - H_1(e^{j\theta}) \right| \tag{5.35}$$

The idea is to spread the error out evenly or in the more general case, in som weighted fashion across the frequency response. This is much more difficult tha the window design problem, but algorithms have been developed to solve it b computer. The Remez algorithm, for example, allows the designer to selec weighting factors throughout the passband and stopbands (Roberts and Mullis 1987).

5.8 LAB: FIR FILTER DESIGN

The UW DigiScope software allows the user to experiment with several FIR design techniques, including direct specification of the pulse response, frequency sampling, window design, and placement of the zeros of the transfer equation. This lab studies some basic FIR filters such as the Hanning filter. It also uses specification of the pulse response and placement of the zeros of the transfer equation as design tools. Refer to Appendix D for information about using UW DigiScope.

5.8.1 Smoothing filters

Generate an ECG file with some random noise with the `(G)enwave` function, then run the Hanning filter by selecting `(F)ilters`, then `(L)oad filter`, then choosing `hanning.fil`. This function loads and immediately executes the filter function. By choosing `(R)un filter`, you can see the effect of cascading another Hanning filter with the first. The filter always operates on the data in the bottom channel.

The filters `poly2.fil`, `poly3.fil`, and `poly4.fil` are the least-squares smoothing filters with $L = 2$, 3, and 4 respectively. To test each of these three filters on the ECG data, first move the original signal from the top channel to the bottom channel with the `(C)opy data` command. Then use `(L)oad filter` to load and run the filters on ECG data. Observe the frequency and phase responses of each filter. How does changing the sampling rate of the ECG data affect the performance of the filters? Load `poly4.fil` and measure the time difference between the peak of the QRS complex in the unfiltered data and in the filtered data using the `(M)easure` command. You will need to change channels using `(A)ctive channel` to make the measurements. How do you account for the delay?

5.8.2 Derivatives

Experiment with the various derivative filters `deriv2.fil`, `deriv3.fil`, and `deriv5.fil` (which are the two-point, three-point central limit, and least-squares with $L = 2$ derivative filters respectively) on ECG and square wave data. Which filter is least suited for use with noisy signals?

5.8.3 Pulse Response

Create a filter that calculates the second derivative by selecting `(F)ilters`, then `(D)esign`, then `(F)IR`. Choose `(P)ulse resp`, and specify a filter length equal to 5. Enter the appropriate coefficients for the transfer equation and observe the frequency response. After saving the filter, return to the main menu, create a triangular wave with the `(G)enwave` function, and then use `(R)un filter` to observe the effects of the filter.

Create a linear phase filter with odd length (even order) and odd symmetry by satisfying Eq. (5.4). Observe its phase response.

5.8.4 Zero placement

The user may arbitrarily place zeros around the unit circle. The program automatically creates complex-conjugate pairs for values off the real axis. It then calculates the frequency, phase, and unit impulse responses for the filter.

Place zeros at $z = -0.5 \pm j0.866$ using the `(Z)ero place` function for a second-order filter, and comment on the frequency response. Run this filter on ECG data with and without 60-Hz noise sampled at 180 samples/s, and summarize the results.

5.8.5 Unit pulse sequence

There is a data file in the `STDLIB` called `ups.dat`. It consists of a single pulse. Reading this file and running an FIR filter will generate the unit impulse output sequence of the filter. Executing `(P)wr Spect` on this resulting output signal will give the magnitude response of the filter. In this way, the frequency response of filter can be measured more accurately than by the graph generated in the filter design utility. Use this method to find the 3-dB frequency of the Hanning filter (i.e., file `hanning.fil`).

5.8.6 Frequency sampling

Study the effect of increasing the width of the transition band by comparing the frequency response FIR filters of length 13 designed using frequency sampling by entering the following sets of values for the responses around the unit circle: {0, 0, –40, –40, –40, –40, –40} and {0, 0, –6, –40, –40, –40, –40}. Optimize the transition value (the third term in the sequence) for minimum stopband ripple.

Design a 60-Hz notch filter with length 21 by using frequency sampling. Run the filter on ECG data with and without 60-Hz noise. Does the filter perform better than `notch60.fil`? Optimize the filter for minimal gain at 60 Hz and minimal attenuation of other frequencies.

5.8.7 Window design

Compare the frequency response of low-pass filters designed with the various windows of the `(W)indow` function. Design the filters to have a cutoff frequency at 25 Hz with a sampling rate of 200 Hz. Make a table comparing main lobe width versus sideband ripple for the various windows. Measure the main lobe width at the first minimum in the frequency response.

5.8.8 Linear versus piecewise-linear phase response

Compare the phase response of the Hanning filter with that of the least-squares derivative filter with $L = 2$ (`deriv5.fil`). Why is the phase response of the least-squares filter not true linear? What causes the discontinuity? See Chapter 7 for additional information.

5.9 REFERENCES

Antoniou, A. 1979. *Digital Filters: Analysis and Design*. New York: McGraw-Hill.
Bogner, R. E. and Constantinides, A. G. 1985. *Introduction to Digital Filtering*, New York: John Wiley and Sons.
Friesen, G. M. et al. 1990. A comparison of the noise sensitivity of nine QRS detection algorithms, *IEEE Trans. Biomed. Eng.*, BME-37(1): 85–98.
Hamming, R. W. 1977. *Digital Filters*. Englewood Cliffs, NJ: Prentice Hall.
Rabiner, L. R. and Rader, C. M. 1972. *Digital Signal Processing* New York: IEEE Press.
Roberts, R. A. and Mullis, R. T. 1987. *Digital Signal Processing*. Reading, MA: Addison-Wesley.

5.10 STUDY QUESTIONS

5.1 What are the main differences between FIR and IIR filters?
5.2 What is the difference between direct convolution and "fast convolution"?
5.3 Why are finite-length register effects less significant in FIR filters than in IIR filters?
5.4 Compute and sketch the frequency response of a cascade of two Hanning filters. Does the cascade have linear phase?
5.5 Derive the phase response for an FIR filter with zeros located at $r\angle\pm\theta$ and $r^{-1}\angle\pm\theta$. Comment.
5.6 What are the trade-offs to consider when choosing the order of a least-squares polynomial smoothing filter?
5.7 Complete the derivation of coefficient values for the parabolic smoothing filter for $L = 2$.
5.8 Give some of the disadvantages of the simple 60-Hz notch filter described in section 5.3.1.
5.9 What are the main differences between the two-point difference and three-point central difference algorithms for approximating the derivative?
5.10 What are the three steps to designing a filter using the window method?
5.11 Explain the relationship between main-lobe width and sideband ripple for various windows.
5.12 How do the free parameters in the transition band of a frequency sampling design affect the performance of the filter?
5.13 What is the fundamental difference between minimax design and the window design method?
5.14 Design an FIR filter of length 15 with passband gain of 0 dB from 0 to 50 Hz and stopband attenuation of 40 dB from 100 to 200 Hz using the window design method. Compare the Hanning and rectangular windows. (Use a sampling rate of 400 sps.)
5.15 Repeat question 5.14 using frequency sampling.

5.16 The transfer function of the Hanning filter is

$$H_1(z) = \frac{1 + 2z^{-1} + z^{-2}}{4}$$

(a) What is its gain *at dc*? (b) Three successive stages of this filter are cascaded together to give a new transfer function [that is, $H(z) = H_1(z) \times H_1(z) \times H_1(z)$]. What is the overall gain of this filter *at dc*? (c) A high-pass filter is designed by subtracting the output of the Hanning filter from an all-pass filter with zero phase delay. How many zeros does the resulting filter have? Where are they located?

5.17 Two filters are cascaded. The first has the transfer function: $H_1(z) = 1 + 2z^{-1} - 3z^{-2}$. The second has the transfer function: $H_2(z) = 1 - 2z^{-1}$. A *unit impulse* is applied to the input of the cascaded filters. (a) What is the output sequence? (b) What is the magnitude of the amplitude response of the cascaded filter 1. at dc? 2. at 1/2 the foldover frequency? 3. at the foldover frequency?

5.18 Two filters are cascaded. The first has the transfer function: $H_1(z) = 1 + 2z^{-1} + z^{-2}$. The second has the transfer function: $H_2(z) = 1 - z^{-1}$. (a) A *unit impulse* is applied to the input of the cascaded filters. What is the output sequence? (b) What is the magnitude of the amplitude response of this cascaded filter at dc?

nfinite Impulse Response Filters

en Zhou

n this chapter we introduce the analysis and design of infinite impulse response IR) digital filters that have the potential of sharp rolloffs (Tompkins and Webster, 981). We show a simple one-pole example illustrating the relationship between ole position and filter stability. Then we show how to design two-pole filters with w-pass, bandpass, high-pass, and band-reject characteristics. We also present alorithms to implement integrators that are all IIR filters. Finally, we provide a labratory exercise that uses IIR filters for ECG analysis.

.1 GENERIC EQUATIONS OF IIR FILTERS

he generic format of transfer function of IIR filters is expressed as the ratio of two olynomials:

$$H(z) = \frac{\sum_{i=0}^{n} a_i z^{-i}}{1 - \sum_{i=1}^{n} b_i z^{-i}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{1 - b_1 z^{-1} - b_2 z^{-2} - \ldots - b_n z^{-n}} = \frac{Y(z)}{X(z)} \tag{6.1}$$

earranging the terms gives

$$Y(z) = b_1 Y(z) z^{-1} + \ldots + b_n Y(z) z^{-n} + a_0 X(z) + a_1 X(z) z^{-1} + \ldots + a_n X(z) z^{-n} \tag{6.2}$$

he $Y(z)$ terms on the right side of this equation are delayed feedback terms. Figure .1 shows these feedback terms as recursive loops; hence, these types of filters are so called recursive filters.

