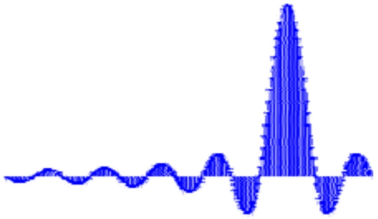
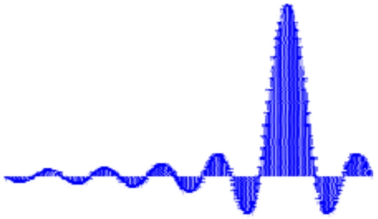


TODAY IN DSP

- Filter Structures (Chapter 8)
  - FIR filters
    - Canonic and Non-canonic forms
    - Linear Phase filter structures
    - Direct Form Type I and Type II realization
  - IIR Filter Structures
    - Direct Forms Type I and Type II realization
    - Cascade realization
    - Parallel realization

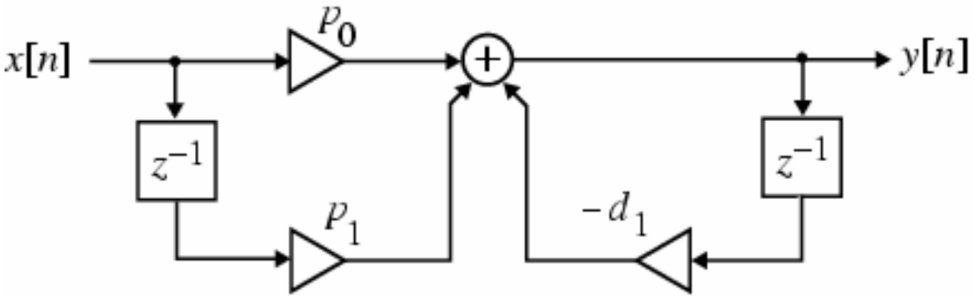


Filter Implementation

- Recall that the filters in real world are implemented in time domain.
  - However, how are they actually implemented in hardware (or software)?
- A block diagram representation of the filter is the first step to its implementation
- For example, the IIR filter

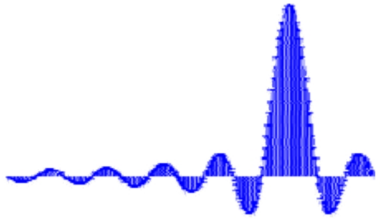
$$y[n] = -d_1 y[n-1] + p_0 x[n] + p_1 x[n-1]$$

can be implemented as follows:



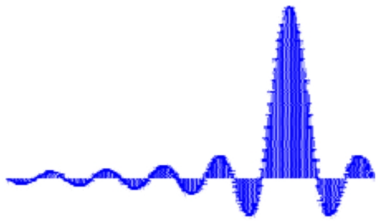
- Knowing the initial condition y[-1], and the input x[n] for n≥-1, we can calculate y[n], n ≥0

$$y[0] = -d_1 y[-1] + p_0 x[0] + p_1 x[-1]$$
$$y[1] = -d_1 y[0] + p_0 x[1] + p_1 x[0]$$
$$y[2] = -d_1 y[1] + p_0 x[2] + p_1 x[1]$$
$$\vdots$$



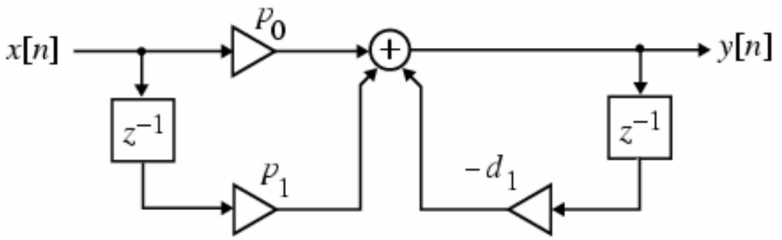
Advantages of such Implementation

- Using a block-diagram based implementation has several advantages:
  - Easy to write down the computational algorithm by inspection
  - Easy to analyze the block diagram to determine the explicit relation between the output and input;
  - Easy to manipulate a block diagram to derive other "***equivalent***" block diagrams yielding different computational algorithms
  - Easy to determine the hardware requirements
  - Easier to develop block diagram representations from the transfer function directly



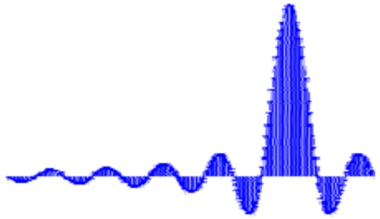
Canonic, NonCanonic & Equivalent Structures

- A digital filter structure is said to be **canonic** if the number of delays in the block diagram representation is equal to the order of the transfer function. Otherwise, it is a **noncanonic** structure.
  - This structure is not canonic, since it uses two delay elements for a first order filter



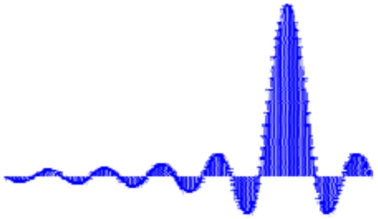
$$y[n] = -d_1 y[n-1] + p_0 x[n] + p_1 x[n-1]$$

- Two digital filter structures are called **equivalent** if they have the same transfer function. One of many equivalent structures is obtained using the **transpose operation**.
  - Reverse all paths
  - Replace pick-off nodes by adders, and vice versa
  - Interchange the input and output nodes



Equivalent Structures

- There are literally an infinite number of equivalent structures realizing the same transfer function. Under **infinite precision** arithmetic any given realization of a digital filter behaves identically to any other equivalent structure
  - However, in practice, due to the finite wordlength limitations, a specific realization may behave very different from its other equivalent realizations. Hence, it is important to choose a structure that has the least quantization effects when implemented using finite precision arithmetic
  - We will see a few of the most popular implementation procedures

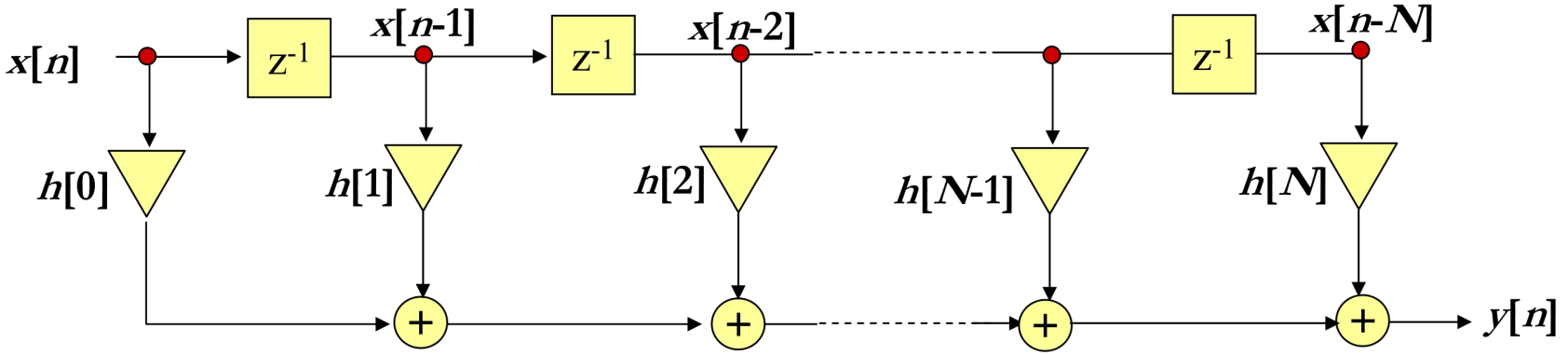


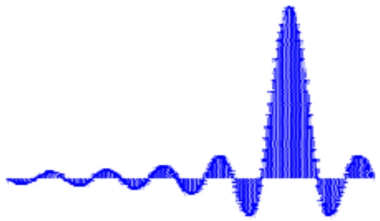
Basic FIR Structures

- A causal FIR filter can be represented in time domain with its CCLDE equation, which is equivalent to its impulse response representation:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{N} h[k]x[n-k]$$
$$= h[0]x[n] + h[1]x[n-1] + h[2]x[n-2] + \cdots + h[N]x[n-N]$$

- This Nth order (N+1-coefficient) filter can be implemented directly by using N+1 multipliers and N two-input adders

