Lecture 3 - Design of Digital Filters

3.1 Simple filters

In the previous lecture we considered the polynomial fit as a case example of designing a smoothing filter. The approximation to an "ideal" LPF can be improved by using a higher-degree polynomial: for example, instead of using a quadratic as in the example given in the previous lecture, we could have fitted a least-squares quartic to the original "noisy" data. The effect of using a higher-degree polynomial is to give both a higher degree of tangency at $\omega = 0$ and a sharper cut-off in the amplitude response.

All these methods though are just examples of *weighted moving average* filters. For example, we consider the the *Hanning* filter, for which:

$$y[k] = \frac{1}{4}(x[k] + 2x[k-1] + x[k-2])$$

This filter produces an output which is a scaled average of three successive inputs, with the centre point of the three weighted twice as heavily as its two adjacent neighbours.

3.1.1 Design by z-domain arguments

Taking the z-transfom we obtain a transfer function of the form

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

which has two zeroes at $z^2 + 2z + 1 = (z + 1)^2 = 0$ i.e. $z = -1$. Remember that this corresponds to a double damping at the Nyquist frequency. This will give attenuation of HF signals i.e. a LPF effect. Alternatively we could argue to put a pole at DC, some fraction $0 < \alpha < 1$ of the way along the real axis. This gives

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

and thus

$$Y(z)(1 - \alpha z^{-1}) = z^{-1}X(z)$$

and in difference terms in the digital time domain

$$y[k] = x[k - 1] + \alpha y[k - 1]$$

which gives a LPF as a *recurrent* filter (which is thus an IIR filter). In general, we may use out knowledge of the Laplace design of transfer functions to argue the design in the z-domain as well. This is simple for low-order filters (as above), but would be tedious at higher orders – there are other ways.

3.2 FIR designs based on window functions

FIR filters can also be designed from a frequency response specification. The equivalent sampled impulse response, which determines the coefficients of the FIR filter, can then be found by inverse (discrete) Fourier transformation (Discrete Fourier Transforms are not covered until later in the course but the example filter design below should still be easy to follow).

Consider an ideal low-pass characteristic (brick-wall filter) $G(j\omega)$ with a cut-off frequency $\omega_c = \pi/4T$ where $T =$ sampling period:

We know that the continuous impulse response $y(t)$ is given by (and shown in

Fig. 3.1):

$$\begin{aligned} g(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} G(j\omega)e^{j\omega t}\,d\omega \\ &= \frac{\omega_c}{\pi}\left(\frac{\sin\omega_c t}{\omega_c t}\right) \end{aligned} \tag{3.1}$$