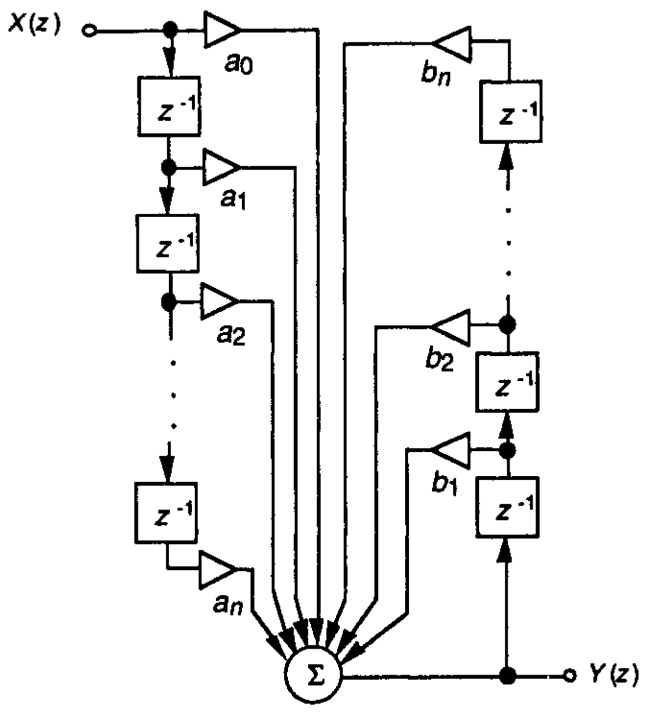


Figure 6.1 The output of an IIR filter is delayed and fed back.

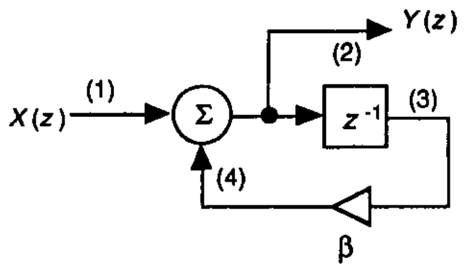
6.2 SIMPLE ONE-POLE EXAMPLE

Let us consider the simple filter of Figure 6.2(a). We find the transfer function b applying a unit impulse sequence to the input $X(z)$. Figure 6.2(b) shows th sequences at various points in the filter for a feedback coefficient β equal to 1/ Sequence (1) defines the unit impulse. From output sequence (2), we write th transfer function

$$H(z) = 1 + \frac{1}{2} z^{-1} + \frac{1}{4} z^{-2} + \frac{1}{8} z^{-3} + \ldots \tag{6.3}$$

Using the binomial theorem, we write the infinite sum as a ratio of polynomials

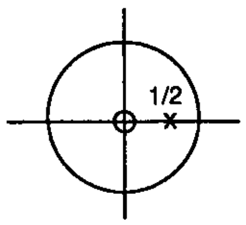
$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \frac{1}{2} z^{-1}} \tag{6.4}$$



(a)

(1) (1, 0, 0, 0, 0, ...)
(2) (1, 1/2, 1/4, 1/8, 1/16 ...)
(3) (0, 1, 1/2, 1/4, 1/8, ...)
(4) (0, 1/2, 1/4, 1/8, 1/16, ...)

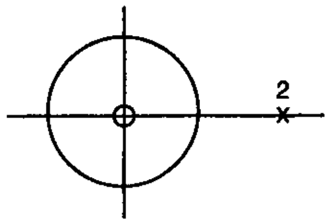
(b)



(c)

(1) (1, 0, 0, 0, 0, ...)
(2) (1, 2, 4, 8, 16 ...)
(3) (0, 1, 2, 4, 8, ...)
(4) (0, 2, 4, 8, 16, ...)

(d)



(e)

Figure 6.2 Simple one-pole recursive filters. (a) Block diagram. (b) Response to a unit pulse for $\beta = 1/2$. (c) Pole-zero plot for $\beta = 1/2$. (d) Response to a unit pulse for $\beta = 2$. (e) Pole-zero plot for $\beta = 2$.

We then rearrange this equation and write the output as a function of the feedback term and the input

$$Y(z) = \frac{1}{2} Y(z)\, z^{-1} + X(z) \tag{6.5}$$

Recognizing that $x(nT)$ and $y(nT)$ are points in the input and output sequences associated with the current sample time, they are analogous to the undelayed z-domain variables, $X(z)$ and $Y(z)$ respectively. Similarly $y(nT - T)$, the output value one sample point in the past, is analogous to the output z-domain output variable delayed by one sample point, or $Y(z)z^{-1}$. We can then write the difference equation by inspection

$$y(nT) = \frac{1}{2}\, y(nT - T) + x(nT) \tag{6.6}$$

Unlike the FIR case where the current output $y(nT)$ is dependent only on current and past values of x, this IIR filter requires not only the current value of the input $x(nT)$ but also the previous value of the output itself $y(nT - T)$. Since past history of the output influences the next output value, which in turn influences the next successive output value, a transient requires a large number or sample points before it disappears from an output signal. As we have mentioned, this does not occur in an FIR filter because it has no feedback.

In order to find the poles and zeros, we first multiply the transfer function of Eq. (6.4) by z/z in order to make all the exponents of z positive.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \frac{1}{2} z^{-1}} \times \frac{z}{z} = \frac{z}{z - \frac{1}{2}} \tag{6.7}$$

By equating the numerator to zero, we find that this filter has a single zero at $z = 0$. Setting the denominator equal to zero in order to locate the poles gives

$$z - \frac{1}{2} = 0$$

Thus, there is a single pole at $z = 1/2$.

The pole-zero plot for this single-pole filter is shown in Figure 6.2(c). In order to find the amplitude and phase responses, we substitute $z = e^{j\omega T}$ into the transfer function

$$H(\omega T) = \frac{1}{1 - \frac{1}{2} e^{-j\omega T}} = \frac{1}{\left[1 - \frac{1}{2}\cos(\omega T)\right] + j\left[\frac{1}{2}\sin(\omega T)\right]} \tag{6.8}$$

Evaluating this function for the amplitude and phase responses, we find that this is a low-pass filter as illustrated in Figure 6.3.

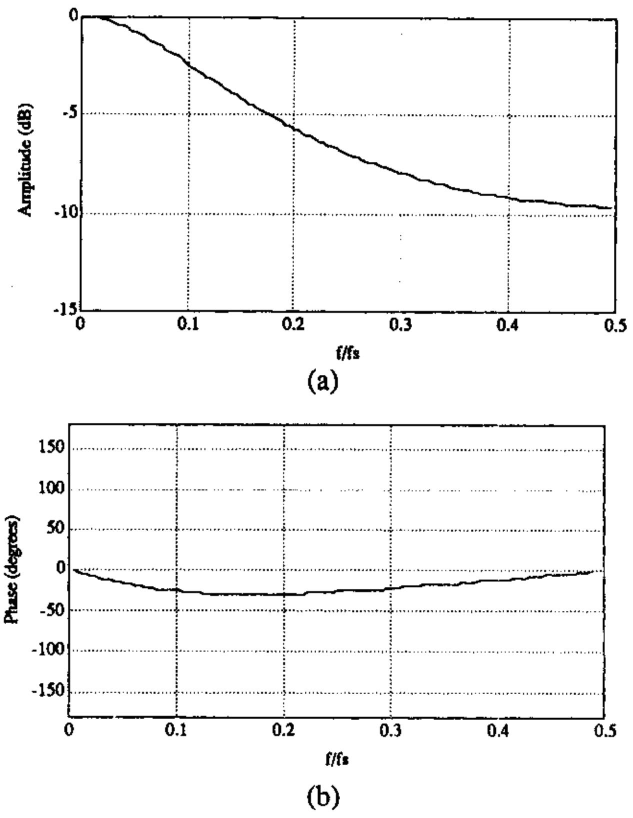


Figure 6.3 Simple one-pole recursive filter of Figure 6.2 with pole located at $z = 1/2$. (a) Amplitude response. (b) Phase response.

If we replace the multiplier constant β of 1/2 in this filter by 2, the output sequence in response to a unit impulse is (1, 2, 4, 8, 16, ...) as shown in Figure 6.2(d). The filter is unstable—the output increases with each successive sample. The response to a unit pulse input not only does not disappear with time, it increases by a factor of 2 each T s. We desire a unit pulse response that decays toward zero. Calculating the location of the pole with the multiplier of 2, we find that the pole is at $z = 2$, as shown in Figure 6.2(e). It is outside the unit circle, and the filter is unstable, as expected.

If we replace the multiplier constant of 1/2 in this filter by 1, the pole is directly on the unit circle at $z = 1$, and the output response to a unit impulse applied to the input is (1, 1, 1, 1, 1, ...). This filter is a special IIR filter, a rectangular integrator (see section 6.3).

6.3 INTEGRATORS

The general form of the integral is

$$A = \int_{t_1}^{t_2} f(t)dt \tag{6.9}$$

where A is the area under the function between the limits t_1 to t_2. The digital implementation for the determination of integral solutions is done by approximating the function using curve fitting techniques at a finite number of points, where

$$A = \sum_{n=t_1}^{t_2} f(n)\Delta t \tag{6.10}$$

The Laplace transform of an integrator is

$$H(s) = \frac{1}{s} \tag{6.11}$$

The amplitude and phase responses are found by substituting into the transfer function the relation, $s = j\omega$, giving

$$H(j\omega) = \frac{1}{j\omega} \tag{6.12}$$

Thus, the ideal amplitude response is inversely proportional to frequency

$$|H(j\omega)| = \left|\frac{1}{\omega}\right| \tag{6.13}$$

and the phase response is

$$\angle H(j\omega) = \tan^{-1}\left(\frac{-1/\omega}{0}\right) = -\frac{\pi}{2} \tag{6.14}$$

A number of numerical techniques exist for digital integration. Unlike the analog integrator, a digital integrator has no drift problems because the process is a computer program, which is not influenced in performance by residual charge on capacitors. We discuss here three popular digital integration techniques—rectangular summation, trapezoidal summation, and Simpson's rule.