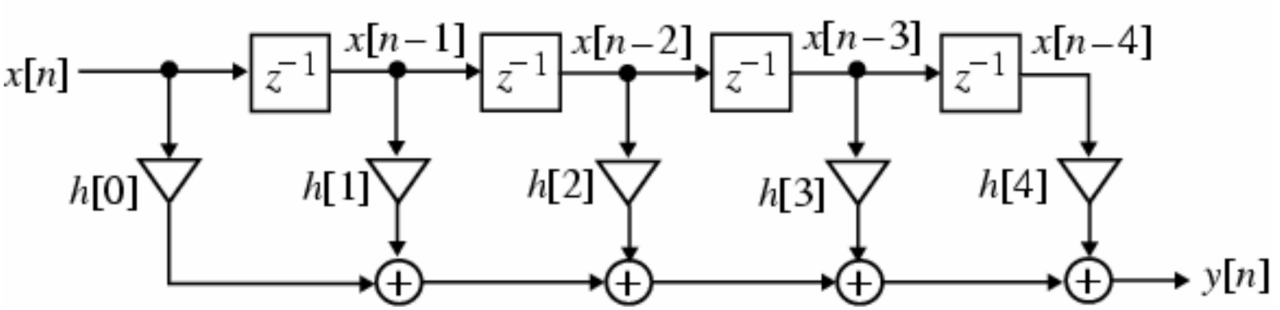




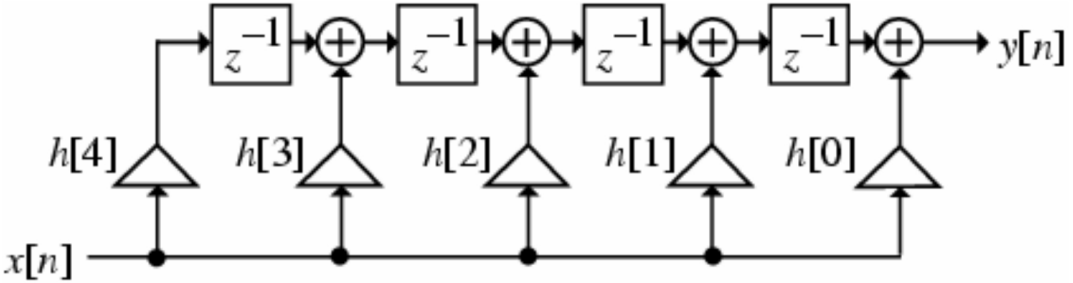
EXAMPLE

- For an order N=4 filter:

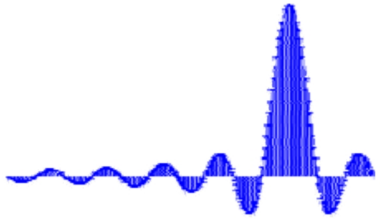
$$y[n] = h[0]x[n] + h[1]x[n-1] + h[2]x[n-2] + h[3]x[n-3] + h[4]x[n-4]$$



- and its transpose:



↳ Both are canonic structures. These structures are also known as ***tapped delay line*** or ***transversal*** filter structures



Linear Phase FIR Filter Structures

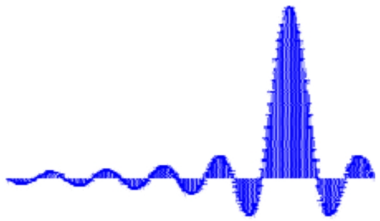
- Recall that a linear phase filter must have either symmetry or anti-symmetry property, which can be exploited to reduce the number of multipliers into almost half of that in the direct form implementations
- Consider a length-7 Type 1 FIR transfer function with a symmetric impulse response:

$$H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[2]z^{-4} + h[1]z^{-5} + h[0]z^{-6}$$

- W can rewrite this expression as

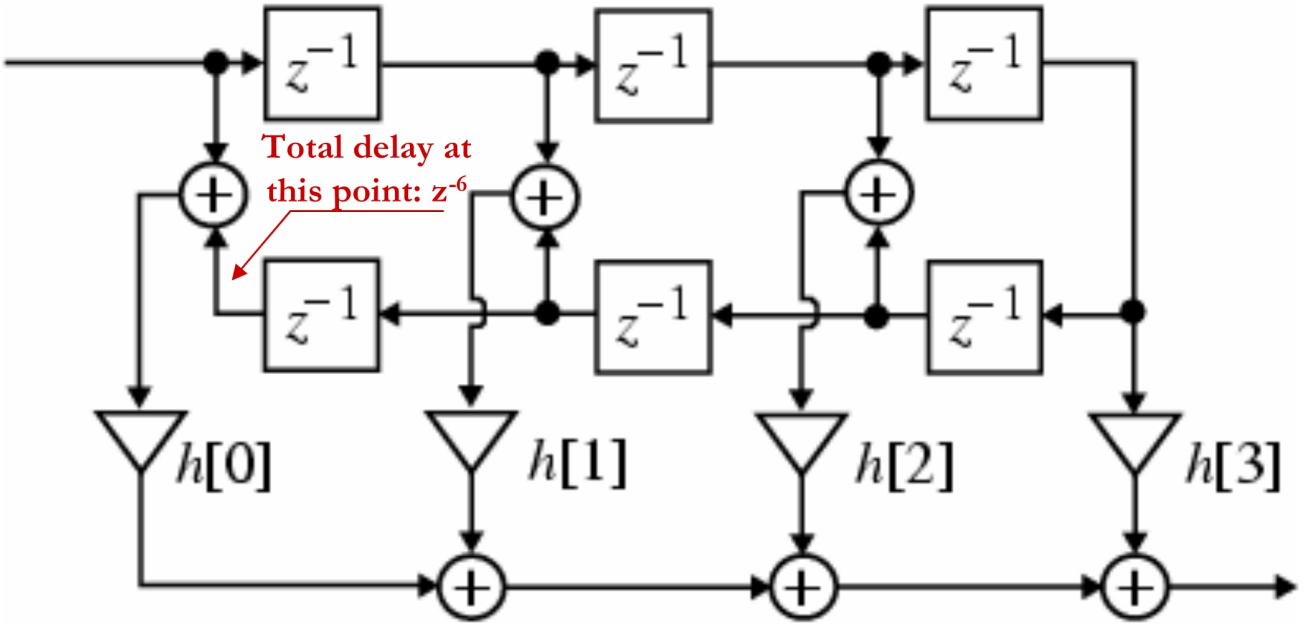
$$H(z) = h[0]\left(1 + z^{-6}\right) + h[1]\left(z^{-1} + z^{-5}\right) + h[2]\left(z^{-2} + z^{-4}\right) + h[3]z^{-3}$$

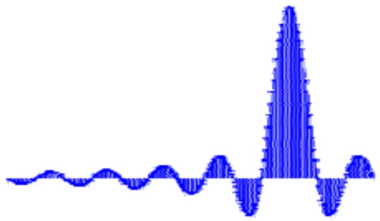
  - Which only needs four multipliers instead of seven



Linear Phase FIR Structures

$$H(z) = h[0]\left(1 + z^{-6}\right) + h[1]\left(z^{-1} + z^{-5}\right) + h[2]\left(z^{-2} + z^{-4}\right) + h[3]z^{-3}$$