The sampled impulse response $g[k]$ (which would be obtained by taking the

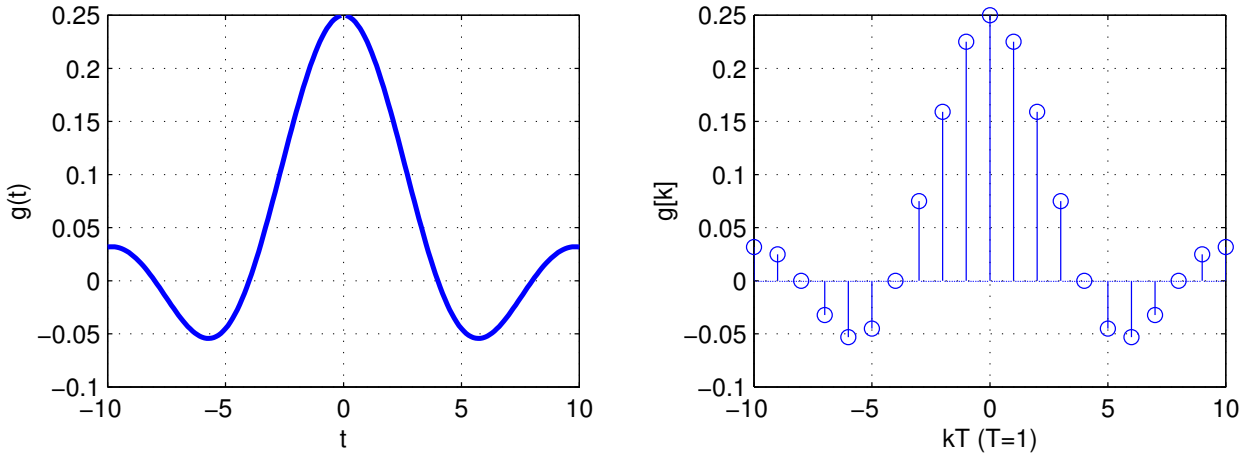


Figure 3.1: *Impulse response of brick wall filter in (left) continuous and (right) discrete time domains.*

inverse DFT of the discrete equivalent "brick-wall" frequency characteristics) is the sampled version of the continuous $\frac{\sin x}{x}$ function. It is not possible to implement the corresponding low-pass filter design because:

- an infinite number of coefficients would be required
- the impulse response is that of a non-causal system ($g[k]$ exists between $k = -\infty$ and $k = 0$).

A first solution

hence we could,

1. Truncate the expression for $g[k]$ at some reasonable value of k, say 10.
2. Shift all the coefficients by the same number.

This is shown in Fig 3.2.

Now that we have the difference equation

$$y[k] = \sum_{i=0}^{20} a_i x[k-i]$$

for the filter, we can also obtain its transfer function

$$G(z) = \sum_{i=0}^{20} a_i z^{-i}$$

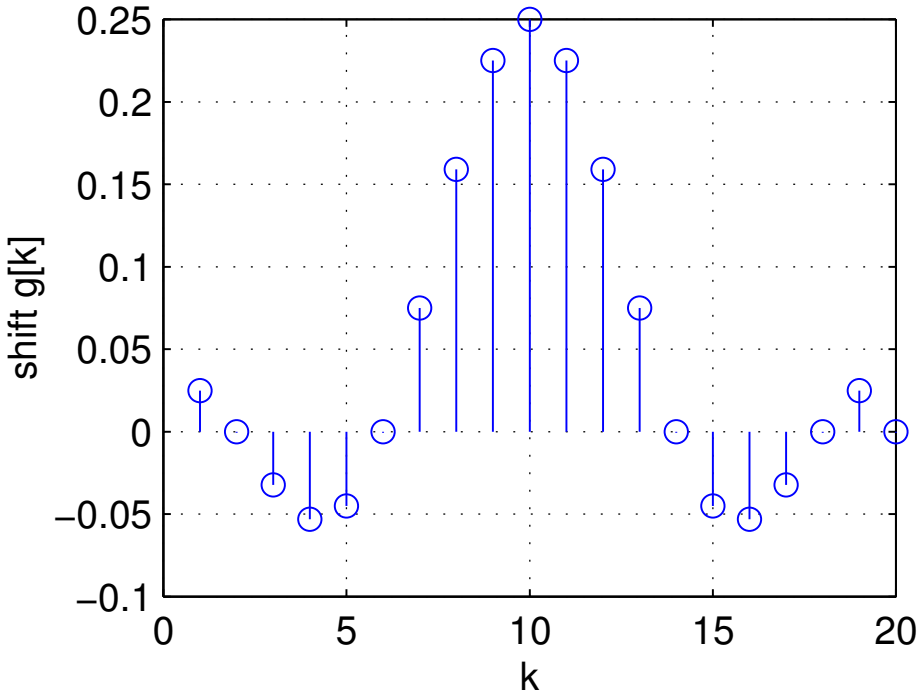


Figure 3.2: *Impulse response of brick wall filter shifted.*

As before, we can obtain the actual frequency response of the filter by evaluating $G(z)$ on the unit circle (i.e. $G(e^{j\omega T})$). This is shown in Fig 3.3 using both linear and logarithmic plots for the amplitude response.

