Chapter 3

Introduction to Digital Filters

3.1 Introduction

A filter is a device that transmits (or rejects) a specific range of frequencies. There are four basic filter types; *Low-pass*, *high-pass*, *band-pass* and *band-stop*. There are also two types of filters; *Finite impulse response* (FIR) and *infinite impulse response* (IIR).

In general FIR filters can be designed to have exact linear phase and there is also great flexibility in shaping their magnitude response. In addition, FIR filters are inherently more stable and the effects of quantisation errors are less severe than IIR filters. Conversely, IIR filters require fewer coefficients than FIR filters for a sharp cut-off frequency response, and analogue filters can *only* be modelled using IIR filters.

3.2 Finite Impulse Response

FIR filters are normally designed to have a linear phase response. Equation 3.1 and Equation 3.2 define the finite difference equation and the z-transfer function for the non-recursive FIR filter respectively. The output of this type of filter is dependent upon the *present* and *previous* inputs.

$$y[n] = \sum_{k=0}^{N-1} b_k \cdot x[n-k] \tag{3.1}$$

$$H(z) = \sum_{k=0}^{N-1} b_k \cdot z^{-k} \tag{3.2}$$

3.3 Infinite Impulse Response

IIR filter design usually concentrates on the magnitude response and regards the phase response as secondary. Equation 3.3 and Equation 3.4 define the finite difference equation and the z-transfer function for the recursive IIR filter respectively. The output of this filter is dependent upon both previous *inputs* and *outputs*.

$$y[n] = \sum_{k=0}^{N} b_k \cdot x[n-k] - \sum_{k=1}^{M} a_k \cdot y[n-k] \tag{3.3}$$

$$H(z) = \frac{\sum_{k=0}^{N} b_k \cdot z^{-k}}{1 + \sum_{k=1}^{M} a_k \cdot z^{-k}} \tag{3.4}$$

3.4 Low-Pass Filter Specifications

A low-pass filter, as depicted in Figure 3.1, is designed to pass low frequencies from zero to a cut off frequency F_p, with approximately unity gain. The frequency range from zero up to F_p is called the *pass band* of the filter. The filter is specifically designed so that any frequencies greater than F_s become attenuated. The frequency range above F_s is referred to as the *stop band* of the filter. Between the pass band and stop band, there is a region called a *transition band* and the exact behaviour of the frequency response in this region is usually of little importance.

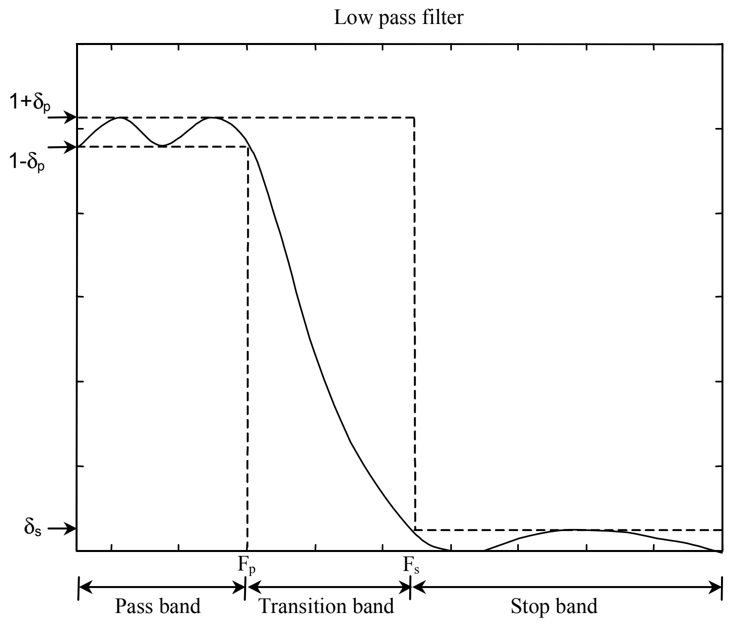

Figure 3.1: Specification of a low-pass filter.

3.5 Realisation of Digital Filters

So far we have discussed both IIR and FIR digital filters as black boxes, whose input-output relationships are well defined but have ignored the internal structure. The internal structure of a digital filter can be represented by a series of block diagrams, or signal flow diagrams, called the *realisation* of a digital filter. The basic operations of a digital filter are illustrated in Figure 3.2. It includes operations such as a unit delay Figure 3.2 (a), additions Figure 3.2 (b), and multiplication of signals by constant coefficients Figure 3.2 (c). The realisation of the delay operator uses the z^{-1} notation since this is the delay in the z-domain.

