Chapter 6: An Introduction to *z*-Space, the *z*-Transform and Digital Filter Design

6.1 Introduction

The *z*-transform is the discrete-space equivalent of the Laplace transform. This transform is most widely applied in the analysis and design of IIR filters, also termed recursive filters because of the feedback nature of their operation. Although it shares many similarities with the Laplace transform, there are certain distinctions between these two transforms of which we should be familiar in order that we can use them effectively. These differences arise due to a seemingly obvious but fundamental reason: because discrete signals must be sampled at some finite rate, they must by definition be band limited. Consequently, any transform that describes a filter system operating in discrete space must be bound by this same limitation. Before we can see what this means in practice, we need to provide a mathematical framework for the subject.

6.2 The *z*-transform: definitions and properties

6.2.1 The *z*-transform and its relationship to the Laplace transform

The *z*-transform is similar in *form* to the DFT, and for a discrete signal $x[n]$, the unilateral version is given by

$$X(z) = Z\{x[n]\} = \sum_{n=0}^{\infty} x[n]z^{-n} \tag{6.1}$$

Where Z denotes the *z*-transform. The bilateral transform is similar, but with n ranging from $\pm \infty$. Because we are dealing with causal systems, the unilateral manifestation is sufficient for our purposes. As with the Fourier transform and the Laplace transform, the original discrete-space time-domain signal may be recovered from the z-transformed signal, using the inverse z-transform thus:

$$x[n] = Z^{-1}\{X(z)\} \tag{6.2}$$

Where Z^{-1} denotes the inverse *z*-transform. Although the variable *z* is more often than not manipulated as a simple algebraic symbol, it is important to remember that it is a complex variable, rather like the Laplace operator *s*. However, an important difference between the Laplace transform and the *z*-transform is that, whilst the former maps the complex frequency space using Cartesian coordinates (ie σ and $j\omega$), the latter employs polar coordinates, where any point is defined by its distance (magnitude) from the origin, and the angle (frequency) that it subtends to it. The relationship between *s* and *z* is therefore given by:

$$z = e^{\sigma T} e^{j\omega T} = e^{(\sigma + j\omega)T} = e^{sT} \tag{6.3}$$

Hence

$$z^{-1} = e^{-\sigma T} e^{-j\omega T} = e^{-(\sigma + j\omega)T} = e^{-sT} \tag{6.4}$$

where T represents the sample period. Furthermore, the polar coordinates of a point $X(r, \Phi)$ in *z*-space are given by

$$\begin{aligned} r &= |z| = e^{\sigma T} \\ \Phi &= \angle z = \omega T \\ \sigma &= \frac{\ln r}{T} \end{aligned} \tag{6.5}$$

Using Equations 6.3 to 6.5, it is possible to map a point in Laplace space into *z*-space . Now this latter domain is usually represented by the *unit circle*, so-called because it has a radius of 1, shown in Figure 6.1. As with Laplace space, the horizontal axis is real and the vertical axis is imaginary. Any point on the circumference on the unit circle denotes a frequency, the angle of which is given by ωT. For example, assume we are sampling a signal at 10 Hz, i.e. T = 0.1 s, and we wish to represent frequencies at 1.25 Hz, 2.5 Hz, 3.75 Hz, 5 Hz and 6.25 Hz. These are indicated by points A to E in Figure 6.1, and Table 6.1.

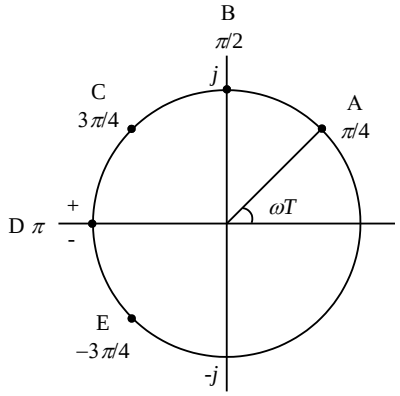


Figure 6.1. The unit circle of *z*-space with five frequency points located on its circumference. The frequency of any given point is determined from the angle ωT that it subtends to the origin, where T is the sample period.

Point on unit circle	Frequency, Hz	Frequency, ω	Angle, ωT, where T = 0.1
A	1.25	7.854	$\pi/4$
B	2.5	15.708	$\pi/2$
C	3.75	23.562	$3\pi/4$
D	5	31.416	π
E	6.25 (3.75)	23.562	$-3\pi/4$

Table 6.1. Frequencies points on the unit circle denoted by angle, where T = 0.1 (see Figure 6.1.).

We know that the highest frequency that can faithfully be represented by a signal sampled at f Hz is $f/2$ Hz. Therefore with the unit circle, the Nyquist point is denoted by π. As a result, although point E represents a frequency of 6.25 Hz and therefore an angle of $5\pi/4$, in fact if the system in our example attempted to digitise it, it would appear as a frequency of 3.75 Hz (ie as point C), with an angle of $-3\pi/4$.

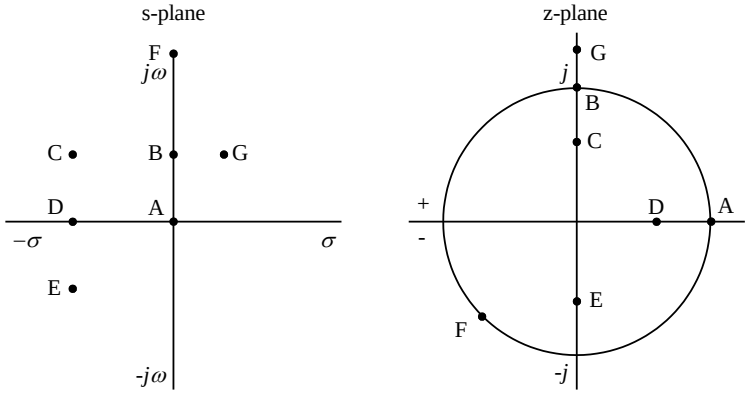


Figure 6.2. Locations on the *s*-plane and their equivalent positions on the *z*-plane.