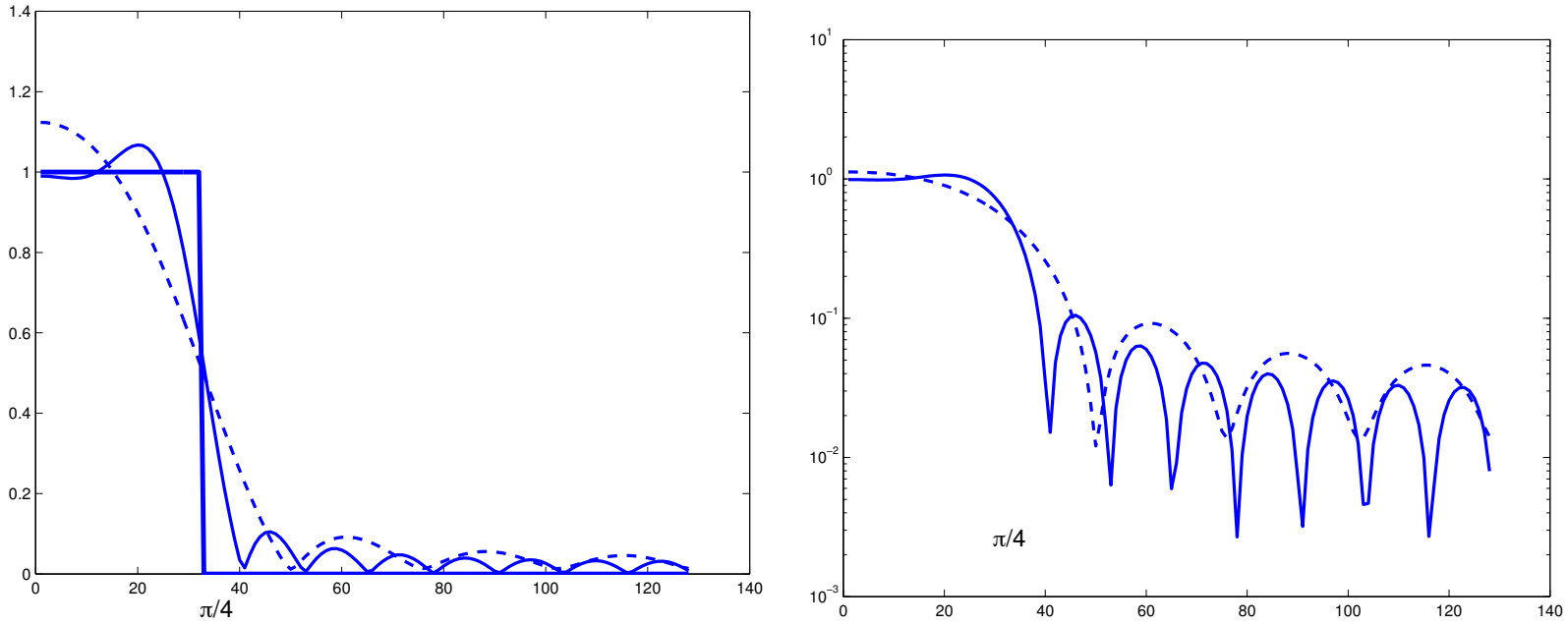


Figure 3.3: *Linear (left) and log (right) responses for 21 and 11 coefficients in the brick wall filter.*

Better solution

The truncation of the impulse response is equivalent to multiplying it by a rectangular "window" function. This leads to an overshoot and ripple before and after the discontinuity in the frequency response – a phenomenom known as *Gibb's phenomenom* (the overshoot is about 9% – see previous Fig). The amplitude of the overshoot does *not* decrease if more and more coefficients are included in the digital filter.