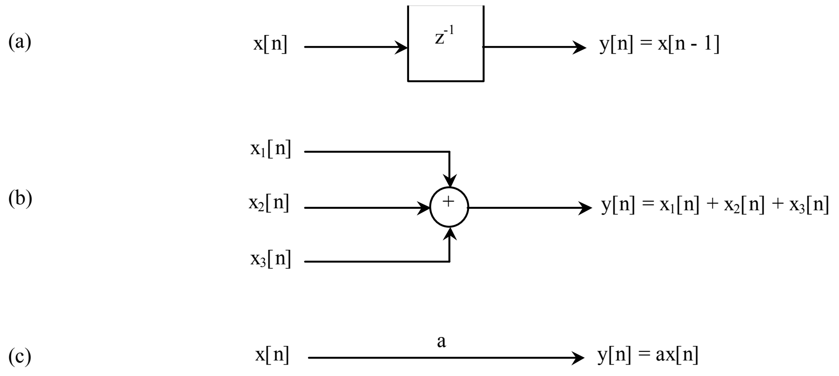

Figure 3.2: Basic building blocks for digital filter realisation.

3.6 FIR Digital Filter Realisation

Consider the first-order FIR filter, as defined by Equation 3.5. Figure 3.3 illustrates a realisation of this filter using one delay element, two multipliers and one adder.

$$H(z) = b_0 + b_1 z^{-1} \tag{3.5}$$

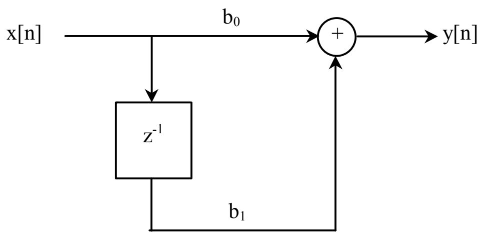

Figure 3.3: Realisation of a first-order non-recursive FIR filter.

The input to the delay element at time *n* is *x*[*n*] and its corresponding output value is *x*[*n*-1], which is also multiplied by a constant value of b_1. Furthermore, the input signal *x*[*n*] is also multiplied by a constant value of b_0. The final output from the system y[*n*] is found by adding these signals together, shown mathematically by Equation 3.6 below.

$$y[n] = b_0 x[n] + b_1 x[n-1] \tag{3.6}$$

It is also worth noting that the output of this device is identical to the time-domain expression for H(z) from Equation 3.5.

3.6.1 Direct Realisation

The direct realisation of a digital FIR system is portrayed in Figure 3.4. Sometimes it is called a tapped delay line, from its chain of *N* delay elements. Equation 3.7 defines the transfer function of a *direct realisation*.

$$H(z) = b_0 + b_1 z^{-1} + \cdots + b_N z^{-N} \tag{3.7}$$

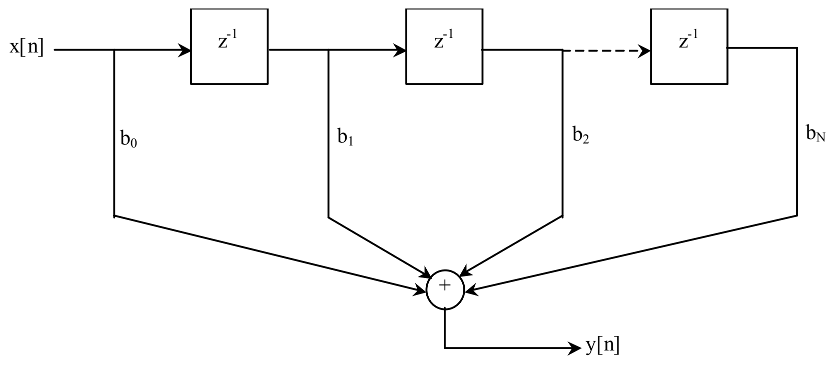

Figure 3.4: Direct realisation of a digital FIR system.

