

The z-transform

- In the analog world we use complex phasors such as $\exp(st)$ or $\exp(j\omega t)$ as eigenfunctions of a system; the modification of these by the system is the Laplace transfer function or Frequency response respectively. In the digital realm we use $f(k)=z^k$ as an eigenfunction;

$$y(k) = \sum_{i=-\infty}^{\infty} h(i)f(k-i) = \sum_{i=-\infty}^{\infty} h(i)z^{k-i} = z^k \sum_{i=-\infty}^{\infty} h(i)z^{-i} = z^k H(z) = f(k)H(z)$$

- Just as Analog Filter design utilises the Laplace transform from time to s -domain, Digital signals make use of the z -transform from i (time) domain to the z -domain.

$$H(z) = \sum_{i=-\infty}^{\infty} h(i)z^{-i}$$

- The z variable (like the s variable is complex valued) so represents a two dimensional space, while the z -transform H , is also complex valued, so each point in z domain holds a complex value.

Properties

- Linearity $Z\{ax(i) + by(i)\} = aX(z) + bY(z)$
- Shifting $Z\{h(i \pm n)\} = z^{\pm n} H(z)$
- Convolution $Z\{x(i) \otimes y(i)\} = X(z)Y(z)$
- Scaling $Z\{a^i h(i)\} = H\left(\frac{z}{a}\right)$
- Multiplication by i $Z\{i^n h(i)\} = \left(-z \frac{d}{dz}\right)^n H(z)$
- [To prove any of these properties simply insert the function into the formula for the z -transform, and manipulate to the rearrangement given]

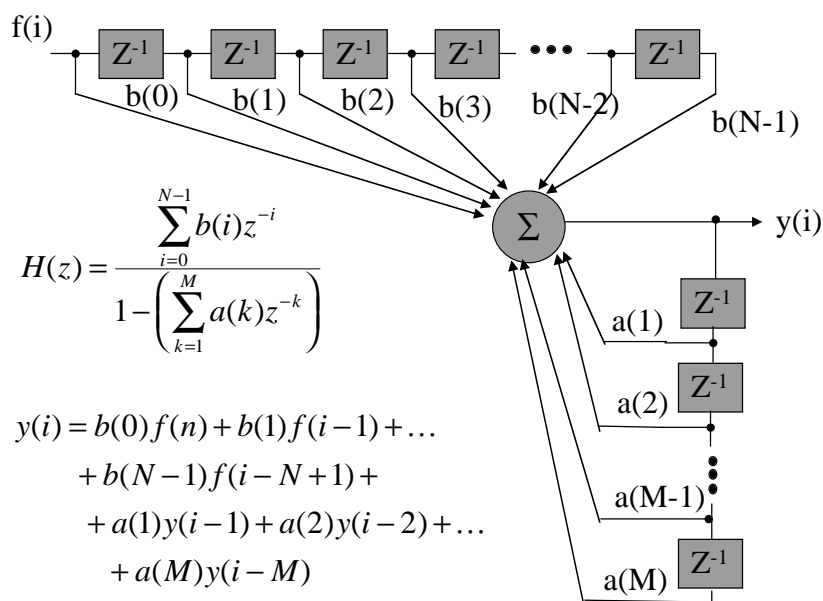
Comments

- Since the z -transform is a power series, it only exists for those areas of the z plane where it takes finite values (Region of Convergence)
- For finite duration signals the z -transform exists for all points except at the origin and infinity
- For z -transforms that are the ratio of polynomials the z -transform exists for all points except at the poles
- If a Fourier transform exists for a signal it is not necessarily the case that the z -transform exists (only that its region of convergence includes the unit circle on the z -plane)
- The inverse z -transform is simpler when the z -transform is a ratio of polynomials but generally;

$$h(i) = \frac{1}{2\pi j} \oint_C H(z) z^{i-1} dz$$

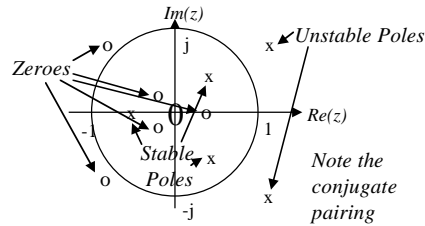
where C is the contour that includes the origin.

Infinite Impulse Response Filter



Poles and Zeroes

- To filter designers and control engineers alike the z -transform is a complex valued function (or surface) draped across the two dimensional z -plane. This surface, $H(z)$, is controlled by the placement of poles and zeroes. (Poles lift the surface or “tent canvas” up, like “tent poles”, and zeroes fasten it to the plane, like “tent pegs”).