A more successful way of designing an FIR filter is to use a finite weighting sequence $w[k]$. There are a number of such sequences, for example the Hamming, Hanning or Kaiser windows.

$$w[k] = \begin{cases} \alpha + (1-\alpha)\cos\frac{k\pi}{N} & -N \leq k \leq N \\ 0 \qquad \text{elsewhere} \end{cases}$$

If $\alpha = 0.54$ this is the *Hamming window*, if $\alpha = 0.5$ this is the *Hanning, or raised cosine, window*. Fig 3.4 shows the 11 point Hamming window. The Fourier transform of these windows consists of a central lobe which contains most of the energy and side lobes which generally decay very rapidly.

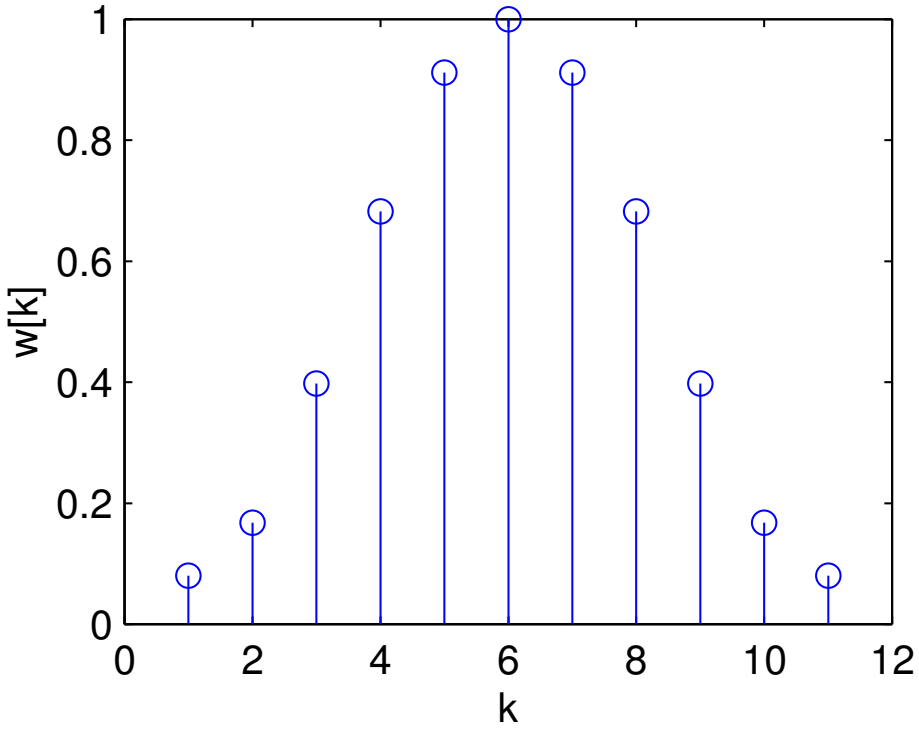


Figure 3.4: *11 point Hamming window.*

The use of such a window to reduce the Fourier coefficients for the higher frequency terms leads to a reduction in ripple amplitude, at the expense of a slightly worse initial cut-off slope. The frequency response of the 21-coefficient FIR filter Fig. 3.3 is shown in Fig. 3.5 together with that of the equivalent "windowed" filter (the filter weights on this case being computed from $g[k] = g[k].w[k]$) using a Hamming window.

