Brief Notes on Digital Filters

Z-transforms

$$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n}$$

For us the x_n are usually samples from a signal.
Remember that terms in the series must approach zero as n tends to $\pm\infty$. This defines the region of convergence.

Most often we use Geometric Progression formulae to do the sums which can then be expressed as: $\sum_n a_o a^n$

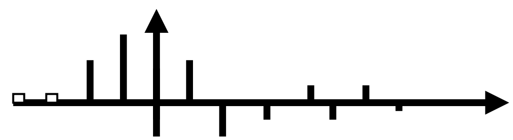
Finite number of terms: sum = $a_o \dfrac{1-a^N}{1-a}$

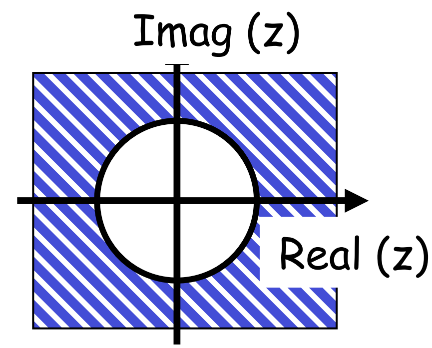
Infinite number of terms: sum = $a_o \dfrac{1}{1-a}$ $\quad |a|<1$

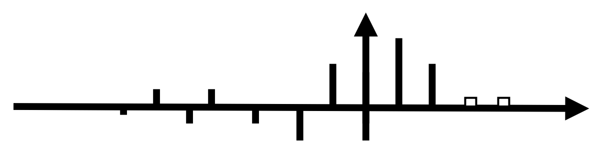
Z-transforms - Continued

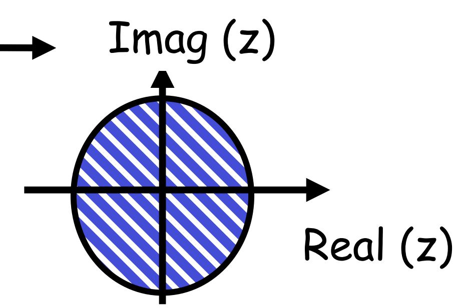
Regions of convergence in the z-plane

Of the form $|z| >$ some value
(defines a region outside a circle)
when the signal is zero for $-\infty < n < n_0$
and then has values for $n_0 + 1 < n < +\infty$









Of the form $|z| <$ some value
(defines a region inside a circle)
when the signal has values for $-\infty < n < n0$
and then is zero for $n_0 + 1 < n < +\infty$

Inverse z-Transforms $X(z)$ to x_n

Definition:

$$x_n = \frac{1}{2\pi j} \oint z^{n-1} X(z)\, dz = \sum residues$$

Contour of integration inside region of convergence

Residues only at poles inside contour of integration

SIMPLE POLE at $z = zo$, residue is:

$$lim\ z \to zo \quad \left((z - zo)\, z^{n-1} X(z) \right)$$

M POLES at $z = zo$, residue is:

$$lim\ z \to zo \quad \frac{1}{(m-1)!} \frac{d^{(m-1)}}{dz^{(m-1)}} \left((z - zo)^m\, z^{n-1} X(z) \right)$$

Inverse z-Transfroms X(z) to xn

Other methods:

1. Express as known z-transforms, through partial fraction expansions.
2. Long division

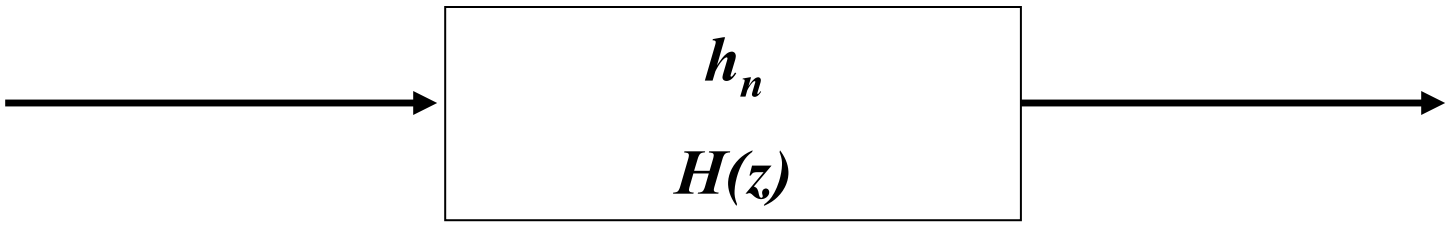
In both cases, use the region of convergence to tell you whether you want to land up with a signal decaying away to zero for:

positive time, $x_n = 0$ for $n < no$

negative time, $y_n =$ for $n > no$

or both, split into two parts a positive and a negative time part.