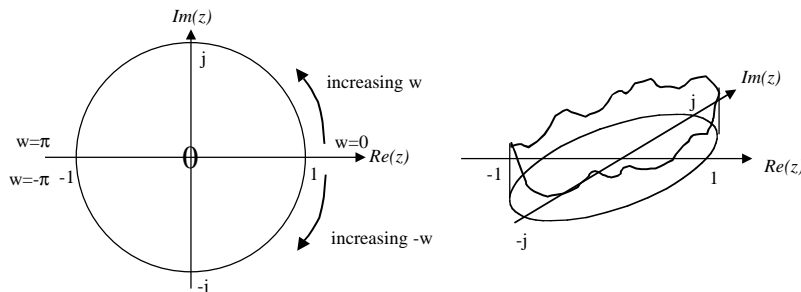
- All uncanceled poles must be within the unit circle for “stability”
- There must be at least as many poles as zeroes for “causality”
- For real valued output poles and zeroes must occur in conjugate pairs

- The factors of the numerator define the location of the zeroes, and the factors of the denominator define the location of the poles.

$$H(z) = \frac{b(0)z^{-0} + b(1)z^{-1} + \dots + b(N-1)z^{-(N-1)}}{1 - a(1)z^{-1} - a(2)z^{-2} - \dots - a(M)z^{-M}} = \frac{(z - z_1)(z - z_2)\dots(z - z_N)}{(z - p_1)(z - p_2)\dots(z - p_M)}$$

Frequency Response

- When z is $\exp(j\omega t)$, (the frequency eigenvector), the magnitude of z is unity. This corresponds to the unit circle on the z -plane, so we can determine the frequency response of the filter by examining the “surface” above the unit circle — A “cookie cutter” profile! So when ω is zero, $z = 1$; the DC component. The normalised angular frequency then increases around the circle to $\pi/2$ when $z=j$, to π (or $-\pi$) at $z=-1$. (As the frequency response of a sampled signal is periodic it is no surprise that going around and around the unit circle results in a repeated frequency spectrum.)



Poles and Zeroes contd.

- It is necessary for “physical realisability” that the number of poles of a filter is equal to, or more than, the number of zeroes. i.e., the power of the denominator is greater than or equal to the power of the numerator.
- FIR filters have a z -transform whose denominator is z^N , so the only cause for $H(z)$ to go infinite is when z is zero, i.e., the origin, (or when z itself is infinite in which case the filter is not causal).

$$H_{FIR}(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + \dots = \frac{h(0)z^N + h(1)z^{N-1} + h(2)z^{N-2} + \dots}{z^N}$$

At these locations the poles make the surface naturally drape so that all frequencies (the unit circle) are affected equally. Therefore their presence is only to ensure “causality” and in fact for each pole at the origin there is one unit delay of the output of the filter.

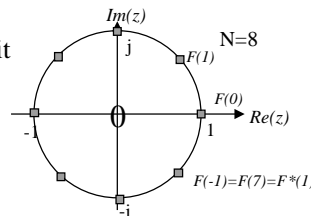
z -transform and the DFT

- As the DFT attempts to provide an estimate of the frequency content of a signal at frequency sample points $F_k = F(2\pi k/N)$, it is not surprising that we can find these samples on the unit circle on the z -transform plane.
- Let $z = \exp(j2\pi k/N)$ in the z -transform formula;

$$H(z) = \sum_{i=-\infty}^{\infty} h(i)z^{-i} = \sum_{i=-\infty}^{\infty} h(i)\exp(-j2\pi i / N)$$

and the similarity to the DFT formula is obvious!

- The N points, $z = \exp(j2\pi k/N)$, are distributed equally around the unit circle of the z -plane. i.e., equally spaced in the frequency domain.



Goertzel's Algorithm

- The DFT can be written as a convolution of the input with a complex valued sequence;

$$F(k) = \sum_{i=0}^{N-1} f(i) \exp(-j2\pi ik / N) = \exp(j2\pi kN / N) \sum_{i=0}^{N-1} f(i) \exp(-j2\pi ik / N)$$

$$= \sum_{i=0}^{N-1} f(i) \exp(j2\pi k(N-i) / N) = (f(i) \otimes \exp(j2\pi i / N)) \Big|_{i=N}$$

and so can be implemented as an FIR filter with impulse response,

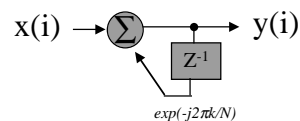
$$h_k(i) = \exp(-j2\pi ki / N)$$

- The frequency coefficient $F(k)$ can be obtained as the N th output value of the filter above when forced with the input signal!

Goertzel's Algorithm contd.

- The FIR filter bank approach still requires N lots of N^2 operations so is not a saving, but if we realise that the z -transform of the FIR filter is actually a geometric series, we can turn it into an IIR for which the sum to N terms is;

$$H(z) = \frac{1}{1 - \exp(-j2\pi k / N)z^{-1}}$$



- This changed z -transform still describes the same frequency response, but is for an IIR filter having a pole on the unit circle at $z = \exp(j2\pi k/N)$, i.e., an oscillator at the frequency represented by $F(k)$!
- Therefore each F_k may be calculated by forcing an oscillator of frequency, $2\pi k/N$, with the signal; taking the N th output sample. Each filter requires just one operation per input sample, so N operations are required before $F(k)$ is obtained — same speed as DFT.
- The real advantage arises if we need only a limited set of $F(k)$ values, perhaps requiring different N for calculation.

Implementation Diagrams

- The block diagram schematics we have drawn are known as the “direct form” implementation of the filter. It is possible, however, to divide the filter into “second order sections” (usually by grouping conjugate pairs together), and implement using less memory, or requiring less MAC operations, or requiring less moving of coefficients from memory to the ALU/DSP core etc. In fact we can specify different implementations that are less affected by quantisation of words.