Point	*s*-plane, Cartesian Coordinates		*z*-plane, polar coordinates		*z*-plane, Cartesian coordinates	
	σ	ω	|z|	$\angle z$	Real	Imaginary
A	0	0	1	0	1	0
B	0	15.707	1	$\pi/2$	0	1
C	-5	15.707	0.606	$\pi/2$	0	0.606
D	-5	0	0.606	0	0.606	0
E	-5	-15.707	0.606	$-\pi/2$	0	-0.606
F	0	39.27	1	$-3\pi/4$	-0.707	-0.707
G	2.5	15.707	1.284	$\pi/2$	0	1.284

Table 6.2. Points in the s-plane and their equivalent positions in the z-plane, for T =0. 1 (see Figure 6.2).

This brings us to an important distinction between the *s*-plane and the *z*-plane. The *s*-plane maps continuous signals, and is therefore not associated with aliasing. In contrast, the unit circle is essentially a periodic mapping technique that describes discrete signals sampled at a finite frequency. Therefore, frequencies that proceed past the Nyquist point simply continue to rotate around the unit circle. This has significant consequences for filter design methodologies that exploit the mapping of *s*-space into equivalent *z*-space. Take a look at Figure 6.2 and Table 6.2. Figure 6.2a depicts the *s*-plane, together with a number of marked positions from A to F. Figure 6.2b shows these points in their equivalent positions in *z*-space. The re-mapping is performed using Equation 6.5. Table 6.2 provides their coordinates in the *s*-plane, and their equivalent polar and Cartesian coordinates in the *z*-plane. Remember that to convert a coordinate from polar to Cartesian form, we simply use the formula:

$$\begin{aligned} re &= e^{\sigma T}\cos(\omega T) \\ im &= e^{\sigma T}\sin(\omega T) \end{aligned} \tag{6.6}$$

If a point moves upwards along the vertical axis on the *s*-plane, it represents an increase in frequency. In contrast, the *z*-plane represents an increase in frequency by the same point moving in a counter-clockwise direction around the unit circle. Point A in Figure 6.2a, for example, which is located at the origins of both axes in the *s*-plane (i.e. zero σ and ω) is located, on the *z*-plane, on the horizontal (real) axis with a magnitude of 1 (Figure 6.2b). Furthermore, a point in the *s*-plane which is non-zero but purely imaginary, such as point B, will lie at position B on the unit circle. Points C, D and E are similarly remapped into the z-plane using the appropriate conversion formulae. Now consider point F. Again this is purely imaginary, with a frequency of 6.25 Hz, or 39.27 radians. In the *s*-plane, this simply appears at a position higher up the frequency axis. However, in the discrete case, in the context of this example where $T = 0.1$ s, this frequency is aliased downwards. From this we can conclude that although the *s*-plane always maps points uniquely, the z-plane does not necessarily do so. Another, and crucial, property of the unit circle is the manner in which it represents instability. Any discrete system that bears a pole with a magnitude greater than 1 will be unstable.

6.2.2 The z-transform as a power series and its role as a delay operator

Imagine we have a causal signal $x[n]$, given by $x[n] = 5, -2, 3, 7, -1$ for $n = 0...4$. Then the *z*-transform is given by

$$\begin{aligned} X(z) &= \sum_{n=0}^{4} x[n]z^{-n} = x[0]z^{0} + x[1]z^{-1} + x[2]z^{-2} + x[3]z^{-3} + x[4]z^{-4} \\ &= 5 - 2z^{-1} + 3z^{-2} + 7z^{-3} - z^{-4} \end{aligned} \tag{6.7}$$

An observation is that *z* is essential a *time shift operator*. Multiplication by *z* advances a signal point by one sample interval, and multiplication by z^{-1} delays it by the same amount. There is a very simple way to demonstrate the time-shifting property of the *z*-transform; simply apply it to an impulse function and a delayed impulse function.

Example 6.1

Find the *z*-transforms of the following:

a. The impulse function $\delta[n]$.

b. The weighted, delayed impulse function $A\delta[n\text{-}3]$.

Solution 6.1

a. For the isolated impulse function, the *z*-transform is given by

$$X(z) = \sum_{n=0}^{\infty} x[n]z^{-n}$$

$$X(z) = (1)z^{-0} + (0)z^{-1} + (0)z^{-2} + \ldots$$

$$\therefore \; X(z) = 1$$

b. For the weighted delayed impulse function, the *z*-transform is

$$X(z) = \sum_{n=0}^{\infty} x[n]z^{-n} = (0)z^{-0} + (0)z^{-1} + (0)z^{-2} + Az^{-3} + (0)z^{-4}\ldots$$

$$\therefore \quad X(z) = Az^{-3}$$

Incidentally, part (b) of the solution above also confirms that the *z*-transform, like the other transforms we have discussed, is a linear operator, i.e.

$$\sum_{n=0}^{\infty} kx[n]z^{-n} = k\sum_{n=0}^{\infty} x[n]z^{-n} \tag{6.8}$$

6.3. Digital filters, diagrams and the *z* transfer function

6.3.1 Digital filter processing blocks

There are two main kinds of linear digital filter that operate in the time domain; the FIR and the IIR variety. There are only ever three operations that a processor uses to execute them: time shift, multiplication and addition. Because these three operations are of such pivotal importance not just to digital filters but to the whole of DSP in general, all real-time DSP devices are designed to execute them as fast as possible.

