9

IIR FILTER DESIGN

"Infinite Impulse Response"

Unlike FIR, they have feedback (recursive)

OUTLINE

OBJECTIVES

1. Use the bilinear transform to create digital filters from analog filters.
2. Use computer tools to find difference equation coefficients.
3. Properly implement IIR designs.

INTRODUCTION

The feedback in IIR difference equations produces very efficient transfer functions. They require fewer multiply and accumulate operations to accomplish as much as an equivalent FIR system. They also approximate the feedback inherent in continuous-time systems. This suggests the possibility that IIR systems could be used to provide digital equivalents to the classic analog filters. It would be very desirable to be able to make use of all the expertise that went into the development of those filters.

There are two popular ways of creating digital filters from analog filters: impulse invariance designs and bilinear transform designs. The impulse invariance approach starts with the impulse response of a desired analog filter, samples it, and creates a digital filter with the same impulse response samples. This ensures that a stable digital filter will result, which is an essential requirement. Since sampling is involved, the sampling theorem tells us that the resulting digital filter will have a frequency response that is an aliased version of the analog filter's response. The impulse invariance process involves some background we will develop in Chapter 12, and it will be further considered at that point.

The bilinear transform uses a frequency mapping approach to guarantee a stable digital filter. It maps the s plane poles of an analog filter into z plane poles that are within the unit circle. It involves an algebraic process similar to that used to create specific filter types from a prototype, and results in filters very similar to their analog counterparts. The bilinear design will be considered in this chapter.

The feedback that makes IIR filters so efficient can also produce problems. Register overflows during additions and processor word size limits on the accuracy to which filter coefficients can be set can have effects that are aggravated by the feedback. These issues must be considered when implementing an IIR filter.

9.1 BILINEAR TRANSFORMATION

The bilinear transformation is well known to mathematicians and other theorists. While it is unlikely we would have stumbled upon it as a tool for creating digital filters, it is easy to show that it has exactly the kinds of features we need. After completing this section you will be able to:

- Use the bilinear transform to create digital filters from analog filters.
- Set filter cutoff frequencies precisely.

The bilinear transformation relates two complex variables s and z as:

$$s = C\frac{z-1}{z+1} \tag{9.1}$$

where C is a real constant. It has unusual properties that have made it of interest in various fields of study. Students of transmission lines may notice a similarity

between the bilinear transformation and the equation of the reflection coefficient. The Smith chart is based on the bilinear transform.

There are two s planes involved in discussing the bilinear transform, and this can be a point of confusion unless it is clear what we are attempting to accomplish with the transformation. We have already defined a z plane that is related to the s plane of a *sampled* signal as $z = e^{sT_s}$. We will temporarily denote this "digital," or discrete-time s, plane with primes. We also have, as our starting point, a desirable transfer function defined in the "analog," or continuous-time, s plane. The bilinear transformation is being used to move the poles and zeros of the desired $H(s)$ into the z plane in the hope that doing so will result in a digital filter similar to the original analog filter. This process is suggested in Figure 9.1.

Solving Equation 9.1 for z, we obtain

$$z = \frac{C + s}{C - s} = \frac{(C + \sigma) + j\omega}{(C - \sigma) - j\omega}$$

$$|z| = \left|\frac{(C + \sigma) + j\omega}{(C - \sigma) - j\omega}\right| = \sqrt{\frac{(C + \sigma)^2 + \omega^2}{(C - \sigma)^2 + \omega^2}} < 1 \qquad \text{if } \sigma < 0 \qquad \textbf{(9.2)}$$

As a consequence of Equation 9.2, poles or zeros in the left half of the analog s plane map into poles or zeros within the unit circle of the z plane. Digital systems whose poles are within the unit circle of the z plane are stable. In other words, a stable $H(s)$ becomes a stable $H(z)$ using the bilinear transformation. Since stability is mandatory for a system to be useful, the bilinear transform passes the first crucial test.

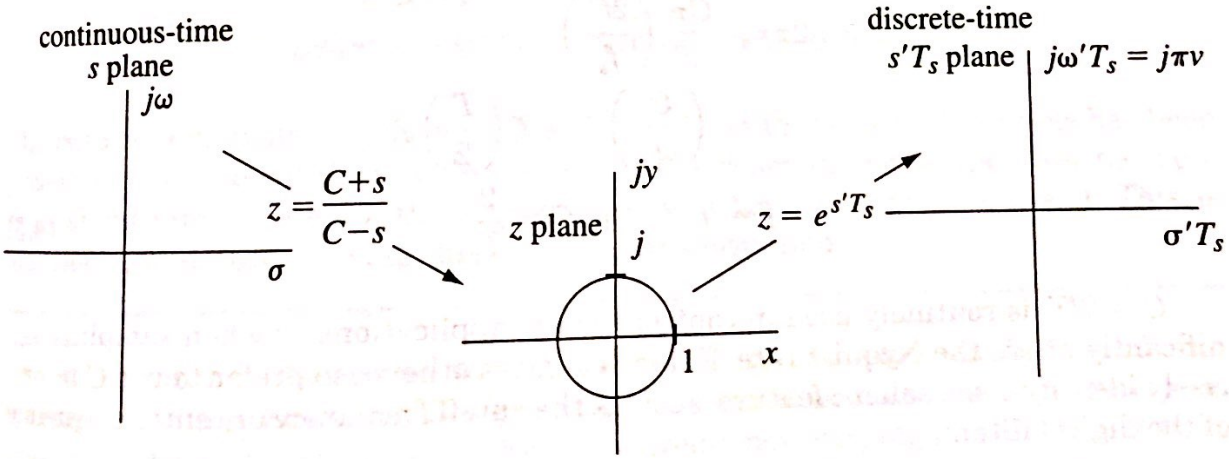


Figure 9.1 The bilinear transformation maps the poles and zeros of continuous-time systems into the z plane in such a way that their frequency response along the $j\omega$ axis becomes compressed onto the unit circle of the z plane. Each full revolution around the unit circle gives one full cycle of the periodic frequency response of the discrete-time system.

Checking the analog frequency response conditions, $s = j\omega$, shows that the entire $j\omega$ axis is mapped onto the unit circle of the z plane.

$$z = \frac{C+s}{C-s} = \frac{C+j\omega}{C-j\omega} = 1\angle 2\arctan\left(\frac{\omega}{C}\right) \tag{9.3}$$

The frequency response of the digital system is also taken along the unit circle of the z plane, and at any given position on the unit circle these frequencies are related by

$$e^{j\pi v} = e^{j2\arctan(\omega/C)} \qquad \therefore \frac{\pi}{2} v = \arctan\left(\frac{\omega}{C}\right)$$

$$\text{or} \qquad \omega = C\tan\frac{\pi v}{2} \tag{9.4}$$

The analog frequency and the digital baseband frequencies are directly related but in a highly nonlinear fashion, since an infinite range of $0 \le \omega \le +\infty$ is compressed into half the circumference of the unit circle, $0 \le v \le 1$. If we start with a known $H(s)$ and apply the bilinear transform to obtain an $H(z)$, each value of $z = e^{j\pi v}$ results in a numerical value for $H(z)$ that would also be obtained from plugging one particular $j\omega$ value in the original $H(s)$. The fact that they are directly related means a low-pass analog filter will transform into a low-pass digital filter, a bandpass filter into bandpass filter, and so on. Further, an analog Chebyshev filter will produce an equiripple digital filter passband, but the maximums and minimums will not have the same frequency spacing in the two systems. The constant C can be used to line up the responses at one particular frequency, such as cutoff.

Alternatively, for $v \ll 1$, $\tan\theta \approx \theta$, and the analog and digital systems can be made to agree in a general sense at low frequencies:

$$2\pi f = \frac{C\pi}{2}\left(\frac{2f'}{f_s}\right)$$

$$f = \left(\frac{C}{2f_s}\right)f' = C\left(\frac{T_s}{2}\right)f'$$

$$f = f' \qquad \text{if } C = \frac{2}{T_s} \tag{9.5}$$

$C = 2/T_s$ is routinely used in control system applications or when sampling significantly above the Nyquist rate. Filter designers otherwise prefer to use C to precisely identify some salient feature, such as the cutoff frequency or center frequency, of the digital filter.

EXAMPLE 9.1

Find $H(z)$ to provide a second-order 2-dB Chebyshev–based low-pass digital filter with $v_{co} = 0.3$ using the bilinear transformation.

Solution

The prototype filter has the following transfer function (Table 5.3):

$$H_p(p) = \frac{0.6538}{p^2 + 0.8038p + 0.8231}$$

C is found from equating cutoff frequencies:

$$\omega_{co} = 1 = C \tan\frac{\pi v_{co}}{2} = C \tan 0.15\pi = 0.5095C$$

So $C = 1.9626$, and, transforming directly from the prototype,

$$p = 1.9626\frac{z-1}{z+1}$$

Then

$$H(z) = \frac{0.6538}{\left[1.9626\frac{z-1}{z+1}\right]^2 + 0.8038\left[1.9626\frac{z-1}{z+1}\right] + 0.8231}$$

$$= \frac{0.6538(z+1)^2}{3.8518(z-1)^2 + 1.5775(z-1)(z+1) + 0.8231(z+1)^2}$$

Multiplying and grouping powers of z gives

$$H(z) = \frac{0.6538(z+1)^2}{6.2524z^2 - 6.0574z + 3.0974} = \frac{0.1046(z+1)^2}{z^2 - 0.9688z + 0.4954}$$

The resulting response is shown in Figure 9.2. Note that the decibel ripple has been preserved and the cutoff is at the desired value. Since the prototype filter has zero gain at infinite frequency, the digital filter will have zero gain at $v = 1$. This is another nice feature of using the bilinear transformation.