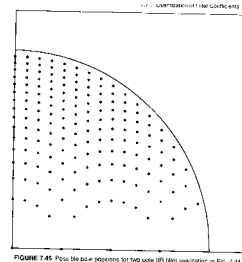
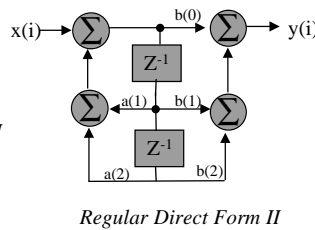
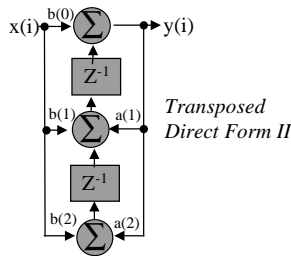
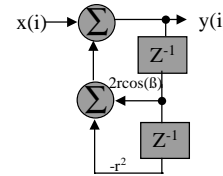


FIGURE 7.45 Pole-Zero Plot for Second-Order Filter with Coefficients as in Fig. 7.44.

Conversions

- One can take a previously designed filter and change it, keeping its form, but changing by substitution for z , its cutoff frequency;

$$z^{-1} \rightarrow \frac{z^{-1} - a}{1 - az^{-1}} \quad a = \frac{\sin[(\omega_p - \omega_p')/2]}{\sin[(\omega_p + \omega_p')/2]}$$

- from low pass to high pass;

$$z^{-1} \rightarrow \frac{z^{-1} - a}{1 - az^{-1}} \quad a = \frac{\cos[(\omega_p - \omega_p')/2]}{\cos[(\omega_p + \omega_p')/2]}$$

- etc.

Cascade or Parallel

- A more complex filter can be implemented by converting it into BIQUAD sections, as these couple complex poles together, and they are most efficient to code.
- The output of one filter is fed into the input of the next in the cascade section, and the transfer function is:

$$H(z) = H_1(z) \cdot H_2(z) \cdots H_n(z)$$

- In a parallel implementation, each filter works on the same input, and the outputs are added. Clearly, the delay through this arrangement is much less than the cascade case. The transfer functions are obtained using a partial fraction expansion (*residue*), and combine per:

$$H(z) = H_1'(z) + H_2'(z) + \cdots + H_n'(z)$$

- Note is any biquad is unstable, the whole filter is also.
- Be aware of quantisation errors accumulating in the cascade implementation.

Group Delay

- $[Gp, W] = \text{grpdelay}(B, A, N)$;
- The group delay is:

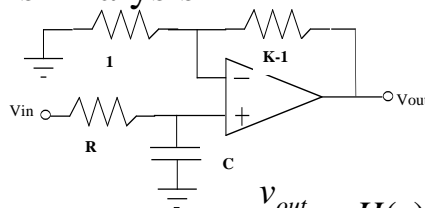
$$GD(\omega) = - \frac{d\Phi(\omega)}{d\omega}$$

- For an FIR filter which has by definition a linear phase response, the group delay should be constant and equal to the centre of the filter coefficients.

Looking Analogue....

First Order Section

- Kirchoff's Analysis



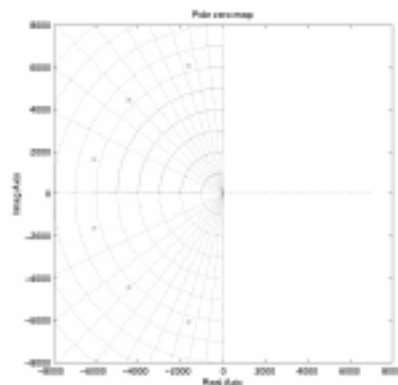
- Transfer function $\frac{v_{out}}{v_{in}} = H(s) = \frac{K}{sRC + 1}$
- Position of poles and zeroes on s-plane
- Advantages of “active” filtering