6.3.1 Rectangular integration

This algorithm performs the simplest integration. It approximates the integral as a sum of rectangular areas. Figure 6.4(a) shows that each rectangle has a base equal in length to one sample period T and in height to the value of the most recently sampled input $x(nT-T)$. The area of each rectangle is $Tx(nT-T)$. The difference equation is

$$y(nT) = y(nT-T) + T\,x(nT) \tag{6.15}$$

where $y(nT-T)$ represents the sum of all the rectangular areas prior to adding the most recent one. The error in this approximation is the difference between the area of the rectangle and actual signal represented by the sampled data. The z transform of Eq. (6.15) is

$$Y(z) = Y(z)\,z^{-1} + T\,X(z) \tag{6.16}$$

and

$$H(z) = \frac{Y(z)}{X(z)} = T\left(\frac{1}{1-z^{-1}}\right) \tag{6.17}$$

Figure 6.4(a) shows that this transfer function has a pole at $z = 1$ and a zero at $z = 0$. The amplitude and phase responses are

$$|H(\omega T)| = \left|\frac{T}{2\sin(\omega T/2)}\right| \tag{6.18}$$

and

$$\angle H(\omega T) = \frac{\omega T}{2} - \frac{\pi}{2} \tag{6.19}$$

Figure 6.5 shows the amplitude response. We can reduce the error of this filter by increasing the sampling rate significantly higher than the highest frequency present in the signal that we are integrating. This has the effect of making the width of each rectangle small compared to the rate of change of the signal, so that the area of each rectangle better approximates the input data. Increasing the sampling rate also corresponds to using only the portion of amplitude response at the lower frequencies, where the rectangular response better approximates the ideal response. Unfortunately, higher than necessary sampling rates increase computation time and waste memory space by giving us more sampled data than are necessary to characterize a signal. Therefore, we select other digital integrators for problems where higher performance is desirable.

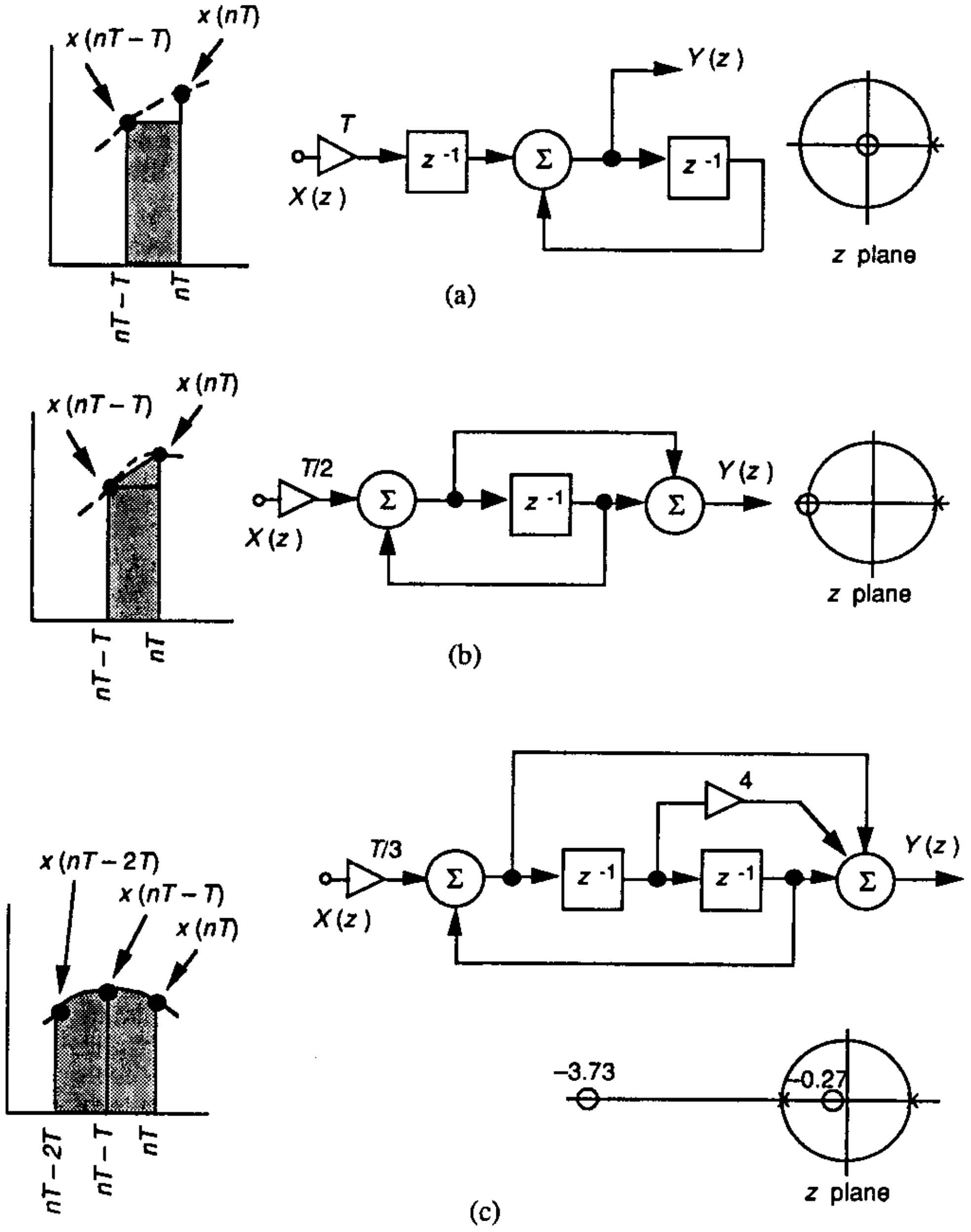


Figure 6.4 Integration. (a) Rectangular. (b) Trapezoidal. (c) Simpson's rule.

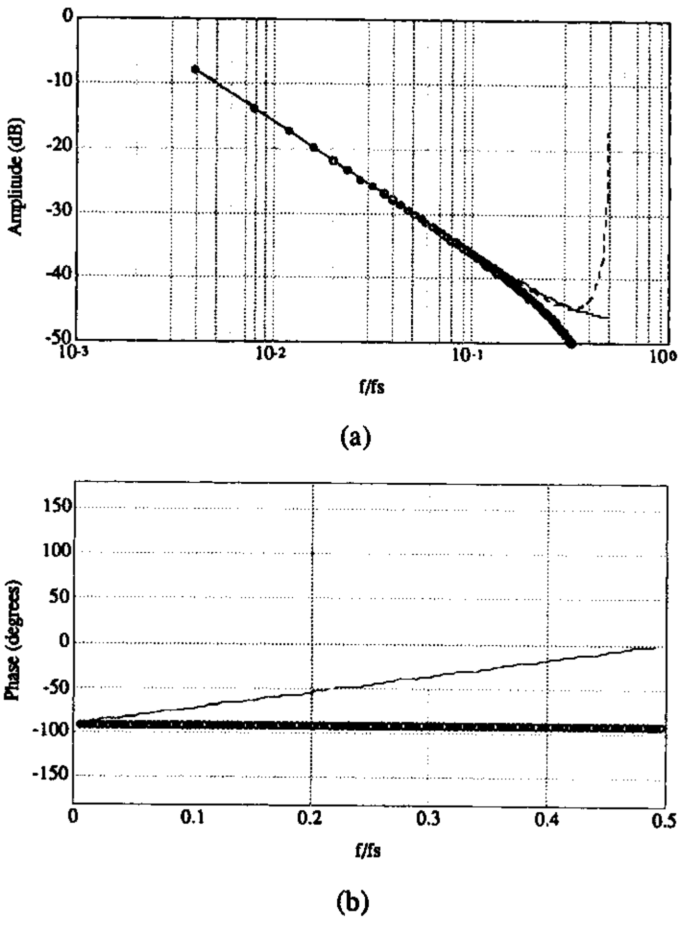


Figure 6.5 Digital integrators. The sampling period T is set equal to 1. (a) Amplitude responses. (b) Phase responses. Solid line: rectangular. Circles: trapezoidal. Dashed line: Simpson's rule.

6.3.2 Trapezoidal integration

This filter improves on rectangular integration by adding a triangular element to the rectangular approximation, as shown in Figure 6.4(b). The difference equation is the same as for rectangular integration except that we add the triangular element.

$$y(nT) = y(nT-T) + Tx(nT-T) + \frac{T}{2}[x(nT) - x(nT-T)]$$
$$= y(nT-T) + \frac{T}{2}[x(nT) + x(nT-T)] \qquad (6.20)$$

This corresponds to the transfer function

$$H(z) = \frac{T}{2}\left(\frac{1+z^{-1}}{1-z^{-1}}\right) \qquad (6.21)$$

For the exact amplitude and phase responses, we use the analytical approach. Since we desire the frequency response, we evaluate the transfer function on the unit circle by simply substituting $e^{j\omega T}$ for every occurrence of z. For the recursive filter

$$H(z) = \frac{1+z^{-1}}{1-z^{-1}} \qquad (6.22)$$

we obtain

$$H(\omega T) = \frac{1+e^{-j\omega T}}{1-e^{-j\omega T}} = \frac{e^{-j\frac{\omega T}{2}}\left(e^{j\frac{\omega T}{2}} + e^{-j\frac{\omega T}{2}}\right)}{e^{-j\frac{\omega T}{2}}\left(e^{j\frac{\omega T}{2}} - e^{-j\frac{\omega T}{2}}\right)} = \frac{2\cos\left(\frac{\omega T}{2}\right)}{2j\sin\left(\frac{\omega T}{2}\right)} = -j\cot\left(\frac{\omega T}{2}\right)$$

The amplitude response is the magnitude of $H(\omega T)$

$$|H(\omega T)| = \left|\cot\left(\frac{\omega T}{2}\right)\right| \qquad (6.23)$$

and the phase response is

$$\angle H(\omega T) = -\frac{\pi}{2} \qquad (6.24)$$

The block diagram and pole-zero plot are shown in Figure 6.4(b). Like the rectangular approach, this function still has a pole at $z = 1$, but the zero is moved to $z = -1$, the location of the folding frequency, giving us zero amplitude response at f_0. The amplitude and phase response are

$$|H(\omega T)| = \left|\frac{T}{2}\cot\left(\frac{\omega T}{2}\right)\right| \qquad (6.25)$$

and

$$\angle H(\omega T) = -\frac{\pi}{2} \tag{6.26}$$

The amplitude response approximates that of the ideal integrator much better than the rectangular filter and the phase response is exactly equal to that of the ideal response. The trapezoidal technique provides a very simple, effective recursive integrator.