Implementing a bandpass digital filter using the bilinear transform involves a bit more planning. Referring to Figure 9.3, we want to be able to specify the digital filter cutoff frequencies v_H and v_L. From the frequency warping formula (Eq. 9.4) we can determine the required analog bandpass filter specifications, ω_c and B:

$$B = \omega_H - \omega_L = C\left[\tan\frac{\pi v_H}{2} - \tan\frac{\pi v_L}{2}\right] \tag{9.6}$$

$$\omega_C^2 = \omega_L\omega_H = C^2 \tan\frac{\pi v_L}{2}\tan\frac{\pi v_H}{2} \tag{9.7}$$

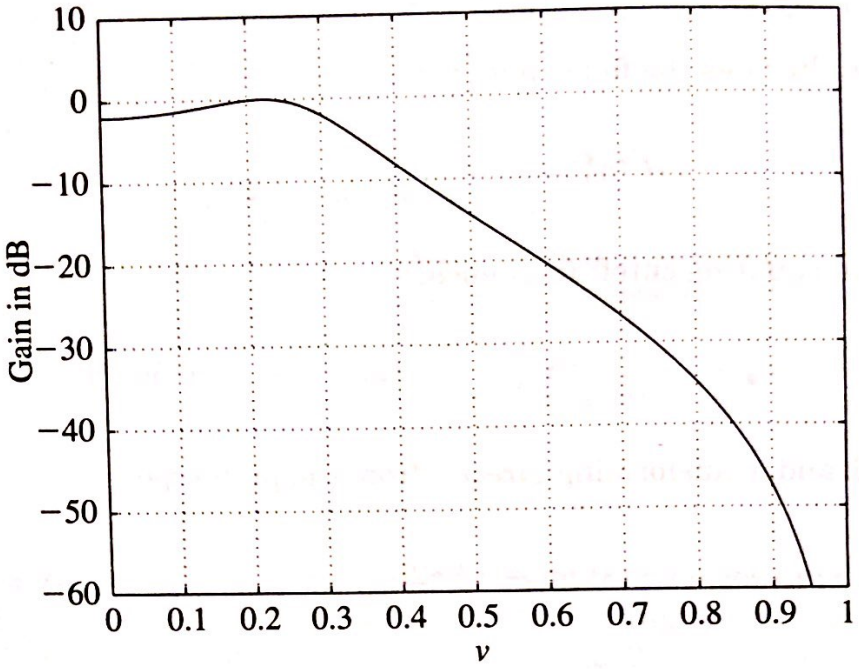


Figure 9.2 Frequency response of a bilinear transformed 2-dB Chebyshev low-pass filter with $\nu_{co} = 0.3$.

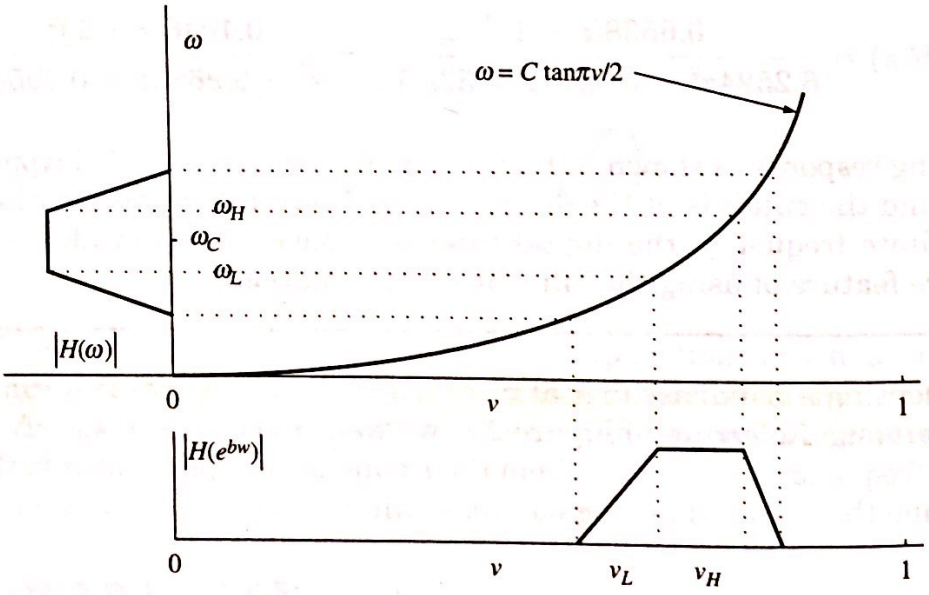


Figure 9.3 Frequency warping in the bilinear transformation as applied to bandpass filter design.

We will arbitrarily set ω_C to unity and calculate the required value for C. This gives the simplest bandpass transformation, but it does not affect the final result in any way. A three-step process is used:

1. Select the desired analog prototype.
2. Transform it to a bandpass filter using

$$p = \frac{s^2 + 1}{Bs}$$

3. Transform the analog bandpass filter to a digital filter using

$$s = C\frac{z - 1}{z + 1}$$

EXAMPLE 9.2

Determine $H(z)$ for a second-order Butterworth–based bandpass filter with $v_H = 0.35$ and $v_L = 0.25$. Use the bilinear transform.

Solution

Using Equation 9.7 we get

$$1 = C^2 \tan\frac{0.25\pi}{2} \tan\frac{0.35\pi}{2} = C^2(0.4142)(0.6128)$$

$$C = 1.9849$$

Then the required B is

$$B = 1.9849(0.6128 - 0.4142) = 0.3942$$

The Butterworth prototype is converted to a bandpass filter:

$$H_p(p) = \frac{1}{p^2 + \sqrt{2}p + 1}$$

$$H_{BP}(s) = \frac{1}{\left[\frac{s^2 + 1}{0.3942s}\right]^2 + \sqrt{2}\left[\frac{s^2 + 1}{0.3942s}\right] + 1}$$

$$H_{BP}(s) = \frac{0.1554s^2}{s^4 + 0.5575s^3 + 2.1554s^2 + 0.5575s + 1}$$

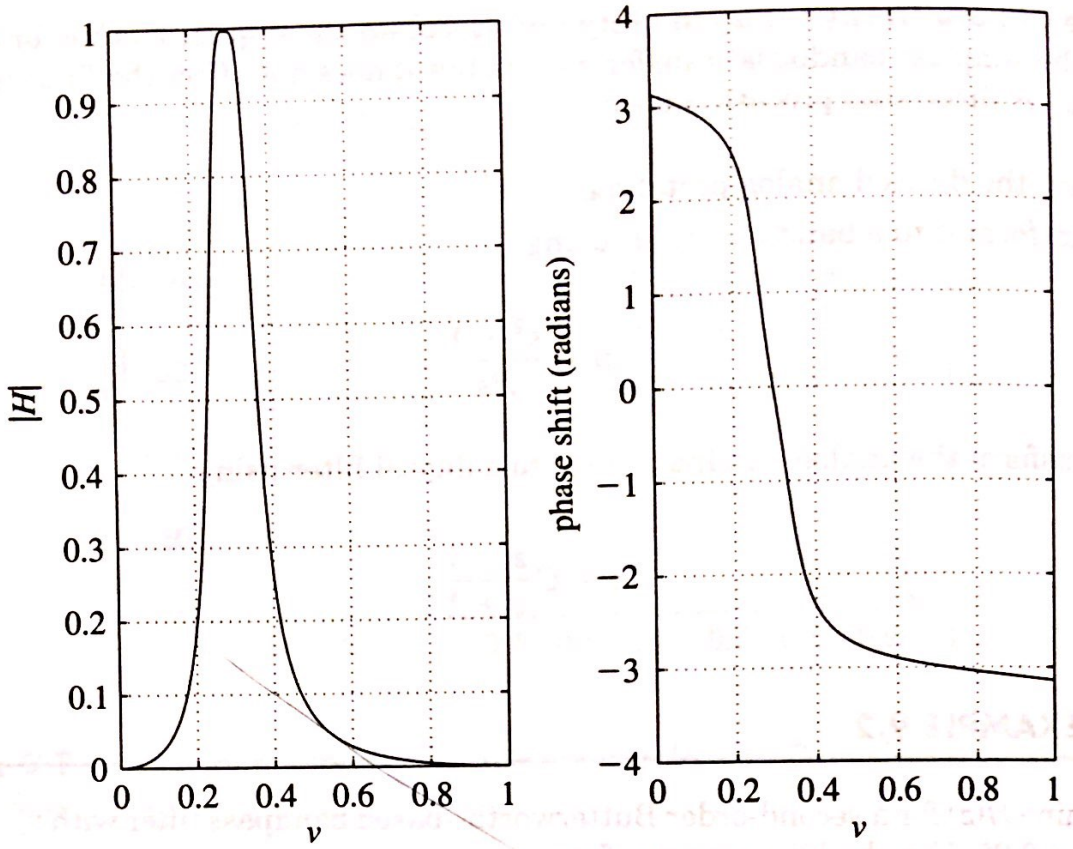


Figure 9.4 Second-order bandpass filter derived from the bilinear transformation.