Transfer function - Generally

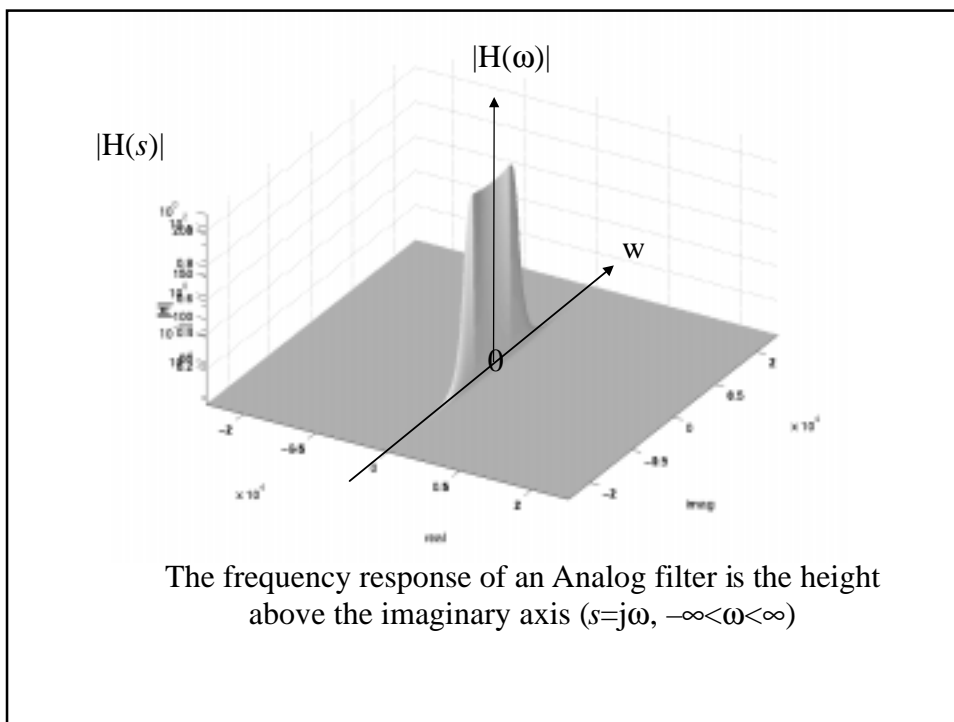
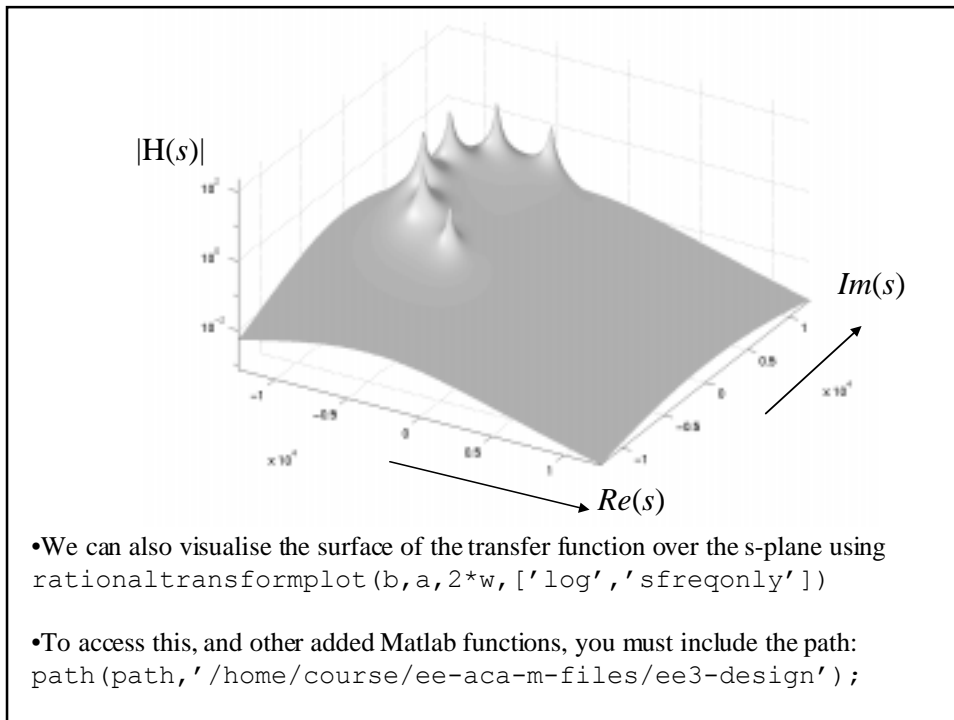
$$H(s) = \frac{b_0 + b_1s + b_2s^2 + \dots + b_ns^n}{a_0 + a_1s + a_2s^2 + \dots + a_ms^m}$$

$$= \frac{(s - z_0)(s - z_1)\dots(s - z_n)}{(s - p_0)(s - p_1)\dots(s - p_m)}$$

- We can express a polynomial in Matlab as a row vector, $a = [a_m \dots a_1 a_0]$ or by its poles as a column vector, $p = [p_0; p_1; \dots; p_m]$ etc.
- We can change between these representations using $p = \text{roots}(a)$ or $a = \text{poly}(p)$
- We can plot the poles and zeroes on the s-plane using $\text{pzmap}(b, a)$ or $\text{pzmap}(p, z)$ and display the frequency response using $\text{bode}(b, a)$ or $\text{freqs}(b, a)$ etc.
- (Note the subtle difference between the `bode` and `freqs` plots.)



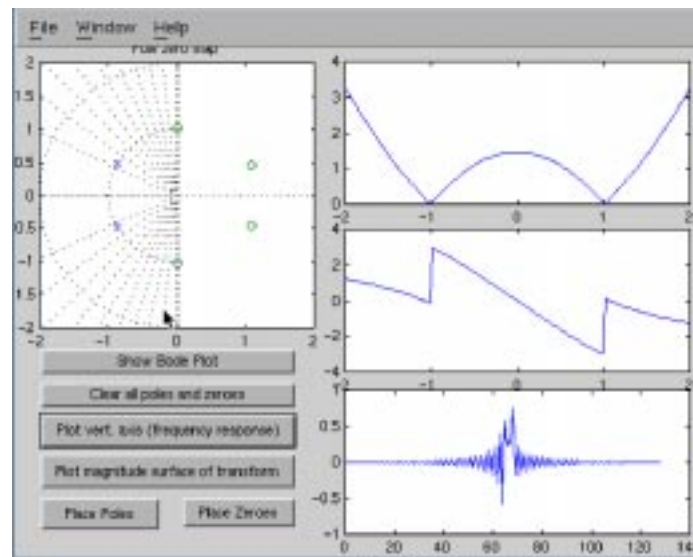
The pole-zero map of an all-pole, sixth order, Butterworth LPF