Figure 6.6 shows a program that performs trapezoidal integration by direct implementation of the difference equation, Eq. (6.20).

```
/*****************************************************************
 *  Turbo C source code implementing a trapezoidal integrator
 *
 *  Assume sampling period is 2 ms, the difference equation is:
 *
 *      y(nT) = y(nT - T) + 0.001x(nT) + 0.001x(nT - T)
 *
 *****************************************************************/
float Trapezoid(signal)
float signal;                    /* x[nT] */
{
  static float  x[2], y[2];
  int count;

  x[1]=x[0];                     /* x[1]=x(nT-T) */
  x[0]=signal;                   /* x[0]=x(nT) */

  y[0]=y[1]+0.001*x[0]+0.001*x[1];/* difference equation */

  y[1]=y[0];                     /* y[1]=y(nT - T) */

  return y[0];
}
```

Figure 6.6 A trapezoidal integrator written in the C language.

6.3.3 Simpson's rule integration

Simpson's rule is the most widely used numerical integration algorithm. Figure 6.4(c) shows that this approach approximates the signal corresponding to three input sequence points by a polynomial fit. The incremental area added for each new input point is the area under this polynomial. The difference equation is

$$y(nT) = y(nT - 2T) + \frac{T}{3}[x(nT) + 4x(nT - T) + x(nT - 2T)] \tag{6.27}$$

The transfer function is

$$H(z) = \frac{T}{3}\left(\frac{1 + 4z^{-1} + z^{-2}}{1 - z^{-2}}\right) \tag{6.28}$$

Figure 6.4(c) shows the block diagram and pole-zero plot for this filter. The amplitude and phase responses are

$$|H(\omega T)| = \left|\frac{T}{2}\left(\frac{2 + \cos\omega T}{\sin\omega T}\right)\right| \tag{6.29}$$

and

$$\angle H(\omega T) = -\frac{\pi}{2} \tag{6.30}$$

Figure 6.5 shows the amplitude response of the Simpson's rule integrator. Like the trapezoidal technique, the phase response is the ideal $-\pi/2$. Simpson's rule approximates the ideal integrator for frequencies less than about $f_s/4$ better than the other techniques. However, it amplifies high-frequency noise near the foldover frequency. Therefore, it is a good approximation to the integral but is dangerous to use in the presence of noise. Integration of noisy signals can be accomplished better with trapezoidal integration.

6.4 DESIGN METHODS FOR TWO-POLE FILTERS

IIR filters have the potential for sharp rolloffs. Unlike the design of an FIR filter, which frequently is based on approximating an input sequence numerically, IIR filter design frequently starts with an analog filter that we would like to approximate. Their transfer functions can be represented by an infinite sum of terms or ratio of polynomials. Since they use feedback, they may be unstable if improperly designed. Also, they typically do not have linear phase response.

6.4.1 Selection method for r and θ

The general design equation has a standard recursive form for the four types of filters: low-pass, bandpass, high-pass, and band-reject:

$$H(z) = \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} + b_2 z^{-2}} \tag{6.31}$$

where the zero locations are

$$z = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2}}{2} \tag{6.32}$$

and the pole locations are

$$z = \frac{b_1 \pm \sqrt{b_1^2 - 4b_2}}{2} \tag{6.33}$$

where

$$b_1 = 2r\cos\theta \qquad \text{and} \qquad b_2 = r^2 \tag{6.34}$$

Also

$$\theta = 2\pi \left(\frac{f_c}{f_s}\right) \tag{6.35}$$

Figure 6.7 shows the structure of the two-pole filter. Assigning values to the numerator coefficients, a_1 and a_2, as shown in Figure 6.8 establishes the placement of the two zeros of the filter and defines the filter type. Values of b_1 and b_2 are determined by placing the poles at specific locations within the unit circle.

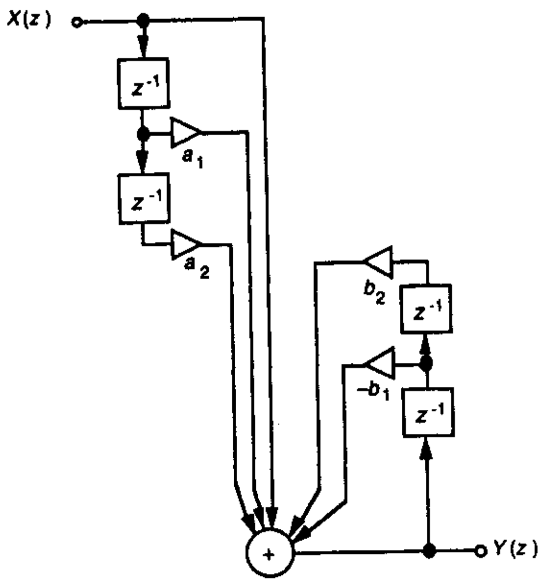


Figure 6.7 Block diagram showing two-pole filter structure.

We start the design by selecting the sampling frequency f_S, which must be at least twice the highest frequency contained in the input signal. Next we choose the critical frequency f_C. This is the cutoff frequency for low- and high-pass filters, the resonant frequency for a bandpass filter, and the notch frequency for a band-reject filter. These two choices establish θ, the angular location of the poles.

We then select r, the distance of the poles from the origin. This is the damping factor, given by

$$r = e^{-aT} \tag{6.36}$$

We know from the amplitude response that underdamping occurs as the poles approach the unit circle (i.e., $r \rightarrow 1$ or $a \rightarrow 0$) and overdamping results for poles near the origin (i.e., $r \rightarrow 0$ or $a \rightarrow \infty$). Moving the poles by increasing r or θ causes underdamping, and decreasing either variable causes overdamping (Soderstrand, 1972). As we have done in other designs, we find the frequency response by substituting $e^{j\omega T}$ for z in the final transfer function.

	a_1	a_2
Low-pass	2	1
Bandpass	0	–1
High-pass	–2	1
Band-reject	2cosθ	1

Figure 6.8 Table of numerator coefficients for the two-pole filter implementations of Eq. (6.31).

We can write the difference equation for the two-pole filter by first rewriting Eq. (6.31):

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} + b_2 z^{-2}} \tag{6.37}$$

Then rearranging terms to find $Y(z)$, we get

$$Y(z) = b_1 Y(z)z^{-1} - b_2 Y(z)z^{-2} + X(z) + a_1 X(z)z^{-1} + a_2 X(z)z^{-2} \tag{6.38}$$

We can now directly write the difference equation using analogous discrete variable terms

$$\begin{aligned} y(nT) &= b_1\, y(nT - T) - b_2\, y(nT - 2T) + x(nT) \\ &\quad + a_1\, x(nT - T) + a_2\, x(nT - 2T) \end{aligned} \tag{6.39}$$

Figure 6.9 shows the pole-zero plots and corresponding rubber membrane representations for each of the four filter types.

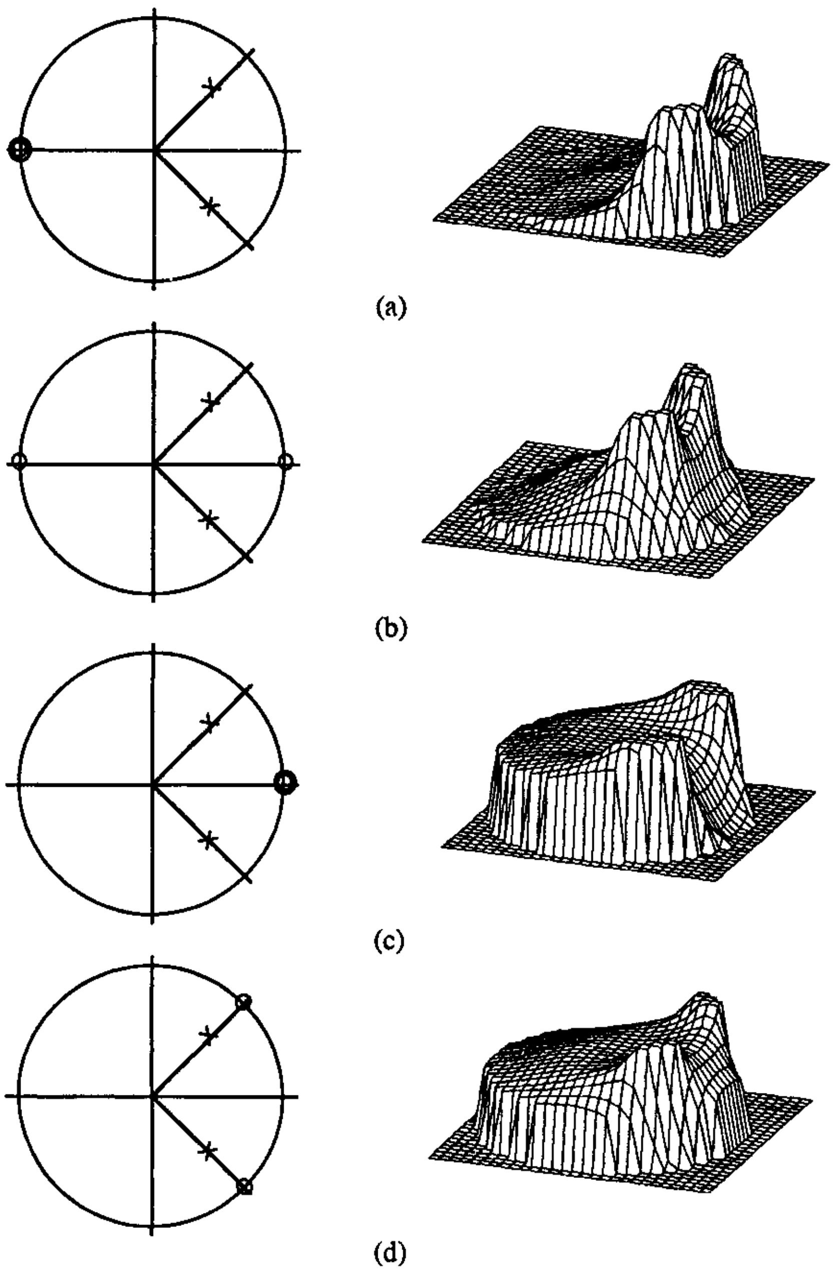


Figure 6.9 Pole-zero plots and rubber membrane views of elementary two-pole recursive digital filters. (a) Low-pass. (b) Bandpass. (c) High-pass. (d) Band-reject (notch).