Now we apply the bilinear transform:

digital

$$H_{BP}(z) = \frac{0.1554\left[1.9849\dfrac{z-1}{z+1}\right]^2}{\left[1.9849\dfrac{z-1}{z+1}\right]^4 + 0.5575\left[1.9849\dfrac{z-1}{z+1}\right]^3 + 2.1554\left[1.9849\dfrac{z-1}{z+1}\right]^2 + 0.5575\left[1.9849\dfrac{z-1}{z+1}\right] + 1}$$

A "little" algebra reduces the result to

$$H(z) = \frac{0.0201(1 - 2z^{-2} + z^{-4})}{1 - 2.1192z^{-1} + 2.6952z^{-2} - 1.6923z^{-3} + 0.6414z^{-4}}$$

and a frequency plot (Figure 9.4) confirms that the desired response has been achieved.

The task of simplifying the transfer functions of digital filters to their final form is tedious, and the numbers involved are never "nice," so either the process will be computerized or a good deal of key punching on the calculator will be necessary.

The bilinear transform has the advantage that the process is at least routine and is identical to the type of procedure used to make a particular analog filter from a prototype analog filter. Stopband and high-pass filters may also be created using the procedures demonstrated in this section.

9.2 MATLAB LESSON 9

The IIR design techniques involve numeric and algebraic calculations that are cumbersome even for low-order filters. Fortunately, MATLAB tools exist that make these techniques practical. After completing this section you will be able to:

- Use computer tools to find difference equation coefficients.
- Try out the Yule–Walker computer-aided IIR design program.

The commands of this chapter expect all z domain polynomials to be in negative powers of z and create z domain polynomials of that same form. Polynomials from the s domain remain in their usual form.

MATLAB EXAMPLES

bilinear

Using a desired $H(s)$ as the starting point, the bilinear derived $H(z)$ may be found. C may be set to $2/T_s$ or used to match an analog frequency to a digital frequency.

```
>anum=[1 0 4]; aden=[1 1 4];                    % notch filter, notch at
                                                  w_o = 2, f_o = 1/π
>freqs(anum,aden)                               % may as well take a look

>[dnum,dden]=bilinear(anum,aden,8/pi,1/pi);     % f_s = 8/π, f_o = 1/π
                                                  (v_o = 1/4)
>freqz(dnum,dden)                               % notch is at v_o = 0.25
>[dnum,dden]=bilinear(anum,aden,8/pi);          % f_s = 8/π, C = 2/T_s
>freqz(dnum,dden)                               % notch is at v_o ≈ 0.236

% bilinear can also work directly with poles, zeros, and a multiplying
  constant
>[az,ap,ak]=cheblap(6,1);                       % Chebyshev 6th-order
                                                  1-dB prototype
>[dz,dp,dk]=bilinear(az,ap,ak,2/pi, 1/(2*pi)); % f_s = 2/π, f_co = 1/(2π),
                                                  v_co = 2f_co/f_s = 0.5

>dnum=dk*poly(dz); dden=poly(dp);
>freqz(dnum,dden)
>dnum                                           % try getting this by hand!
```

butter cheby1 cheby2 ellip

Bilinear transformed versions of the standard filters are available directly in single commands. For example, the sixth-order Chebyshev 1-dB digital filter polynomials could have been found from the following:

```
>[dnum,dden]=cheby1(6,1,.5);          % 6th-order Chebyshev, 1-dB ripple,
                                        v_co = 0.5
>freqz(dnum,dden)

>[dnum,dden]=butter(8, .4);           % 8th-order Butterworth, v_co = 0.4
>freqz(dnum,dden)

> w=[.14 .34];
>[n,d]=ellip(5, 1, 30, w);           % 5th-order Cauer bandpass filter, 1-dB
                                        passband ripple,
                                      % 30-dB min. stopband atten., passing
                                        from 0.14 ≤ v ≤ 0.34
>freqz(n,d)                           % help ellip for other options
```

yulewalk

The *Yule–Walker* program designs IIR filters based on the desired magnitude response for its transfer function. The user specifies a set of frequencies starting at $v = 0$ and running to $v = 1$ in one vector and the desired magnitude response at each of these frequencies in a second vector. The user also sets the order of the filter. The Yule–Walker program then seeks numerator and denominator polynomials that provide an optimum fit to the points specified within the order constraint.

Multiple passbands and passbands of differing gain are possible with Yule–Walker designs. Common sense suggests that few points should be specified if only low-order polynomials are allowed. Experience with the classical filters also suggests that rapid changes from pass to stop regions require either high-order filters or lots of rippling in the pass- and stop-bands.

Designing a digital filter based on one of the classical analog filters and using the Yule–Walker program are fundamentally different processes. With the classical approaches, the designer basically knows what the filter characteristics are going to be from the start. With Yule–Walker, it is expected that many trial specifications may need to be made and the resulting filter characteristics observed before a final design is decided on. In cases where the standard filters are viable options, it may be very difficult to find a Yule–Walker design that is preferable. In comparing potential filter designs, the number of multiply and accumulate operations should be the same.

The Yule–Walker program provides a very useful alternative design approach for IIR digital filters. The polynomials that give an "optimum" match are those that produce the least error at the specified response points. This does not mean it is an optimum design or even that it produces a good filter. The user of the program is responsible for achieving that. The user must also realize that it is possible to specify very poor filters and to achieve them with great accuracy using computer-aided design.

EXAMPLE 9.3

Use the Yule–Walker program to design a digital low-pass filter with $v_{co} = 0.22$.

Solution

One of the difficulties in using Yule–Walker is setting the cutoff frequency. Since the program will try to match the filter response to the points specified in the design, specifying the −3-dB point is one possibility, provided no other points are specified nearby. On the other hand, the −3-dB point may not be the proper cutoff criterion if rippling occurs in the passband. The result for one set of specifications is shown for three different filter orders in Figure 9.5a.

As the order of the filter increases, rippling appears in the passband. The response never does pass right through the −3-dB point specified. Looking at the third-order filter on a decibel scale (Figure 9.5b) shows it to be very similar to the Chebyshev type 2 filters, which we have not particularly favored in the past. Both designs require seven multiplications. Further exploration would be needed to see if other v and m vectors would yield better results.

```
>v=[ 0 .1 .2 .22 .3 .7 1];
>m=[ 1 1 1 .707 0 0 0];
>[n,d]=yulewalk(3,v,m);
>h=freqz(n,d);
>vv=(0:511)/512;
>plot(v,m,'o')
>hold
Curent plot held
>plot(vv, abs(h))
>grid
```

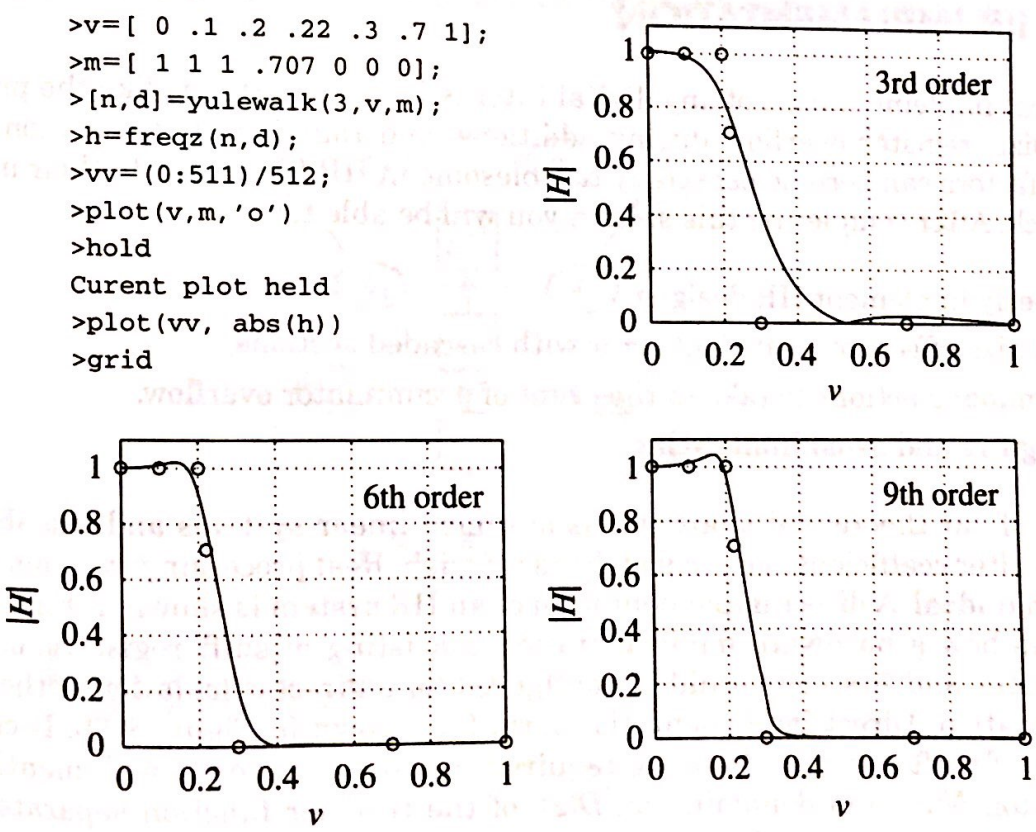


Figure 9.5a Yule–Walker filter designs of varying order. Circles show filter specification points.