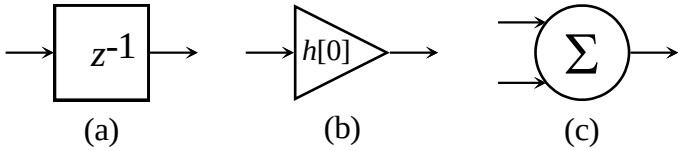


Figure 6.3. Filter block operations; (a) delay input by one sample interval; (b) multiply input by term; (c) sum inputs.

A digital filter may be represented as an equation, sometimes called a *difference formula*, or it may also be depicted as a diagram, the visual nature of which often aids in the understanding of the flow of signals and coefficients, and hence how the filter operates. Filter block diagrams, as they are called, portray the three key operations of digital filters as shown by Figure 6.3. A square enclosing the symbol z^{-1} (or sometimes T) denotes that a delay operation is applied to the incoming signal (Figure 6.3a). A triangle enclosing a symbol or number indicates that the incoming signal is multiplied by this term or value (Figure 6.3b). Finally, addition or summation is represented by two or more input signals applied to a circle enclosing a Σ symbol (the + sign is also widely used).

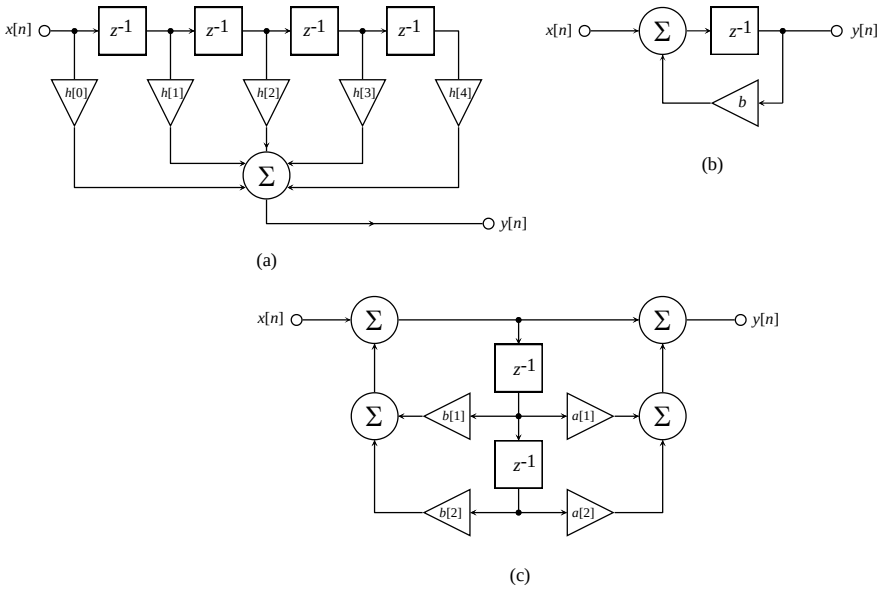


Figure 6.4. (a) FIR, (b) purely recursive and (c) typical IIR filter block diagrams. The final form shown is also a canonic biquad IIR filter stage, or direct form II representation (see text).

The FIR filter employs an impulse response of finite length, and the output is obtained simply by convolving this with the incoming signal. Consequently, there is no *feedback* in the system. Such filters, are typically visualised as shown in Figure 6.4a. This depicts a 5-point FIR structure. In contrast, if the filter is of the IIR type, it *must* include feedback. Figure 6.4b shows a very simple IIR filter that feeds back a fraction *b* of the output, which is then summed with the next input value to produce a new output value. This filter is purely recursive, since it does not involve any convolution with multiple input signal values, i.e. it has no transversal path. Most IIR filters, however, comprise a recursive and non-recursive part, one such diagram being shown in Figure 6.4c. The right hand side of the diagram denotes the transversal or feed-forward section, involving coefficients *a*[1] and *a*[2], and the left-hand side represents the recursive part, with coefficients *b*[1] and *b*[2]. In general with this kind of IIR filter, it is considered that the signal is first processed by the recursive section, and the processed result is fed back into the transversal, or FIR section. This is also known as a Direct Form II implementation.

6.3.2 Details of difference equations and the *z* transfer function

The convolution equations or difference formula for non-recursive FIR and recursive IIR systems have been studied earlier. To recapitulate, the non-recursive FIR discrete-time equation is given by

$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k] \tag{6.9}$$

In contrast, the IIR version involves a feedback structure and is therefore given by

$$y[n] = \sum_{k=0}^{\infty} h[k]x[n-k] = \sum_{k=0}^{M} a[k]x[n-k] - \sum_{k=1}^{N} b[k]y[n-k] \tag{6.10}$$

In the case of Equation 6.9, the impulse response is time limited, i.e. it comprises *M* coefficients, or taps. However, Equation 6.10 reveals that although there is a finite number (*M* + 1) of transversal coefficients and a finite number *N* of recursive coefficients, because of feedback the impulse response is infinite, or eternal.

As with the Fourier transform and the Laplace transform, the principle of duality operates between the time domain the *z* domain. Therefore, just as the output signal *y*[*n*] can be calculated by convolving the input signal *x*[*n*] with the impulse response *h*[*n*], so to the *z*-transform of the output signal is obtained by multiplying the z-transform of *x*[n] and *h*[*n*], i.e.

$$Y(z) = H(z)X(z) = Z\{h[n]*x[n]\} \tag{6.11}$$

Now the z-transform of *h*[*n*] is known as the *z*-transfer function of the system or sometimes just the transfer function, and is written as *H*(*z*). It is a most important property, and a critical goal, particularly in the design of IIR filters. Rearranging Equation 6.11, we find that in general, the transfer function is given by

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{n=0}^{\infty} h[n]z^{-n} \tag{6.12}$$

If we are dealing with an FIR filter, then the transfer function is simply a single polynomial of z^{-1}, i.e.