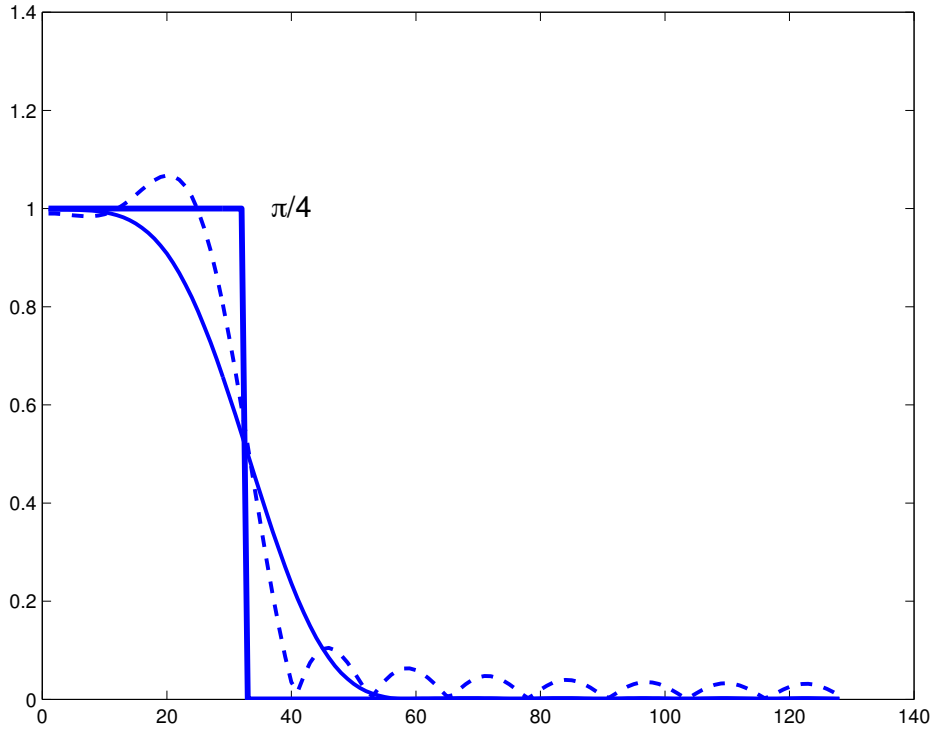


Figure 3.5: *Hamming window removing overshoot.*

3.2.1 FIR filter design – conclusion

FIR filters are usually found in applications where waveform distortion due to non-linear phase is harmful. As the 2 examples of filters studied illustrate, FIR filters with exactly linear phase can be designed (they must have an impulse response which is either symmetrical – i.e. palindromic coefficients – or purely anti-symmetrical). FIR filters are mostly realised as non-recursive structures; such filters are always stable. However, if a sharp cut-off in the amplitude response is required, a large number of coefficients are needed (usually > 100).

3.3 Design of IIR filters

Most recursive filters have an infinite impulse response, because of the feedback of previous outputs. Practical Infinite-Impulse-Response (IIR) filters are usually based upon analogue equivalents (Butterworth, Chebyshev, etc.), using a transformation known as the *bilinear transformation* which maps the s-plane poles and zeros of the analogue filter into the z-plane. However, it is quite possible to design an IIR filter without any reference to analogue designs, for example by choosing appropriate locations for the poles and zeroes on the unit circle (Remember: $G(e^{j\omega T}) = 0$ wherever there is a zero *on* the unit circle, i.e. complete attenuation of that frequency; on the other hand, $G(e^{j\omega T}) \to \infty$ when there is a pole *near* the unit circle, i.e. high gain at that frequency).

3.4 Bilinear transformation

The technique of digitizing an analogue design is the most popular IIR filter design technique, since there is a large amount of theory on standard analogue filters available (some of which was explored in the first half of this lecture course).

The bilinear z-transform is a mathematical transformation from the s-domain to the z-domain which preserves the frequency characteristics and is defined by:

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \quad \text{where } T = \text{ sampling period}$$

Under this mapping, the entire $j\omega$ axis in the s-plane is mapped onto the unit circle in the z-plane; the left-half s-plane is mapped *inside* the unit circle and the right-half s-plane is mapped outside the unit circle.

