Chapter 9: The Design and Implementation of Infinite Impulse Response Filters

9.1 Introduction

Poles have a much more powerful effect on the frequency response than zeros, and for this reason, an IIR filter comprising *n* coefficients may be designed with a much sharper transition zone than an FIR version with the same number. This implies that the IIR filter is desirable when speed is the critical determinant in selecting the kind of filter required. The recursion equation which characterises this filter is given by

$$y[n] = \sum_{k=0}^{M} a[k]x[n-k] - \sum_{k=1}^{N} b[k]y[n-k] \tag{9.1}$$

and its transfer function by

$$H(z) = \frac{a[0] + a[1]z^{-1} + ... + a[m]z^{-m}}{1 + b[1]z^{-1} + ... + b[n]z^{-n}} = \frac{\sum_{m=0}^{M} a[m]z^{-m}}{1 + \sum_{n=1}^{N} b[n]z^{-n}} \tag{9.2}$$

It is probably true to say that IIR filters are more difficult to design than FIR types. They are also less flexible, since arbitrary IIR filters are particularly challenging to design with sufficient accuracy, respecting some desired frequency response template. Over the years, two main design approaches have emerged, as follows:

- Direct digital synthesis, eg pole-zero placement (as we have seen).
- Conversion of analogue filters into their discrete equivalents, eg the *bilinear z-transform*, or BZT method.

9.2 The bilinear z-transform: definitions

The BZT allows us to convert an entire analogue filter transfer function based on the Laplace transform into a discrete transfer function based on the z-transform, and is perhaps the most important re-mapping technique for the design of digital filters. It works as follows: wherever *s* appears in the transfer function, it is replaced by the expression

$$s = \frac{2}{T}\left[\frac{1 - z^{-1}}{1 + z^{-1}}\right] = \frac{2}{T}\left[\frac{z-1}{z+1}\right] \tag{9.3}$$

This process seems beautifully simple, and indeed it is. However, there is a fundamental difference between the Laplace space and *z*-space, because the former deals with continuous frequencies that extend to infinity, whereas the later is bounded by the sample rate and hence is associated with a Nyquist point. When we use the BZT to remap a continuous-time frequency response into a discrete-time equivalent, we are remapping a function from a straight line onto the locus of a circle (remember that the frequency response for a Laplace transform lies on the vertical axis, whereas for a *z*-transform it lies on the unit circle). Distortion therefore occurs between the two mappings, particularly at higher frequencies; this can readily be demonstrated if we substitute *s* by $j\Omega$ and *z* by $e^{j\omega T}$ in Equation 9.3, where Ω and ω represent continuous and discrete frequencies respectively. Therefore:

$$\begin{aligned} j\Omega &= \frac{2}{T}\left[\frac{e^{j\omega T} - 1}{e^{j\omega T} + 1}\right] = \frac{2}{T}\left[\frac{e^{j\omega T/2}(e^{j\omega T/2} - e^{-j\omega T/2})}{e^{j\omega T/2}(e^{j\omega T/2} + e^{-j\omega T/2})}\right] \\ &= \frac{2}{T}\left[\frac{(\cos(\omega T/2) + j\sin(\omega T/2))(\cos(\omega T/2) + j\sin(\omega T/2) - \cos(\omega T/2) + j\sin(\omega T/2))}{(\cos(\omega T/2) + j\sin(\omega T/2))(\cos(\omega T/2) + j\sin(\omega T/2) + \cos(\omega T/2) - j\sin(\omega T/2))}\right] \\ &= \frac{2}{T}\left[\frac{2j\sin(\omega T/2)}{2\cos(\omega T/2)}\right] \end{aligned} \tag{9.4}$$

i.e.

$$\Omega = \frac{2}{T}\tan(\omega T/2) \tag{9.5}$$

Hence the discrete frequency ω is related to the continuous frequency Ω by the relationship

$$\omega = \frac{2}{T}\tan^{-1}\left(\frac{\Omega T}{2}\right) \tag{9.6}$$

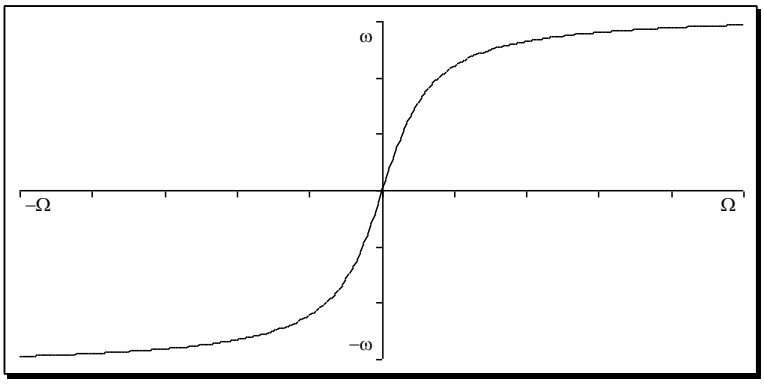


Figure 9.1. Frequency warping curve, describing the relationship between frequencies in continuous and discrete space.

A plot of this function is shown in Figure 9.1. As the continuous frequency increases, so the discrete equivalent tends towards a flat line. When converting from a continuous to a discrete transfer function using the BZT, it is desirable to introduce a *prewarping factor* into the proceedings. If our analogue filter has a cut-off frequency given by Ω, then we adjust it to a prewarped frequency Ω^* given by

$$\Omega^* = \frac{2}{T}\tan(\Omega T/2) \tag{9.7}$$

This progressively increases the frequency we feed into out conversion equations as the continuous frequency approaches the Nyquist point of our discrete, DSP system. To see how this works in practice, we'll start with a very simple analogue filter, a simple 1st order low-pass, whose transfer function is given by

$$H(s) = \frac{1}{1+sRC} \tag{9.8}$$

To obtain the discrete IIR equivalent, we commence by using the BZT without prewarp, which yields

$$H(z) = \frac{1}{\frac{2RC}{T}\left[\frac{1-z^{-1}}{1+z^{-1}}\right]+1} \tag{9.9}$$