$$H(z) = \sum_{n=0}^{M-1} h[n]z^{-n} = h[0] + h[1]z^{-1} + h[2]z^{-2} + \ldots + h[M-1]z^{-(M-1)} \tag{6.13}$$

In contrast, the transfer function of an IIR filter involves the ratio of two polynomials (Ifeachor and Jervis, 1993) thus:

$$H(z) = \frac{a[0] + a[1]z^{-1} + \ldots + a[m]z^{-m}}{1 + b[1]z^{-1} + \ldots + b[n]z^{-n}} = \frac{\sum_{m=0}^{M} a[m]z^{-m}}{1 + \sum_{n=1}^{N} b[n]z^{-n}} \tag{6.14}$$

In Equation 6.14 the *numerator polynomial* represents the non-recursive, or FIR part of the filter, and the *denominator polynomial* represents the recursive part. This can be verified with respect to

Equation 6.13; if there is no feedback, then the denominator becomes 1 and we are left with a single polynomial. The second feature of interest is the change in sign (+) in the denominator, respecting the $y[n]$ coefficients, $b[n]$; in Equation 6.10, the $y[n]$ terms are shown as negative. This may readily be understood using the following reasoning. Say we have a recursive system, given by

$$\begin{aligned} &y[n] = a[0]x[n] + a[1]x[n-1] + a[2][x[n-2] \\ &- b[1]y[n-1] - b[2]y[n-2] - b[3]y[n-3] \end{aligned} \tag{6.15}$$

Rearranging to group $y[n]$ an $x[n]$ yields

$$\begin{aligned} &y[n] + b[1]y[n-1] + b[2]y[n-2] + b[3]y[n-3] \\ &= a[0]x[n] + a[1]x[n-1] + a[2][x[n-2] \end{aligned} \tag{6.16}$$

Taking the z-transform of both sides (and using the law of linearity), we obtain

$$\begin{aligned} &Y(z) + b[1]Y(z)z^{-1} + b[2]Y(z)z^{-2} + b[3]Y(z)z^{-3} \\ &= a[0]X(z) + a[1]X(z)z^{-1} + a[2]X(z)z^{-2} \end{aligned} \tag{6.17}$$

ie

$$Y(z)\left(1 + b[1]z^{-1} + b[2]z^{-2} + b[3]z^{-3}\right) = X(z)\left(a[0] + a[1]z^{-1} + a[2]z^{-2}\right) \tag{6.18}$$

So finally,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a[0] + a[1]z^{-1} + a[2]z^{-2}}{1 + b1[1]z^{-1} + b[2]z^{-2} + b[3]z^{-3}} = \frac{\sum_{m=0}^{2} a[m]z^{-m}}{1 + \sum_{n=1}^{3} b[n]z^{-n}} \tag{6.19}$$

The change in sign of the denominator polynomial is something that must not be overlooked when moving from the transfer function to expressing the filter in terms of its difference equation.

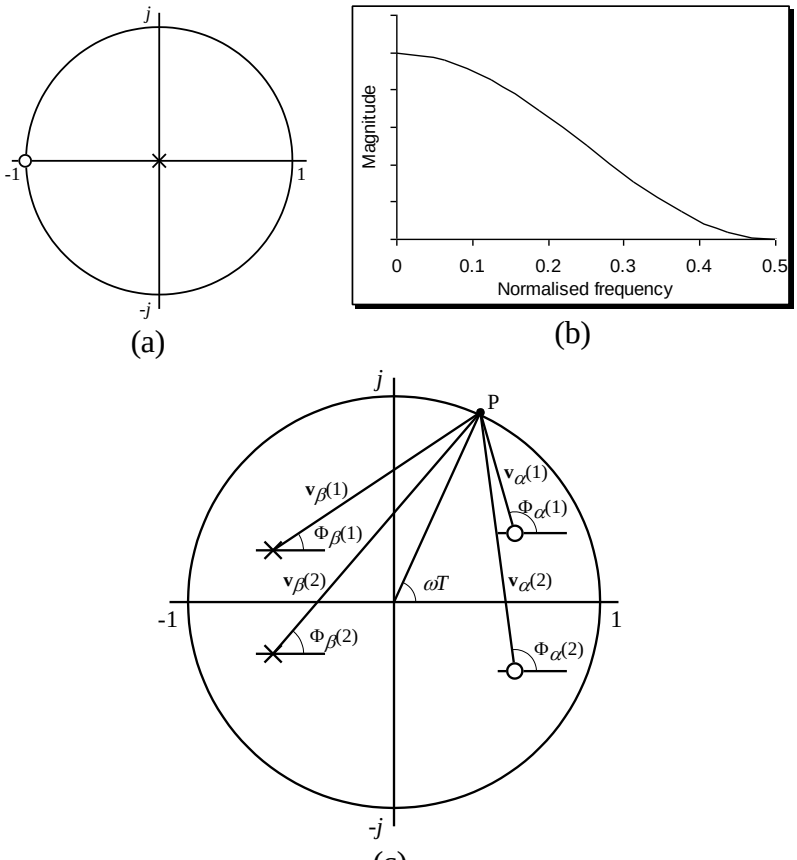


Figure 6.5. (a) Pole-zero plot and (b) frequency response of a Hanning filter with an impulse response given by 0.5, 1.0, 0.5. (c) Obtaining the magnitude and phase of the frequency response using the pole-zero plot.