Digital Systems (IIR Filters)



Often $H(z)$ is in a polynomial form:

$$H(z) = \frac{b_o + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots \quad b_{NB} z^{-NB}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots \quad a_{NA} z^{-NA}} = \frac{Y(z)}{X(z)}$$

This is an infinite impulse response (IIR) filter.

Difference Equation: [recall $Z\{x_{n-q}\} = z^{-q} X(z)$]

$$\begin{aligned} y_n = {} & b_o x_n + b_1 x_{n-1} + b_2 x_{n-2} + \ldots\ldots \quad b_{NB} x_{n-NB} \\ & - a_1 y_{n-1} - a_2 y_{n-2} - \ldots\ldots \quad a_{NA} y_{n-NA} \end{aligned}$$

Digital Systems (FIR Filters)

Some digital systems have transfer functions [H(z)] like:

$$H(z) = b_{-M}z^{+M} + b_{-M+1}z^{+M-1} + \ldots\ldots\ \ b_{-1}z^{+1} + b_o + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots\ b_M z^{-M} = \frac{Y(z)}{X(z)}$$

This is a Finite Impulse Response (FIR) Filter.

Difference equation is:

$$y_n = b_{-M}x_{n+M} + b_{-M+1}x_{n+M-1} + \ldots\ldots\ \ b_{-1}x_{n+1} + b_o x_n + b_1 x_{n-1} + b_2 x_{n-2} + \ldots\ldots\ \ b_M x_{n-M}$$

Note that this 2M+1 length filter is non-causal, and y_n depends on future as well as past values of x_n.

Digital Systems (FIR Filters) continued

Coefficients give the impulse response of these FIR filters:

$h_n = b_n$ for $-M \le n \le +M$; $h_n = 0$ for $|n| > M$.

Note that the response is a convolution of xn with the impulse response $h_n = b_n$:

$$y_n = \sum_{k=-M}^{M} b_k x_{n-k} = \sum_{m=n-M}^{n+M} b_{n-m} x_m$$

Fastest way to implement this, unless M is very small, is through convolution via FFTs, not forgetting to zero pad appropriately.

Non Causal IIR Digital Filters

You can have non-causal IIR filters too,

(response now is a function of future values of the response as well as current and future values of the input)

but they are tricky to implement.

We split the transfer function into causal and acausal parts:

$H(z) = H_c(z) \,.\, H_{ac}(z)$ where:

$$H_c(z) = \frac{b_o + b_1 z^{-1} + \ldots\ldots}{1 + a_1 z^{-1} + \ldots\ldots} \qquad H_{ac}(z) = \frac{c_o + c_1 z^{+1} + \ldots\ldots}{1 + d_1 z^{+1} + \ldots\ldots}$$

Non Causal IIR Digital Filters (cont)

$H_c(z)$ is implemented in the usual way

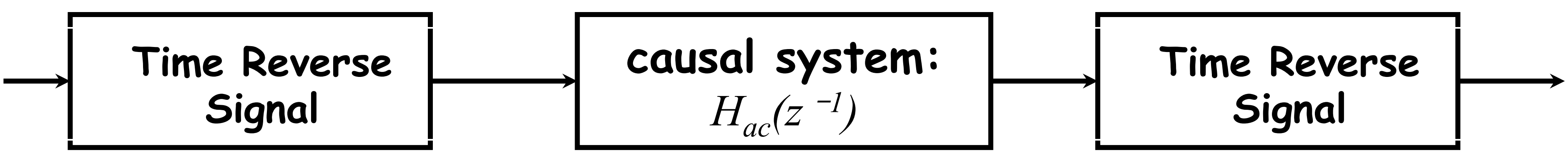
(see difference equation on slide 5)