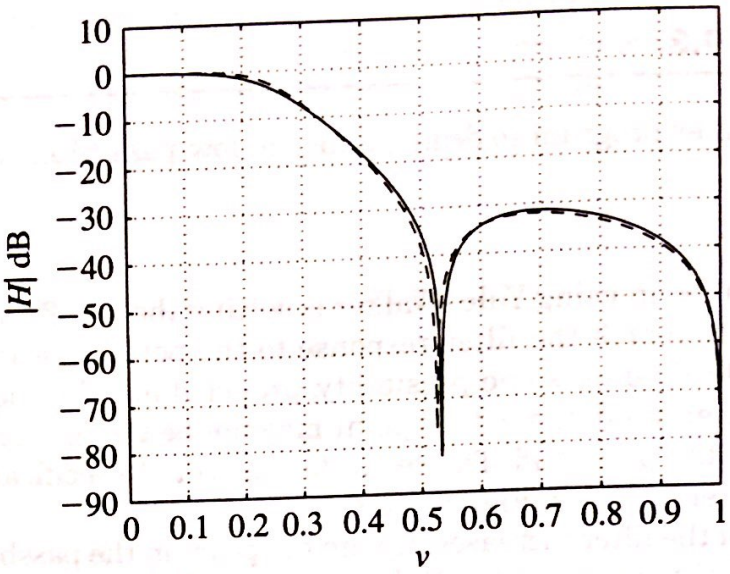


Figure 9.5b Comparison of the 3rd order Yule–Walker design (solid line) specified in Figure 9.5a with a 3rd order Chebyshev type 2 filter (dashed line) having a minimum stop-band attenuation of 32 dB @ $\nu = 0.48$.

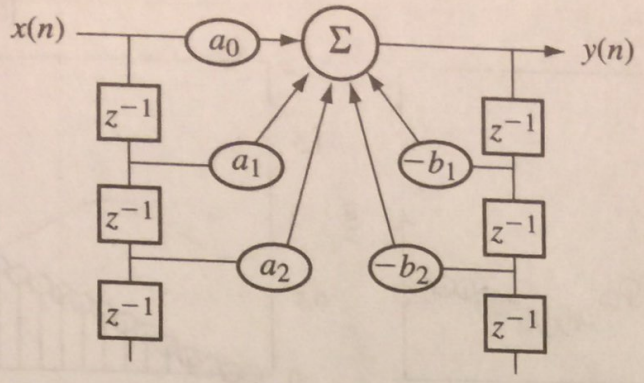
9.3 IIR IMPLEMENTATION

Practical implementation of any digital filter is limited by things like the processor word size, register overflow during additions, and round-off or truncation errors. These factors can become especially troublesome in IIR filters due to their inherent feedback. After completing this section you will be able to:

- Properly implement IIR designs.
- Minimize effects of coefficient error with cascaded sections.
- Recommend actions to take in the event of accumulator overflow.
- Recognize and avoid limit cycles.

All of our theoretical analysis has assumed linear systems and the ability to specify filter coefficients as accurately as we wish. Real processing systems are not quite that ideal. A direct implementation of an IIR system is shown in Figure 9.6a. It shows how a hardware implementation consisting of shift registers, constant multipliers, and summers could be configured for a direct calculation of the difference equation. Direct implementation form 2 is shown in Figure 9.6b. It cuts the number of shift registers that are required in half by cleverly implementing the numerator, $N(z)$, and denominator, $D(z)$, of the transfer function separately and cascading them.

$$H(z) = [1/D(z)][N(z)]$$

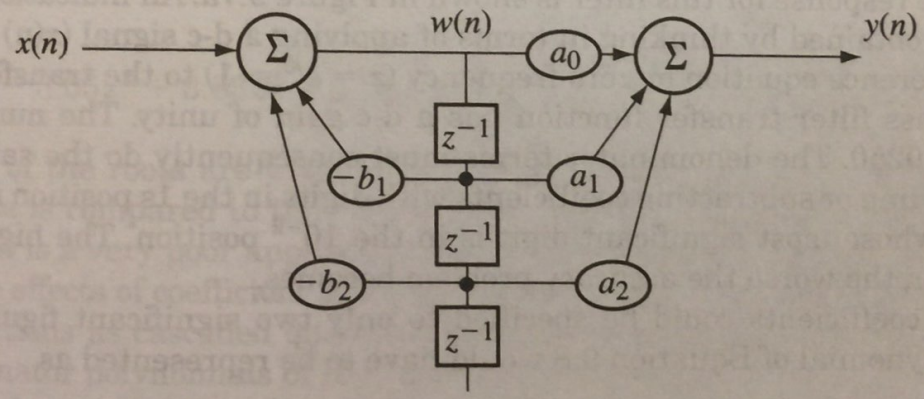


$$\frac{\hat{Y}}{\hat{X}} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots}$$

$$y(n) = a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) \cdots - b_1 y(n-1) - b_2 y(n-2) \cdots$$

Figure 9.6a Direct implementation form 1 for a difference equation.

Neither form of direct implementation of a transfer function works well if the accuracy with which the filter coefficients can be specified is limited excessively by the processor word size. That, of course, is not a surprise. The trouble is that the feedback inherent to IIR filters compounds the inaccuracy, and can also lead to



$$\hat{Y} = \hat{X}\,\frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots}$$

$$\hat{W} = \frac{\hat{X}}{1 + b_1 z^{-1} + b_2 z^{-2} + \cdots} \quad \text{and} \quad \hat{Y} = \hat{W}(a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots)$$

$$w(n) = x(n) - b_1 w(n-1) - b_2 w(n-2) - \cdots$$

$$y(n) = a_0 w(n) + a_1 w(n-1) + a_2 w(n-2) + \cdots$$

Figure 9.6b Direct implementation form 2 for a difference equation.

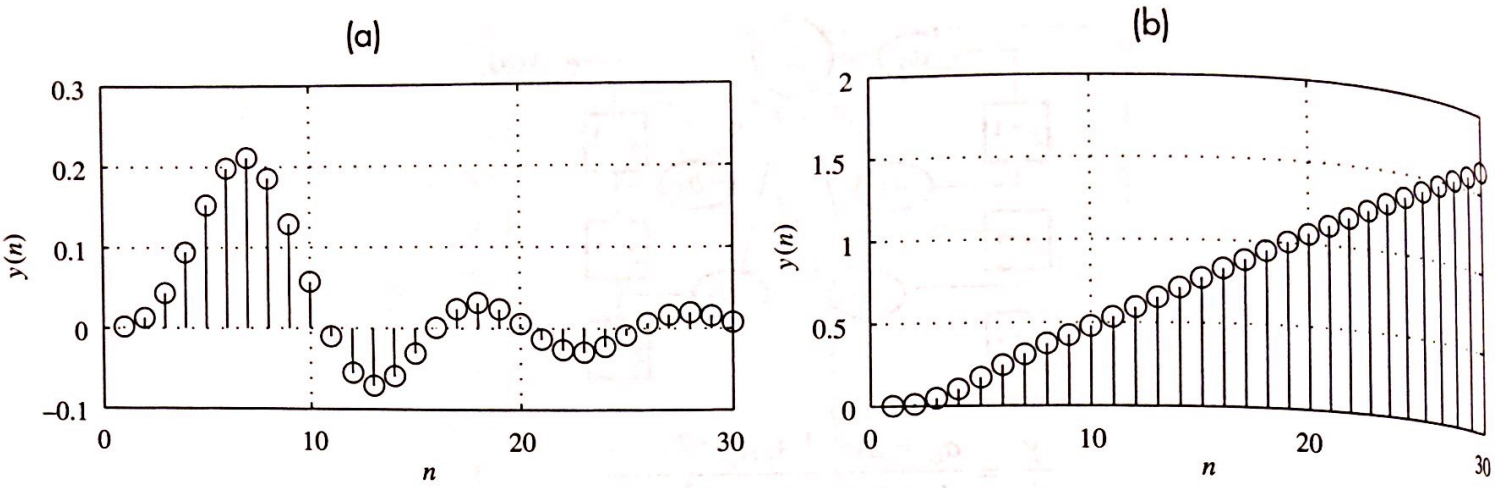


Figure 9.7 Impulse responses for a fourth-order, 1-dB, $v_{co} = 0.2$ bilinear transformed Chebyshev low-pass filter: (a) uses full coefficient accuracy and is a stable system; (b) limits the coefficients to two significant figures and is unstable.

instability. To demonstrate this problem suppose we wish to implement a fourth-order Chebyshev–based low-pass filter (1-dB ripple, $v_{co} = 0.20$) obtained from the bilinear transformation:

$$H(z) = \frac{0.0018z^4 + 0.0073z^3 + 0.0110z^2 + 0.0073z + 0.0018}{z^4 - 3.0543z^3 + 3.8290z^2 - 2.2925z + 0.5507} \tag{9.8}$$

The impulse response for this filter is shown in Figure 9.7a. An indication of the problem can be obtained by thinking in terms of applying a d-c signal ($x(n) = 1$ for all n) to the difference equation or zero frequency ($z = e^{j0} = 1$) to the transfer function. The low-pass filter transfer function has a d-c gain of unity. The numerator terms sum to 0.0250. The denominator terms must consequently do the same, but that involves adding or subtracting coefficients with digits in the 1s position in order to get a result whose most significant digit is in the 10^{-2} position. The higher the order of the filter, the worse the accuracy problem becomes.