Let us now design a filter and write a C-language program to implement it. First we choose a low-pass filter. This establishes the locations of the two zeros and establish the values for $a_1 = 2$ and $a_2 = 1$ (see Figure 6.8). Then we decide on the locations of the poles. For this design, we choose locations at $r = 0.5$ and $\theta = \pm 45°$. This establishes the values for $b_1 = 0.707$ and $b_2 = 0.25$ [see Eq. (6.34)].

Software implementation of this digital filter is now fairly straightforward. We write a filter subroutine directly from the difference equation. Figure 6.10 shows how this low-pass filter is executed in five steps:

1. Pass in the current input signal.
2. Shift array elements to get delays of past input signal and store them.
3. Accumulate all past and/or current input and past output terms with multiplying coefficients.
4. Shift array elements to get delays of past output signal and store them.
5. Return with output signal.

```
/****************************************************************
 *  Turbo C source code for implementing an IIR low-pass filter
 *
 *  Assume we use the r and q method where r = 0.5, q = 45°.
 *  The difference equation is:
 *
 *  y(nT) = 0.707y(nT-T)-0.25y(nT-2T)+x(nT)+2x(nT-T)+x(nT-2T)
 *  y(nT) = 0.707*y[1] - 0.25*y[2] + x[0] + 2*x[1] + x[2]
 *
 ****************************************************************/
float LowPass(signal)
float signal;                    /* x[nT] */
{
  static float  x[3], y[3];
  int count;

  for ( count=2; count>0; count-- )
    x[count]=x[count-1];              /* shift for x term delays */

  x[0]=signal;                        /* x[0] = x[nT] */
                                      /* difference equation */

  y[0]=0.707*y[1]-0.25*y[2]+x[0]+2.0*x[1]+x[2];

  for( count = 2; count > 0; count--)
    y[count]=y[count-1];              /* shift for y term delays */

  return y[0];
}
```

Figure 6.10 An IIR low-pass filter written in the C language.

The same algorithm can be used for high-pass, bandpass, and band-reject filters as well as for integrators. The only disadvantage of the software implementation is its limited speed. However, this program will execute in real time on a modern PC with a math coprocessor for biomedical signals like the ECG.

These two-pole filters have operational characteristics similar to second-order analog filters. The rolloff is slow, but we can cascade several identical sections to improve it. Of course, we may have time constraints in a real-time system which limit the number of sections that can be cascaded in software implementations. In these cases, we design higher-order filters.

6.4.2 Bilinear transformation method

We can design a recursive filter that functions approximately the same way as a model analog filter. We start with the s-plane transfer function of the filter that we desire to mimic. We accomplish the basic digital design by replacing all the occurrences of the variable s by the approximation

$$s \approx \frac{2}{T}\left[\frac{1-z^{-1}}{1+z^{-1}}\right] \tag{6.40}$$

This substitution does a nonlinear translation of the points in the s plane to the z plane. This warping of the frequency axis is defined by

$$\omega' = \frac{2}{T}\left[\tan\left(\frac{\omega T}{2}\right)\right] \tag{6.41}$$

where ω' is the analog domain frequency corresponding to the digital domain frequency ω. To do a bilinear transform, we first prewarp the frequency axis by substituting the relation for ω' for all the critical frequencies in the Laplace transform of the filter. We then replace s in the transform by its z-plane equivalent. Suppose that we have the transfer function

$$H(s) = \frac{\omega_c'}{s^2 + \omega_c'^2} \tag{6.42}$$

We substitute for ω_c' using Eq. (6.41) and for s with Eq. (6.40), to obtain

$$H(z) = \frac{\frac{2}{T}\left[\tan\left(\frac{\omega T}{2}\right)\right]}{\left\{\frac{2}{T}\left[\frac{1-z^{-1}}{1+z^{-1}}\right]\right\}^2 + \left\{\frac{2}{T}\left[\tan\left(\frac{\omega T}{2}\right)\right]\right\}^2} \tag{6.43}$$

This *z* transform is the description of a digital filter that performs approximately the same as the model analog filter. We can design higher-order filters with this technique.

6.4.3 Transform tables method

We can design digital filters to approximate analog filters of any order with filter tables such as those in Stearns (1975). These tables give the Laplace and *z*-transform equivalents for corresponding continuous and discrete-time functions. To illustrate the design procedure, let us consider the second-order filter of Figure 6.11(a). The analog transfer function of this filter is

$$H(s) = \frac{A}{LC}\left[\frac{1}{s^2 + (R/L)s + 1/LC}\right] \tag{6.44}$$

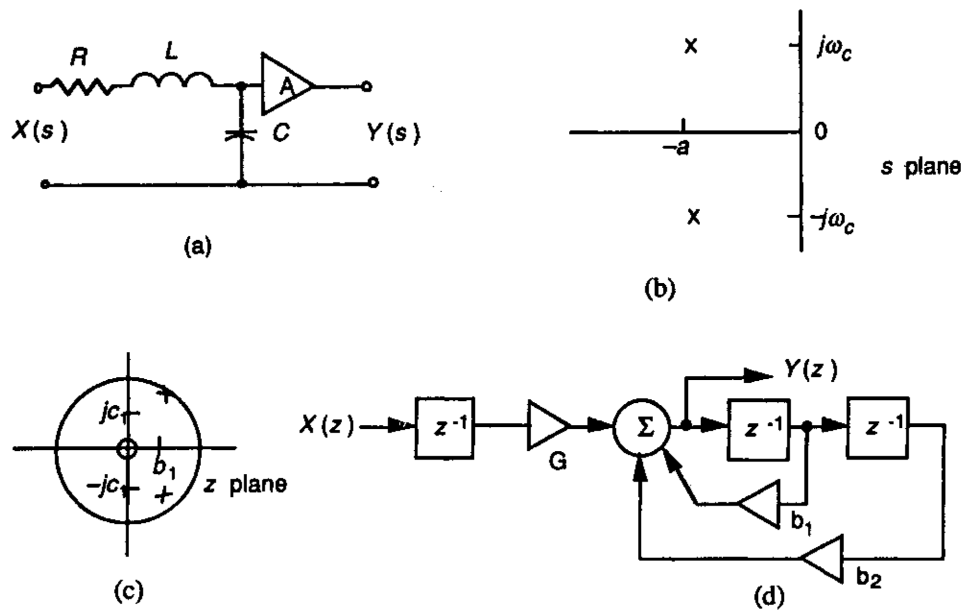


Figure 6.11 Second-order filter. (a) Analog filter circuit. (b) Transfer function pole-zero plot for the analog filter. (c) Pole-zero plot for digital version of the second-order filter. (d) Block diagram of the digital filter.

Solving for the poles, we obtain

$$s = -a \pm j\omega_c \tag{6.45}$$

where

$$a = \frac{R}{2L} \tag{6.46}$$

$$\omega_c = \left[\frac{1}{LC} - \frac{R^2}{4L^2}\right]^{1/2} \tag{6.47}$$

We can rewrite the transfer function as

$$H(s) = \frac{A}{LC}\left[\frac{1}{(s+a)^2 + \omega_c^2}\right] \tag{6.48}$$

Figure 6.11(b) shows the s-plane pole-zero plot. This s transform represents the continuous time function $e^{-at}\sin\omega_c t$. The z transform for the corresponding discrete-time function $e^{-naT}\sin n\omega_c T$ is in the form

$$H(z) = \frac{Gz^{-1}}{1 - b_1 z^{-1} - b_2 z^{-2}} \tag{6.49}$$

where

$$b_1 = 2e^{-aT}\cos\omega_c T \tag{6.50}$$

and

$$b_2 = -e^{-2aT} \tag{6.51}$$

Also

$$G = \frac{A}{\omega_c LC} e^{-aT}\sin\omega_c T \tag{6.52}$$

Variables a, ω_c, A, L, and C come from the analog filter design. This transfer function has one zero at $z = 0$ and two poles at

$$z = b_1 \pm j(b_1^2 + 4b_2)^{1/2} = b_1 \pm jc_1 \tag{6.53}$$

Figure 6.11(c) shows the z-plane pole-zero plot. We can find the block diagram by substituting the ratio $Y(z)/X(z)$ for $H(z)$ and collecting terms.

$$Y(z) = GX(z)z^{-1} + b_1Y(z)z^{-1} + b_2Y(z)z^{-2} \tag{6.54}$$

The difference equation is

$$y(nT) = Gx(nT - T) + b_1\, y(nT - T) + b_2\, y(nT - 2T) \tag{6.55}$$

From this difference equation we can directly write a program to implement the filter. We can also construct the block diagram as shown in Figure 6.11(d).

This transform-table design procedure provides a technique for quickly designing digital filters that are close approximations to analog filter models. If we have the transfer function of an analog filter, we still usually must make a substantial effort to implement the filter with operational amplifiers and other components. However, once we have the z transform, we have the complete filter design specified and only need to write a straightforward program to implement the filter.

6.5 LAB: IIR DIGITAL FILTERS FOR ECG ANALYSIS

This lab provides experience with the use of IIR filters for processing ECGs. In order to design integrators and two-pole filters using UW DigiScope, select `(F)ilters` from the main menu, choose `(D)esign`, then `(I)IR`.

6.5.1 Integrators

This chapter reviewed three different integrators (rectangular, trapezoidal, and Simpson's rule). Which of these filters requires the most computation time? What do the unit-circle diagrams of these three filters have in common?

Run the rectangular integrator on an ECG signal (e.g., `ecg105.dat`). Explain the result. Design a preprocessing filter to solve any problem observed. Compare the output of the three integrators on appropriately processed ECG data with and without random noise. Which integrator is best to use for noisy signals?

6.5.2 Second-order recursive filters

Design three two-pole bandpass filters using $r = 0.7$, 0.9, and 0.95 for a critical frequency of 17 Hz (sampling rate of 200 sps). Measure the Q for the three filters. Q is defined as the ratio of critical frequency to the 3-dB bandwidth (difference between the 3-dB frequencies). Run the three filters on an ECG file and contrast the outputs. What trade-offs must be considered when selecting the value of r?

6.5.3 Transfer function (Generic)

The `(G)eneric` tool allows you to enter coefficients of a transfer function of the form

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}} \qquad \text{where } m <= n$$

Using the example from the transform tables method discussed in section 6.4.3 with $R = 10\ \Omega$, $L = 10$ mH, and $C = 1\ \mu$F, calculate and enter the coefficients of the resulting $H(z)$. Use a sampling rate of 200 sps. What is the Q of this filter?