The output of this becomes the input to $H_{ac}(z)$

$H_{ac}(z)$ is implemented by using:

$$y_n = c_o x_n + c_1 x_{n+1} + c_2 x_{n+2} + \ldots\ldots\ldots \quad c_{NC} x_{n+NC}$$
$$- d_1 y_{n+1} - d_2 y_{n+2} - \ldots\ldots\ldots \quad d_{NA} y_{n+ND}$$

To implement this you start at the end of the input series and work back towards the start. This is equivalent to:



Frequency Response of Digital Filters

Evaluate H(z) around the unit circle

$$H(z) = \left. \frac{b_o + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots \quad b_{NB} z^{-NB}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots \quad a_{NA} z^{-NA}} \right|_{z=exp(j2\pi f\Delta)}$$

and $f = k.fs/N,\ k = 0, 1, \ldots. N\text{-}1$.

This can be time-consuming. Note: you are actually doing:

$$H(z) = \frac{DFT\{b_o, b_1, b_2, \ldots. b_{NB}, 0, 0, 0 \ldots. 0_{N-1}\}}{DFT\{1, a_1, a_2, \ldots\ldots., a_{NA}, 0, 0, \ldots. 0_{N-1}\}}$$

and hence this can be done efficiently in MATLAB by using:

H_freq_resp=fft{b,N}./fft{a,N}

Make N very large (and a power of 2 for efficiency) to get a finely resolved spectrum.

Digital Filter Design

All digital systems are filters but we usually we design filters to:

- remove noise from a signal
- differentiate or integrate a signal
- calculate the Hilbert transform of a filter

We also sometimes design filters

- to simulate physical systems
- to act as controllers (not in this class)

FIR Filter Design (Brief Overview)

Method 1: (Sample in time) N = 2M+1 point filter

- Start with H(f), the desired frequency response of an analog filter
  E.g. High-pass:
  $H(f) = 1$ for $|f| > f_c$, and $H(f) = 0$ for $|f| < f_c$
  This is a severe change at $|f| = f_c$, which is not really a good idea, having a gentler transition is desirable.

- Band limit putting $H(f) = 0$ for $|f| > fs/2$.
- Inverse Fourier Transform (analytically) to obtain $h(t)$.
- Sample and scale to obtain expressions for $\Delta h(n\Delta)$.
- Window to have finite sequence $-M \le n \le M$ and evaluate to obtain coefficients: b_j for $j = -M, -M+1, \ldots, -1, 0, 1, \ldots, M$.
  A window that → zero smoothly at $\pm M$ is desirable.

FIR Filter Design (continued)

Method 2: (Sample in frequency) N point filter

- Start with H(f) the desired frequency response for $0<f<fs/2$. Beware of sharp transitions as with method 1.
- Sample $H(f)$ at $f=k.fs/N$ $k=0,1,\ldots .N/2$ to get $Bf(k)$ for $k=1,2,\ldots(N/2)+1$
- Specify $Bf(k+N/2+1) =$ complex conjugate of $Bf(N/2+1-k)$ for $k=1,2,\ldots(N/2)-1$. *(Symmetry condition for real filter coefficients).*
- Inverse Discrete Fourier Transform (IFFT) to get coefficients.
- Rearrange, if you didn't do the phase adjustment in frequency, to move the last $N/2$ points of the filter to the start of the filter. Now the coefficients in the vector correspond to times: $-(N/2)\,\Delta \le t \le ((N/2)-1)\,\Delta$ instead of $0 \le t \le (N-1)\,\Delta$.