Poles and Zeroes (Analog Filters)

- Poles and zeroes must occur in *conjugate pairs* to implement a “real” system unless they exist on the real axis.
- Poles must fall on the *left* side of the s-plane for stability
- You must have *more poles than zeroes* for physical realisability (causality)
- Typically low pass filters have zeroes only at infinity, bandpass has zeroes at both $s=0$ and infinity, and high pass involves zeroes at $s=0$.
- For each pole introduced the cutoff frequency may fall by 20 dB/decade or 6 dB/octave
- The *order* of the filter is the number of finite poles
- Try the positioning of poles and zeros using mySpacement
(Concentrate on poles in particular.)

The GUI of the mySpacement tool.



Transfer function (2nd order)

- One can determine the transfer function of a circuit using Kirchoff's analysis
- For the equal component low pass Sallen Key implementation

$$H(s) = \frac{K / (R_1 R_2 C_1 C_2)}{s^2 + \left[\frac{1}{R_2 C_1} + \frac{1}{R_1 C_1} + (1-K) \frac{1}{R_2 C_2} \right] s + \frac{1}{R_1 R_2 C_1 C_2}}$$

- In general, a high pass filter will have a transfer function;

$$H(s) = \frac{Ks^2}{s^2 + d\omega s + \omega^2}$$

where K is the DC gain, d is the "damping" or bluntness of the knee of the bode plot.

Higher Order Filters

- Repeated poles, or "cascaded" second order sections. $H(s) = H_1(s) \cdot H_2(s) \cdot H_3(s)$. Just collect poles and zero pairs, design second order sections for each pair, then feed the output of the first section into the second, etc...
- Alternatively implement in "parallel" so $H(s) = H_1(s) + H_2(s) + H_3(s)$. Use `residue` to do the partial fraction expansion.
- Watch out for impedance mismatch between sections.
- SIDE NOTE:
 - Discrete capacitor values will limit the design. Once designed and implemented, you can use the chosen values in a further Matlab analysis to determine changes to pole positions etc.

Common Analog Designs

- There are many respected and well used filter designs which you can adapt to your situation:
- Butterworth filters give maximally smooth response
`butter(N,w,'s'[,type])`
- Chebyshev filters trade off E dB of ripple in pass or stop band for sharper transitions: `cheby1(N,w,E,'s')` and `cheby2(N,w,E,'s')`. The order of the filter, N , determines the number of ripples.
- To determine the order, N , of a Chebyshev filter required for a certain transition roll-off, use `chebyord`
- If you have a vector of desired (complex valued) frequency response H , at a set of specified frequencies, W , you can get Matlab to design the filter using `invfreqs(H,W,'s')`

Impulse Invariant Technique

- A Laplace transform of the form $H(s) = \frac{A_i}{s - p_i}$ results in an impulse response of;

$$h(t) = \begin{cases} A_i e^{p_i t} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

- If we sample this impulse response at period T , we obtain the z -transform of form;

$$H(z) = \sum_{i=0}^{\infty} A_i e^{ip_i T} z^{-k}$$

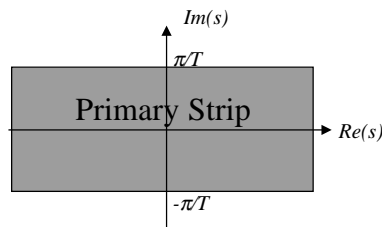
- Which is a geometric power series, for which we can write the convergent sum;

$$H(z) = \frac{A_i}{1 - e^{p_i T} z^{-1}} = \frac{A_i z}{z - e^{p_i T}}$$

- i.e. a pole at p in the s plane becomes a zero at $z=0$, and a pole at $z=e^{pT}$

Impulse Invariant Tech. Contd.

- So we take an analog transfer function, rearrange it by a partial fractions expansion into single pole filters (filters in parallel), perform the pole mapping, and reassemble as a z -transform expression.
- There is one drawback to this, in that it relies on us adequately sampling the analog impulse response. Thus only points on the s -plane with an imaginary component between $-\pi/T$ to π/T (the ‘primary strip’) will be mapped correctly. Values outside this range will be aliased back into the primary strip.



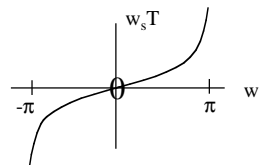
Bilinear Transformation

- We can perform a simple transformation from s -domain to z -domain that avoids the aliasing problem

$$H(z) = H(s) \Big|_{s=\frac{2(1-z^{-1})}{T(1+z^{-1})}}$$

- The transformation uses a non-linear mapping from the imaginary axis of the Laplace domain to the unit circle of the z plane, with infinity being mapped to π . Therefore low frequency features transfer reasonably well but the higher frequencies are compressed.
- To alleviate this ‘frequency warping’ we can pre-warp the filter prior to the transformation.

$$\omega_z = 2 \tan^{-1} \left(\omega_s \frac{T}{2} \right)$$



Getting back to Digital Filters....