6.5.4 Pole-zero placement

Create a 60-Hz notch FIR filter for a sampling rate of 180 sps by placing two zeros on the unit circle (i.e., $r = 1.0$) at angles of ±120° as shown in Figure 6.12(a). Create an IIR filter by adding poles inside the circle at angles of ±110° and ±130° at $r = 0.9$, as shown in Figure 6.12(b). Compare the amplitude response of this filter with that of the FIR filter. Measure the Q of both filters. Can you further increase the Q of the IIR filter? What happens to the phase response?

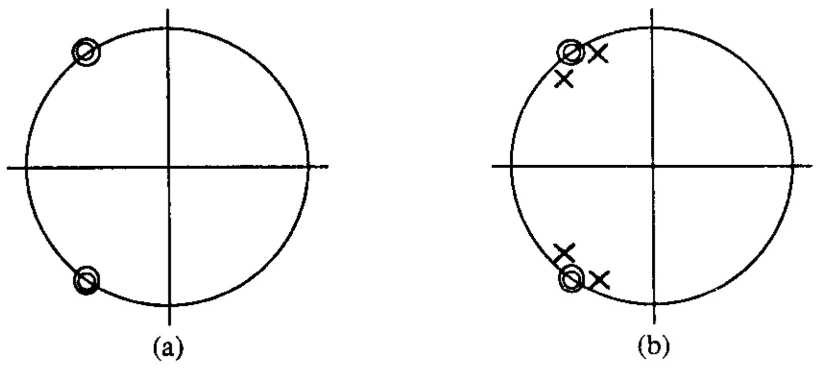


Figure 6.12 Notch filters. (a) FIR filter. (b) IIR filter.

6.6 REFERENCES

Soderstrand, M. A. 1972. On-line digital filtering using the PDP-8 or PDP-12. *Computers in the Neurophysiology Laboratory*, 1: 31-49. Maynard, MA: Digital Equipment Corporation.
Stearns, S. D. 1975. *Digital Signal Analysis*. Rochelle Park, NJ: Hayden.
Tompkins, W. J. and Webster, J. G. (eds.) 1981. *Design of Microcomputer-based Medical Instrumentation*. Englewood Cliffs, NJ: Prentice Hall.

6.7 STUDY QUESTIONS

6.1 Describe the characteristics of the generic transfer function of recursive filters.
6.2 What is the difference between IIR filters and recursive filters?
6.3 How do you derive the frequency response of a recursive filter?

6.4 How do you write the difference equation from the transfer function of a recursive filter?

6.5 Design a low-pass recursive filter using the r and θ method.

6.6 Design a high-pass filter using bilinear transformation method.

6.7 Design a filter using transform tables method.

6.8 A digital filter has a *unit impulse* output sequence {i.e., 1, 0, 0,...} when a *unit step* {i.e., 1, 1, 1,...} is applied at its input. What is the transfer function of the filter?

6.9 Write a rectangular integrator in C.

6.10 Using the difference equation from the result of question 6.5, run the filtering progran provided in section 6.4.

6.11 Draw the z-plane po'e-zero plot for a filter with the z transform:

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

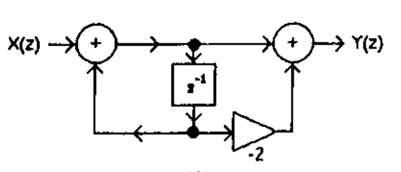
6.12 A filter has the following output sequence in response to a unit impulse: {–2 ,4, –8, 16, ...}. Write its z transform in closed form (i.e., as a ratio of polynomials). From the following list, indicate all the terms that describe this filter: recursive, nonrecursive, stable, unstable, FIR, IIR.

6.13 A digital filter has the following transfer function:

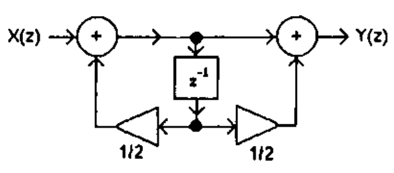
$$H(z) = \frac{1 - bz^{-1}}{1 + bz^{-1}}$$

(a) What traditional type of filter is this if b = (1) 0.8; (2) –0.8; (3) 1; (4) –2; (5) –1/2? (b) If $b = -1/2$, what is the filter gain? (c) If $b = 1/2$, what is the difference equation for $y(nT)$?

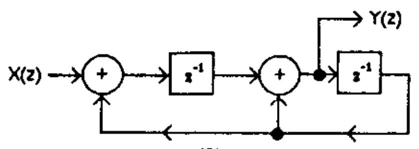
6.14 The block diagrams for four digital filters are shown below. Write their (a) transfer functions, (b) difference equations.



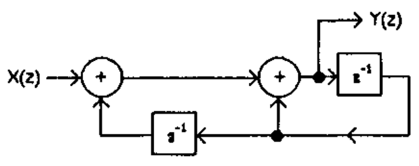
(1)



(2)



(3)



(4)

6.15 Write the transfer function and comment on the stability of a filter with the following output sequence in response to a unit impulse: (a) {3, –6, 12, –24, ...}. (b) {2, –1, 1, –1/2, 1/4, –1/8, ...}.

6.16 A digital filter has a difference equation: $y(nT) = y(nT - 2T) + x(nT - T)$. What is its output sequence in response to a unit impulse applied to its input?

6.17 A filter's difference equation is: $y(nT) = x(nT) + 3y(nT - T)$. What is its output sequence in response to a unit impulse?

6.18 The difference equation of a filter is: $y(nT) = x(nT) + x(nT - 2T) + y(nT - 2T)$. Where are its poles and zeros located?

6.19 The general equation for a two-pole digital filter is:

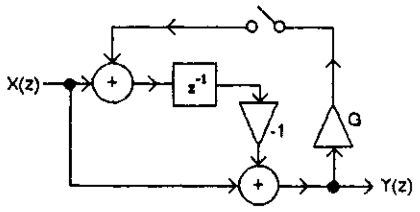
$$H(z) = \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} + b_2 z^{-2}}$$

where $\quad b_1 = 2r\cos(\theta) \quad$ and $\quad b_2 = r^2$

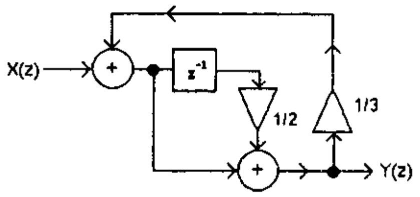
(a) What traditional type of filter is this if (1) $a_1 = 2$, $a_2 = 1$, $-2 < b_1 < 2$, $0 < b_2 < 1$, (2) $a_1 = -2\cos(\theta)$, $a_2 = 1$, $r = 1$, $\theta = 60°$? (b) What is the difference equation for $y(nT)$ if $a_1 = 0$, $a_2 = -1$, $r = 1/2$, $\theta = 60°$?

6.20 A digital filter has two zeros located at $z = 0.5$ and $z = 1$, and two poles located at $z = 0.5$ and $z = 0$. Write an expression for (a) its amplitude response as a function of a single trigonometric term, and (b) its phase response.

6.21 The block diagram of a digital filter is shown below.
(a) Consider the case when the switch is **open**. (1) What is the filter's transfer function? (2) Write its difference equation. (3) What is the magnitude of its amplitude response at one-half the sampling frequency?
(b) Consider the case when the switch is **closed**. (1) What is the filter's transfer function? (2) Write its difference equation. (3) If G is a negative fraction, what traditional filter type best describes this filter? (4) If $G = 1$, where are its poles and zeros located? (5) If a unit-amplitude step function is applied to the input with $G = 1$, what is the output sequence? (6) If a unit-amplitude step function is applied to the input with $G = -1$, what is the output sequence?



6.22 A digital filter has the block diagram shown below. (a) Write its transfer function. (b) Where are its poles and zeros located?



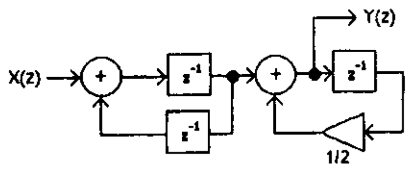
6.23 The difference equation for a filter is: $y(nT) = 2y(nT - T) + 2x(nT) + x(nT - T)$. Draw its z-plane pole-zero plot, indicating the locations of the poles and zeros.

6.24 Application of a unit **impulse** to the input of a filter produces the output sequence {1, 0, 1, 0, 1, 0, ...}. What is the difference equation for this filter?

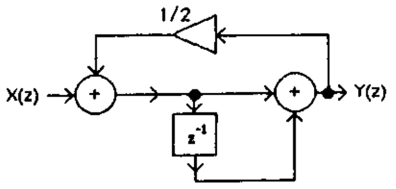
6.25 Application of a unit step to the input of a filter produces the output sequence {1, 1, 2, 2, 3, 3, ...}. What is the difference equation for this filter? HINT: The z transform of a unit step is

$$\frac{1}{1 - z^{-1}}$$

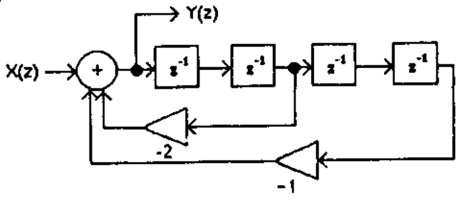
6.26 Write the transfer functions of the following digital filters:



(a)



(b)



(c)

6.27 What is the phase response for a digital filter with the transfer function

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

6.28 Write the amplitude response of a filter with the transfer function:

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

6.29 A filter has two poles at $z = 0.5 \pm j0.5$ and two zeros at $z = 0.707 \pm j0.707$. What traditional filter type best describes this filter?

6.30 A filter operating at a sampling frequency of 1000 samples/s has a pole at $z = 1/2$ and a zero at $z = 3$. What is the magnitude of its amplitude response at dc?

6.31 A filter is described by the difference equation: $y(nT) = x(nT) + x(nT - T) - 0.9y(nT - T)$. What is its transfer function?

6.32 A filter has the difference equation: $y(nT) = y(nT - 2T) + x(nT) + x(nT - T)$. What traditional filter type best describes this filter?

6.33 In response to a unit impulse applied to its input, a filter has the output sequence: {1, 1/2, 1/4, 1/8,...}. What is its transfer function?