If the filter coefficients could be specified to only two significant figures, the denominator polynomial of Equation 9.8 would have to be represented as

$$1.0z^4 - 3.1z^3 + 3.8z^2 - 2.3z + 0.6 = (z - 1)^2(z - 0.55 \pm j0.54)$$

which has a repeated pole on the unit circle. The system is unstable and has the impulse response shown in Figure 9.7b. (The calculated roots assume the polynomial coefficients are exact, and therefore the roots have been allowed more significant figures than the coefficients. We also assume that order of magnitude changes between the numerator and denominator coefficients can be handled separately, or we would need at least five significant figures to represent both sets of coefficients in exactly the same way.)

Since the filter is unstable, we will have to provide for a larger word size if we want to implement it. Suppose we can represent the filter coefficients to three significant figures. Then the denominator becomes

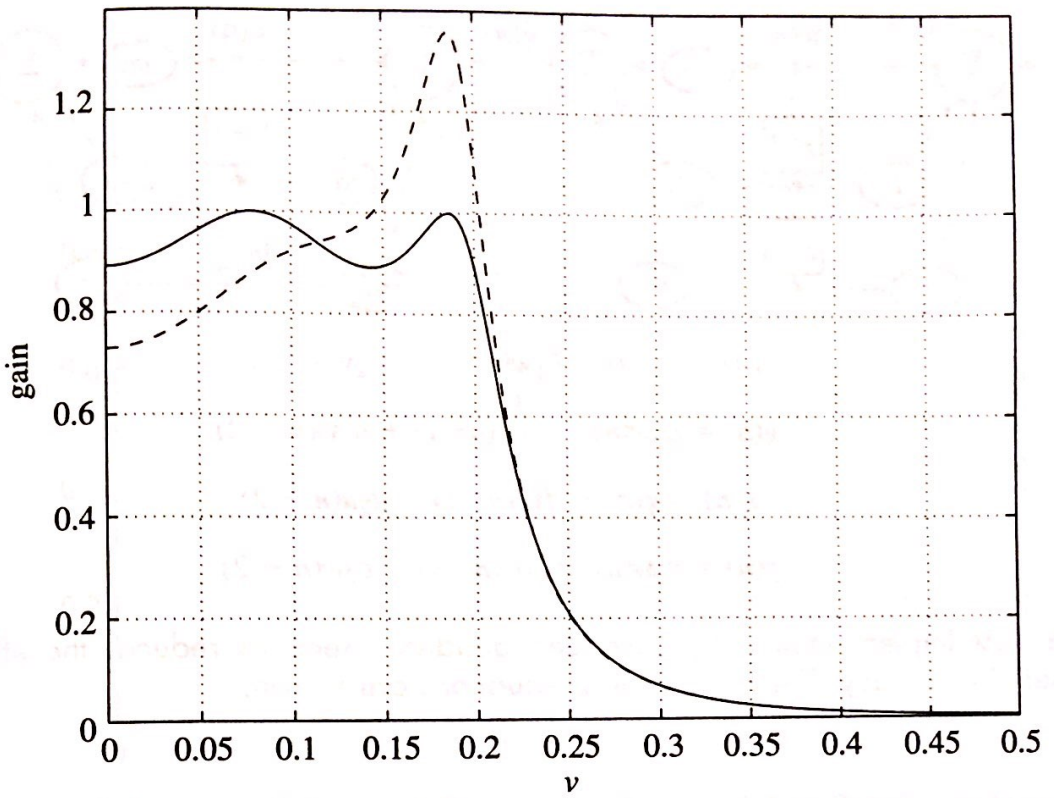


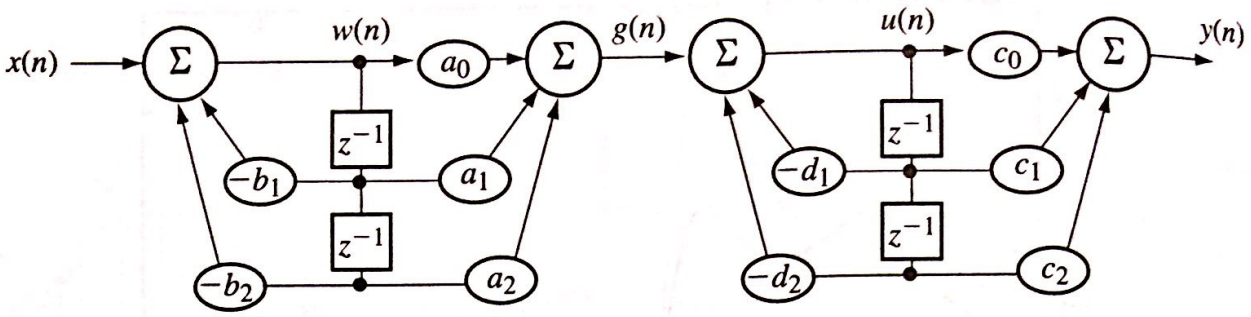
Figure 9.8 Fourth-order Chebyshev–based filter response with full (solid line) and three-significant-figure (dashed line) coefficient accuracy. The poles of each system are within the unit circle.

$$1.00z^4 - 3.05z^3 + 3.83z^2 - 2.29z + 0.55 = (z - 0.77 \pm j0.53)(z - 0.75 \pm j0.24)$$

and all of the roots are within the unit circle. The system is stable. Its frequency response is compared to that of the ideal filter in Figure 9.8. Although the filter is stable, it is a very poor approximation to the Chebyshev response desired.

The effects of coefficient accuracy can be further reduced by implementing transfer functions as cascaded quadratic sections. To accomplish this the numerator and denominator polynomials of $H(z)$ are factored. Complex roots are always recombined with their conjugates so that all filter coefficients remain real. Any quadratic numerator term can be combined with any quadratic denominator term. In high-order filters, many combinations are possible and some may work better than others. Normally each quadratic section would be implemented using the direct form 2 sections shown in Figure 9.9. Since the extra shift registers are not an issue for our demonstration, a direct form 1 implementation will be used. Our $H(z)$ factors as follows:

$$H(z) = H_1(z)H_2(z) = \underbrace{0.00183\left(\frac{z^2 + 2z + 1}{z^2 - 1.4996z + 0.8482}\right)}_{H_1(z) = \tilde{G}/\tilde{X}}\underbrace{\left(\frac{z^2 + 2z + 1}{z^2 - 1.5548z + 0.6493}\right)}_{H_2(z) = \tilde{Y}/\tilde{G}}$$



$$w(n) = x(n) - b_1 w(n-1) - b_2 w(n-2)$$

$$g(n) = a_0 w(n) + a_1 w(n-1) + a_2 w(n-2)$$

$$u(n) = g(n) - d_1 u(n-1) - d_2 u(n-2)$$

$$y(n) = c_0 w(n) + c_1 w(n-1) + c_2 w(n-2)$$

Figure 9.9 Implementation by cascaded quadratic sections reduces the effects of coefficient inaccuracy. The direct form 2 equations are shown.

Allowing three significant figures for the coefficients of the cascaded sections, the direct form 1 difference equations are:

$$g(n) = 0.00183x(n) + 0.00366x(n-1) + 0.00183x(n-2) + 1.50g(n-1) - 0.85g(n-2)$$

$$y(n) = g(n) + 2g(n-1) + g(n-2) + 1.55y(n-1) - 0.65y(n-2)$$

The results for the cascaded implementation are shown in Figure 9.10. The result is significantly better than the direct implementation, but it is still not a true Chebyshev equiripple response. Increasing the processor word size to handle four significant figures would probably work nicely for this fourth-order filter.

Two other problems that occur in implementing IIR filters are associated with accumulator operations. Accumulator overflow is a nonlinear effect that may override the predictions of linear systems. If two numbers are added in a digital processor, the result may overflow the accumulator. When that happens, an overflow flag is asserted. While the result may still be properly interpreted, there is no way to accommodate a larger number size. If the result is going to a D/A converter, for instance, the converter expects a fixed number of input bits and has no means of receiving or responding to an extra one. The overflow will consequently cause an incorrect value to be delivered to the output. What may have been a large negative number could end up as a large positive number when using a 2's complement number format. Probably the best thing the programmer can do in this situation is to use the flag information to set the output to the maximum positive or negative value, as appropriate. The D/A converter will then show the same kind of saturated output produced by an overdriven amplifier, rather than the sporadic appearances of random values.