3.6.2 Fast Convolution

The FFT algorithm described in Chapter 2, can be used to implement frequency-domain equivalents of the time-domain digital filter. Rather than convolving an input signal with the impulse response of the desired filter, the signal is transformed using an FFT and then multiplied by the equivalent frequency response. A final inverse transform of Y[k] = X[k]H[k] yields the filtered signal *y*[n]. Although this method sounds lengthy, it often proves faster than time-domain convolution. This is due to the relative simplicity of multiplication, and the speed with which an FFT can perform the necessary forward and inverse transforms. A block diagram structure for fast convolution is illustrated in Figure 3.5.

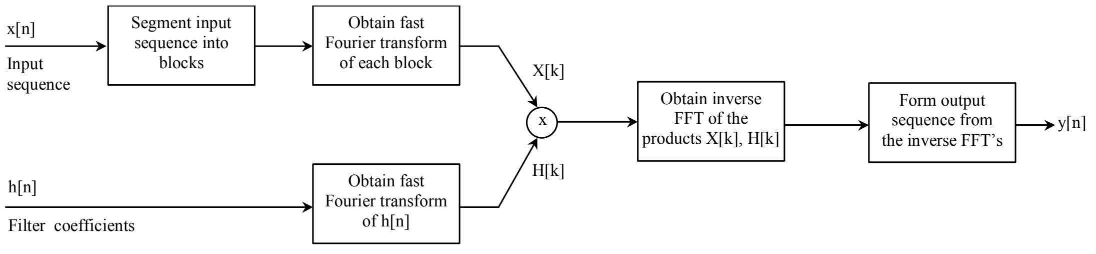
x[n]
Input sequence
Segment input sequence into blocks
Obtain fast Fourier transform of each block
X[k]
x
Obtain inverse FFT of the products X[k], H[k]
Form output sequence from the inverse FFT's
y[n]
h[n]
Filter coefficients
Obtain fast Fourier transform of h[n]
H[k]

Figure 3.5: Fast convolution.

In this exact form, there will be discontinuities in the data stream at the boundaries of the blocks. It is therefore necessary to use the approach called *overlap-add*. This involves the following steps:

(a) Segment the data sequence into blocks of length N_1 – call the i^{th} block $x_i[n]$.

(b) Zero-pad $x_i[n]$ and $h[n]$ to a length $N_1 + N_2 - 1$ (N_2 is the length of $h[n]$).

(c) Perform the operations shown in Figure 3.5 using as input the two zero-padded sequences formed in (b), and calling the output sequence:

$$z_i[n] = u_i[n] + v_i[n]$$

where: $u_i[n]$ is non-zero for $N_1 i \le n \le N_1(i+1) - 1$

$v_i[n]$ is non-zero for $N_1(i+1) \le n \le N_1(i+1) + N_2 - 2$

(d) Construct the i^{th} output block $y_i[n]$ from:

$$y_i[n] = u_i[n] + v_{i-1}[n]$$

A simple illustration of how this works is given in figure 3.5(a) [1] for the parameters N_1=5, N_2=4. This gives an FFT block length of 5+4-1 = 8. For each FFT there will therefore be 8 output samples, the first 5 of which contribute to this output segment, and the last 3 of which contribute to the first 3 samples of the next output segment.

It turns out that "fast convolution" is faster than direct convolution for $N_2 \ge 19$. By analysing the number of multiplications and additions, it is possible to establish the optimum values of N_1 and N_2, and these are given the table below.