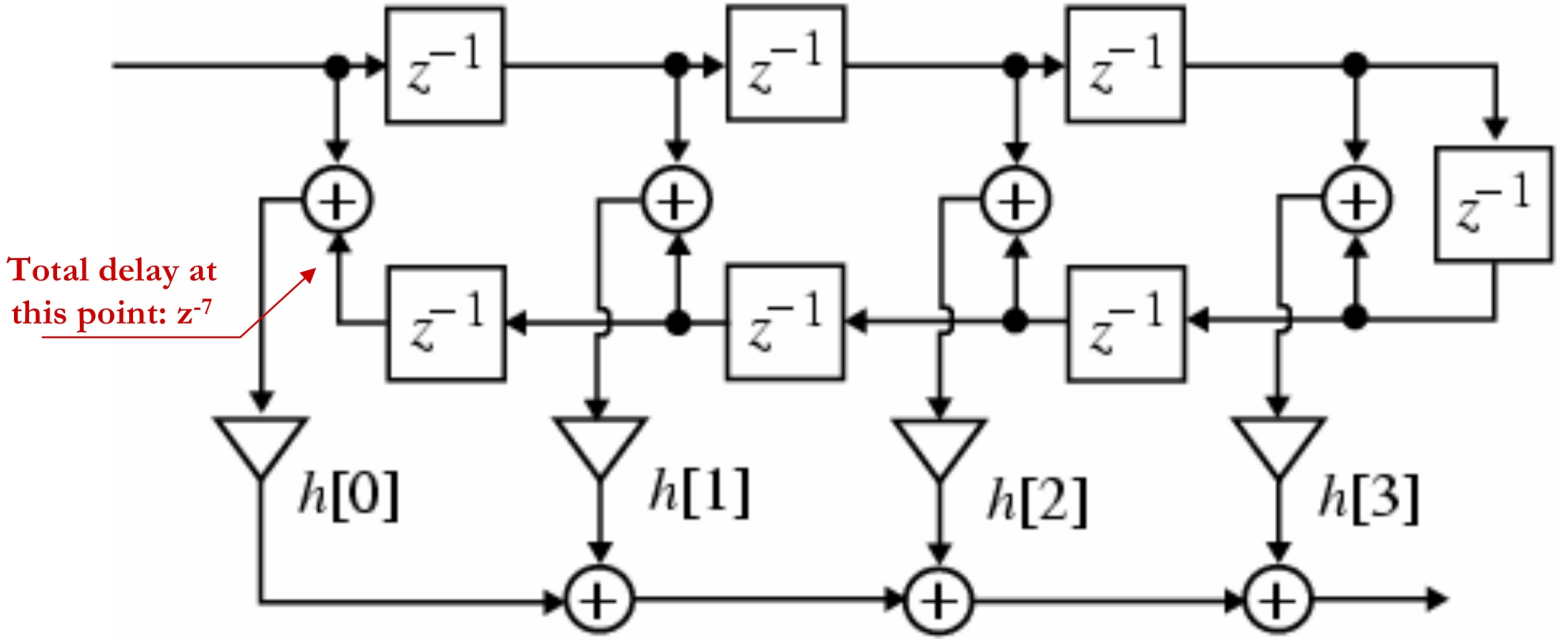


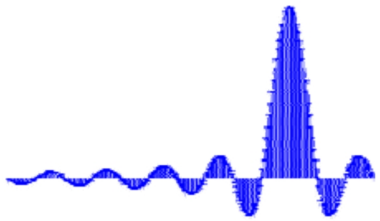
Linear Phase FIR Structures

➲ Similarly, and FIR II filter (with even length, say 8)

$$H(z) = h[0](1 + z^{-7}) + h[1](z^{-1} + z^{-6}) + h[2](z^{-2} + z^{-5}) + h[3](z^{-3} + z^{-4})$$

can be implemented with 4 multipliers, instead of 8. Note however, it still needs 8 delay elements





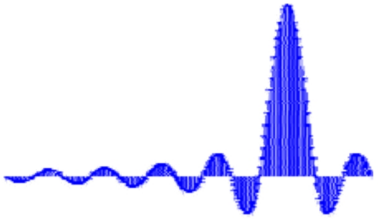
IIR Filter Structures

- The causal IIR digital filters are characterized by a real rational transfer function of z^{-1}, or equivalently, by a constant coefficient difference equation.
- From the difference equation representation, it can be seen that the realization of the causal IIR digital filters requires some form of feedback. Furthermore,
  - An N^{th} order IIR digital transfer function is characterized by $2N+1$ unique (***a*** and ***b***) coefficients, and in general, requires $2N+1$ multipliers and $2N$ two-input adders for implementation

$$\sum_{k=0}^{N-1} a_0\, y[n-k] = \sum_{l=0}^{M-1} b_0\, x[n-l]$$

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_{M-1} x[n-M+1] - a_1 y[n-1] - \cdots - a_{N-1} y[n-N+1]$$

- Given the filter CCLDE, we can implement it directly using the multiplier coefficients. This is called **Direct Form I** implementation.

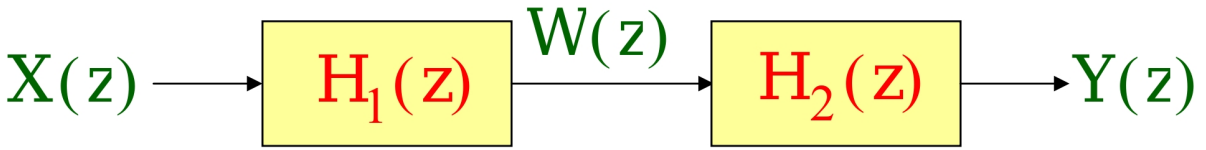


IIR Filter Structures

➲ Consider a 3rd order example:

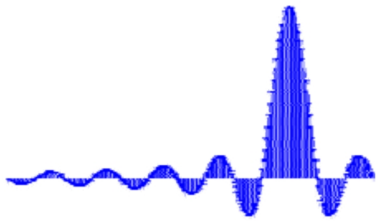
$$H(z) = \frac{P(z)}{D(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3}}{1 + d_1 z^{-1} + d_2 z^{-2} + d_3 z^{-3}}$$

↳ Which can be split into two systems of H_1 (numerator) and H_2 (denominator):



$$H_1(z) = \frac{W(z)}{X(z)} = P(z) = p_0 + p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3}$$

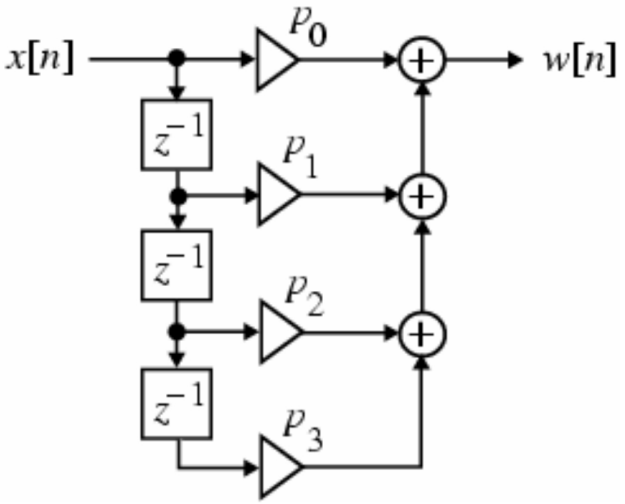
$$H_2(z) = \frac{Y(z)}{W(z)} = \frac{1}{D(z)} = \frac{1}{1 + d_1 z^{-1} + d_2 z^{-2} + d_3 z^{-3}}$$



IIR Filter Structures

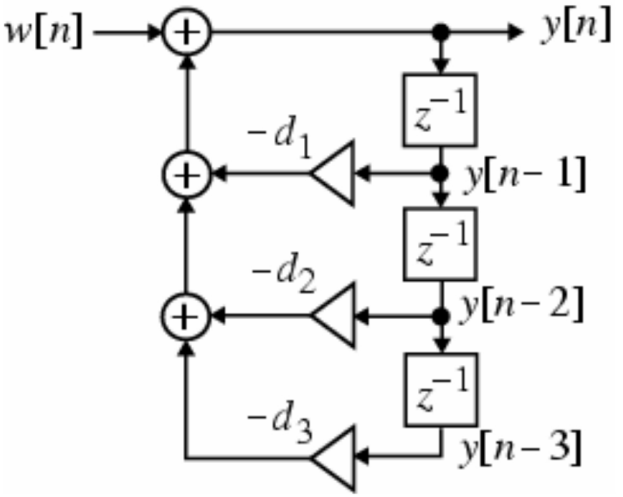
- Here is $H_1(z)$, whose input is x[n] and output is w[n]

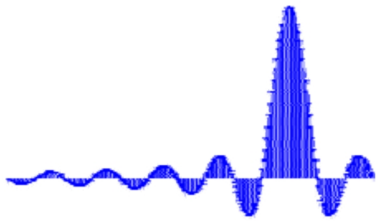
$$w[n] = p_0 x[n] + p_1 x[n-1] + p_2 x[n-2] + p_3 x[n-3]$$



- And, here is $H_2(z)$, whose input is w[n] and output is y[n]