The bilinear transformation gives a non-linear relationship between analogue frequency ω_a and digital frequency ω_d. Since the frequency response of a digital filter is evaluated by setting $z = e^{j\omega T}$:

$$s_a = j\omega_a = \frac{2}{T}\frac{1 - e^{-j\omega_d T}}{1 + e^{-j\omega_d T}} = \frac{2}{T}\frac{e^{\frac{j\omega_d T}{2}} - e^{\frac{-j\omega_d T}{2}}}{e^{\frac{j\omega_d T}{2}} + e^{\frac{-j\omega_d T}{2}}} = \frac{2}{T}\tanh\frac{j\omega_d T}{2}$$

$$\text{ie.} \quad \omega_a = \frac{2}{T} \tan \frac{\omega_d T}{2}$$

The form of this non-linearity is shown in Fig. 3.6 for the case $T = 2$. For small values of ω_d, the mapping is almost linear; for most of the frequency scale, however, the mapping is highly non-linear.

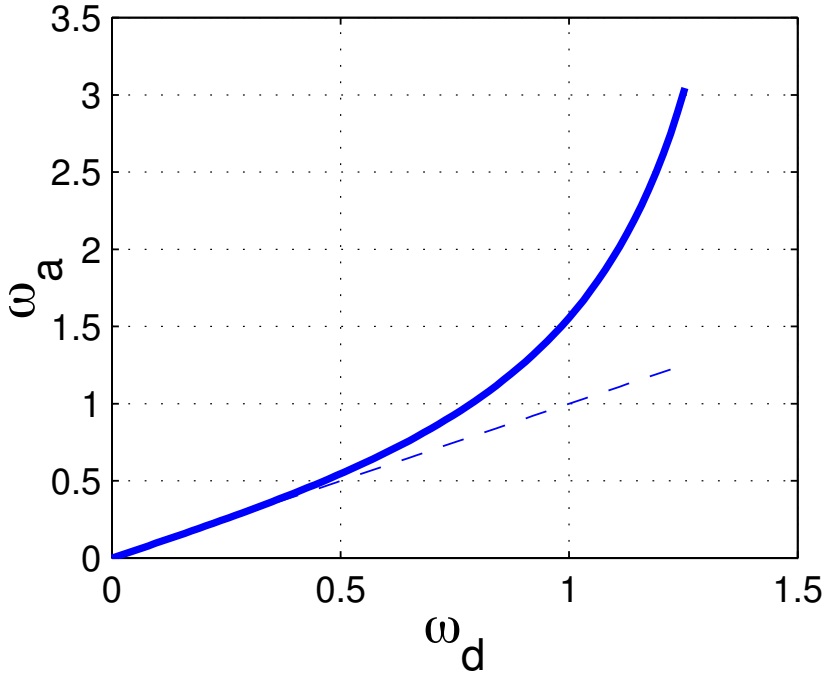


Figure 3.6: *The bilinear mapping function, with* $T = 2$

The cut-off frequencies of a digital filter will therefore be tangentially warped compared with those of the analogue filter from which it was designed. In order to compensate for this undesirable effect, it is necessary to *pre-warp* the required cut-off frequencies *before* designing the analogue filter. Thus

- the desired set of digital filter cut-off frequencies is determined first. For example, if there are four critcial cut-off frequencies, $\omega_{d1}, \omega_{d2}, \omega_{d3}$ and ω_{d4}). Using the frequency warping relationship derived above, the filter cut-off frequencies are converted to a new set of analogue cut-off frequencies, $\omega_{a1}, \omega_{a2}, \omega_{a3}$ and ω_{a4}.
- Finally, an analogue filter is designed with the appropriate warped cut-off frequencies. Applying the bilinear transformation to this analogue filter gives a digital filter with the desired cut-off frequencies.

3.4.1 Example: design of IIR filter using bilinear z-transform

Design a digital low-pass Butterworth filter with a 3dB cut-off frequency of 2kHz and minimum attenuation of 30dB at 4.25kHz for a sampling rate of 10kHz.

Answer

$f_s = 10^4$ Hz; hence $T = 10^{-4}$ sec.
$\omega_{d1} = 2\pi \times 2 \times 10^3$ rads/sec; $\omega_{d2} = 2\pi \times 4.25 \times 10^3$ rads/sec.