6.3.3 The poles and zeros of digital filters

The zeros are the values of z for which the transfer function becomes equal to zero, and are in essence the roots of the numerator polynomial. Similarly, the poles represent the roots of the denominator polynomial, i.e. they are the values of z for which the transfer function becomes infinite. As long as the poles of a filter lie within the unit circle, it will be stable. In order to avoid confusion with the z operator, we will use the letter α to denote a zero and β to denote a pole. We have already stated that FIR filters are unconditionally stable; thinking about them in terms of poles and zeros allows us to see why. For example, say we have a filter given by $h[n]$ = 0.5, 1, 0.5, for n = 0...2. This will have a transfer function of

$$H(z) = 0.5 + z^{-1} + 0.5z^{-2} \tag{6.20}$$

Clearly, this has two zeros, i.e. α = -1 twice, but only a single pole given by β = 0. The pole-zero diagram for this filter is shown in Figure 6.5, together with its frequency response. No matter how many terms an FIR filter has, it only ever has a single pole, always located at the origin, i.e. β = 0.

IIR filters can have poles anywhere within the unit circle; as we approach a pole the magnitude of the filter's output increases, and as we approach a zero, it falls. Using all the poles and zeros within the unit circle it is possible to obtain the frequency response of the transfer function To calculate the response at a given frequency, say ω_1, we proceed as follows. First, locate the point P on the circumference of the unit circle that corresponds to this frequency. Draw vectors from each of the poles to this point. These vectors we will call $\mathbf{v}_\beta(1)\ldots\mathbf{v}_\beta(M)$. Next, produce similar vectors for the zeros, $\mathbf{v}_\alpha(1)\ldots\mathbf{v}_\alpha(N)$. The *magnitude* of the frequency response at the frequency ω_1 is given by the product of the lengths of all the zero vectors divided by the product of the lengths of all the pole vectors, i.e.

$$\left|H(z)\right| = \frac{\prod_{n=1}^{N} \mathbf{v}_\alpha(n)}{\prod_{m=1}^{M} \mathbf{v}_\beta(m)} \tag{6.21}$$

This is illustrated in Figure 6.5c (note: use radian, not hertz). To obtain the *phase* of the frequency response at the frequency ω_1, we sum the phases of all the zero vectors, sum the phases of the pole vectors, and subtract the second sum from the first, i.e.

$$\Phi(z) = \sum_{n=1}^{N} \Phi_\alpha(n) - \sum_{m=1}^{M} \Phi_\beta(m) \tag{6.22}$$

This is also illustrated in Figure 6.5c. An alternative method of obtaining the frequency response is to replace z^{-1} with the term $e^{-j\omega T}$ wherever it appears in the transfer function; a brief inspection of Equation 6.1 shows that if we do this then we are simply taking the Fourier transform of the impulse response. Although both the pole-zero method and the substitution method may in principle be used to find the frequency response of any filter, they are very laborious and rarely used in practice. A much more efficient and reliable method is simply to stimulate the difference equation with an impulse, and compute the DFT of the resulting function (which is of course the impulse response).

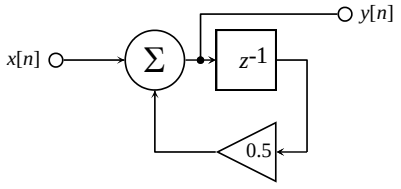


Figure 6.6. Simple recursive filter with a feedback gain of 0.5.

Figure 6.6, for example, shows a recursive filter whose transfer function is given by

$$H(z) = \frac{1}{1 - 0.5z^{-1}} \tag{6.23}$$

and whose difference equation is therefore

$$y[n] = x[n] + 0.5y[n-1] \tag{6.24}$$

Clearly this has an impulse response given by $h[n]$ = 1, 0.5, 0.25, 0.125… It is therefore a simple matter to compute that $h[100] = 7.889 \times 10^{-31}$.

6.3.4 Factored forms of the transfer function

If we know the poles and zeros of a filter, we can express it directly in diagram form (direct form type II) or as a difference equation. For example the filter of Figure 6.7 has a transfer function given by

$$H(z) = \frac{1 - a[2]z^{-2}}{1 + b[1]z^{-1} - b[2]z^{-2} + b[4]z^{-4}} \tag{6.25}$$

hence a difference equation expressed as

$$y[n] = x[n] - a[2]x[n-2] - b[1]y[n-1] + b[2]y[n-2] - b[4]y[n-4] \tag{6.26}$$

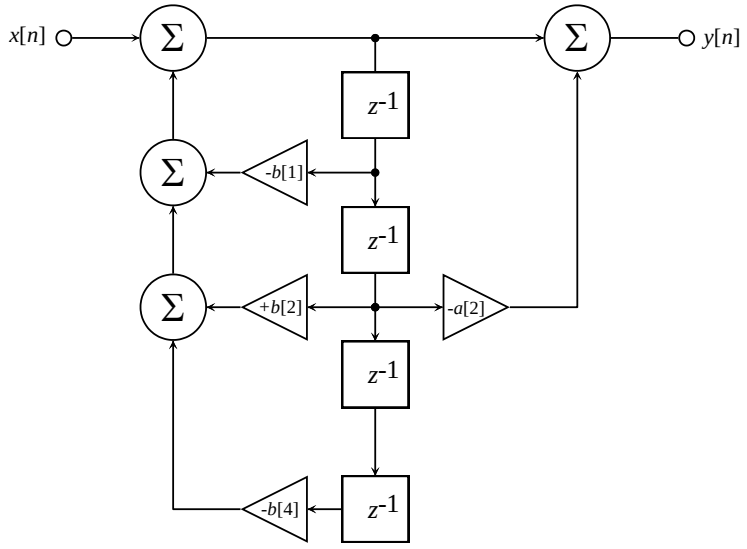


Figure 6.7. IIR Filter diagram of Equation 6.26.