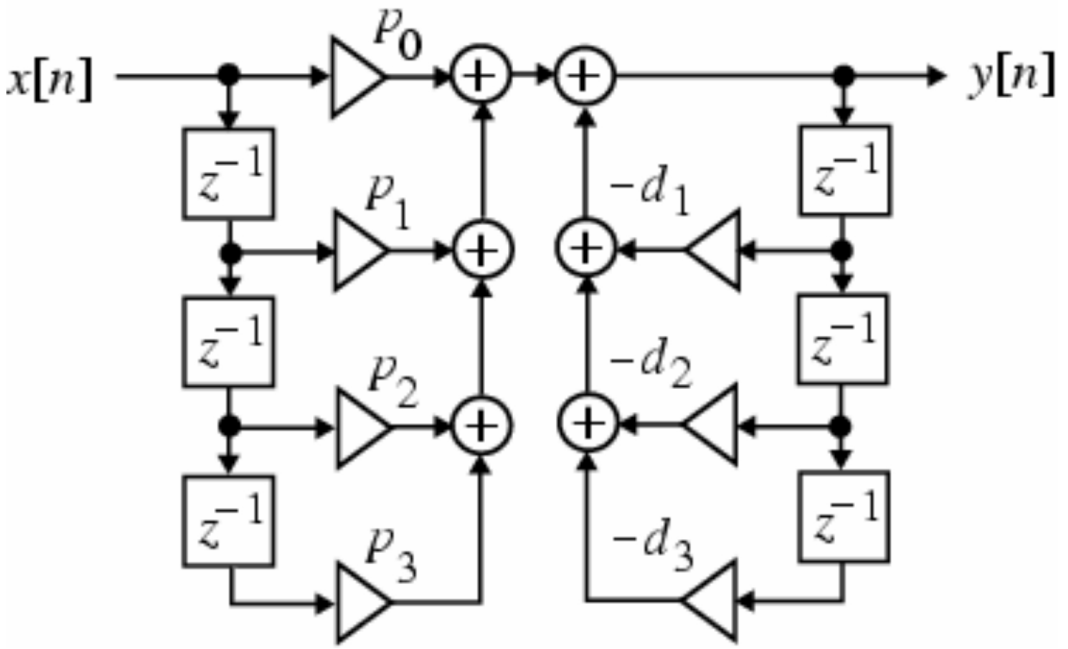
$$y[n] = w[n] - d_1 y[n-1] - d_2 y[n-2] - d_3 y[n-3]$$



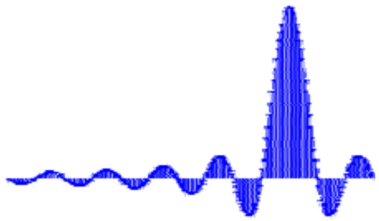


IIR Filter Structures: Direct Form I

- A cascade of the two then gives us the overall H(z), whose implementation is known as **Direct Form I** implementation

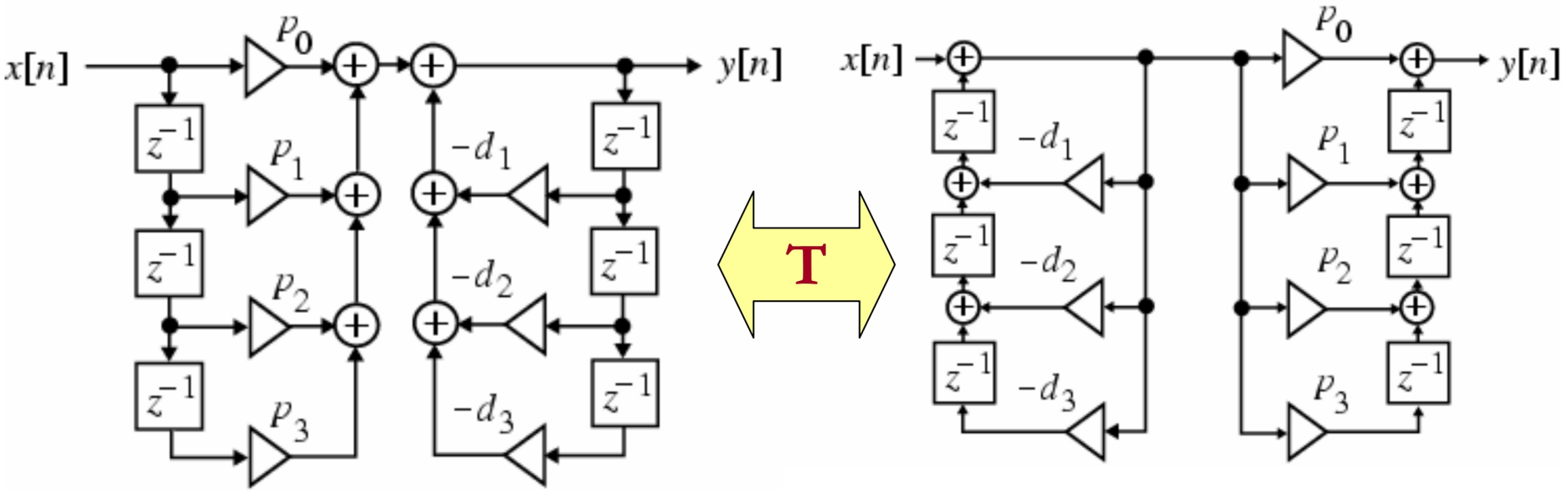


  - Note that this structure is ***noncanonic*** since it employs 6 delays to realize a 3rd-order transfer function



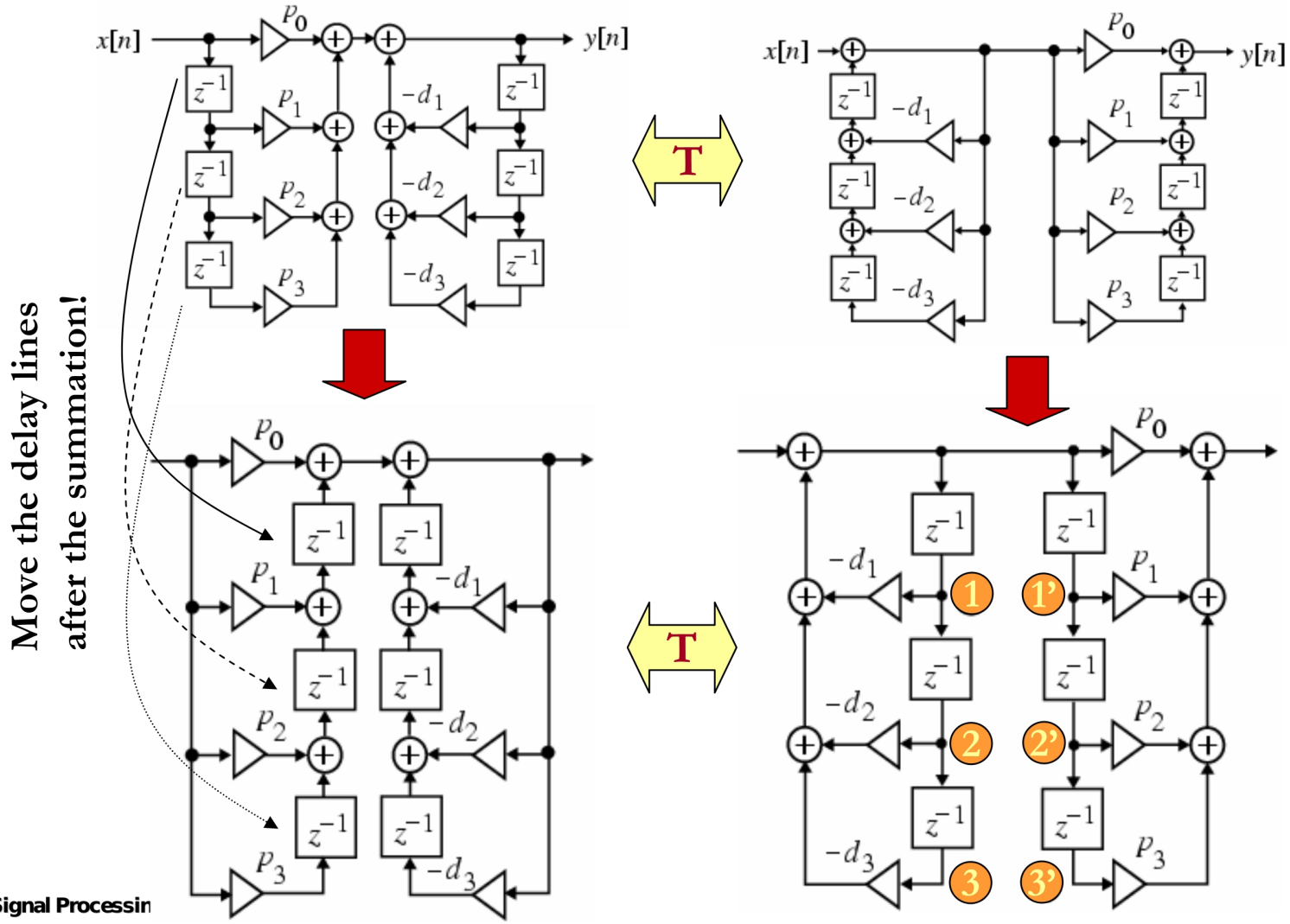
IIR Filter Structures: Direct Form I_T

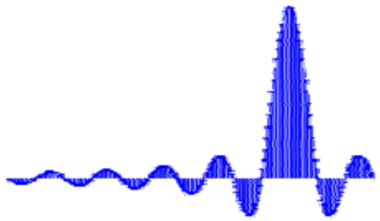
➲ The transpose of this implementation can also be obtained:



Other Non-Canonic Implementations

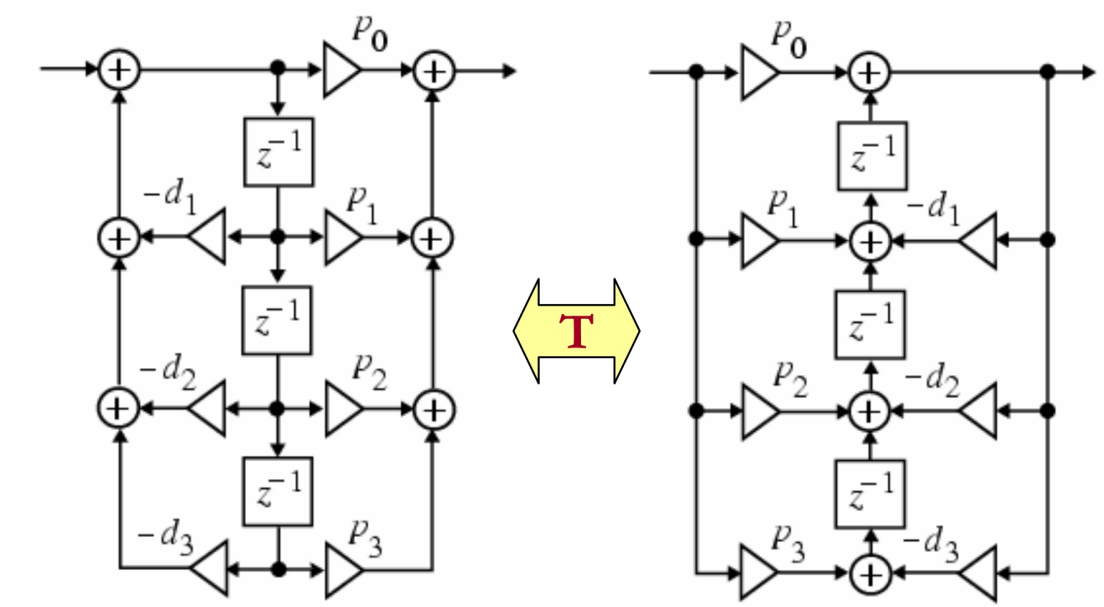
➲ Notice that we can also implement these structures as follows:



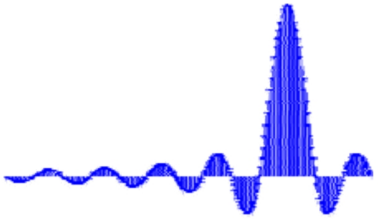


IIR Direct Form II Implementation

- Now notice that the points indicated as (1) and (1'), (2) and (2'), (3) and (3') are really indistinguishable from each other. Therefore, the delay elements can be shares.
- Hence, we obtain the following equivalent structures:



- This particular implementation is called the **Direct Form II realization**, and requires half the number of delay elements!

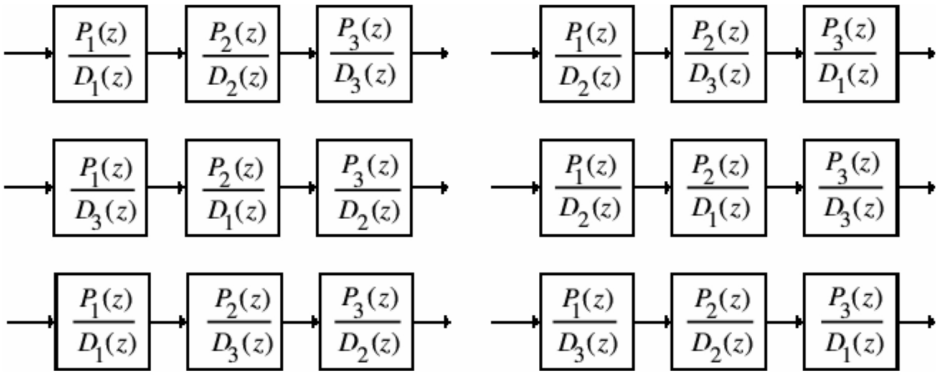


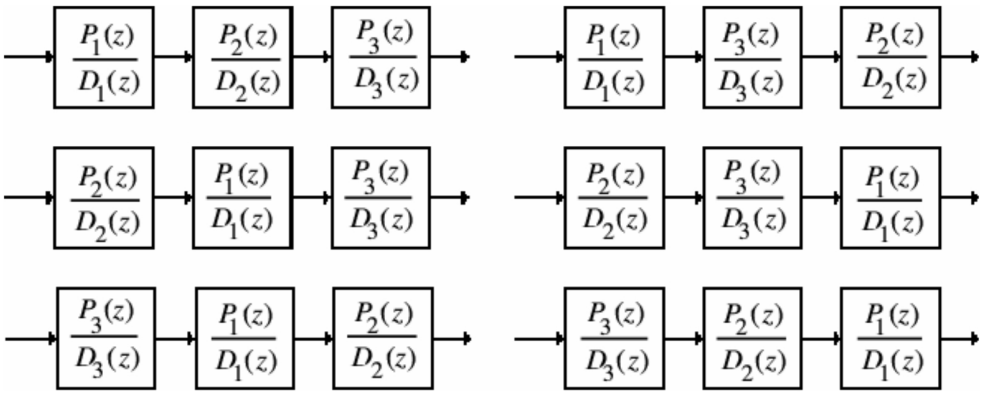
Cascade Forms

- By expressing the numerator and the denominator polynomials of the transfer function as a product of polynomials of lower degree, a digital filter can be realized as a cascade of low-order filter sections

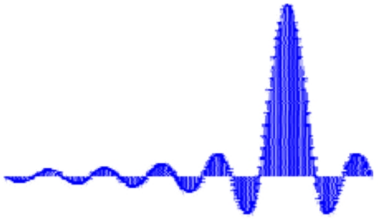
$$H(z) = \frac{P(z)}{D(z)} = \frac{P_1(z)P_2(z)P_3(z)}{D_1(z)D_2(z)D_3(z)}$$

- Consider, for example,
which can be implemented in one of 36 different forms, based on the pole-zero orderings and factorings!





• If we were to implement these structures using infinite precision hardware components, they would all results in the identical filter realization. However, each will actually realize a different filter due to finite word-length effects of finite precision.



Cascade Form

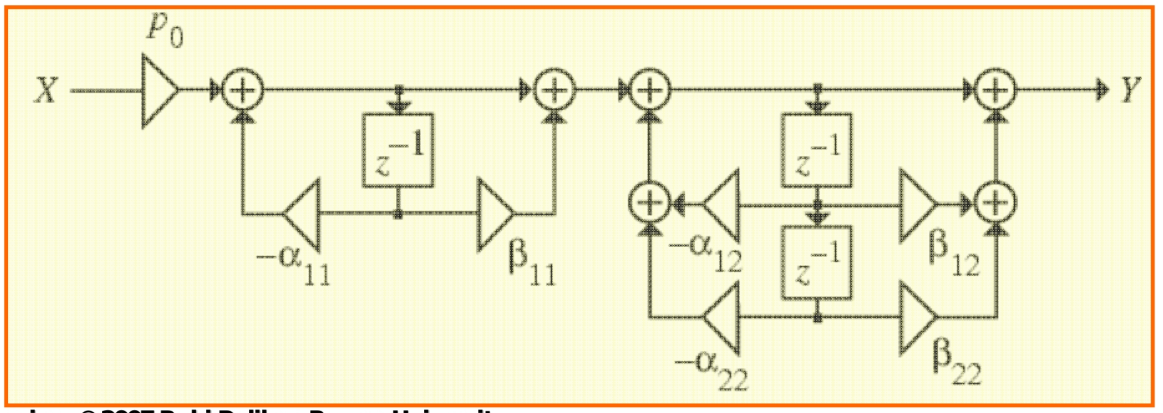
- The polynomials are typically factored into first or second order forms

$$H(z) = p_0 \prod_k \left(\frac{1 + \beta_{1k} z^{-1} + \beta_{2k} z^{-2}}{1 + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}} \right)$$

  - Where $\alpha_{2k} = \beta_{2k} = 0$ for first order forms.