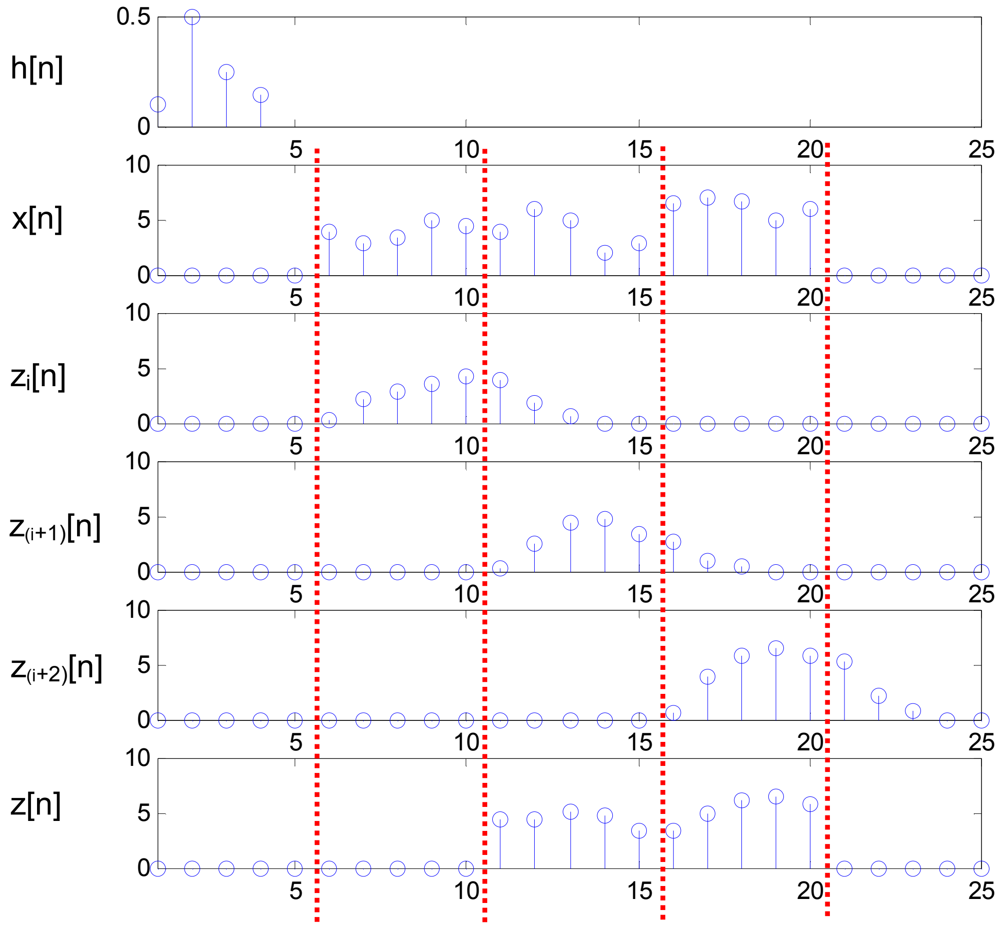

Figure 3.5(a): Illustration of *overlap-add*.

1 "A Course in Digital Signal Processing", Boaz Porat, Wiley, pp.148-151.

Range of N_2	Optimal Block Length = N_1+N_2-1
19 – 26	128
27 – 47	256
48 – 86	512
87 – 158	1024

For example, with the filter length N_2=80:

- The data segment block length N_1 should be 512-80+1 = 433.
- The sequences $x_i[n]$ and $h[n]$ will be zero-padded to a length of 433+80-1=512.
- The non-zero part of $u_i[n]$ will be of length 433.
- The non-zero part of $v_i[n]$ will be of length 78 (i.e. the number of samples that need to be *overlapped* and *added* into the next block).

3.7 IIR Digital Filter Realisation

Now let us consider a first-order recursive IIR filter, as defined by Equation 3.11 below. The realisation of this filter using one delay element, one multiplier and one adder is also illustrated in Figure 3.6.

$$H(z) = \frac{1}{1 + a_1 z^{-1}} \tag{3.11}$$

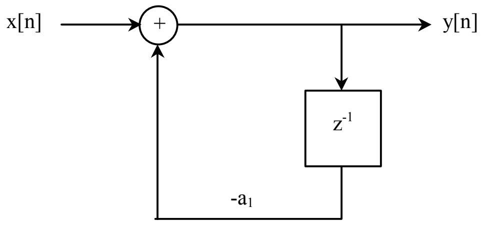

Figure 3.6: Realisation of a first-order recursive IIR filter.

The input to the delay element at time n is $y[n]$, and its corresponding output is therefore $y[n\text{-}1]$, which is multiplied by a constant value of $-a_1$. The output of the realisation structure is therefore defined by Equation 3.12 below.

$$y[n] = -a_1 y[n-1] + x[n] \tag{3.12}$$

Again, this is exactly the same as the time-domain expression for H(z) of Equation 3.11. Also notice how the recursive IIR filter builds the output value of $y[n]$ from its previous values and from the input signal.