However, many filter designers choose to reconfigure high-order filters as a series cascaded 2nd order sections. This is known as *biquad design*, and is desirable because the filter involves low negative index values of *z*, the dangers associated with instability are minimised. Additionally, it is possible to derive a general-purpose algorithm that can process each and every stage of the filter, obviating the necessity of changing the code each time the filter is redesigned. Biquad design commences by using the poles and zeros to express the transfer function in factored form. Given an n^{th} order filter with n poles and n zeros, the factored form of the transfer function may be written as

$$H(z) = \prod_{n=0}^{N-1} \frac{z - \alpha_n}{z - \beta_n} = \frac{(z-\alpha_0)(z-\alpha_1)}{(z-\beta_0)(z-\beta_1)} \times \frac{(z-\alpha_2)(z-\alpha_3)}{(z-\beta_2)(z-\beta_3)} \times \ldots \times \frac{(z-\alpha_{N-2})(z-\alpha_{N-1})}{(z-\beta_{N-2})(z-\beta_{N-1})} \tag{6.27}$$

It is often the case when designing IIR filters that the poles and zeros are represented as conjugate pairs, i.e. $a \pm jb$. If we make this reasonable assumption here, then for each biquad section given by

$$H(z) = \frac{(z-\alpha_0)(z-\alpha_1)}{(z-\beta_0)(z-\beta_1)} \tag{6.28}$$

the complex values of the zeros and poles are

$$\begin{aligned} \alpha_0 &= a_0 + jb_0 \qquad \alpha_1 = a_0 - jb_0 \\ \beta_0 &= a_1 + jb_1 \qquad \beta_1 = a_1 - jb_1 \end{aligned} \tag{6.29}$$

Making the simplification

$$\begin{aligned} \varepsilon_0 &= a_0^2 + b_0^2 \\ \varepsilon_1 &= a_1^2 + b_1^2 \end{aligned} \tag{6.30}$$

Then each 2nd order stage is given by

$$H(z) = \frac{[z-(a_0+jb_0)][z-(a_0-jb_0)]}{[z-(a_1+jb_1)][z-(a_1-jb_1)]} = \frac{1-2a_0z^{-1}+\varepsilon_0 z^{-2}}{1-2a_1z^{-1}+\varepsilon_1 z^{-2}} \quad (6.31)$$

and the difference equation for each 2nd order stage is therefore

$$y[n] = x[n] - 2a_0x[n-1] + \varepsilon_0 x[n-2] + 2a_1y[n-1] - \varepsilon_1 y[n-2] \quad (6.32)$$

The nice thing about Equation 6.32 is that to use it, we simply insert different coefficient values, based on the poles and zeros of our high order filter.

6.4 IIR filter deign using pole-zero placement

6.4.1 Simple design strategies

Now that we have established some general-purpose technique for encoding cascaded biquad sections, it would be appropriate to deal first with one simple but widely used method for designing IIRs: that of pole-zero placement. We will discuss a program called *ztransfer.exe* that demonstrates this method of IIR filter design. A screenshot of the program is shown in Figure 6.8. This program allows the user to enter the poles and zeros of a digital filter, whereupon it will calculate the impulse and frequency response of the IIR design. Since it uses Equation 6.32, it assumes that for an n^{th} order filter there are n zeros and n poles, grouped in conjugate pairs (this is indeed often the case with IIR filters). For example, say you had a 2nd order filter whose zeros and poles were given by:

$$\begin{aligned} \alpha_0 &= 0.7 + j0.7 \quad \beta_0 = 0.6 + j0.6 \\ \alpha_1 &= 0.7 - j0.7 \quad \beta_1 = 0.6 - j0.6 \end{aligned} \quad (6.33)$$

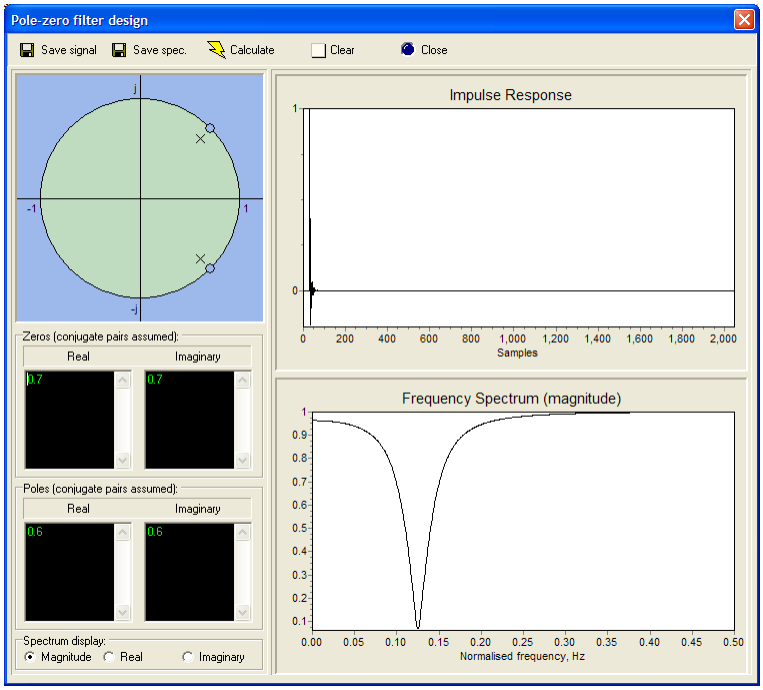


Figure 6.8. Screenshot of the program *ztrasnfer.exe.*

Given these values, the program displays a pole-zero plot as shown in Figure 6.9a, together with an impulse and frequency response as depicted in Figures 6.9b and 6.9c. This is moderately sharp notch filter, and an examination of the pole-zero plot, together with Equation 6.21 reveals why. The frequency of a system using the unit circle is given by the angle of the point on the circumference that we are measuring. Since the real and imaginary values of the zeros are equal in magnitude, then these must lie at an angle of 45° on the circle, i.e. a frequency of $\pi/4$ radians. In other words, this is 0.125 of the sampling frequency.