- Then, for example, a third order system can be written and realized as

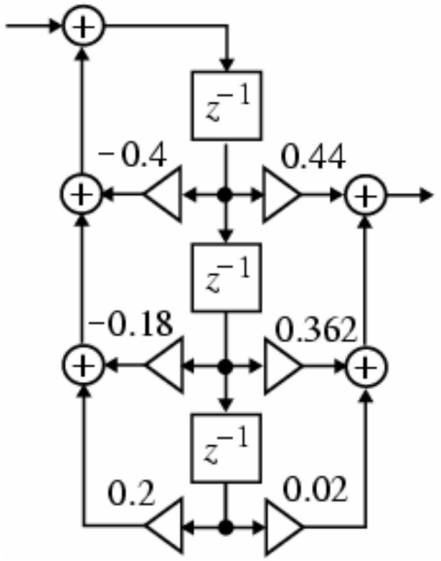
$$H(z) = p_0 \left(\frac{1 + \beta_{11} z^{-1}}{1 + \alpha_{11} z^{-1}} \right) \left(\frac{1 + \beta_{12} z^{-1} + \beta_{22} z^{-2}}{1 + \alpha_{12} z^{-1} + \alpha_{22} z^{-2}} \right)$$



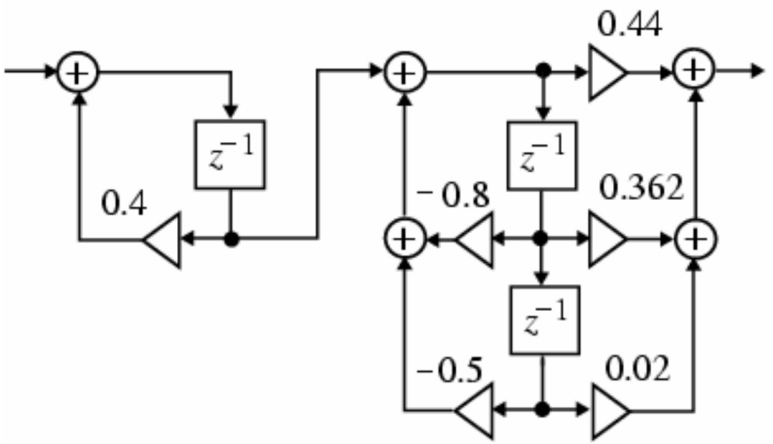
Cascade Direct Form II Realization

An Example

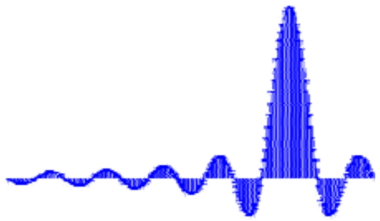
$$H(z) = \frac{0.44z^{-1} + 0.362z^{-2} + 0.02z^{-3}}{1 + 0.4z^{-1} + 0.18z^{-2} - 0.2z^{-3}} = \left(\frac{0.44 + 0.362z^{-1} + 0.02z^{-2}}{1 + 0.8z^{-1} + 0.5z^{-2}}\right)\left(\frac{z^{-1}}{1 - 0.4z^{-1}}\right)$$



Direct form II



Cascade form

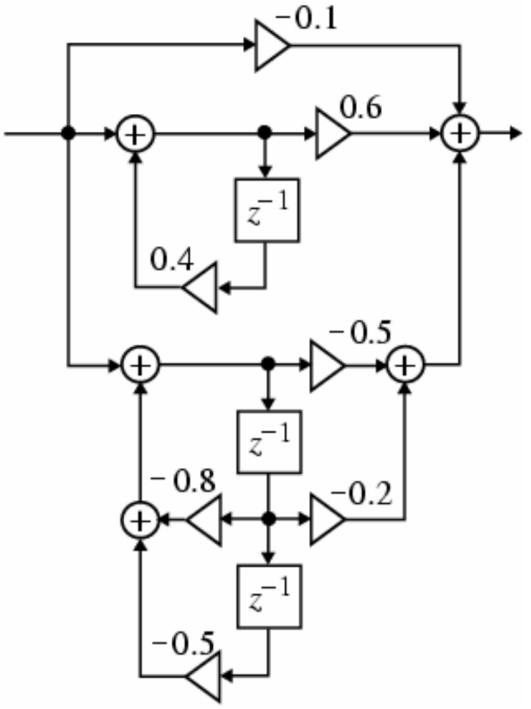


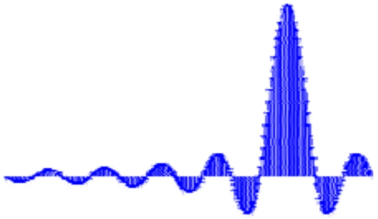
Parallel Form Realizations

- We can also realize IIR filters through direct partial fraction expansion, where each term is then implemented separately.
  - If the partial fraction expansion is done in terms of z^{-1}, ***parallel form I*** realization is obtained.
  - If the partial fraction expansion is done in terms of z, ***parallel form II*** realization is obtained.
- As an example, consider the same H(z) as in the previous example:

$$H(z)=\frac{0.44z^{-1}+0.362z^{-2}+0.02z^{-3}}{1+0.4z^{-1}+0.18z^{-2}-0.2z^{-3}}$$

$$H(z)=-0.1+\frac{0.6}{1-0.4z^{-1}}+\frac{-0.5-0.2z^{-1}}{1+0.8z^{-1}+0.5z^{-2}}$$



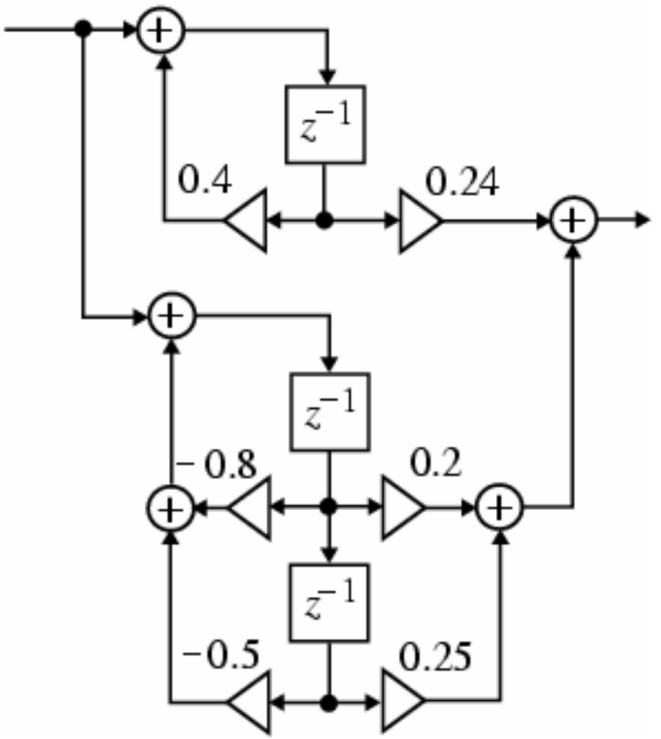


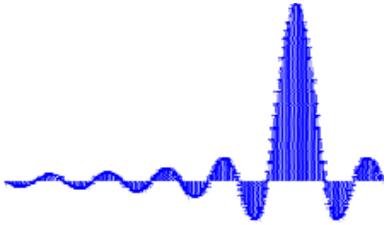
Parallel Form Realizations

➲ For Parallel Form II implementation:

$$H(z) = \frac{0.44z^{-1} + 0.362z^{-2} + 0.02z^{-3}}{1 + 0.4z^{-1} + 0.18z^{-2} - 0.2z^{-3}}$$

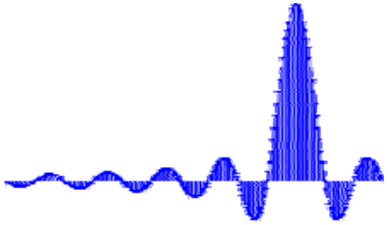
$$= \frac{0.24z^{-1}}{1 - 0.4z^{-1}} + \frac{0.2z^{-1} + 0.25z^{-1}}{1 + 0.8z^{-1} + 0.5z^{-2}}$$