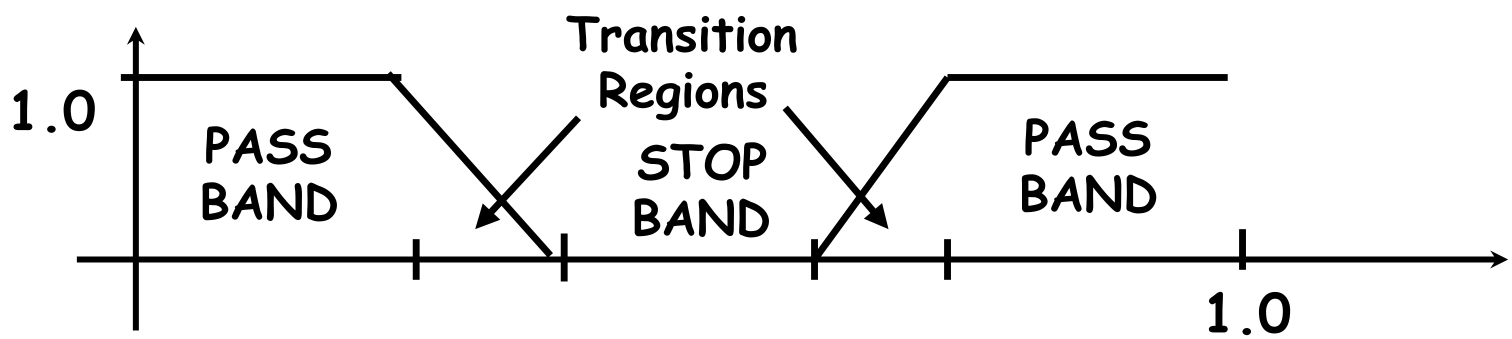
FIR Filter Design (continued)

Method 3 (Remez Exchange/McCellan Parks Algorithm)

$N=2M+1$ point FIR filter.

– This is a nonlinear optimization algorithm that tries to ensure the frequency response error is uniform. (Doesn't always converge.)
– You specify a frequency and an amplitude vector that specifies the desired frequency response from 0 to *fs*/2.

Example: Band-stop filter. F=[0 0.3 0.35 0.65 0.7 1.0]; A=[1 1 0 0 1 1].



Normalized Frequency: 1.0 corresponds to fs/2.

– Can weight the importance of error in each band. E.g., w=[0.9 0.1 0.9]

FIR Filter Design-Validation

Always check out the frequency response of designed FIR filter by zero-padding h_n to about 8 times its length and taking the DFT (fft in Matlab is fast if zero padded length is a power of 2.)

Remember the DFT always assumes that the data starts at t=0. You have to adjust the phase yourself to account for time-delays or advances.

The impulse response should → zero at the filter ends.
It not doing so indicates there are problems with the design. You will see large ripples in the frequency response when this happens.
Rectify by:

- Increasing the length of the filter
- Smoothing transition regions in the frequency domain
- Windowing the impulse response with a smoother window

Differentiators etc.

Differentiator: $H(f) = j\, 2\, \pi\, f$

Amplifies high frequency noise.

Digital filter will have a sharp transition at fs/2.

Often a good idea to combine this with a low-pass filter…… also to smooth function at fs/2.

Integrator: $H(f) = 1/\{j\, 2\, \pi\, f\}$

Amplifies low frequency noise.

Digital filter will have a sharp transition at 0 Hz.

Often a good idea to combine this with a high-pass filter…… also to smooth function at 0.

Hilbert Transformer: $H(f) = -j \operatorname{sign}(f)$.

Digital filter has sharp transition regions at f=0 & fs/2.

Good idea to smooth function in these regions.

IIR Filter Design

Analog Design Mapped into Digital Design

- Impulse invariant mapping
  H(s) → *h(t)* → *Δh(nΔ)* → *H(z)*
  - aliasing an issue, therefore not a good idea for design of, e.g., high-pass filters.
  - Stable in s-plane (poles in LHP) →
    stable in z-plane (poles inside unit circle)

- Bilinear mapping $s = (2/\Delta).(1 - z^{-1})/(1 + z^{-1})$
  Use *H(s)* and substitute every *s* with this function of *z*.
  - Frequency distortion, must pre-warp frequencies (digital design → analog design frequencies) to account for this in the design.

$$\omega_{analog} = (2/\Delta).tan(\omega_{digital}.\Delta/2)$$

  - Stability conserved, entire left half plane maps into unit circle.

IIR Filter Design (continued)

We only looked at an analog Butterworth low-pass filter

$$H(s).H(-s) = \frac{1}{1 + \left(\frac{s}{j\omega_c}\right)^{2N}}$$

Poles equispaced around a circle in the s-plane.
Radial positions of poles: $|s_k| = \omega_c$
Angular positions of $H(s)$ poles in Left Half Plane (LHP) are:

$$\angle s_k = \frac{\pi}{2} + (1 + 2k)\frac{\pi}{2N} \text{ rads, } k = 0,1,2,\ldots N-1.$$

LHP poles correspond to $H(s)$, RHP poles correspond to $H(-s)$.