Now from Equation 4.11, the -3dB point of a filter, Ω, is given by $1/RC$. Using the prewarp formula of Equation 9.7, obtain Ω^*. Equation 9.9 therefore becomes

$$H(z) = \frac{1}{a\left[\frac{1-z^{-1}}{1+z^{-1}}\right]+1} \tag{9.10}$$

where

$$a = \frac{2}{\Omega^* T} \tag{9.11}$$

Rationalise Equation 9.10 by multiplying the denominator and numerator by $1+z^{-1}$. This yields

$$H(z) = \frac{1+z^{-1}}{a(1-z^{-1})+1+z^{-1}} \tag{9.12}$$

The objective now must be to group the z terms, since these determine the nature of the difference equation (remember that multiplying a signal by z^{-1} is equivalent to delaying it by one sampling interval). Therefore Equation 9.12 becomes

$$H(z) = \frac{1+z^{-1}}{(1+a)+z^{-1}(1-a)} \tag{9.13}$$

For the purposes of gain normalisation, the first term in the denominator must be unity; in addition, if we say that

$$b = (1+a)^{-1}$$
$$c = (1-a)/(1+a) \tag{9.14}$$

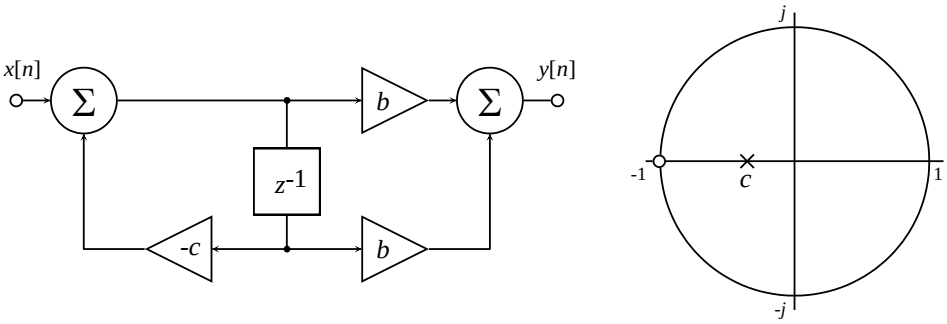
Then Equation 9.13 becomes

$$H(z) = \frac{b+bz^{-1}}{1+cz^{-1}} = \frac{b(1+z^{-1})}{1+cz^{-1}} \tag{9.15}$$

Equation 9.15 allows us to write the difference equation (also known as the recurrence formula) directly, and is given by

$$y[n] = -cy[n-1] + bx[n] + bx[n-1]$$
$$= -cy[n-1] + b(x[n] + x[n-1]) \tag{9.16}$$

Note the change in sign of the recursive coefficient *c*, as we discussed earlier. This filter may be represented in block-diagram form as shown in Figure 9.2, following the convention we outlined in Chapter 6. The zero of a 1st order low pass filter, from Equation 9.15, is purely real and is given by -1. The pole is also real and is given by *–c* (it should be noted that *c* will always evaluate as negative for a low-pass filter, so the coefficient is actually a real, positive number). A typical pole-zero plot for such a filter is given in Figure 9.2b.



(a) (b)

Figure 9.2. (a) IIR filter diagram and (b) pole-zero plot of the simple low-pass filter given in Equation 9.16.

A program called *Passive_filter.exe* is shown in Figure 9.3 which uses the BZT to obtain the difference equations of a simple passive filters comprising resistors, capacitors and inductors. It uses these equations in IIR filter routines to compute their associated impulse and frequency responses.

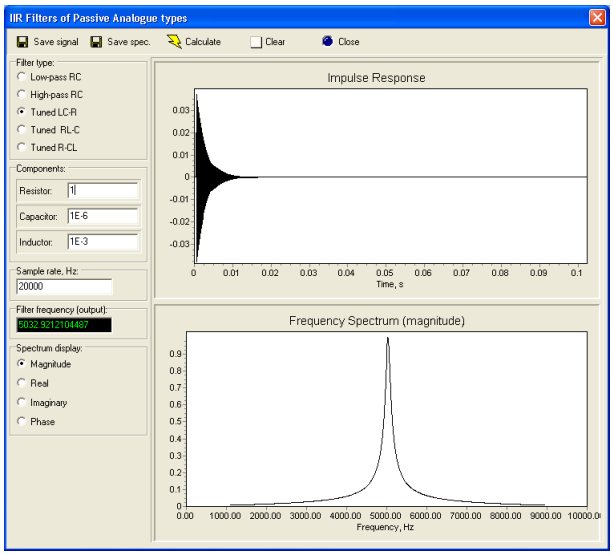


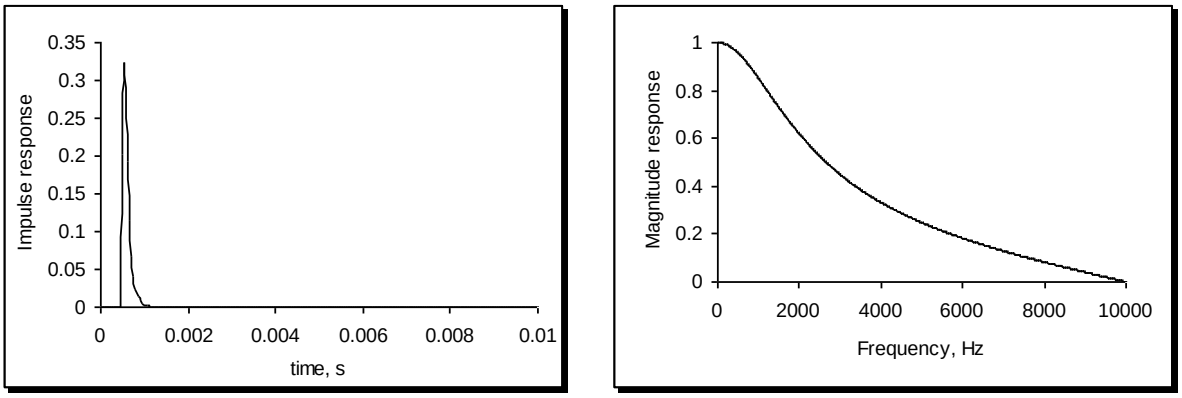
Figure 9.3. Screenshot of *Passive_filter.exe*.

The code for the impulse response of the 1st order low-pass *RC* filter is given in Listing 9.1.

```
x[5]:=1;
.
.
omega:=1/(res*cap);
fc:=omega/(2*pi);
edit5.Text:=floattostr(fc);
omega1:=(2/t)*tan(omega*t/2);
a:=2/(omega1*t);
b:=1/(1+a);
c:=(1-a)/(1+a);
for n:=5 to N1-1 do y[n]:=b*x[n]+b*x[n-1]-c*y[n-1];
```

Listing 9.1.

The array holding the input signal, `x[n]`, is set to zero (elsewhere in the code), and a single impulse is loaded into `x[5]`. Next, the cut-off frequency omega is calculated from an input obtained elsewhere in the program, and this is used to compute the prewarp cut-off, given by omega1. The final four lines of Listing 9.1 employ Equations 9.11, 9.14 and 9.16 directly. The output signal `y[n]`, i.e. the impulse response, is used as an input to an FFT routine later in the program, and both are plotted in their respective graphical output windows. Figures 9.4a and 9.4b show respectively the impulse and frequency responses obtained from an IIR filter with R = 1 kΩ, C = 0.1 μF and a sample rate of 20 kHz. The (analogue) -3dB (or 0.7071 in linear scale) point of such a filter is 1591.55 Hz.



(a) (b)

Figure 9.4. (a) Impulse and (b) frequency response of simple low-pass *RC* filter produced using the IIR program *Passive_filter.exe* (impulse response has been delayed for purposes of clarity).

Now that we have established the basic mechanism of the BZT, it would be worth trying some examples just to fix the method firmly in our minds.