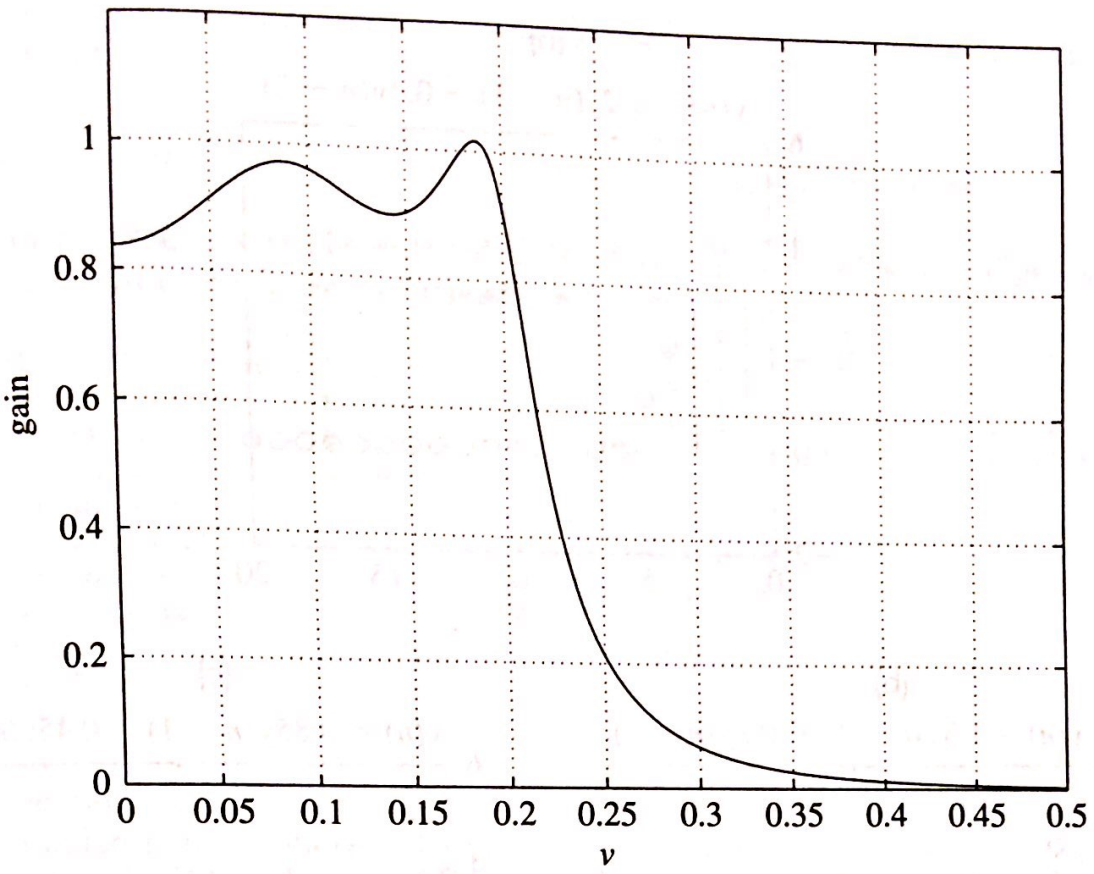


Figure 9.10 Cascade implementation of a fourth-order Chebyshev 1-dB low-pass filter using filter coefficients of three significant figures.

Another type of problem occurs due to rounding. In an 8-bit processor, for instance, the 8-bit filter coefficient is multiplied by an 8-bit representation of the signal, producing a 16-bit result. Since we are limiting the results to 8 bits, either we simply throw the lower byte of the multiplication result away (truncate), or we can consult its most significant bit and round the upper byte to the appropriate value. Rounding would give the most accurate result, but it can lead to *limit cycles* or *deadband* effects.

To investigate these effects, think of our signal as taking on integer values: −128 to 127 in an 8-bit machine. Neither the order of the filter nor the accuracy of its coefficients is at issue. The three difference equations of Figure 9.11 all have poles within the unit circle of the z plane, and checking their impulse response would demonstrate that they are indeed stable. However, since we want to consider the discrete levels of our signal and use rounding, we will need to write our own program. (An iteration table would be a second choice.) A MATLAB program will be demonstrated, although any standard programming language would do as well. We will supply the program with values for $y(1)$ and $y(2)$. These can be arbitrary integer values, since we can always find a two-sample input pulse that would cause them. In MATLAB, a function file **round.m** will be called to look up the rounded-off result after each multiplication. It rounds to the closest integer value.

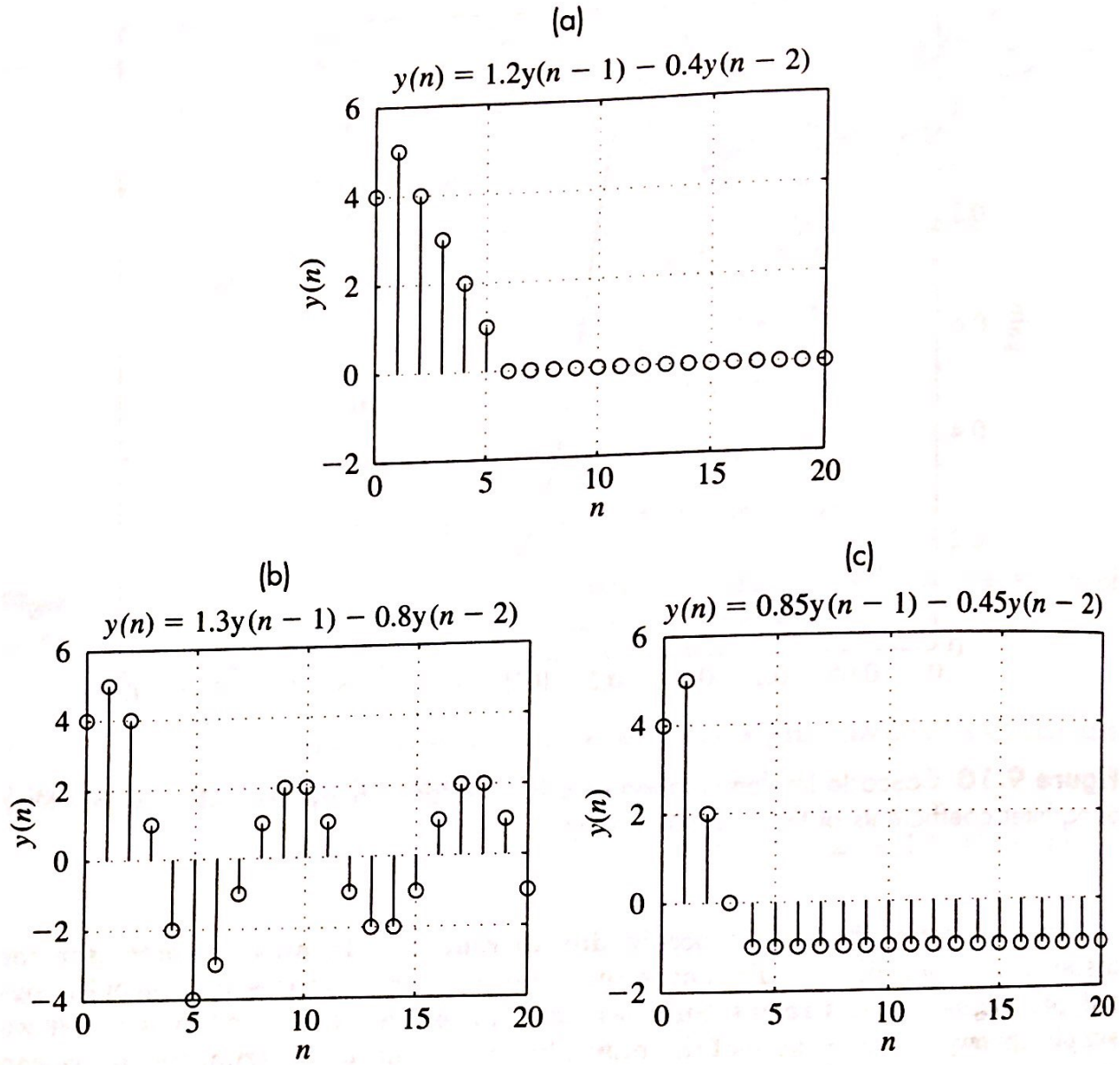


Figure 9.11 Impulse responses for stable second-order systems where round-off is used after each multiply. (a) The response settles to zero, as expected in a linear system. (b) The response falls into a limit cycle and oscillates forever. (c) The response settles at a nonzero value due to deadband loss of any correction signals. Initial values for all responses are based on a four-unit impulse at $n = 0$, which results in a five-unit response at $n = 1$.

File: round.m

```
function [n]=round(y)
% rounds an input value between -127 and 128
% to the nearest integer.
i=1; k=1;
if y<0
     k=-1;                          % save sign of y
end
z=abs(y);
```

```
x=linspace(0,128,129)+.5;          % look up table for integers 0-128
while z>x(i)
  i=i+1;
end
n=k*(x(i)-.5);                     % return with nearest integer
```

The following program can be entered in the Command Window; but if you want to explore using different filter coefficients, put it in a script M-file.

```
% MATLAB 2nd Order Difference Equation Program
>y(1)=4; y(2)=5;      % 2 initial values are needed to start the program
>for i=3:30           % use as many terms as you want, but start with y(3)
a = 1.3*y(i-1);       % multiply
z = round(a);         % call the rounding subroutine
b = -0.8*y(i-2);      % multiply the next terms
zz=round(b);          % call the rounding subroutine
y(i) = z + zz;
end                   % continue for 30 values
>t=0:29;
>stem(t,y)           % see the result! (Fig. 9.11b)
```

Truncation does not create the problems associated with rounding. From a simple-minded point of view, truncation always decreases the value of the output, and therefore it adds a damping factor to the difference equation response. Rounding, on the other hand, may contribute to an overshoot of the output value, which effectively lowers the damping. While these effects are diminished as larger word sizes are used, an IIR design should not be considered complete until it has been simulated and checked for limit cycling.