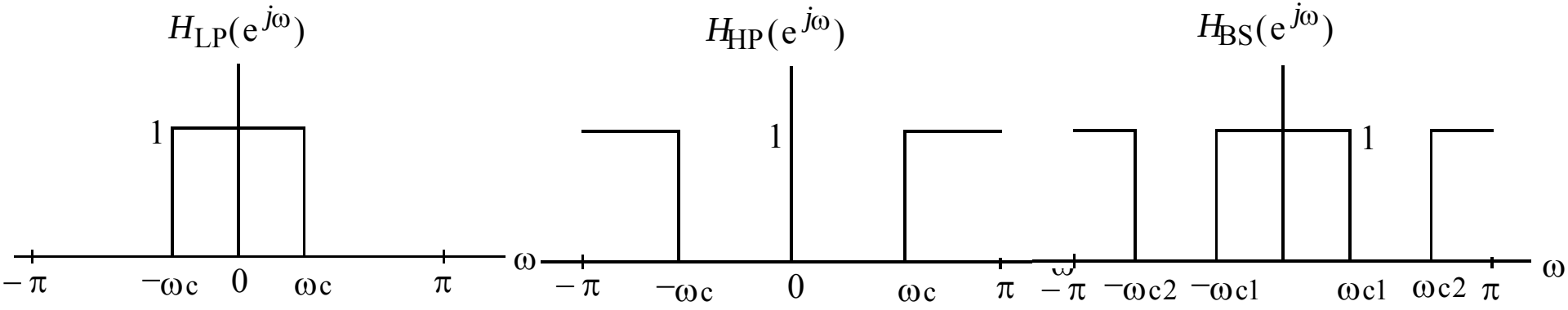
DIGITAL FILTERS

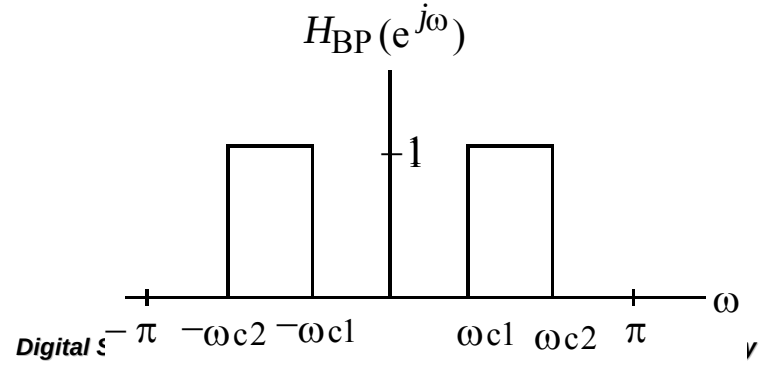
- So far we have seen several basic filter architectures
  - Impulse response based: FIR & IIR
  - Magnitude response based: Lowpass, highpass, bandpass, bandstop (notch), allpass, comb
  - Phase response based: zero-phase, linear phase, generalized linear phase, non-linear
- FIR examples:
  - 1sr order Moving average filter – lowpass
  - 1st order Highpass by replacing "z" with "-z" in MAF
  - High order FIR by cascading
- IIR examples:
  - 1st order lowpass, highpass, 2nd order bandpass, bandstop, notch, Lth order comb
  - Allpass, minimum phase, maximum phase
- The transfer functions provided for these are fixed, not allowing much flexibility in the design of these filters
  - Once you have picked the structure and order of the filter, all filter specs – including cutoff frequencies are automatically determined.
- How to design a filter with specific desired passband and stopband characteristics?
  - Filter design techniques



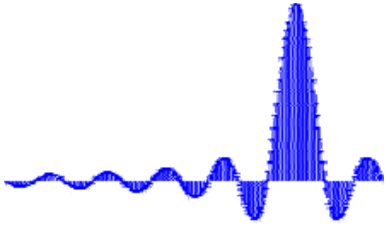
FILTER DESIGN

- **Objective:** Obtain a realizable ***transfer function*** H(z) approximating a desired frequency response.
- Digital filter design is the process of deriving this transfer function
  - Typically **magnitude** (and sometimes phase) response of the desired filter is specified.
  - Recall that there are four basic types of ideal digital filter, based on magnitude response





Since these filters cannot be realized, we need to relax (i.e., smooth) the sharp filter characteristics in the passband and stop band by providing *acceptable tolerances*



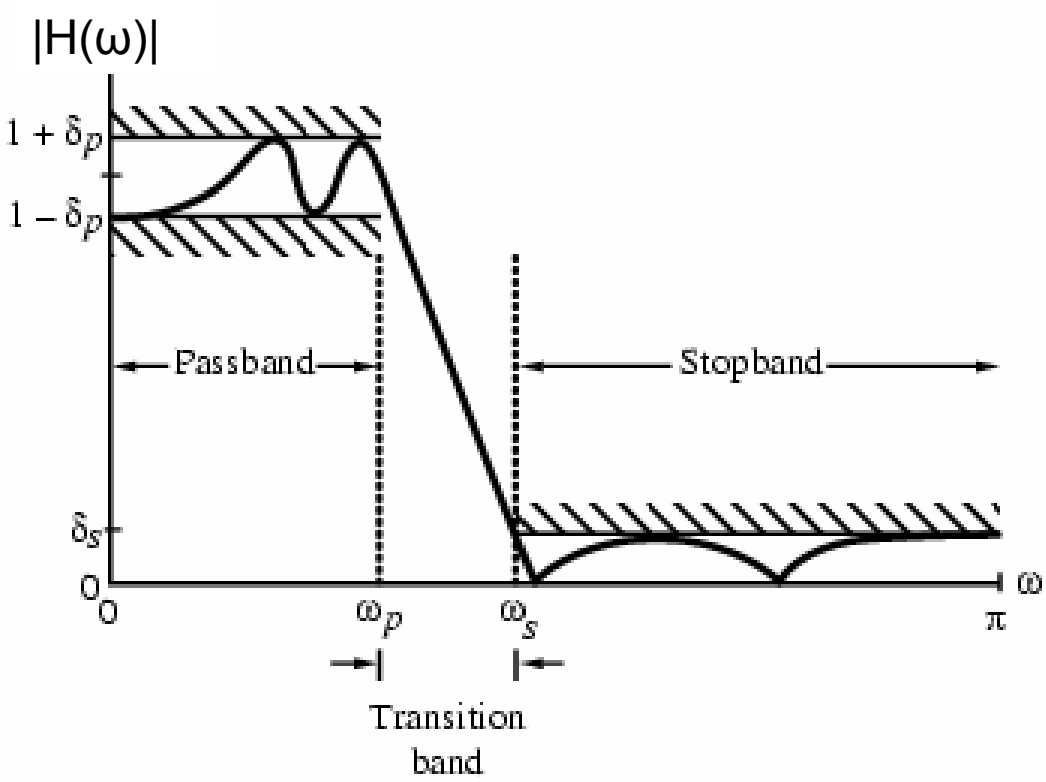
Filter Specifications

- $|H(e^{j\omega})| \approx 1$, with an error $\pm\delta_p$ in the *passband*, i.e.,

$$1-\delta_p \le |H(\omega)| \le 1+\delta_p, \quad |\omega| \le \omega_p$$

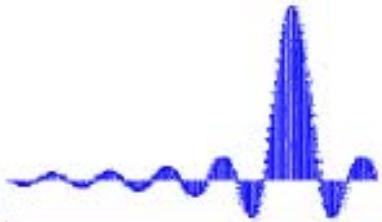
- $|H(e^{j\omega})| \approx 0$, with an error $\pm\delta_s$ in the *stopband*, i.e.,

$$|H(\omega)| \le \delta_s, \quad \omega_s \le |\omega| \le \pi$$



ω_p - passband edge frequency
ω_s - stopband edge frequency
δ_p - peak ripple value in the passband
δ_s - peak ripple value in the stopband

We will assume that we are dealing with filters with real coefficients, hence the frequency response is periodic with 2π, and symmetric around 0 and π.