Example 9.1

Using the BZT, obtain the discrete transfer difference function of the 1st order high-pass *RC* filter shown in Figure 9.5. Express this as a difference equation suitable for computer processing.

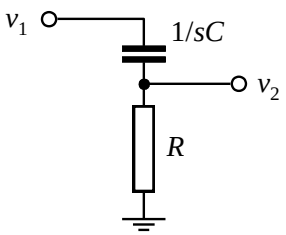


Figure 9.5. 1st order *RC* high-pass.

Solution 9.1

The continuous-time Laplace transfer function of this filter is given by

$$H(s) = \frac{sRC}{sRC + 1} \tag{9.17}$$

Before applying pre-warp, the BZT substitution yields

$$H(z) = \frac{\frac{2RC}{T}\left[\frac{1 - z^{-1}}{1 + z^{-1}}\right]}{\frac{2RC}{T}\left[\frac{1 - z^{-1}}{1 + z^{-1}}\right] + 1} \tag{9.18}$$

As with the low-pass filter, $\Omega = 1/RC$ and $a = \frac{2}{\Omega * T}$, where Ω* is obtained from Equation 9.7. Equation 9.18 therefore becomes

$$H(z) = \frac{a\left[\frac{1 - z^{-1}}{1 + z^{-1}}\right]}{a\left[\frac{1 - z^{-1}}{1 + z^{-1}}\right] + 1} \tag{9.19}$$

Rationalising using $1+z^{-1}$ and grouping the z terms gives

$$H(z) = \frac{a - az^{-1}}{(1 + a) + (1 - a)z^{-1}} \tag{9.20}$$

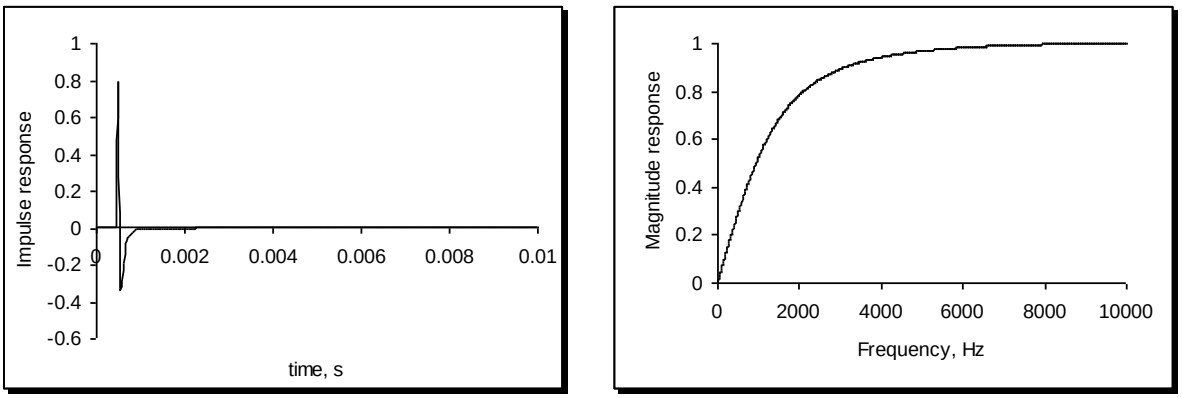
If $b = a/(1+a)$ and $c = (1-a)/(1+a)$ then Equation 9.20 becomes

$$H(z) = \frac{b - bz^{-1}}{1 + cz^{-1}} \tag{9.21}$$

Hence the difference equation is given by

$$y[n] = b(x[n] - x[n-1]) - cy[n-1] \tag{9.22}$$

The program above has provision for calculating the impulse and frequency response of this IIR filter; the code is very similar to that of the low-pass version discussed above. An impulse and frequency response of this filter, using the same component values as in the low-pass example, is depicted in Figure 9.6. In this case, the zero is given by 1 and the pole by –*c* (which is again negative).



(a) (b)

Figure 9.6. (a) Impulse and (b) frequency response of a simple high-pass *RC* filter produced using the IIR program *Passive_filter.exe*.

9.3 The BZT and 2nd order passive systems

The program *Passive_filter.exe* can also deal with 2nd order filters, as shown in Figure 9.7. These we will call *LCR*, *RLC* and *RCL*.

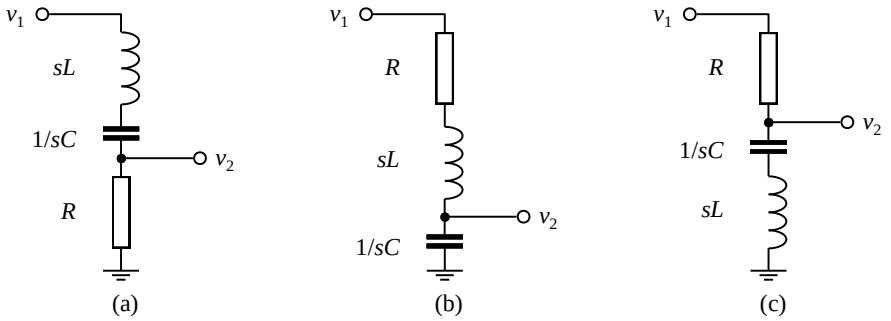


Figure 9.7. (a) *LCR*, (b) *RLC* and (c) *RCL* filters.

In all cases, the resonant point of the filters is given by

$$\Omega = \frac{1}{\sqrt{LC}} \tag{9.23}$$

It is important to know this because it is from Ω that we obtain the prewarp frequency Ω*. Let's start by finding the z-transfer function of the first case, the *LCR* filter. Its Laplace transfer function is of course

$$H(s) = \frac{sRC}{s^2LC + sRC + 1} \tag{9.24}$$

Using the BZT substitution, we obtain

$$H(z) = \frac{\frac{2}{T}RC\left[\frac{1-z^{-1}}{1+z^{-1}}\right]}{\left(\frac{2}{T}\right)^2 LC\left[\frac{1-z^{-1}}{1+z^{-1}}\right]^2 + \frac{2}{T}RC\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1} \tag{9.25}$$

Using the identities

$$a = 2\frac{RC}{T} \qquad b = \left(\frac{2}{T\Omega *}\right)^2 \tag{9.26}$$

and substituting into Equation 9.25, we obtain

$$H(z) = \frac{a\left[\frac{1-z^{-1}}{1+z^{-1}}\right]}{b\left[\frac{1-z^{-1}}{1+z^{-1}}\right]\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + a\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1} \tag{9.27}$$

Once again, the objective of the algebraic manipulation must be to express the transfer function as the ratio of two polynomials in *z*. Multiplication of both the numerator and denominator of Equation 9.27 by $(1 + z^{-1})^2$ and grouping the *z* terms produces

$$H(z) = \frac{a - az^{-2}}{(a+b+1) + (2-2b)z^{-1} + (b-a+1)z^{-2}} \tag{9.28}$$

For the purposes of filter gain normalisation, the constant term in the denominator must be unity; to simplify the algebra, we make use of the following identities:

$$\varepsilon_0 = \frac{a}{(a+b+1)} \qquad \varepsilon_1 = \frac{(2-2b)}{(a+b+1)} \qquad \varepsilon_2 = \frac{(b-a+1)}{(a+b+1)} \tag{9.29}$$