3.7.1 Direct Realisation

The direct realisation of an IIR filter is portrayed in Figure 3.7. Similar to the FIR structure, it too is passed through a chain of delay elements. Equation 3.13 gives the transfer function of the direct realisation of an IIR filter.

$$H(z) = \frac{b(z)}{a(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{3.13}$$

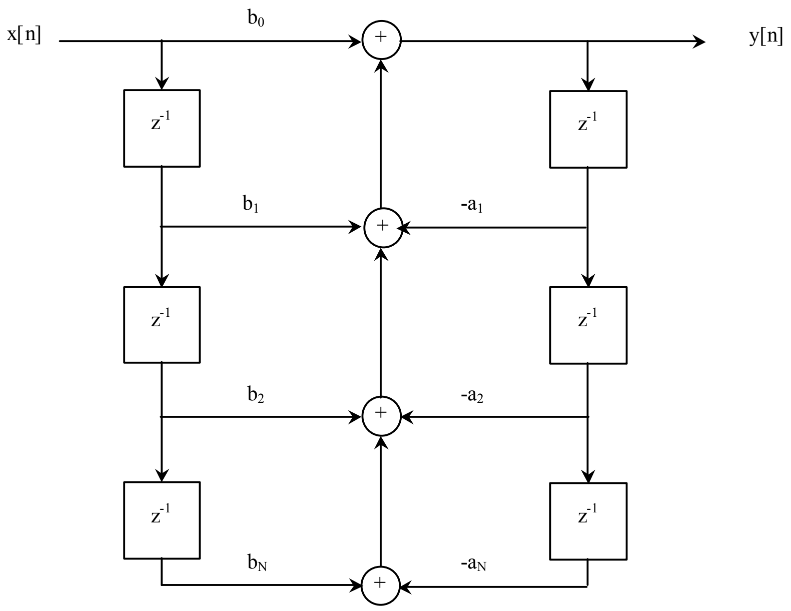

Figure 3.7: Direct realisation of a digital IIR system

3.7.2 Canonic Realisation

This structure can be re-arranged to reduce the number of registers required as follows:

$$Y(z) = X(z)\cdot H(z) = X(z)\cdot \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

$$Y(z) = \left[\frac{X(z)}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}\right]\cdot\left[b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}\right] \tag{3.13a}$$

The term in the first square bracket is an intermediate variable that has the values stored in the registers (delays) in Figure 3.8. This type of structure is often known as the *canonic* structure.

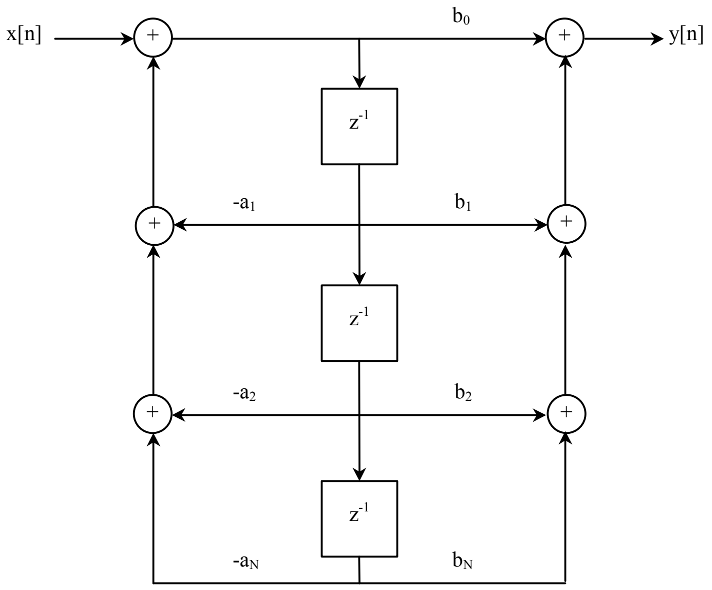

Figure 3.8: Canonic realisation of a digital IIR system.

Both of the above types of structure are known to suffer from severe coefficient sensitivity problems, especially for higher order filters.