6.34 The difference equation for a digital filter is: $y(nT) = x(nT) - ax(nT - T) - by(nT - T)$. Variables a and b are positive integers. What traditional type of filter is this if $a = 1$ and (a) $b = 0.8$, (b) $b > 1$?

.35 Write the (a) amplitude response, (b) phase response, and (c) difference equation for a filter with the transfer function:

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

.36 Write the (a) amplitude response, (b) phase response, and (c) difference equation for a filter with the transfer function:

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

.37 A filter operating at a sampling frequency of 1000 samples/s has a pole at $z = 1$ and a zero at $z = 2$. What is the magnitude of its amplitude response at 500 Hz?

.38 A filter operating at a sampling frequency of 200 samples/s has poles at $z = \pm j/2$ and zeros at $z = \pm 1$. What is the magnitude of its amplitude response at 50 Hz?

.39 A filter is described by the difference equation: $2y(nT) + y(nT - T) = 2x(nT)$. What is its transfer function $H(z)$?

.40 A filter has the difference equation: $y(nT) = y(nT - T) - y(nT - 2T) + x(nT) + x(nT - T)$. What is its transfer function?

.41 In response to a unit impulse applied to its input, a filter has the output sequence: {1, 1, 1/2, 1/4, 1/8,...}. What is its transfer function?

.42 A filter has a transfer function that is identical to the z transform of a unit step. A unit step is applied at its input. What is its output sequence?

.43 A filter has a transfer function that is equal to the z transform of a ramp. A unit impulse is applied at its input. What is its output sequence? HINT: The equation for a ramp is $x(nT) = nT$, and its z transform is

$$X(z) = \frac{Tz^{-1}}{(1 - z^{-1})^2}$$

6.44 A ramp applied to the input of digital filter produces the output sequence: {0, T, T, T, T, ...}. What is the transfer function of the filter?

6.45 A digital filter has a unit step {i.e., 1, 1, 1, 1, ...) output sequence when a unit impulse {i.e., 1, 0, 0, ...} is applied at its input. How is this filter best described?

6.46 A discrete impulse function is applied to the inputs of four different filters. For each of the output sequences that follow, state whether the filter is recursive or nonrecursive. (a) {1, 2, 3, 4, 5, 6, 0, 0, 0,...}, (b) {1, -1, 1, -1, 1, -1,...}, (c) {1, 2, 4, 8, 16,...}, (d) {1, 0.5, 0.25, 0.125,...}.

6.47 What similarities are common to all three integrator algorithms discussed in the text (i.e., rectangular, trapezoidal, and Simpson's rule)?

6.48 A differentiator is cascaded with an integrator. The differentiator uses the two-point difference algorithm:

$$H_1(z) = \frac{1 - z^{-1}}{T}$$

The integrator uses trapezoidal integration:

$$H_2(z) = \frac{T}{2}\left[\frac{1 + z^{-1}}{1 - z^{-1}}\right]$$

A unit impulse is applied to the input. What is the output sequence?

6.49 A differentiator is cascaded with an integrator. The differentiator uses the three-poi central difference algorithm:

$$H_1(z) = \frac{1 - z^{-2}}{2T}$$

The integrator uses rectangular integration:

$$H_2(z) = T\left[\frac{1}{1 - z^{-1}}\right]$$

(a) A unit impulse is applied to the input. What is the output sequence? (b) What tradition filter type best describes this filter?

6.50 A digital filter has two zeros located at $z = 0.5$ and $z = 1$, and a single pole located at z 0.5. Write an expression for (a) its amplitude response as a function of a single trigonome ric term, and (b) its phase response.

6.51 In response to a unit impulse applied to its input, a filter has the output sequence: {2, –1, –1/2, 1/4, –1/8,...}. What is its transfer function?

6.52 The difference equation for a filter is: $y(nT - T) = x(nT - T) + 2x(nT - 4T) + 4x(nT - 10T$ What is its transfer function, $H(z)$?

6.53 What is the transfer function $H(z)$ of a digital filter with the difference equation

$$y(nT - 2T) = y(nT - T) + x(nT - T) + x(nT - 4T) + x(nT - 10T)$$

6.54 A digital filter has the following output sequence in response to a *unit impuls* {1, –2, 4, –8,...}. Where are its poles located?

6.55 A digital filter has a single zero located at $z = 0.5$ and a single pole located at $z = 0.5$. Wh are its amplitude and phase responses?

6.56 The difference equation for a filter is: $y(nT) = 2y(nT - T) + 2x(nT) + x(nT - T)$. What a the locations of its poles and zeros?

6.57 What traditional filter type best describes the filter with the z transform:

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

6.58 A discrete impulse function is applied to the inputs of four different filters. The outp sequences of these filters are listed below. Which one of these filters has a pole outside th unit circle? (a) {1, 2, 3, 4, 5, 6, 0, 0, 0,...} (b) {1, –1, 1, –1, 1, –1,...} (c) {1, 2, 4, 8 16,...} (d) {1, 0.5, 0.25, 0.125,...}

6.59 Draw the block diagram of a filter that has the difference equation:

$$y(nT) = y(nT - T) + y(nT - 2T) + x(nT) + x(nT - T)$$

6.60 What is the transfer function $H(z)$ of a filter described by the difference equation:

$$y(nT) + 0.5y(nT - T) = x(nT)$$

6.61 A filter has an output sequence of {1, 5, 3, –9, 0, 0,...} in response to the input sequence o {1, 3, 0, 0,...}. What is its transfer function?

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 $2\cos(\theta)$ 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

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

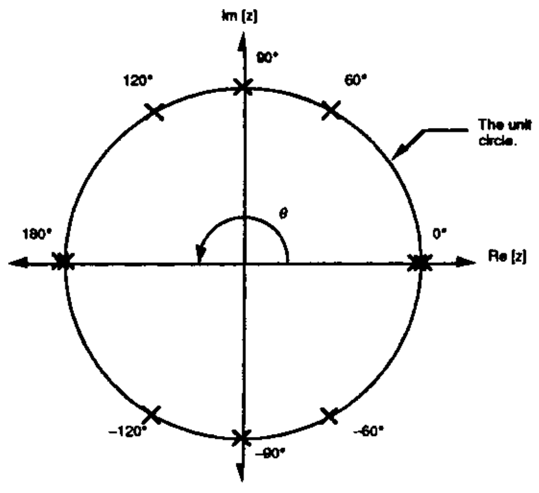
Solving for z, we arrive at

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

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 $n = \{0, 1, 2, \dots, m-1\}$.

θ	$\cos(\theta)$	$2\cos(\theta)$
0°	1	2
±60°	+1/2	1
±90°	0	0
±120°	−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

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

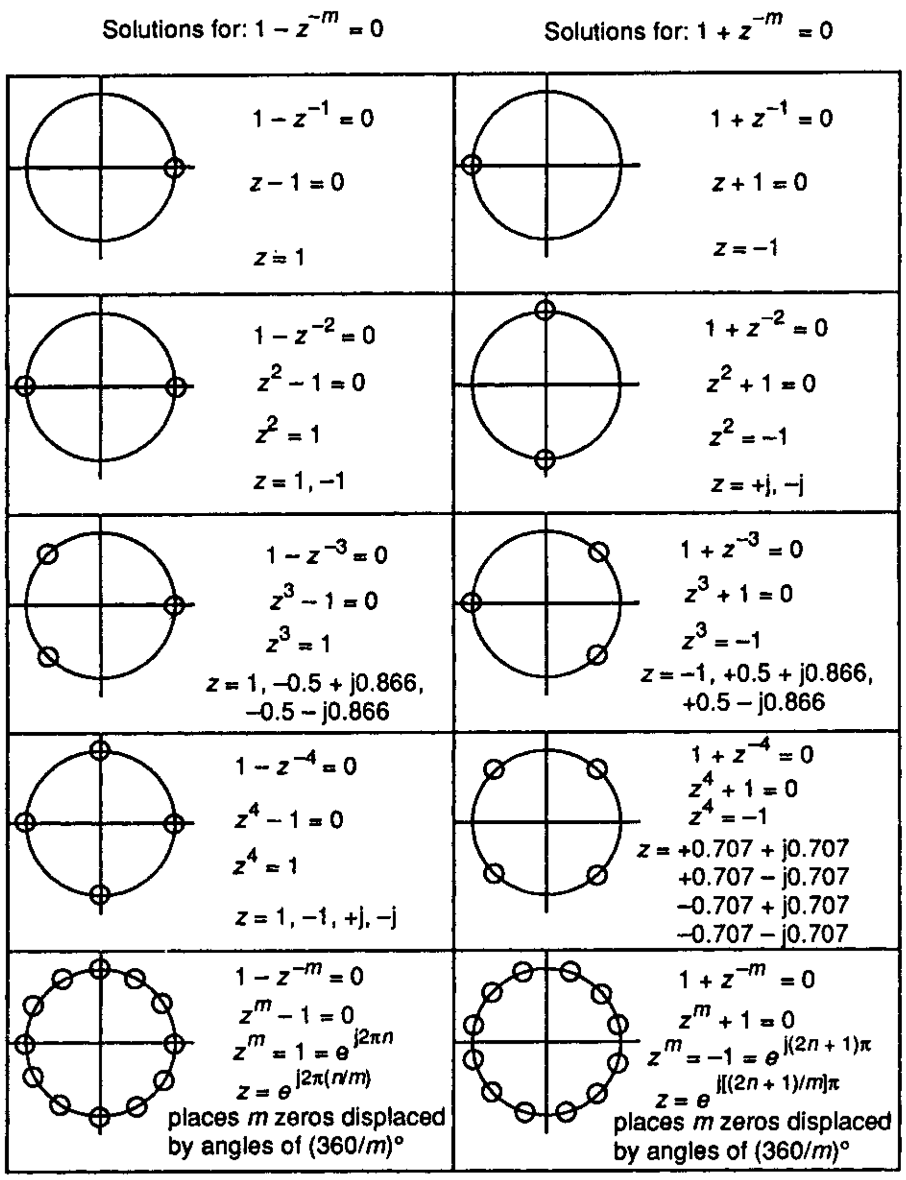


Figure 7.2 A comparison between the factors $(1 - z^{-m})$ and $(1 + z^{-m})$ for different values of m. When m is odd, the zeros appear "flipped over" from each other.