Equation 9.28 therefore reduces to

$$H(z) = \frac{\varepsilon_0(1-z^{-2})}{1 + \varepsilon_1 z^{-1} + \varepsilon_2 z^{-2}} \tag{9.30}$$

Finally, the difference equation of the IIR filter is obtained directly from Equation 9.30, i.e.

$$y[n] = \varepsilon_0 x[n] - \varepsilon_0 x[n-2] - \varepsilon_1 y[n-1] - \varepsilon_2 y[n-2] \qquad (9.31)$$

Listing 9.2 shows the fragment of code from *Passive_filter.exe* that generates the IIR filter and associated frequency response, using this *LCR* filter. If you study it, you will find that it makes use of Equations 9.7, 9.26, 9.29 and 9.31.

```
x[5]:=1;
.
.
omega:=1/(sqrt(ind*cap));
omega1:=(2/t)*tan(omega*t/2);
fc:=omega/(2*pi);
edit5.Text:=floattostr(fc);
a:=(2/t)*res*cap;
b:=sqr(2/(t*omega1));
c:=a/(a+b+1);
d:=(2-2*b)/(a+b+1);
e:=(1+b-a)/(a+b+1);
for n:=5 to N1-1 do y[n]:=c*x[n]-c*x[n-2]-d*y[n-1]-e*y[n-2];
```

Listing 9.2.

Figure 9.8 shows an IIR output produced by the program for the discrete equivalent of the *LCR* filter; here, R = 2 ohms, C = 1μF and L = 1mH, with a sample rate of 20 kHz. The program uses Equation 9.23 to calculate the resonance point of the analogue filter, which is 5.033 kHz. Figure 9.8b shows that the IIR equivalent does indeed peak precisely at this value.

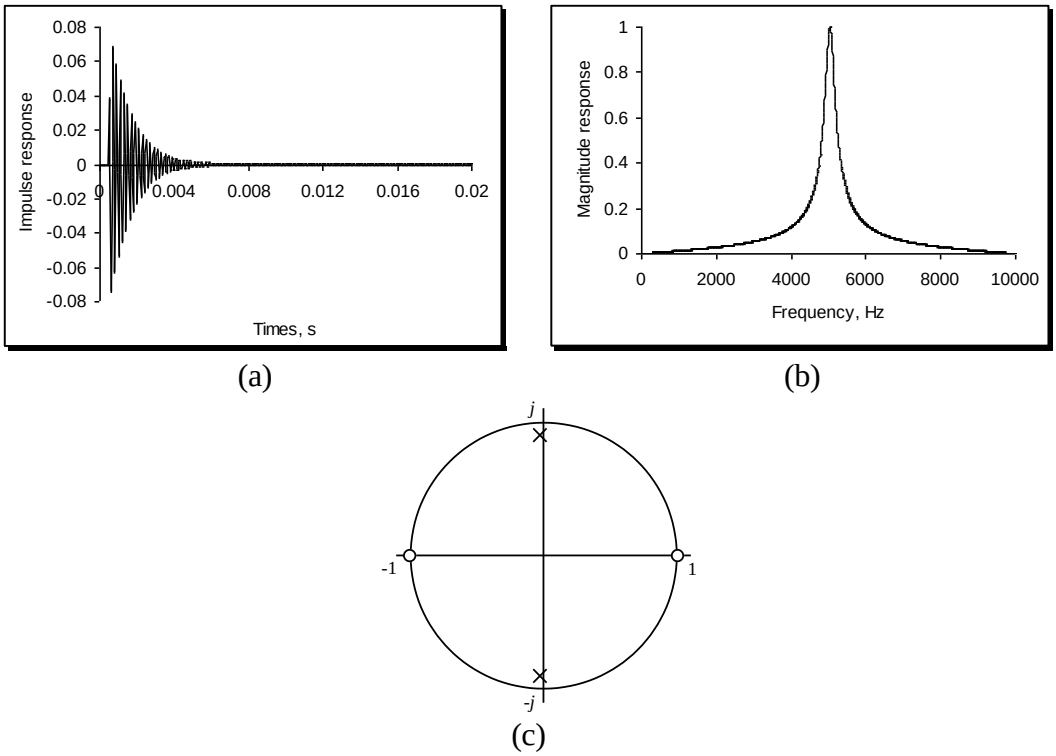


Figure 9.8. (a) impulse and (b) frequency response of *LCR* filter designed by the IIR filter program *Passive_filter.exe*; (c) its pole-zero diagram.

By inspecting the circuit diagram for this filter, Figure 9.7a, we may deduce that it must have zero gain both at DC and at infinite frequency. Analysis of its *z*-transfer function, Equation 9.30, corroborates this since the zeros of the filter, α_0 and α_1, are always at ± 1, regardless of the component values. It also has two poles, since it is a second order system, and if we look at the denominator of the *z*-transfer function we see that the poles are given by

$$\beta_0 = \frac{-\varepsilon_1 + \sqrt{\varepsilon_1^2 - 4\varepsilon_2}}{2} \qquad \beta_1 = \frac{-\varepsilon_1 - \sqrt{\varepsilon_1^2 - 4\varepsilon_2}}{2} \tag{9.32}$$

The angle that the poles subtend to the origin will always correspond to the resonance point; however, their radii (remember that they are conjugates) will depend on the degree of damping of the system, i.e. they will be controlled by the value of *R*. Figure 9.8c shows a typical pole-zero plot for this kind of filter. Having established the method for obtaining the IIR difference equation for the *LCR* filter, it is a straightforward procedure to derive the same for the *RLC* and *RCL* variants.

Example 9.2

a. Obtain IIR difference equations for the *RLC* circuit shown in Figure 9.7b.
b. Obtain an expression for its poles and zeros.

Solution 9.2

Part a.

The Laplace transfer function of the *RLC* circuit is given by

$$H(s) = \frac{1}{s^2 LC + sRC + 1} \tag{9.33}$$

Using the BZT substitution, the *z*-transfer function before prewarping is given by

$$H(z) = \frac{1}{\left(\frac{2}{T}\right)^2 LC\left[\frac{1-z^{-1}}{1+z^{-1}}\right]^2 + \frac{2}{T}RC\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1} \tag{9.34}$$

Using the identities given in Equation 9.26, we obtain

$$H(z) = \frac{1}{b\left[\frac{1-z^{-1}}{1+z^{-1}}\right]^2 + a\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1} \tag{9.35}$$

after rationalising, this gives

$$H(z) = \frac{\varepsilon_0(1 + 2z^{-1} + z^{-2})}{1 + \varepsilon_1 z^{-1} + \varepsilon_2 z^{-2}} \tag{9.36}$$

where

$$\varepsilon_0 = \frac{1}{(a+b+1)} \qquad \varepsilon_1 = \frac{(2-2b)}{(a+b+1)} \qquad \varepsilon_2 = \frac{(b-a+1)}{(a+b+1)} \tag{9.37}$$

the IIR difference equation is therefore

$$y[n] = \varepsilon_0(x[n] + 2x[n-1] + x[n-2]) - \varepsilon_1 y[n-1] - \varepsilon_2 y[n-2] \tag{9.38}$$

Part b.