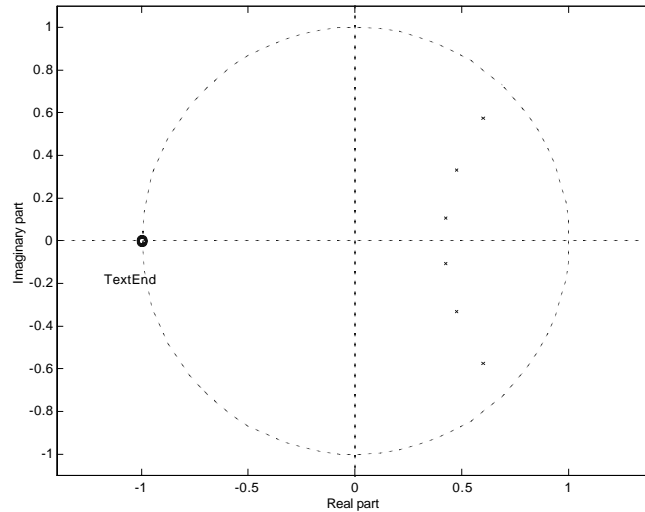
- The previous few slides indicate to you how analogue filters may be designed using the Laplace transform.
- Be aware there are differences from the z -transform:
 - We evaluate $H(f)$ around the unit circle
 - Poles are stable if within the unit circle
 - Poles at $z=0$ just delay the output by one sample
 - The frequency domain is from $-\pi$ to π , with aliases ad-infinitum (round and round the unit circle)
 - Of course, we use $H(z)$ to plot the surface.

Transfer function - Generally

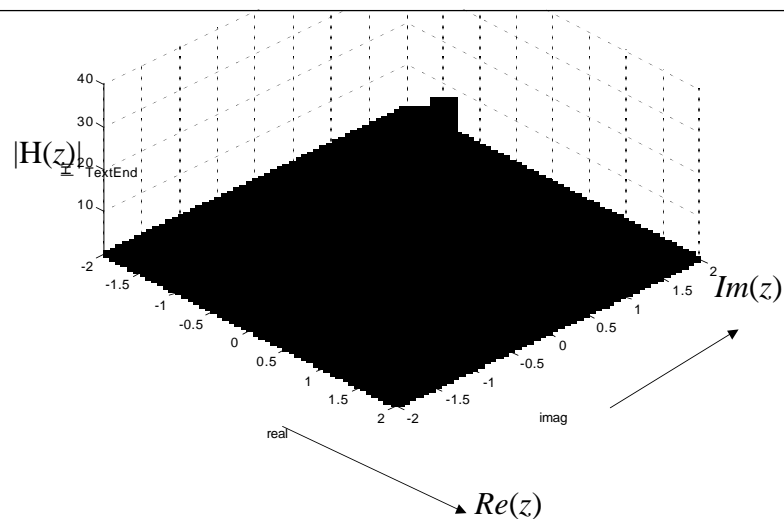
$$H(z) = \frac{b_0 + b_1z + b_2z^2 + \dots + b_nz^n}{a_0 + a_1z + a_2z^2 + \dots + a_mz^m}$$

$$= \frac{(z - z_0)(z - z_1)\dots(z - z_n)}{(z - p_0)(z - p_1)\dots(z - p_m)}$$

- We can express a polynomial in Matlab as a row vector, $a = [a_m \dots a_1 a_0]$ or by its poles as a column vector, $p = [p_0; p_1; \dots; p_m]$ etc.
- We can change between these representations using $p = \text{roots}(a)$ or $a = \text{poly}(p)$
- We can plot the poles and zeroes on the s -plane using $\text{pzmap}(b, a)$ or $\text{pzmap}(p, z)$ and display the frequency response using $\text{bode}(b, a)$ or $\text{freqz}(b, a)$ or $\text{zplane}(b, a)$ or $\text{zplane}(p, z)$ etc.
- (Note the subtle difference between the bode and freqz plots.)