Apply *pre-warping* transformation:

$$\omega_{a1} = \frac{2}{T}\tan\frac{\omega_{d1}T}{2} = 2 \times 10^4 \tan 0.2\pi = 0.72654 \times \frac{2}{T} = 14.53 \times 10^3 \text{ rads/sec}$$

$$\omega_{a2} = \frac{2}{T}\tan\frac{\omega_{d2}T}{2} = 2 \times 10^4 \tan 0.425\pi = 83.31 \times 10^3 \text{ rads/sec}$$

(ie. $f_{a1} = 2.3$kHz and $f_{a2} = 13.3$kHz – this shows the warping effect near $\frac{f_s}{2}$)
The order of filter required can now be worked out as before:

$$n \approx \frac{\log A}{\log \frac{\omega_a}{\omega_c}} = \frac{log_{10} 10\sqrt{10}}{log_{10} \frac{83.31}{14.53}} = 1.98$$

$n = 2$ **meets the specification**

For a second-order Butterworth LPF,

$$G(s) = \frac{\omega_c^2}{s^2 + s\sqrt{2}\omega_c + \omega_c^2}$$

Substitute $\omega_{a1} = \omega_c = 0.72654\frac{2}{T}$ in above equation and use the bilinear transformation $s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$:

$$G(z) = \frac{(\frac{2}{T})^2 0.72654^2}{(\frac{2}{T})^2(\frac{1-z^{-1}}{1+z^{-1}})^2 + \sqrt{2}(\frac{2}{T})^2 0.72654(\frac{1-z^{-1}}{1+z^{-1}}) + (\frac{2}{T})^2(0.72654)^2}$$

(**NB:** $(\frac{2}{T})$ factor cancels out)

$$G(z) = \frac{0.52786}{(\frac{1-z^{-1}}{1+z^{-1}})^2 + 1.02749(\frac{1-z^{-1}}{1+z^{-1}}) + 0.52786}$$

$$G(z) = 0.52786\frac{1 + 2z^{-1} + z^{-2}}{2.55535 - 0.94427z^{-1} + 0.50038z^{-2}}$$

$$G(z) = 0.20657\frac{1 + 2z^{-1} + z^{-2}}{1 - 0.36953z^{-1} + 0.19582z^{-2}}$$

from which we can finally write the following difference equation:

$$y[k] = 0.20657x[k] + 0.41314x[k-1] + 0.20657x[k-2] + 0.36953y[k-1] - 0.19582y[k-2]$$

In order to check the magnitude characteristics of the frequency response, re-write $G(z)$ as:

$$G(z) = 0.20657\frac{z^2 + 2z + 1}{z^2 - 0.36953z + 0.19582}$$

- dc gain: $\omega_d T = 0 \rightarrow z = 1$ which gives $|G(1)| = 1$

- 3dB cut-off frequency $\omega_{d1} T = 0.4\pi$

  Thus

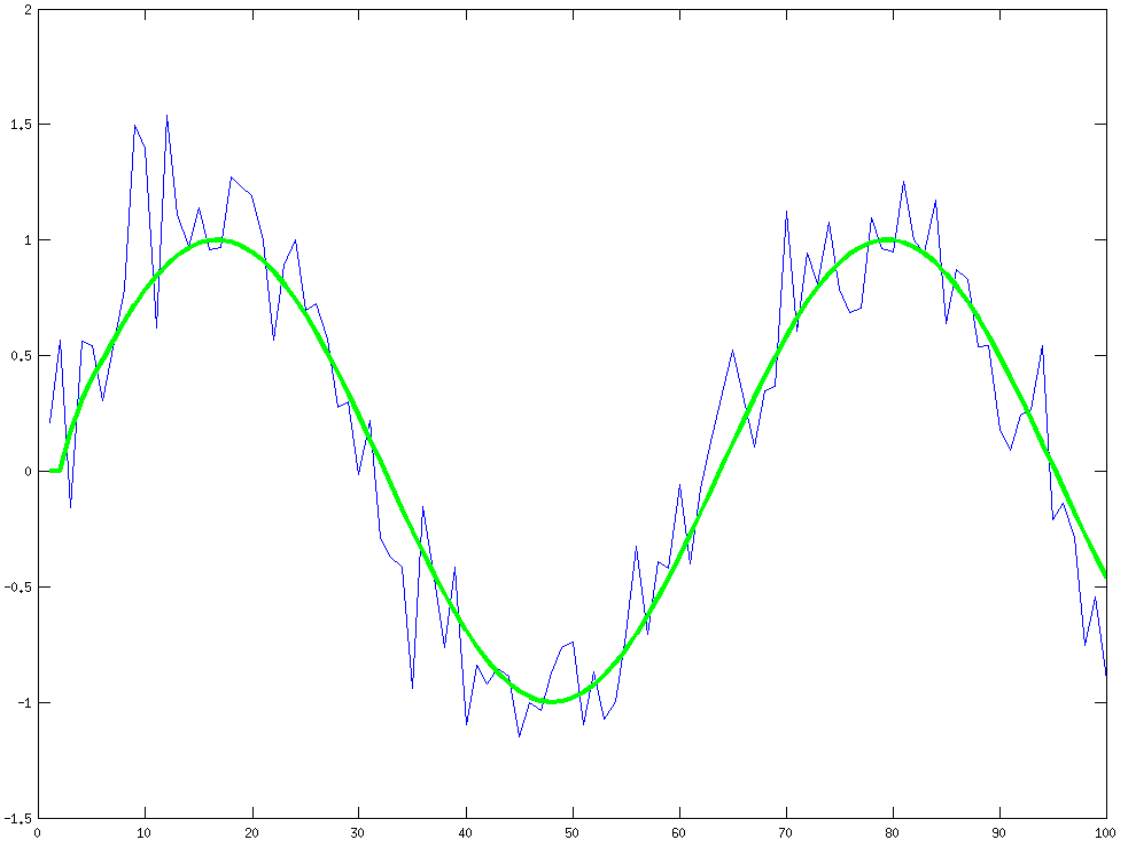
  $$G(e^{0.4\pi j}) = 0.20657\frac{\cos 0.8\pi + j \sin 0.8\pi + 2(\cos 0.4\pi + j \sin 0.4\pi) + 1}{\cos 0.8\pi + j \sin 0.8\pi - 0.36953(\cos 0.4\pi + j \sin 0.4\pi) + 0.19582}$$

ie. $G(e^{0.4\pi j}) = 0.20657\dfrac{0.8092 + 2.4899j}{-0.72739 + 0.23634j}$ which gives $|G| = 0.707$

You can also check the 30dB minimum attenuation requirement by working out $G(e^{0.85\pi j})$.

Implementation – is easy

```
>> x = sin([1:100]/10);
>> plot(x)
>> xn = x + randn(1,100)*0.2;
>> plot(xn)
>> y = zeros(1,100);
>> for n=3:100,
y(n) = 0.20657*x(n)+0.41314*x(n-1)+0.207*x(n-2)+0.36953*y(n-1)-0.19582*y(n-2);
end;
>>
>> plot(y)
>> plot(xn)
>> hold on
>> plot(y,'g','linewidth',2)
```



3.4.2 Conclusion

With recursive IIR filters, we can generally achieve a desired frequency response characteristic with a filter of lower order than for a non-recursive filter (especially if elliptic designs are used). A recursive filter has both poles and zeroes which can be selected by the designer, hence there are more free parameters than for a non-recursive filter of the same order (only zeroes can be varied). However, when the poles of an IIR filter are close to the unit circle, they need to be specified very accurately (typically 3 to 6 decimal places) if instability is to be avoided.