The expression for the poles is the same as that of the *LCR* circuit, given by Equation 9.32. Both the zeros are -1, i.e. they are given by

$$\alpha_0 = \frac{-2\varepsilon_0 + \sqrt{4\varepsilon_0^2 - 4\varepsilon_0^2}}{2\varepsilon_0} \qquad \alpha_1 = \frac{-2\varepsilon_0 - \sqrt{4\varepsilon_0^2 - 4\varepsilon_0^2}}{2\varepsilon_0} \tag{9.39}$$

Both the *LCR* and *RLC* circuits are essentially tuned band-pass filters; the major difference between them is that the latter does not block the DC level. You can deduce this from the component arrangement in Figure 9.7b – there is a direct current path from the source to the output.

The final filter we wish to discuss here is the *RCL* type, which has a Laplace transfer function of

$$H(s) = \frac{s^2 LC + 1}{s^2 LC + sRC + 1} \tag{9.40}$$

a *z*-transfer function of

$$H(z) = \frac{\varepsilon_0 + \varepsilon_1 z^{-1} + \varepsilon_0 z^{-2}}{1 + \varepsilon_1 z^{-1} + \varepsilon_2 z^{-2}} \tag{9.41}$$

and an IIR difference equation of

$$y[n] = \varepsilon_0 x[n] + \varepsilon_1 x[n-1] + \varepsilon_0 x[n-2] - \varepsilon_1 y[n-1] - \varepsilon_2 y[n-2] \tag{9.42}$$

where

$$\varepsilon_0 = \frac{(b+1)}{(a+b+1)} \qquad \varepsilon_1 = \frac{(2-2b)}{(a+b+1)} \qquad \varepsilon_2 = \frac{(b-a+1)}{(a+b+1)} \tag{9.43}$$

This is a tuned notch filter, and as you might expect, the selectivity of the notch increases as the value of the resistor in the circuit falls. Figure 9.9 shows frequency responses for both the *RLC* and *RCL* filters calculated using the IIR difference equations and produced by the program *Passive-filter.exe*.

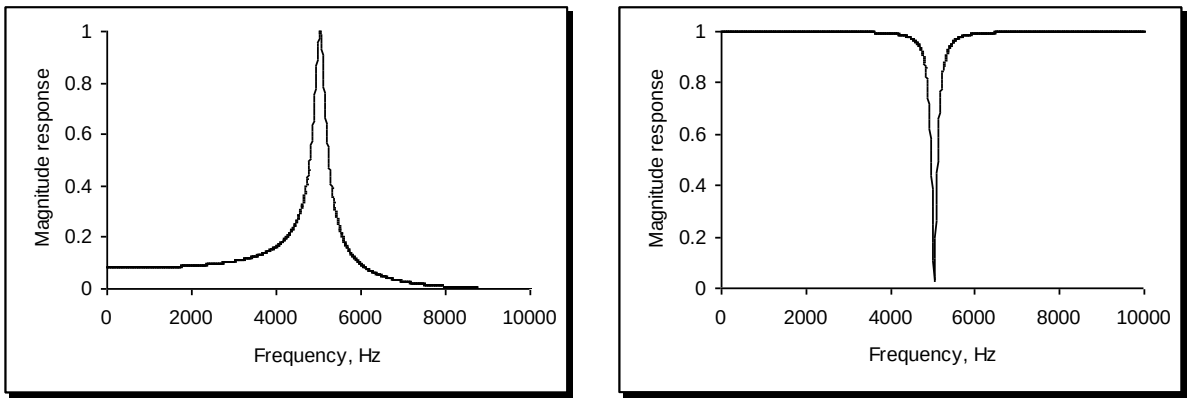


Figure 9.9. Frequency responses for the (a) *RLC* and (b) *RCL* IIR filters.

Second order IIR systems can normally be represented using the filter block diagram shown in Figure 9.10, the direct form II representation. This is also often referred to as a 2nd order canonic structure, and is very widely encountered in DSP filter design. To obviate problems with instability and unpredictable performance, high order IIR filters are often constructed from a series of cascaded 2nd order sections. This biquad design approach demands that we know the poles and zeros of a filter in advance, since the filter is expressed in factored form and a general purpose expression may therefore be employed for each stage. If we want to design IIR filters based on active analogue counterparts, it is imperative that we obtain the poles and zeros. Doing this from first principles can be a lengthy and error-prone task. Fortunately, standard equations are available that allow us to compute them without too much effort.

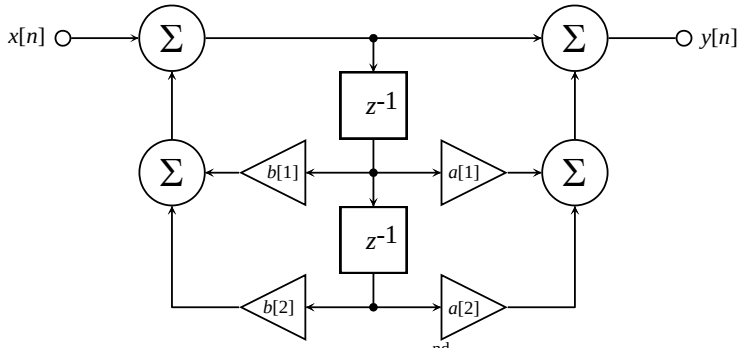


Figure 9.10. Direct form II or canonic 2nd order stage of an IIR filter.

9.4 Digital Butterworth and Chebyshev IIR filters

Whether a Butterworth or a Chebyshev is applied in a given circumstance depends upon a number of factors, including pass and stop band ripple, and transition zone performance. The Butterworth filter is maximally flat in the passband, i.e. it contains no ripples, and has a transition zone characteristic, as

we have discussed, of 6*n* dBs per octave, where *n* represents the order of the filter. In contrast, for a given order of filter the Chebyshev has a steeper transition zone; however, the price paid for this is that the Chebyshev manifests ripples in the passband. The magnitude of these ripples can be controlled by a ripple magnitude δ, and as we reduce their magnitude, so the transition zone widens. A typical Chebyshev frequency response is shown in Figure 9.11.

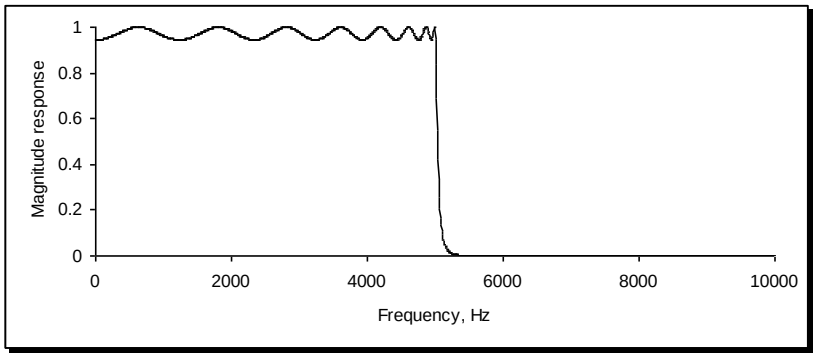


Figure 9.11. Typical low-pass Chebyshev frequency response.