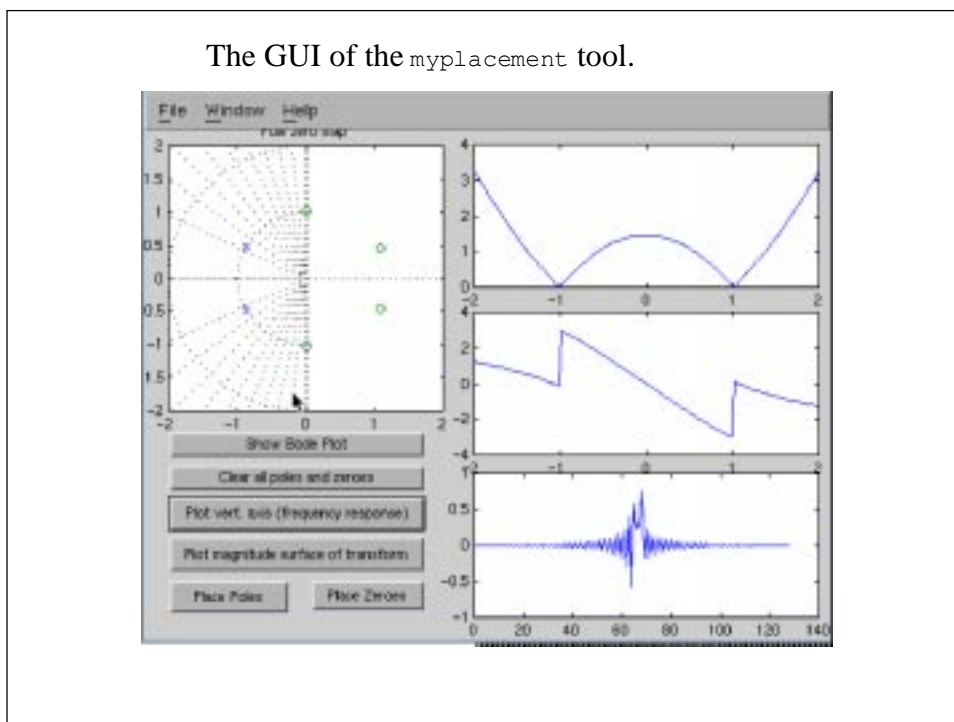
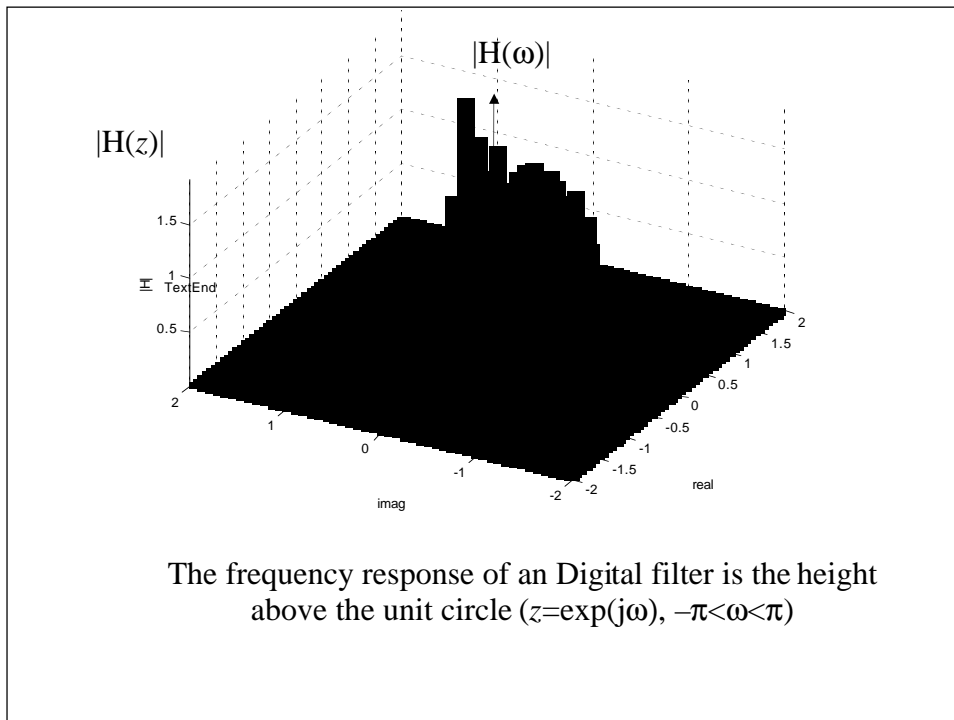


The pole-zero map of an all-pole, sixth order, digital Butterworth LPF



•We can also visualise the surface of the transfer function over the z-plane using `rationaltransformplot(b,a,2*w,['log','zring'])`

•To access this, and other added Matlab functions, you must include the path:
`path(path,'/home/course/ee-aca-m-files/ee3-design');`



Phase distortion

- The advantage of using feedback in the IIR filter over using a FIR filter is clear! However, one of the sacrifices made is that the filter is no longer “linear phase”, i.e., we can no longer just delay the taps of the filter and obtain the same (albeit delayed) response.
- It is therefore important that we investigate how the phase response of the filter is going to distort the signal.
- One way around this distortion is to use a two pass method (“*filtfilt*”) involving time reversal of the signal (This of course means we must truncate the impulse response of the filter so we may reverse it.)
 - Step 1: Pass the signal through the IIR filter
 - Step 2: Reverse the output
 - Step 3: Pass this signal through the same filter
 - Step 4: Reverse the output
- The result obtained has the desired linear phase, and the designed magnitude response.