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 $\cos(\theta) = 0°$. 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 $\theta = 0°$ 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.

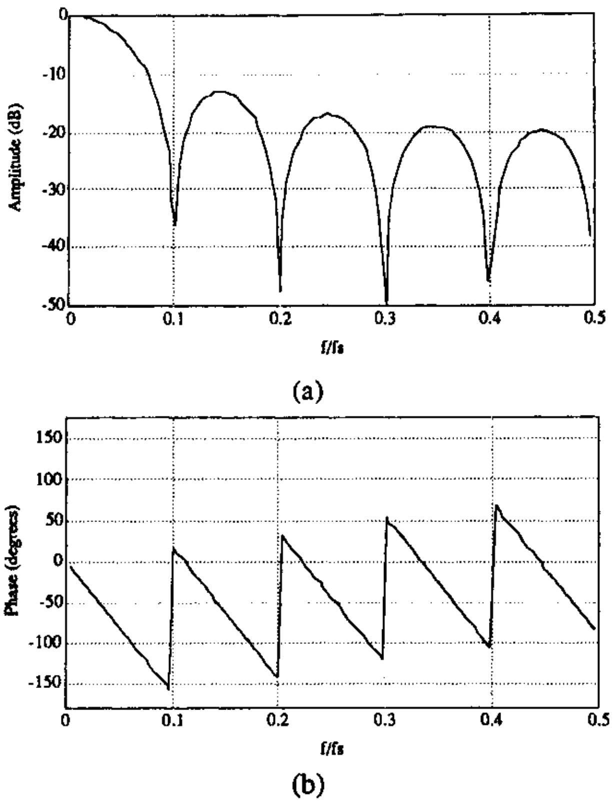


Figure 7.3 Low-pass filter with $m = 10$. (a) Amplitude response. (b) Phase response.

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 $\theta = 180°$ 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:

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

or

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

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 $\theta = 180°$. 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 $\angle H_a(z) = \angle H_{low}(z)$. 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.

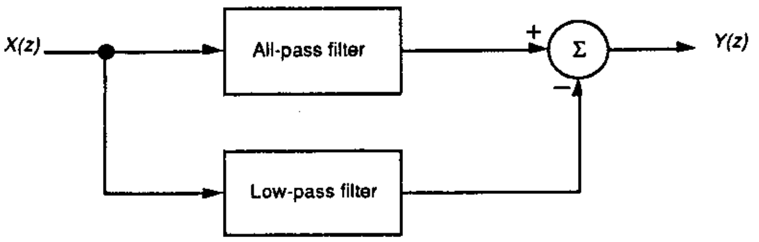


Figure 7.4 A block diagram of the low-pass filter being subtracted from a high-pass filter.

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, *2nd* order, *4th* order, etc., the sidelobes become smaller and smaller.

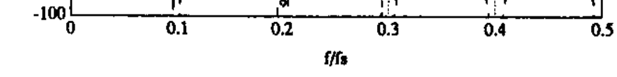


Figure 7.5 The effect of the order of a filter on its gain. This is an example of a high-pass filter with m = 10. Solid line: first order. Same as filter of Figure 7.3. Circles: second order. Dashed line: third order.

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.

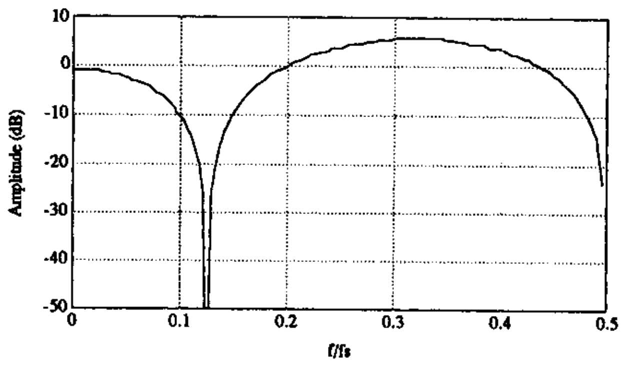
A piecewise-linear phase response is acceptable if the filter displays significant attenuation in the regions where the phase response has changed to a new value. If the stopband is not well attenuated, phase distortions will occur whenever the signals that are being filtered have significant energy 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.

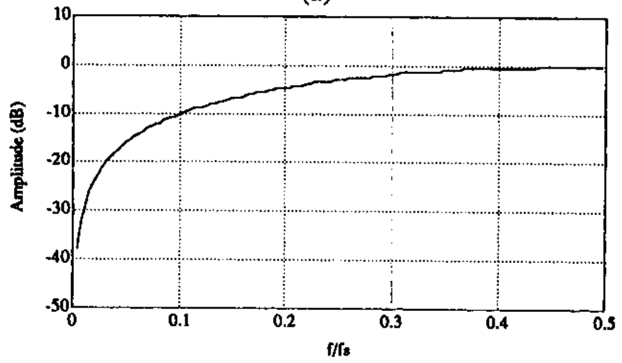
7.6 OTHER FAST-OPERATING DESIGN TECHNIQUES

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.

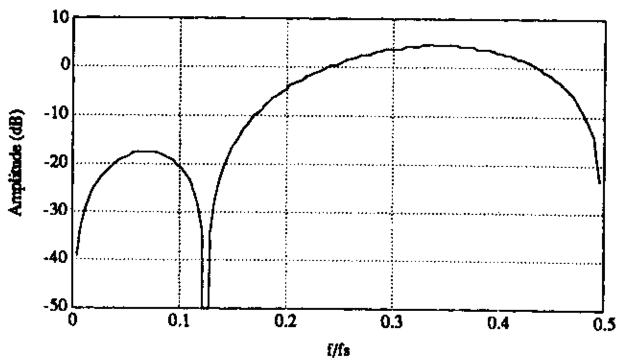
An example of this is shown in Figure 7.6. Since all the poles are located at the origin, these FIR filters are guaranteed to be stable. Placing zeros to attenuate gain is more flexible than placing poles to cancel zeros on the unit circle. Zeros are placed in conjugate pairs by using the factor $1 -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

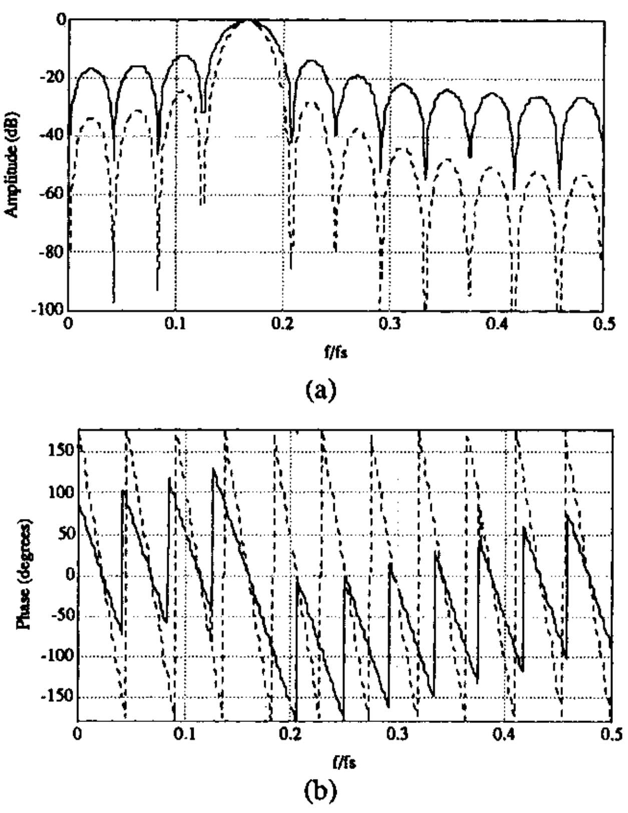


Figure 7.7 Piecewise-linear FIR bandpass filters. (a) Magnitude responses. Solid line: first-order filter with transfer function of $H(z) = (1 - z^{-24})/(1 - z^{-1} + z^{-2})$. Dashed line: second-order filter with transfer function of $H(z) = (1 - z^{-24})^2/(1 - z^{-1} + z^{-2})^2$. (b) Phase responses.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

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

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

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

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

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

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

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

$$\angle H(\omega T) = \pi - 22\omega T \qquad (7.18)$$

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

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

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

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

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: $x(t) = 100\,\sin(2\pi 10t)$. 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?

8

Adaptive Filters

Steven Tang

This chapter discusses how to build adaptive digital filters to perform noise cancellation and signal extraction. Adaptive techniques are advantageous because they do not require a priori knowledge of the signal or noise characteristics as do fixed filters. Adaptive filters employ a method of learning through an estimated synthesis of a desired signal and error feedback to modify the filter parameters. Adaptive techniques have been used in filtering of 60-Hz line frequency noise from ECG signals, extracting fetal ECG signals, and enhancing P waves, as well as for removing other artifacts from the ECG signal. This chapter provides the basic principles of adaptive digital filtering and demonstrates some direct applications.

In digital signal processing applications, frequently a desired signal is corrupted by interfering noise. In fixed filter methods, the basic premise behind optimal filtering is that we must have knowledge of both the signal and noise characteristics. It is also generally assumed that the statistics of both sources are well behaved or wide-sense stationary. An adaptive filter learns the statistics of the input sources and tracks them if they vary slowly.

8.1 PRINCIPAL NOISE CANCELER MODEL

In biomedical signal processing, adaptive techniques are valuable for eliminating noise interference. Figure 8.1 shows a general model of an adaptive filter noise canceler. In the discrete time case, we can model the primary input as $s(nT) + n_0(nT)$. The noise is additive and considered uncorrelated with the signal source. A secondary reference input to the filter feeds a noise $n_1(nT)$ into the filter to produce output $\zeta(nT)$ that is a close estimate of $n_0(nT)$. The noise $n_1(nT)$ is correlated in an unknown way to $n_0(nT)$.

The output $\zeta(nT)$ is subtracted from the primary input to produce the system output $y(nT)$. This output is also the error $\varepsilon(nT)$ that is used to adjust the taps of the adaptive filter coefficients $\{w(1,\ldots,p)\}$.

$$y(nT) = s(nT) + n_0(nT) - \zeta(nT) \tag{8.1}$$