9.4.1 The Butterworth low-pass filter in detail

Keeping to the convention we have established in this chapter, the frequency response for an n^{th} order analogue Butterworth low-pass filter is given by

$$\left|H(\Omega)\right| = \frac{1}{\sqrt{1+\left(\frac{\Omega}{\Omega_c}\right)^{2n}}} \tag{9.44}$$

Where Ω_c denotes the cut-off frequency. Prewarping dictates, therefore, that the frequency response of the equivalent digital Butterworth filter is expressed as

$$\left|H(\omega)\right| = \frac{1}{\sqrt{1+\left(\frac{\tan(\omega/2)}{\tan(\omega_c/2)}\right)^{2n}}} \tag{9.45}$$

where ω_c denotes the cut-off frequency, i.e. $2\pi f_c/T$. Now the poles, β, of such an analogue filter may be obtained from the expression (Jong, 1982)

$$\beta_k = \Omega e^{j\pi(2k+n-1)/2n}, \qquad k = 1,2,\ldots n \tag{9.46}$$

Furthermore, an n^{th} order Butterworth has *n* zeros all located at infinity. With considerable work, we could apply the BZT to transform the Laplace-space poles into their *z*-space equivalents. Fortunately, there is a set of equations that allows us to achieve this for any order of low-pass Butterworth filter:

$$\beta_{r,m} = d^{-1}\left[1-\tan^2\left(\frac{\omega_c}{2}\right)\right]; \qquad \beta_{i,m} = d^{-1}\left[2\tan\left(\frac{\omega_c}{2}\right)\sin\theta\right]$$

$$d = 1-2\tan\left(\frac{\omega_c}{2}\right)\cos\theta+\tan^2\left(\frac{\omega_c}{2}\right) \tag{9.47}$$

$$\theta = \pi m/n, \qquad n = \text{odd}$$

$$\theta = \pi(2m+1)/2n, \qquad n = \text{even}$$

Where $m = 0, 1\ldots(2n\text{-}1)$. If you study these equations, you will find that they generate $m = 2n$ poles, whereas we require only *n*. This is because the poles of such a filter lie on the locus of a circle, exactly half of which will be outside the unit circle of the *z*-plane. Therefore, when we calculate the *m* poles, we simply retain only those that are located within the unit circle. As for the zeros, an n^{th} order digital low-pass Butterworth filter has *n* zeros, all located at -1. A program is shown in Figure 9.12 which calculates the poles and zeros of a variety of low-pass, high-pass, band-pass and band-stop IIR filters for both Butterworth and Chebyshev families.

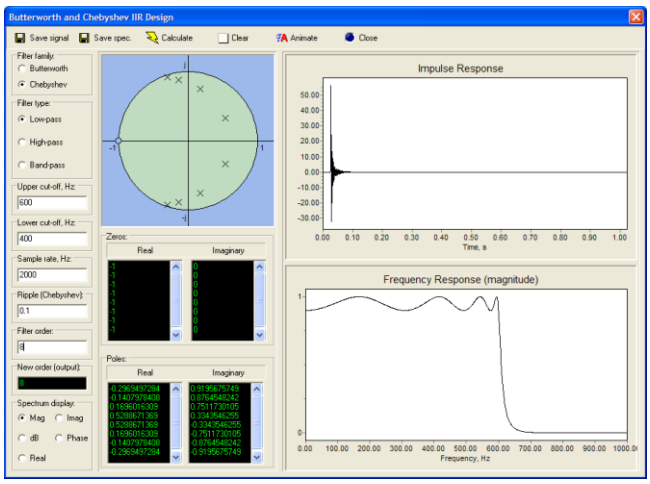


Figure 9.9. Screenshot of the program *active_filter.exe*.

This program obtains the response of a high-order filter by cascading together canonic 2nd order stages. To compute a Butterworth, the user enters the cut-off frequencies, the order required and the sample rate. The program does the rest. Figure 9.13 shows pole-zero plots for a range of such filters.

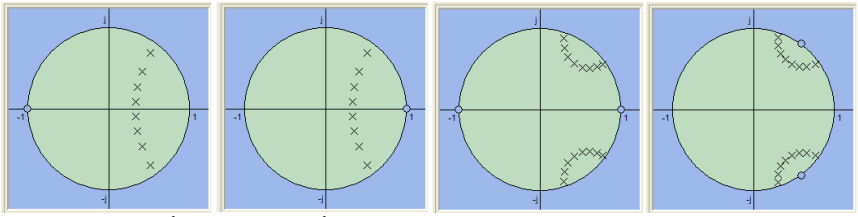


Figure 9.13. Pole-zero plots for 8th order and 16th order Butterworth filters; (a) low-pass, with a cut-off of 300 Hz; (b) high-pass, cut-off of 300 Hz, (c), band-pass, 200 – 400 Hz; (d) band-stop, 200 – 400 Hz. All sample rates at 2 kHz.

```
case (filter_type.ItemIndex) of
  0,1: begin
       if (filter_type.ItemIndex=0) then f:=fu else f:=fl;
       k:=n*2-1;
       for m:=0 to k do
       begin
         d1:=tan(pi*f*t);
         d2:=1-(2*d1*cos(pi*(2*m+1)/(2*n)))+(d1*d1);
         pole_r:=(1-d1*d1)/d2;
         pole_i:=(2*d1*sin(pi*(2*m+1)/(2*n)))/d2;
         if(sqrt(sqr(pole_r)+sqr(pole_i))<1) then
         begin
           if (filter_type.ItemIndex=0) then
           zero_real.lines.add('-1') else zero_real.lines.add('1');
           zero_imag.lines.add('0');
           pole_real.lines.add(floattostrf(pole_r,fffixed,10,10));
           pole_imag.lines.add(floattostrf(pole_i,fffixed,10,10));
         end;
       end;
     end;
```

Listing 9.3.

Listing 9.3 shows how the poles and zeros of a low-pass Butterworth filter are generated. In fact, it also calculates these for a high pass filter, covered later. Assume that the variable `filter_type.ItemIndex` is zero, since this informs the code a low-pass has been selected. You will see that the code follows Equation 9.47 precisely. Once a pole has been calculated, it checks that it lies within the unit circle – if it does, it enters it into string grids that store the real and imaginary values of the poles. It also fills string grids associated with the zeros with *n* values of -1,

0. The contents of these string grids are used later in the program in a canonic biquad cascade algorithm, defined by Equation 9.31.