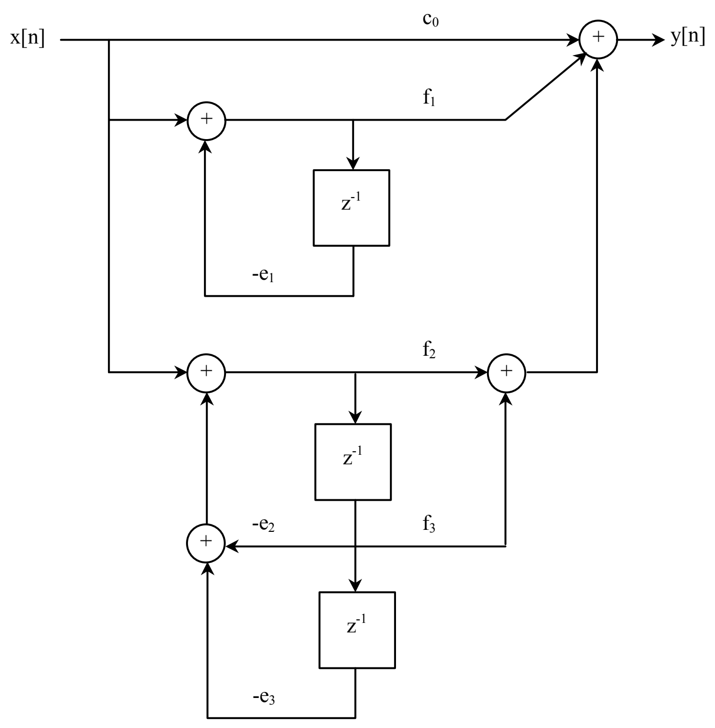

Figure 3.9: Parallel realisation of a digital IIR system.

3.7.3 Cascade Realisation

The transfer function of *cascade realisation* is described by Equation 3.15. The real numbers $\{h_i, g_i, 1 \le i \le N\}$ are more commonly known as the coefficients of cascade realisation.

$$H(z) = b_0 \prod_{i=1}^{N/2} \frac{1 + h_{2i-1} z^{-1} + h_{2i} z^{-2}}{1 + g_{2i-1} z^{-1} + g_{2i} z^{-2}} \tag{3.15}$$

If we assume that N is even, Equation 3.15 can be implemented as a cascade connection of $N/2$ sections, each of order 2. Each section can be realised by two *direct* or *canonic* realisations. The *cascade* realisation of an IIR filter is portrayed in Figure 3.10, for the case of $N = 4$. If we assume that N is odd, then the realisation can be extended by adding an extra first-order section in the cascade.

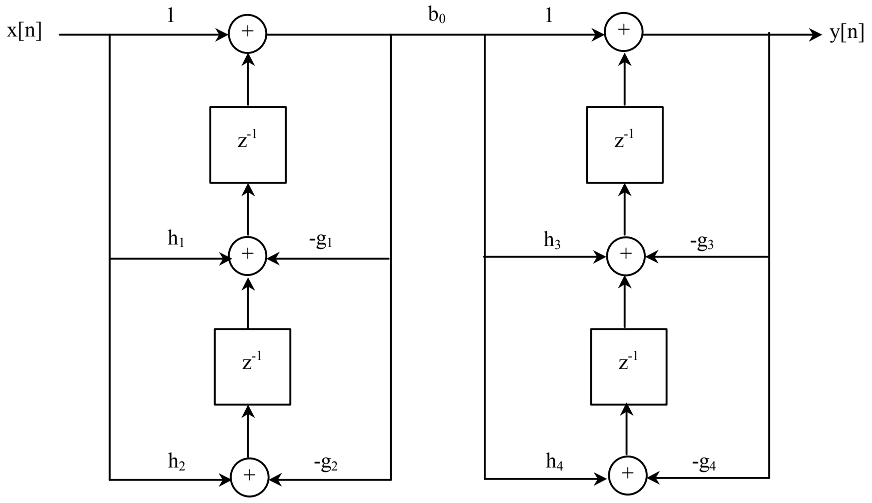

Figure 3.10: Cascade realisation of a digital IIR system.

3.7.4 Parallel Realisation

The transfer function of *parallel realisation* is described by Equation 3.14, where N_1 is the number of real poles and $2N_2$ is the number of complex poles. It is derived from partial fraction decomposition into a sum of terms that are either of first order (i.e. one pole) or second order (i.e. with complex conjugate poles). The real numbers $\{e_i, f_i, 1 \le i \le N\}$ are more commonly known as the coefficients of parallel realisation.

$$H(z) = c_0 + \sum_{i=1}^{N_1} \frac{f_i}{1 + e_i z^{-1}} + \sum_{i=1}^{N_2} \frac{f_{N_1+2i-1} + f_{N_1+2i} z^{-1}}{1 + e_{N_1+2i-1} z^{-1} + e_{N_1+2i} z^{-2}} \qquad (3.14)$$

The parallel realisation of an IIR filter is portrayed in Figure 3.9, for $N_1 = 1$, $N_2 = 1$. The first and second-order terms can be implemented by a *direct* or *canonic* realisation and the constant term c_0 is realised by simple multiplication.