CHAPTER SUMMARY

The bilinear transformation creates z domain filters from s domain filters through a simple algebraic relationship. It converts stable $H(s)$ transfer functions into stable $H(z)$ transfer functions while retaining identical frequency response characteristics except for a warping of the frequency axis. The frequency axes of each domain can be matched at one frequency or provided a general correspondence at low frequencies. Through the bilinear transform, all of the classical filter types remain available for digital processing as IIR systems.

Practical application of the bilinear transformation for high-order filters would be a nightmare without computer assistance. A computer-aided design technique known as the Yule-Walker program creates IIR filter designs that minimize the error between desired and actual transfer function magnitudes at a set of points selected by the designer.

The feedback inherent in IIR filters will lead to unstable operation unless proper attention is paid to implementation techniques, having adequate processor word size for establishing filter coefficients, or handling of accumulator overflows. A nonlinear oscillation called a limit cycle can result from using accumulator round-off.

PROBLEMS

Section 9.1

1. The standard one-pole low-pass filter is given by $H(s) = 1/(s + 1)$. Find $H(z)$ for an equivalent digital filter using the bilinear transform. Express the results in terms of the cutoff frequency v_{co}.
2. An analog notch filter with its notch at $\omega_o = 1$ rad/s has the transfer function

$$H(s) = \frac{s^2 + 1}{s^2 + 0.1s + 1}$$

   Use the bilinear transformation to find an equivalent digital filter with its notch at $v_o = 0.25$.
3. Use the bilinear transform to design a 0.5-dB second-order Chebyshev–based low-pass digital filter having $v_{co} = 0.35$.
4. Find the bilinear–based digital equivalent for the analog transfer function $H(s) = 10{,}000/(s + 100)^2$. Use $C = 100$ for the bilinear transformation constant. Sketch the pole-zero plots for $H(s)$ and $H(z)$. Do they have the same number of poles and zeros?
5. Design a second-order Butterworth–based high-pass digital filter having $v_{co} = 0.50$ using the bilinear transform.
6. Obtain an analytical expression for and sketch the $|H(e^{j\pi v})|$ obtained from the difference equation $y(n) = x(n) + ky(n - m)$ for the following values.
   a. $k = 1, m = 4$
   b. $k = -1, m = 4$
   c. $k = 0.95, m = 6$
7. An nth-order Butterworth–based bandpass filter with $v_L = 0.3$ and $v_H = 0.45$ is to be designed using the bilinear transform. Given that the design starts from a Butterworth bandpass filter with a center frequency of 1 rad/s, determine that filter's bandwidth and the C value needed in the bilinear transformation.
8. Use the bilinear transform to design a first-order Butterworth–based bandpass digital filter to have $v_L = 0.5$ and $v_H = 0.7$.
9. Design a first-order bilinear–based stopband digital filter to have $v_L = 0.1$ and $v_H = 0.3$. Use a 0.5-dB Chebyshev prototype.

Section 9.2

10. Design a tenth-order 2-dB Chebyshev–based low-pass digital filter with $v_{co} = 0.20$ using the bilinear transform.
    a. Plot the filter response over a sensible decibel range.
    b. Use the filter to process 250 samples of the waveform

$$x(n) = \sin 0.20\pi n + 10 \sin 0.25\pi n + 100 \sin 0.55\pi n$$

Plot the output, and relate it to the filter response.

11. Use the bilinear transform to design a Butterworth–based fifth-order band-pass filter to pass frequencies of $0.15 \leq v \leq 0.25$.
    a. Determine the filter transfer function, and plot its magnitude and phase response.
    b. Use the filter to process 200 samples of the signal

$$x\,(n) = 10 \cos 0.06\pi n + \cos 0.20\pi n + 50 \cos 0.4\pi n$$

    Plot $x(n)$ and output $y(n)$. Discuss the results in terms of the filter's response curve.

12. Given the transfer function

$$H_1(z) = \frac{0.0181 - 0.0543z^{-2} + 0.0543z^{-4} - 0.0181z^{-6}}{1 - 2.9419z^{-1} + 4.7023z^{-2} - 4.6341z^{-3} + 3.0774z^{-4} - 1.2466z^{-5} + 0.2781z^{-6}}$$

    a. Plot the magnitude and phase of $H_1(z)$.
    b. Create a new filter $H_2(z) = H_1(-z)$, and plot its magnitude and phase.
    c. Compare the passbands and group delays of the two filters.

13. Sinusoids of frequency v may be generated digitally by hitting the following difference equations with an impulse. Demonstrate this for $v = 0.02$. What is the difference between the waveforms generated by the two equations?
    a. $y(n) = (\sin \pi v)x(n-1) + (2 \cos \pi v)y(n-1) - y(n-2)$
    b. $y(n) = x(n) - (\cos \pi v)x(n-1) + (2 \cos \pi v)y(n-1) - y(n-2)$

14. Plot the characteristic of a Butterworth–based bilinear design for a third-order low-pass filter with $v_{co} = 0.2$. Design a third-order Yule–Walker filter to compete with the Butterworth–based design. Show the frequency and magnitude vectors that give what you regard to be the best result, and plot the final Yule–Walker and Butterworth characteristics on the same graph. Discuss the advantages of the two filters in terms of flatness of passband, accuracy of the cutoff frequency, stopband attenuation, and the number of multiply/accumulate operations required.

15. Repeat Problem 14 using a type 1 Chebyshev–based design with a passband ripple of 1 dB as the target characteristic.

Section 9.3

16. Express the given transfer functions as products of first- or second-order functions with real coefficients.
    a. $H(z) = \dfrac{0.3(1 - z^{-3})}{1 + 0.2z^{-1} - 0.6z^{-2} + 0.4z^{-3}}$

b. $H(z) = \dfrac{1 + 0.2z^{-1} - 0.6z^{-2} + 0.4z^{-3}}{1 - 0.5z^{-3}}$

c. $H(z) = \dfrac{(1 - 0.4512z^{-1})(1 - 0.2201z^{-1})\,(1 - 0.512z^{-1} + 0.3131z^{-2})}{1 + 0.1234z^{-2} - 0.4321z^{-4}}$

17. A third-order Chebyshev–based filter design results in

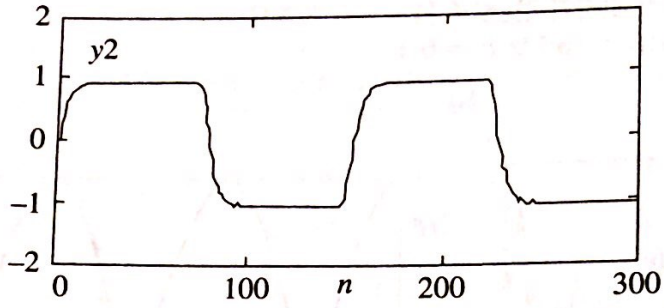
$$H(z) = \frac{0.01(1.15 + 3.44z^{-1} + 3.44z^{-2} + 1.15z^{-3}}{1 - 2.1378z^{-1} + 1.7693z^{-2} - 0.5398z^{-3}}$$

    a. Can the filter be implemented in direct form if its coefficients are limited to two significant figures (the ones and tenths decimal position values)? Show the impulse response of the filter under that implementation.
    b. Provide a cascade implementation of the filter, again limiting the coefficients to two significant figures. Plot $|H|$ vs. v.

18. Use the bilinear transform to design a fourth-order Butterworth–based low-pass filter with $v_{co} = 0.35$.
    a. Can a direct form implementation be obtained for this filter if only two significant figures are allowed in its coefficients?
    b. Show a cascaded form for this filter using two-significant-figure coefficients.
    c. Provide frequency response plots for each of the implementations that is stable.

19. Calculate the first 10 outputs of the difference equations $y(n) = \pm 0.8y(n - 1)$ if y is rounded to integer values. Assume $y(0) = 10$.

20. Write a program to plot the first 30 outputs from the difference equation $y(n) = 0.4y(n - 1) - 0.4y(n - 2) - 0.5y(n - 3)$ if each multiplication is rounded to the closest integer value. Use nonzero initial values of your choice in the range $-20 \le y \le 20$.