9.4.2 The Chebyshev low-pass filter in detail

An n^{th} order Chebyshev analogue low-pass filter has a frequency response given by

$$|H(\Omega)| = \frac{1}{\sqrt{1+\varepsilon^2 C_n^2\left(\frac{\Omega}{\Omega_c}\right)}} \tag{9.48}$$

where ε is the "ripple parameter", related to the linear ripple magnitude, δ, in the pass band by

$$\varepsilon = \sqrt{\frac{1}{(1-\delta)^2} - 1} \tag{9.49}$$

and C_n is the Chebyshev polynomial of the n^{th} order.. After prewarping, the digital Chebyshev frequency response becomes

$$|H(\omega)| = \frac{1}{\sqrt{1+\varepsilon^2 C_n^2\left(\frac{\tan(\omega/2)}{\tan(\omega_c/2)}\right)}} \tag{9.50}$$

The equations which yield its poles are as follows:

$$\beta_{r,m} = 2d^{-1}\left[1 - a\tan\left(\frac{\omega_c}{2}\right)\cos\theta\right] - 1; \qquad \beta_{i,m} = 2bd^{-1}\left[2\tan\left(\frac{\omega_c}{2}\right)\sin\theta\right]$$

$$d = \left[1 - a\tan\left(\frac{\omega_c}{2}\right)\cos\theta\right]^2 + \left[b\tan\left(\frac{\omega_c}{2}\right)\sin\theta\right]^2$$

$$a = 0.5(c^{1/n} - c^{-1/n}); \qquad b = 0.5(c^{1/n} + c^{-1/n}); \qquad c = \sqrt{(1+\varepsilon^{-1}+\varepsilon^{-2})} \tag{9.51}$$

$$\theta = \pi m/n, \qquad n = \text{odd}$$

$$\theta = \pi(2m+1)/2n, \qquad n = \text{even}$$

Again, an n^{th} order low-pass Chebyshev has n zeros, all situated at -1. The code within the program that computes the Chebyshev response does so in much the same way as the code for the Butterworth filter; the only difference is that we substitute Equation 9.47 with Equation 9.51. Like the Butterworth algorithm, the equations will always generate $2n$ poles, and the system must reject those that lie outside of the unit circle. The poles of a Chebyshev filter lie on the locus of a cardioid rather than a circle. In addition, for the same order, the poles of a Chebyshev lie closer to the unit circle than do those of a Butterworth, so the filter has a sharper transition zone. Figure 9.14 shows pole-zero plots for different Chebyshev filters with a ripple factor of 0.1.

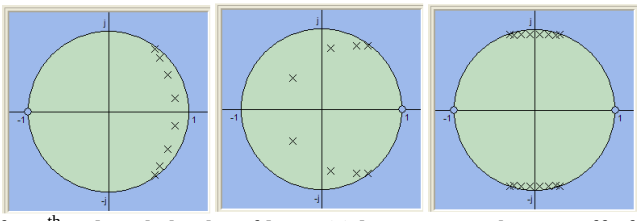


Figure 9.14. Pole-zero plots for 8th order Chebyshev filters; (a) low-pass, with a cut-off of 300 Hz; (b) high-pass, cut-off of 300 Hz, (c), band-pass, 400 – 600 Hz. All sample rates at 2 kHz and a ripple factor of 0.1.

9.4.3 Frequency transformations

IIR low-pass filters are significant, not just because we may wish to use them directly for filtering purposes, but because they form the basis of other types of filter, through a process termed *frequency transformation*. Typically, we compute the poles and zeros of what is called the *low-pass prototype*, and remap them using a system of equations to generate high-pass, band-pass and band-stop variants.

To remap a low-pass to a high-pass is simplicity itself. We take the zeros at -1 and place them at 1. If you return to the code shown in Listing 9.3, you can see that if `filter_type.ItemIndex` is set equal to 1, then the string grids associated with the zeros have *n* zeros added of value 1, 0. Note that the positions of the poles do not change – only the zeros. Figure 9.15 depicts an 8th order high-pass Chebyshev filter calculated in this manner.

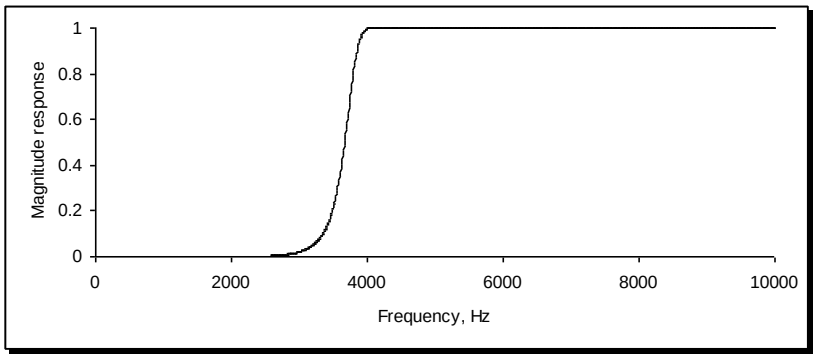


Figure 9.15. 8th order high-pass Chebyshev with a cut-off of 4 kHz, a ripple of 0.002 and a sample rate of 20 kHz.

To compute the poles and zeros of a band-pass filter with lower and upper cut-off frequencies of ω_1 and ω_2, we first generate those for a low-pass prototype with a cut-off of $\omega_c = \omega_2 - \omega_1$. Next, we take each of its poles and zeros, denoted generally by λ, and compute *two* new ones, using the transformation

$$z = 0.5A(1+\lambda) \pm \sqrt{0.25A^2(1+\lambda)^2 - \lambda}$$
$$A = \cos\left(\frac{\omega_2 + \omega_1}{2}\right) / \cos\left(\frac{\omega_2 - \omega_1}{2}\right) \qquad (9.52)$$

As a result of this transformation, the original n^{th} order prototype is converted to a $2n^{th}$ order band-pass filter. Again, this is easy to achieve using the program. In the code, there is a procedure called `transform` which performs the remapping, using Equation 9.52. Remember that λ is complex, so we need to use a routine that calculates the root of a complex number. We can use a similar remapping procedure to generate a band-stop filter. In this case, ω_c = p/T + ω_1 – ω_2. The transformation equation is

$$z = 0.5A(1-\lambda) \pm \sqrt{0.25A^2(1-\lambda)^2 + \lambda} \qquad (9.53)$$

with A as before. Unfortunately, in many instances this does not work very well. In general, it is harder to convert a low-pass prototype into an equivalent band-stop, mainly because it is harder to control the gain on either side of the stop-band region. Instead, Gaydecki (2004) describes a slightly different remapping is used for Butterworth band-stop filters, which performs excellently with precise gain control and perfect symmetry of the lower and upper transition zones. Here is how it works: first, a low-pass prototype is calculated with a cut-off frequency equal to

$$f_c = (f_2 - f_1)/2 \qquad (9.54)$$

Next, the centre frequency of the band-stop filter is calculated using

$$\theta = (\omega_1 + \omega_2)/2 \qquad (9.55)$$

Now, each pole is rotated around the unit circle by $\pm\theta$ using a rotation matrix thus:

$$\lambda_{r,0} = \beta_{r,0}\cos\theta - \beta_{i,0}\sin\theta \qquad \lambda_{i,0} = \beta_{i,0}\cos\theta + \beta_{r,0}\sin\theta$$
$$\lambda_{r,1} = \lambda_{r,0} \qquad \lambda_{i,1} = -\lambda_{i,0} \qquad (9.56)$$

Finally, 2*n* zeros are placed on the unit circle at θ, corresponding to the stop-band centre frequency. This is a simple technique, but an elegant one. You can see an example of how beautifully this works if you look at Figure 9.16. One of the reasons it is successful is because the poles of a Butterworth

filters lie on the locus of a circle, and their position with respect to the zeros dictate well-behaved pass and stop-band regions. For the same reason, it cannot be applied to Chebyshev filters.