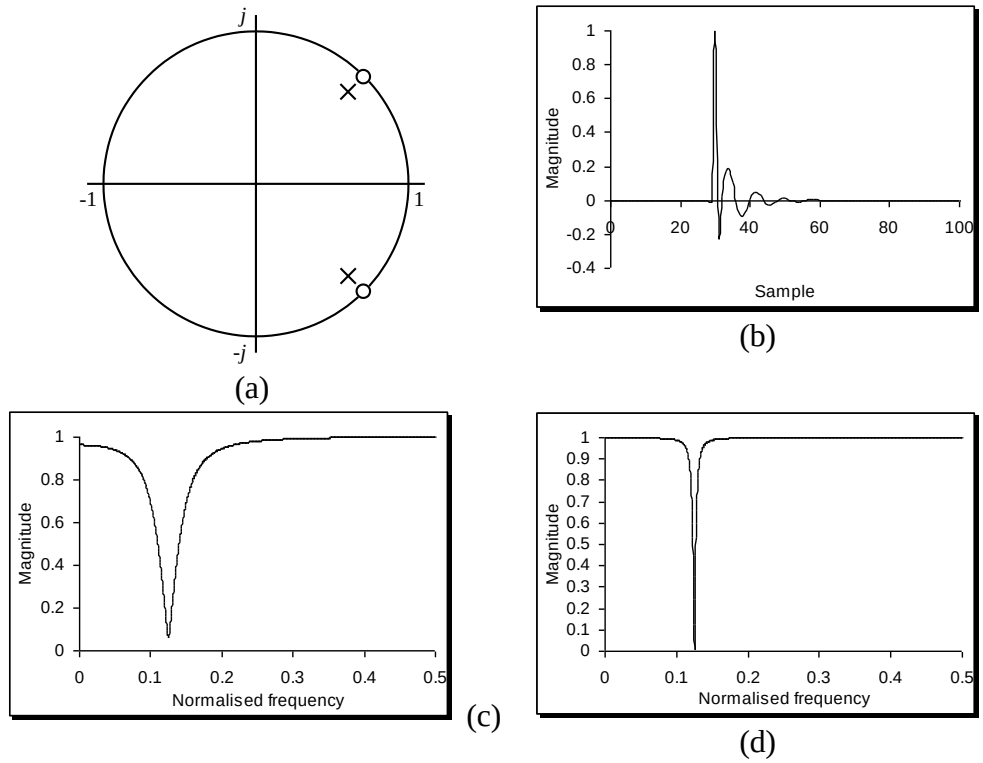


Figure 6.9. A simple IIR notch filter; (a) pole-zero plot with poles of 0.6 ± *j*0.6 and zeros of 0.7 ± *j*0.7; (b) impulse and (c) frequency response. For purposes of clarity, the impulse response has been shifted to the right. (d) Frequency response with poles of 0.69 ± *j*0.69 and zeros of 0.707 ± *j*0.707;

At a substantial distance away from the poles and zeros the lengths of the pole and zero vectors are approximately equal, so the frequency response is flat in these regions. As we approach the frequency corresponding to the position of zero, then the zero vectors become progressively shorter than those of the poles. Hence, when we take the ratio of the zero to the pole vectors, we find a trough in the response. A remarkable property of the IIR filter is its efficiency; to increase the sharpness of the response, simply move the zeros exactly on to the unit circle and move the poles closer to the zeros. Now the filter is much more selective with respect to the rejection band. It is important to emphasise that this improvement in selectivity has been realised *without* increasing the order of the design, and therefore requires no more computational effort. However, the risk of instability is greater, since the poles are now approaching the unit circle. Figure 6.10 shows what happens when the poles lie outside of the unit circle.

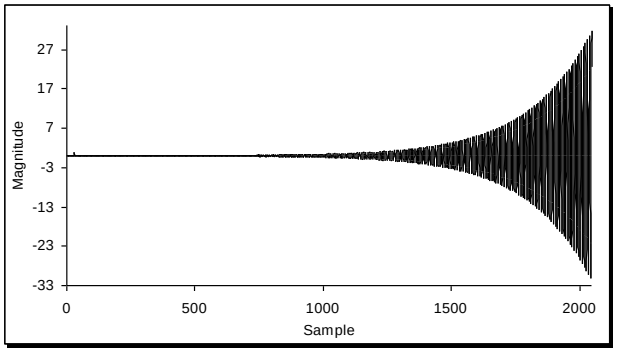
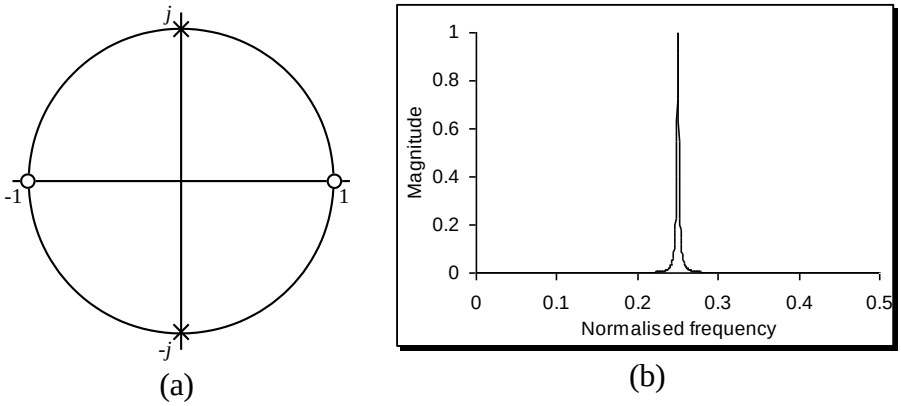


Figure 6.10. Output of unstable filter, with poles located outside of the unit circle.