E.g., N=3,

$$H(s) = \frac{1}{\left(1 - \frac{s}{s_1}\right)} \cdot \frac{1}{\left(1 - \frac{s}{s_2}\right)} \cdot \frac{1}{\left(1 - \frac{s}{s_3}\right)}$$

IIR Filter Design (continued)

Finding N and ω_c

Specify desired digital characteristics

- at $\omega_{digital\text{-}1}$ $20 \log 10 |H_{digital}| = -K1$ dB
- at $\omega_{digital\text{-}2}$ $20 \log 10 |H_{digital}| = -K2$ dB

Transform digital design to analog design (pre-warp)

- $\omega_{analog\text{-}1} = (2/\Delta) \tan (\omega_{digital\text{-}1} \Delta/2)$
- $\omega_{analog\text{-}2} = (2/\Delta) \tan (\omega_{digital\text{-}2} \Delta/2)$

- at $\omega_{analog\text{-}1}$ $10 \log 10 |H_{analog}|^2 = -K1$ dB (**)
- at $\omega_{analog\text{-}2}$ $10 \log 10 |H_{analog}|^2 = -K2$ dB (***)

IIR Filter Design (continued)

Use (**) and (***) to solve for N and ω_c

Round up N to make it integer.

Write down locations of poles in LHP of Butterworth filter with this N and ω_c:

s_1, s_2, s_3 etc.

Form $H(s)$, E.g., N=3

$$H(s) = \frac{1}{\left(1-\frac{s}{s_1}\right)}\cdot\frac{1}{\left(1-\frac{s}{s_2}\right)}\cdot\frac{1}{\left(1-\frac{s}{s_3}\right)}$$

Note that s_3 will be the complex conjugate of s_1, and when combined for a 2nd order filter with real coefficients.

Apply bilinear transform and rearrange to put $H(z)$ in standard form or as a cascade of 1st and 2nd order filters each in standard form:

$H(z) = H_1(z)H_2(z)$

$$s = \frac{2}{\Delta}\cdot\frac{(1-z^{-1})}{(1+z^{-1})},$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}.$$

Check out the resulting frequency response of H(z).

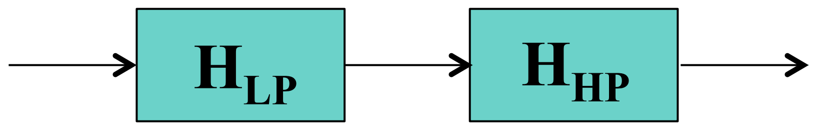
Implement using the difference equation(s).

For high order filters, refer to literature for robust implementations of the difference equations to avoid built up of rounding errors.

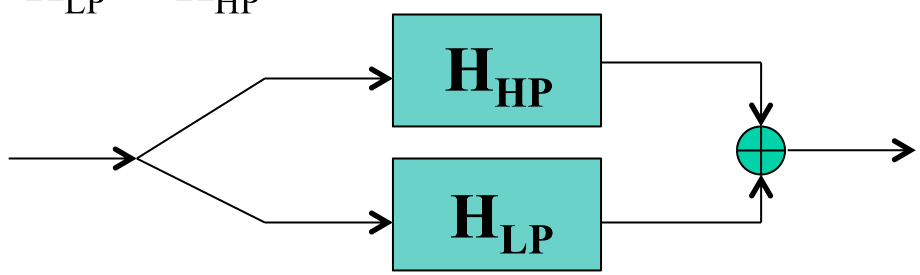
IIR High-pass, Notch and Band-pass Filters

You can transform a low-pass Butterworth filter into a high-pass filter by replacing (*s/jwc*) with (*jwc/s*) in the original design.

You can combine a low- and high-pass in series to produce a band-pass filter:
$H = H_{LP} \cdot H_{HP}$



You can combine a low- and high-pass in parallel to produce a band-stop (notch) filter: $H = H_{LP} + H_{HP}$



Or, you can use mappings to create analog Butterworth band-pass and band-stop filters in the first stage of the design.
(Similar to the high-pass mapping in the first bullet, but more complicated. See Oppenheim and Schafer or most any other Digital Filtering book.)