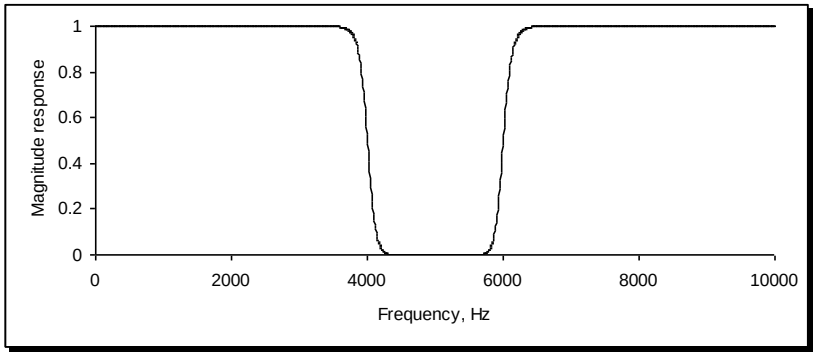


Figure 9.16. 16th order Butterworth stop-band filter obtained from a low-pass prototype and a rotation matrix (Equation 9.56). The cut-off region lies between 4 kHz and 6 kHz, and the sample rate is 20 kHz.

9.7 FIR expression of IIR responses

Previously we noted that an advantage of the IIR filter over the FIR was that, it is possible to design the digital equivalents of analogue filters. Although we saw how we could use the magnitude frequency response of an analogue filter to obtain an FIR filter with the same frequency response, it did not result in a filter with the same impulse response. There are two solutions to this problem. The first, and obvious solution, is to use a sampled version of the complex frequency response of the analogue filter and apply it to the frequency sampling method. The other is to store the impulse response produced by the IIR design software, and use this directly in an FIR convolution expression, rather than the IIR difference equation.

This may seem a rather pointless (not to say inefficient) thing to do, but if the processor can accommodate the conditional extra computational burden, there are several advantages to be gained from this method of processing. Not only is the risk of instability associated with high-order IIR designs completely negated, but more importantly, the effects of word-length sensitivity are mitigated considerably. FIR-to-IIR conversion is most useful in real-time applications, where, if the speed of the processor is sufficient, no degradation in performance will result.

9.8 Observations on IIR and FIR filters

9.8.1 Comparisons of efficiency

We have established that because of the existence of poles (in other words, feedback), IIR filters require fewer coefficients in the difference equation to elicit the same transition zone sharpness. Exactly how much more efficient are they? Well, the answer depends on a number of factors, including the filter family, the sample rate and word length, but it is possible to get an approximate idea by designing an FIR equivalent of a Butterworth filter, reducing the taps until the realised frequency response deviates significantly from the ideal. Figure 9.17 shows the ideal frequency response of a 10th order low-pass Butterworth as a bold curve, and two FIR equivalent using 63 taps and 127 taps (and a Hanning window in both cases). Clearly, there is a considerable difference between the ideal curve and the 63-tap FIR equivalent, in contrast to the FIR filter comprising 127 taps.

Now each biquad stage of an IIR filter requires a total of 5 MMACs, hence 25 in total for a 10th order filter. In this case therefore, we can say that approximately, the IIR is more efficient by a factor of five. This of course means that using this IIR filter we can process a real-time signal sampled five times faster than the limit imposed by the FIR design.

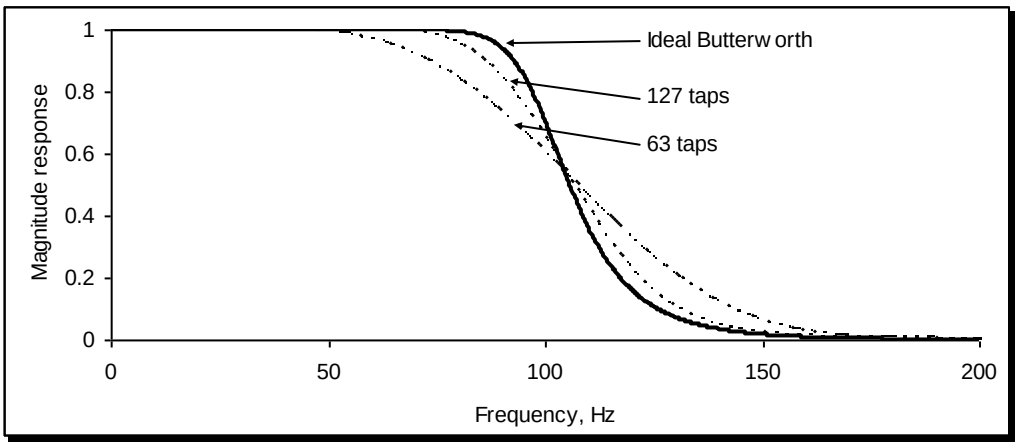


Figure 9.17. Ideal frequency response curve of a 10th order low-pass Butterworth filter and FIR equivalents using 63 and 127 taps.

9.8.2 Word length sensitivity

The closer a pole lies to the unit circle, the greater the potential for instability or deviations in the expected frequency response. This is particularly the case when using floating-point arithmetic on a PC to obtain the coefficients, and then converting them into fixed-point format to run on a real-time processor. The processor will normally have a lower resolution than the PC (typically 16 or 24 bits), and so the least significant bits of the values must be discarded upon conversion. This will lead to a slight repositioning of the poles and zeros, normally manifested in a degraded frequency response. Additionally, if normalisation has not been applied to the biquad stages, then they may generate values too large for the arithmetic registers of the processor, leading to overflow. In this case, the filter will fail completely. PC-based programs for IIR design exploit floating point arithmetic, so pre-scaling of the coefficients is required if they are to run on real-time fixed-point processors (which use integer arithmetic).

Because FIR filters contain only zeros in the transfer function, they are far more robust to changes in resolution with respect to the frequency response. Furthermore, gain normalisation is a very simple matter – all the taps of the impulse response are weighted by a constant factor to ensure overflow does not occur.

9.8.3 Phase

To gain an appreciation of the phase nonlinearity of IIR filters, it is necessary only to inspect their impulse responses, which of course display neither even nor odd symmetry. Now it is possible to overstate the case here, since in their pass bands both Butterworth and Chebyshev are pretty well behaved; moreover, in many applications the phase is not really relevant. However, the crucial issue is that with FIR designs, it is possible *explicitly* to control this parameter.