Now consider the 4th order filter whose zeros and poles are given by

$$\alpha_0 = 1 + j0 \quad \alpha_1 = 1 - j0 \quad \alpha_2 = -1 + j0 \quad \alpha_3 = -1 - j0$$
$$\beta_0 = 0 + j0.99 \quad \beta_1 = 0 - j0.99 \quad \beta_2 = 0 + j0.99 \quad \beta_3 = 0 - j0.99 \qquad (6.34)$$

This produces a very sharp band-pass filter, with a centre frequency of 0.25 the sample rate. The filter's pole-zero plot and frequency response is shown in Figure 6.11.



(a) (b)

Figure 6.11. (a) Pole-zero plot and (b) frequency response of filter given in Equation 6.34.

```
x[30]:=1;
m:=0;
N2:=N1 div 2;
repeat
  a:=strtofloat(zero_real.Lines.Strings[m]);
  b:=strtofloat(zero_imag.Lines.Strings[m]);
  c:=strtofloat(pole_real.Lines.Strings[m]);
  d:=strtofloat(pole_imag.Lines.Strings[m]);
  e:=a*a+b*b;
  f:=c*c+d*d;
  for n:=20 to N1-1 do
    y[n]:=x[n]-2*a*x[n-1]+e*x[n-2]+2*c*y[n-1]-f*y[n-2];
  for n:=0 to N1-1 do x[n]:=y[n];
  inc(m);
until(zero_real.Lines.Strings[m]='');
fft(forwards,rectangular,y,dummy,Yr,Yi,N1);
```

Listing 6.1

To understand how the software works, take a look at Listing 6.1, which shows the most important code fragment from the program. Initially, the input signal array `x[n]` is set to zero (elsewhere in the code), and then a single value at `x[30]` is set equal to 1. This represents a shifted impulse function with which the impulse response will be stimulated. It is shifted merely to avoid array indexing errors in the difference equation we need to commence our calculations sometime after time zero. In the code, `N1` is set to 2048, i.e. the length of the time domain signal, and `N2` is half of this, i.e. the length of the spectrum for positive frequencies. Within the `repeat...until` loop, the poles and zeros are read. This loop repeats m times, where m is equal to the order of the filter divided by two. Hence within each loop the program is computing the impulse response of each 2^{nd} order stage, i.e. it is following a biquad cascade design process. After gathering the zero/pole information, the program simply implements Equations 6.29, 6.30 and 6.32. Finally, it obtains uses the FFT procedure to calculate the frequency response of the filter.

6.4.2 Standard filters using biquad cascading

As it stands so far, the pole-zero placement method represents a somewhat trial-and-error approach to filter design. Knowing that a zero gives rise to a dip in the frequency response and a pole produces a peak, the designer moves these around until a suitable filter is produced. Are there more systematic approaches, and can the biquad cascade method be used to implement them? The answer to both of these questions is yes, and later we will look in detail at a couple of solid IIR design strategies.

6.5 FIR and IIR filters: merits and disadvantages

Although we have not yet analysed in detail how FIR filters are designed, we have already found that in some respects, FIR and IIR filters are radically different. Arising from this there are important consequences and behaviours associated with these two approaches to digital filtering, which are summarised in Table 6.3. Filter selection criteria are predicated on efficiency, phase linearity, transition zone performance, stability, ease of design and word length immunity. Because of their recursive nature, IIR filters impose less computational burden on the processor than FIR types, requiring fewer coefficients to effect the same cut-off performance; in addition, formulae have been established over many years to replicate the filter characteristics of traditional analogue designs (actually, there is a way around this problem with FIR types, as we shall see). However, it is not practicable with IIR filters to explicitly determine the phase of harmonics in the transition zone, resulting in signal shape distortion in the time-domain for high-order designs. In addition, the presence of feedback introduces the risk of instability and degraded levels of performance if a filter designed on a floating-point processor is ported to an architecture supporting a shorter fixed-point format.

Property/Filter Type	**FIR**	**IIR**
Unconditional stability	Yes	No
Phase distortion	No	Yes
Design ease	Easy	Difficult
Arbitrary response	Easy	Difficult
Computational load	High	Low
Word length immunity	Good	Poor
Analogue equivalent	No	Yes
Real-time operation	Yes	Yes

Table 6.3. Some common properties of FIR and IIR filters.

In contrast, whilst FIR filters require more coefficients for a given transition zone performance, they are unconditionally stable, manifest good word length immunity and can, if desired, operate with zero-phase distortion, a feature desirable for high-fidelity audio systems and biomedical signal processing. Linear-phase is guaranteed if the impulse response of the filter is symmetrical, i.e. if it obeys the relationship given by

$$h(n) = h(N\text{-}n\text{-}1), \qquad n = 0,1..(N\text{-}1)/2 \qquad (N \text{ odd}) \tag{6.35}$$

Similarly, by employing the frequency sampling method for FIR design, it is a simple task to specify the phase angle of any given harmonic or group of harmonics to a level of precision far beyond that which is possible with analogue or IIR filters. A particular advantage of the FIR over the IIR realisation is that filters with completely arbitrary frequency responses may be designed and implemented with great facility. In fact, as we shall see, this allows us to replicate the behaviour of *any* linear system, and is more properly termed linear systems emulation, rather than filtering. Arbitrary filters of this kind are quite impossible to design using analogue means, and whilst in theory IIR filters may be designed with multiple stop and pass bands, the difficulties associated with the computations and the non-linear characteristics of their phase effectively limits their use in this context.

Despite the many merits of FIR filters, IIR types are nevertheless widely used, not only because of their speed, but because many natural processes – such as acoustic reverberation, the behaviour of an *LCR* circuit or indeed a guitar string – may be modelled using the Laplace and z-transform systems of equations.