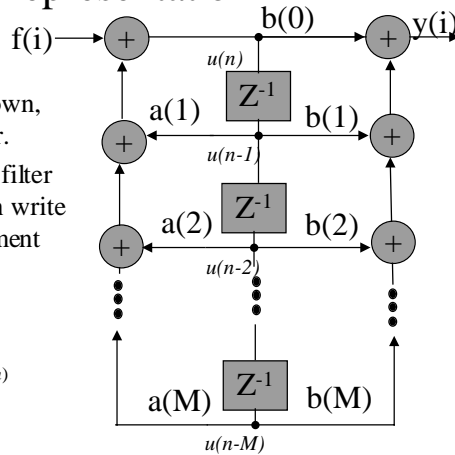
State Space Representation

- The regular direct form II for filter implementation can be drawn as shown, where the $u(n)$ are the states of filter.
- Let $s_k(n) = u(n-k)$ be the states of the filter on the k th time through, then we can write a “state equation” and an “measurement equation” to represent the system;

$$\begin{bmatrix} s_1(n+1) \\ s_2(n+1) \\ \vdots \\ s_M(n+1) \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & \dots & a_M \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} s_1(n) \\ s_2(n) \\ \vdots \\ s_M(n) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} x(n)$$

$$y(n) = [c_1 \quad c_2 \quad \dots \quad c_N] \begin{bmatrix} s_1(n) \\ s_2(n) \\ \vdots \\ s_N(n) \end{bmatrix} + b_0 x(n)$$

where $c_k = b_k + b_0 a_k$



$$H(z) = \frac{\sum_{i=0}^M b(i)z^{-i}}{1 - \left(\sum_{i=1}^M a(i)z^{-i}\right)}$$

State Space Representation (contd)

- Or more tidily put... $\mathbf{s}(n + 1) = \mathbf{A}\mathbf{s}(n) + \mathbf{B}x(n)$
 $y(n) = \mathbf{C}\mathbf{s}(n) + \mathbf{D}x(n)$

- These matrix equations suggest many permutations of the implementation diagram, but the main advantage of this representation is that the impulse response may be determined without resorting to the z-transform or partial fraction decompositions...

$$s(n) = \begin{cases} 0 & n \leq 0 \\ \sum_{k=0}^{M-1} \mathbf{A}^{M-k-1} \mathbf{B}x(k) & n > 0 \end{cases} \quad y(n) = \begin{cases} 0 & n \leq 0 \\ \mathbf{D}x(0) & n = 0 \\ \sum_{k=0}^{M-1} \mathbf{C}\mathbf{A}^{M-k-1} \mathbf{B}x(k) + \mathbf{D}x(n) & n > 0 \end{cases}$$

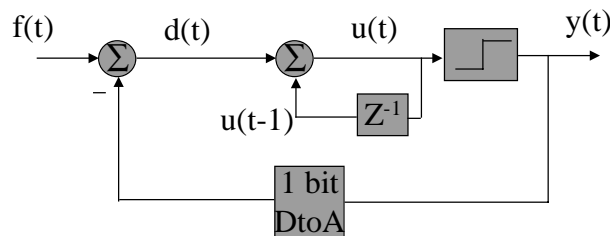
$$H(z) = \mathbf{C}(z\mathbf{I}_{M \times M} - \mathbf{A})^{-1} \mathbf{B} + \mathbf{D} = \mathbf{C} \frac{\text{adj}(z\mathbf{I}_{M \times M} - \mathbf{A})}{\det(z\mathbf{I}_{M \times M} - \mathbf{A})} \mathbf{B} + \mathbf{D}$$

- Finally, this representation is easily transmuted into a recursive filter such as the Kalman filter...

A to D continued

- OVERSAMPLING ($\Sigma-\Delta$)

$$S/N = 20 \log_{10} \left(\frac{3}{\sqrt{2\pi}} \right) + 9 \log_2(M)$$

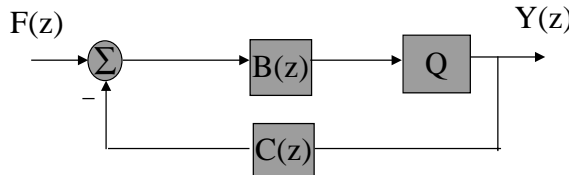


$$B(z) = 1 + z^{-1}$$

$$H_{ns}(z) = 1/(1 + z^{-1}) = z/(z + 1)$$

OVERSAMPLING ($\Sigma-\Delta$) AtoD contd.

$$Y(z) = B(z)[F(z) - C(z)Y(z)] + Q = \frac{B(z)F(z) + Q}{1 + B(z)C(z)}$$



- Now let $C(z) = 1 - \frac{1}{B(z)}$

and $B(z) = \frac{1}{H_{ns}(z)}$

then $Y(z) = F(z) + H_{ns}(z) \cdot Q$

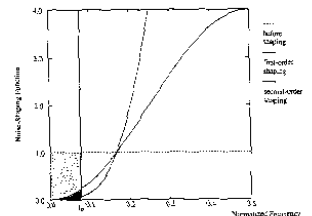


FIGURE 5.10 Effect of oversampling on the signal-to-noise ratio. The curves show the magnitude of the signal and the magnitude of the noise after reconstruction. The noise is the area under the curve. The signal is the area under the curve. The noise is the area under the curve. The signal is the area under the curve.

Cepstrum

- The spectrum of the cepstrum is:

$$X_{cep}(z) = \ln\{X(z)\}$$

ensuring the phase of $X(z)$ is unwrapped.

- So the cepstrum (rceps) is:

$$x_{cep}(i) = ZT^{-1}\{X_{cep}(z)\}$$

- So what is it useful for? Consider corruption that is convolved with the signal (e.g. channel) – taking \ln of the z -transform product results in additive components:

$$\ln\{X(z)H(z)\} = X_{cep}(z) + H_{cep}(z)$$