Chapter 9

9.1 $H(z) = \dfrac{1}{\left[C\dfrac{z-1}{z+1}\right]+1} = \dfrac{z+1}{(C+1)z-(C-1)} =$

$$\frac{z+1}{[1+\cot(\pi v_{co}/2)]z+[1-\cot(\pi v_{co}/2)]}$$

9.3 $H_p(s) = \dfrac{1.4314}{s^2+1.4256s+1.5162}, C = 1.6319, H(z) = \dfrac{0.2200(z^2+2z+1)}{z^2-0.3525z+0.2848}$

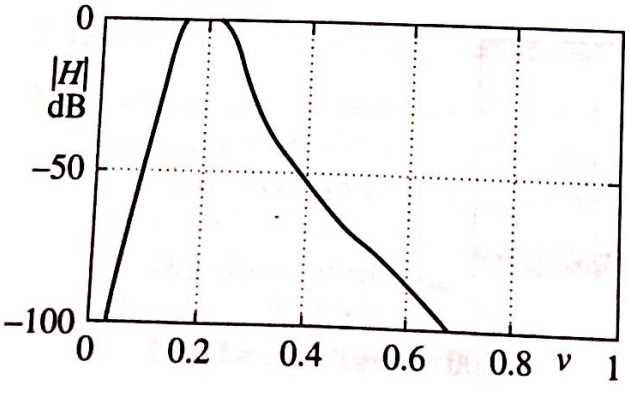
9.5 $H_p(p) = \dfrac{1}{p^2+1.4142p+1}, H_{HP}(s) = \dfrac{s^2}{s^2+1.4142s+1}, C = 1,$

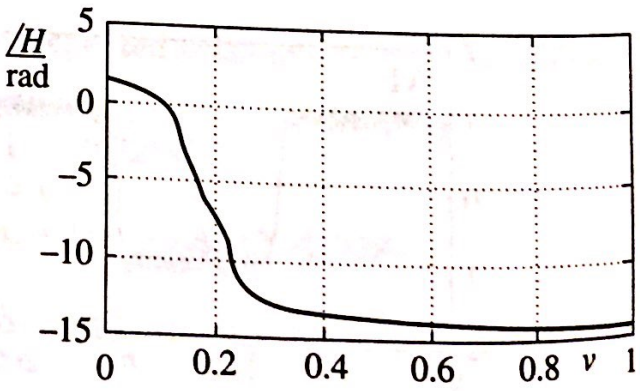
$$H(z) = \frac{0.2929(z^2-2z+1)}{z^2+0.1716}$$

9.7 $C = 1.5158, B = 0.3446\text{C} = 0.5224$

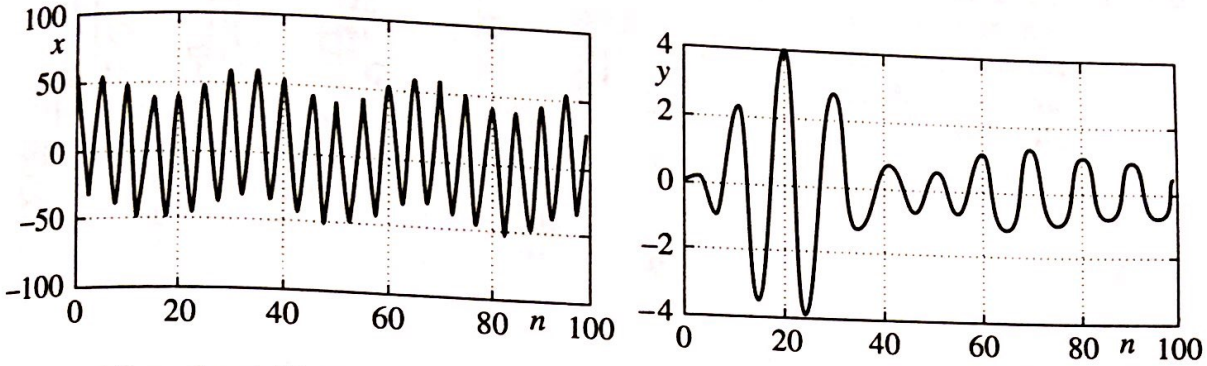
9.9 $C = 3.5200, B = 1.2359, H_{BS}(z) = \dfrac{0.8981-1.5279z^{-1}+0.8981z^{-2}}{1-1.5279z^{-1}+0.7961z^{-2}}$

9.11 a)



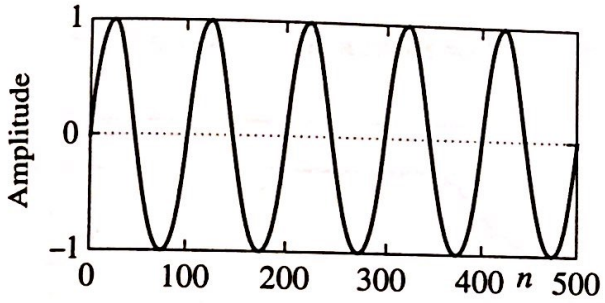


b)

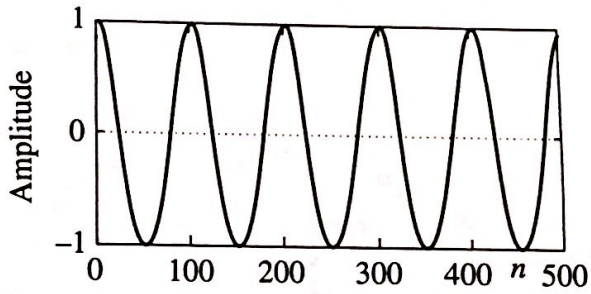


After about 50 transient samples, the output is essentially the $v = 0.2$ signal (10 samples/cycle).

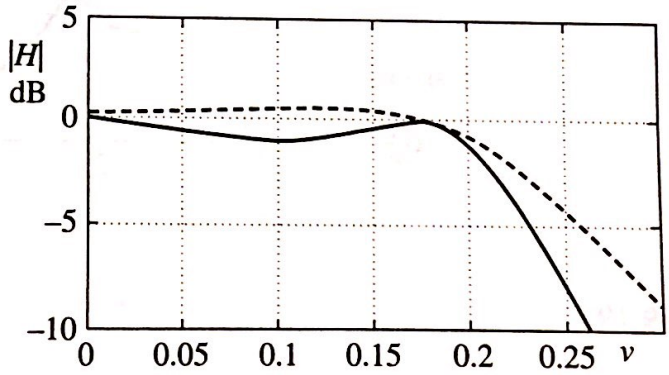
9.13 a)

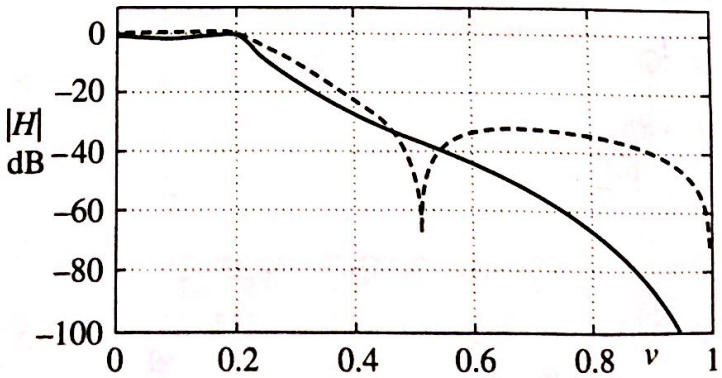


b)

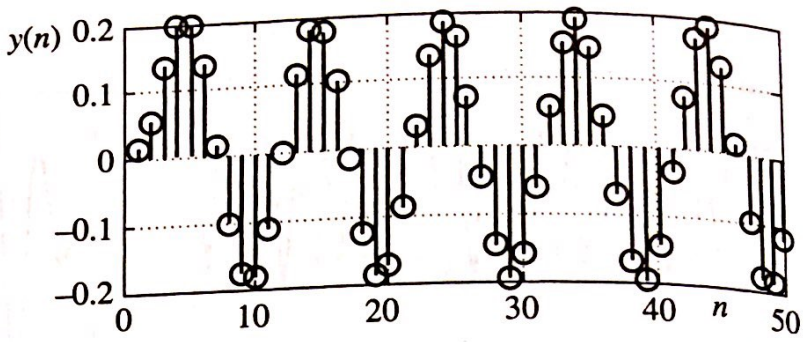


9.15 One solution is:
v = [0 0.1 0.20 0.20 1];
m = [1 1 1.5 0 0];
The Yule–Walker design does not compare well with a Chebyshev type 1 filter (solid). Other vectors may do better or worse. Seven multiplications are required in each case.



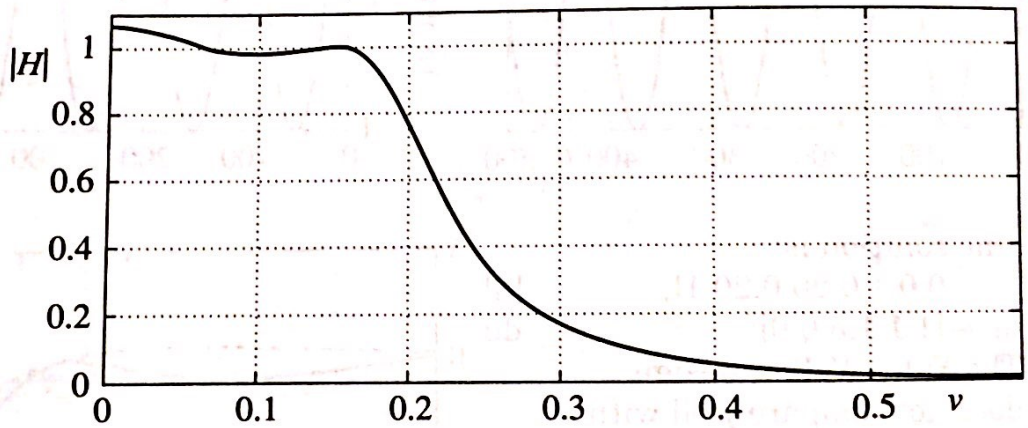


9.17 a) Using two significant figures puts poles on the unit circle—conditionally stable.



b) $H(z) = \left(\frac{0.0115(z + 1)}{z - 0.7233}\right)\left(\frac{z^2 + 2z + 1}{z^2 - 1.4145z + 0.7462}\right)$

Two-significant-figure cascade implementation is stable and resembles the desired Chebyshev response.



9.19