FILTER SPECIFICATIONS

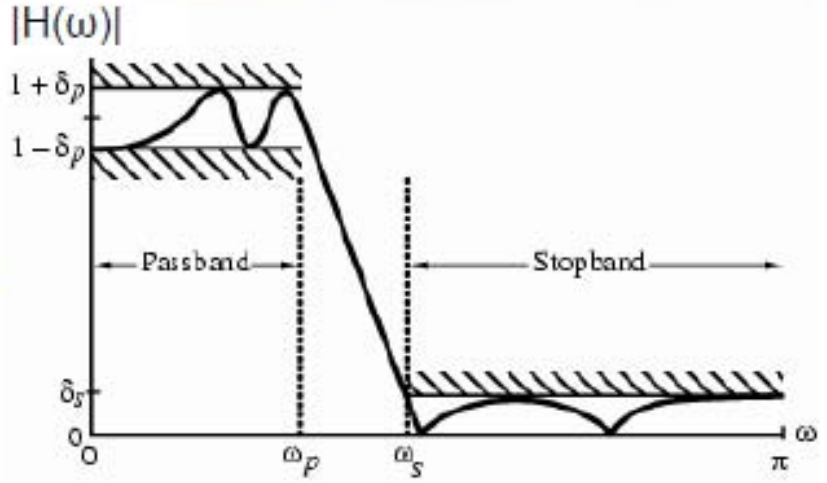
- Filter specifications are often given in decibels, in terms of loss of gain:

$$G(\omega) = -20\log_{10}\left|H(e^{j\omega})\right|$$

with peak pass and minimum stopband ripple

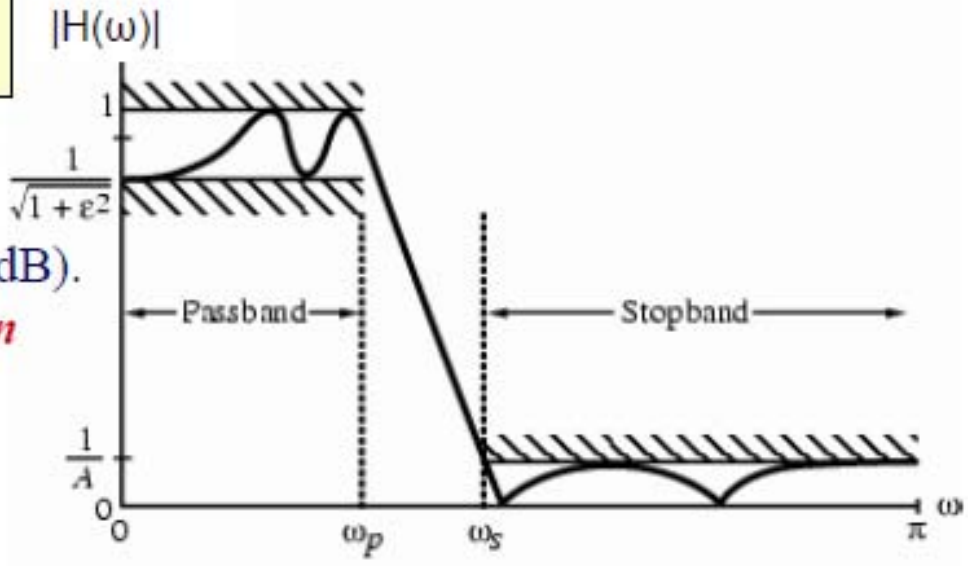
$$\alpha_p = -20\log_{10}(1-\delta_p) \quad \text{dB}$$

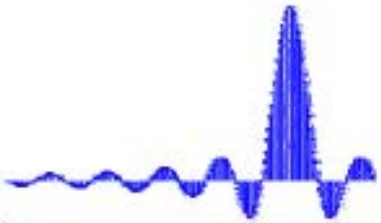
$$\alpha_s = -20\log_{10}(\delta_s) \quad \text{dB}$$



- Magnitude specs can also be given in a normalized form, where the max. passband value is normalized to "1" (0 dB). Then, we have ***max. passband deviation*** and ***max. stopband magnitude***

$1/\sqrt{1+\varepsilon^2}$ $\quad \frac{1}{A}$





REMEMBER!

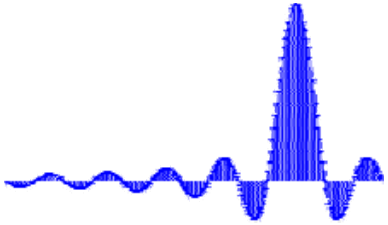
- The following must be taken into consideration in making filter selection
  - H(z) satisfying the frequency response specifications must be causal and stable (poles inside unit circle, ROC includes unit circle, h[n] one-sided)
  - If the filter is FIR, then H(z) is a polynomial in z^{-1} with real coefficients

$$H(z) = \sum_{n=0}^{N} b[n] z^{-n} = \sum_{n=0}^{N} h[n] z^{-n}$$

    - If linear phase is desired, the filter coefficients h[n] (also the impulse response) must satisfy symmetry constraints: h[n] = ±h[M-n]
    - For computational efficiency, the minimum filter order M that satisfies design criteria must be used.
  - If the filter is IIR, then H(z) is a real rational function of z^{-1}

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}}$$

    - Stability must be ensured!
    - Minimum (M,N) that satisfies the design criteria must be used.



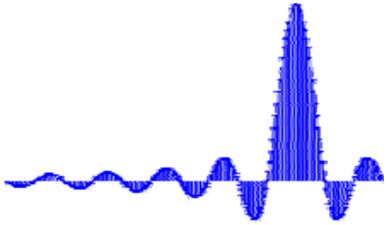
Cutoff Frequencies in Digital Domain

- In general, filter specifications are given in Hz, but most digital filters are designed in normalized frequencies, where π is $f_t/2$.
  - p.s. We will use the subscript "s" to refer to stop-band. In order not to confuse with the subscript "s" indicating samling frequency, let us refer to sampling frequency as f_t
  - Then, the normalized passband and stopband edge frequencies can be obtained from linear frequencies as follows:
    - f_p: linear pass band edge frequency
    - f_s : linear stop band edge frequency
    - f_t: sampling frequency
    - T: sampling period
    - ω_p: normalized (angular) pass band edge frequency
    - ω_s: normalized (angular) stop band edge frequency

$$\omega_p = \frac{2\pi f_p}{f_t} = 2\pi f_p T$$

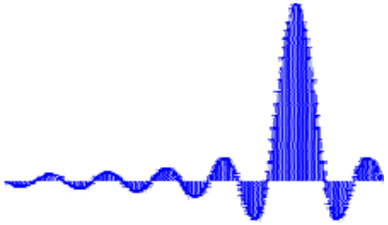
$$\omega_s = \frac{2\pi f_s}{f_t} = 2\pi f_s T$$

  - E.g. If the sampling frequency is f_t=10kHz, and we want f_p=3kHz and f_s=4kHz
    - Then $\omega_p = 2\pi 3\text{kHz}/10\text{kHz} = 0.6\pi$, and $\omega_s = 2\pi 4\text{kHz}/10\text{kHz} = 0.8\pi$. Of course f_t=10kHz➔2π



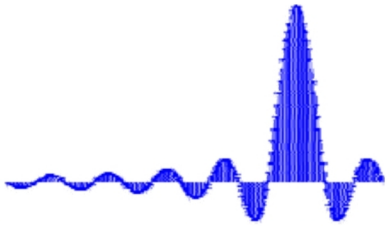
FIR OR IIR???

- Several advantages to both FIR and IIR type filters.
- Advantages of FIR filters (disadvantages of IIR filters):
  - Can be designed with exact linear phase,
  - Filter structure always stable with quantized coefficients
  - The filter startup transients have finite duration
- Disadvantages of FIR filters (advantage of IIR filters)
  - Order of an FIR filter is usually much higher than the order of an equivalent IIR filter meeting the same specifications ➔ higher computational complexity
  - In fact, the ratio of orders of a typical IIR filter to that of an FIR filter is in the order of tens.
- The nonlinear phase of an IIR filter can be minimized using an appropriate allpass filter, however, by that time the computational advantage of IIR is lost.
- However, in most applications that does not require real-time operation, phase is not an issue. Why…?



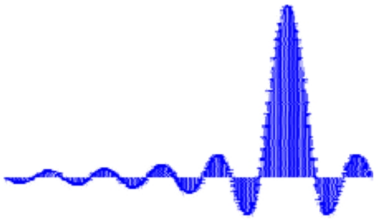
Basic Design Approaches

- (Classic) IIR filter design: **(Chapter 9)**
  1. Convert the digital filter specifications into an analog prototype lowpass filter specifications
  2. Determine the analog lowpass filter transfer function $|H(\Omega)|$
  3. Transform $|H(\Omega)|$ into the desired digital transfer function $H(z)$
     - Analog approximation techniques are highly advanced
     - They usually yield closed-form solutions
     - Extensive tables are available for analog filter design
     - Many applications require digital simulation of analog systems
- FIR filter design is based on a direct approximation of the specified magnitude response, with the often added requirement that the phase be linear (or sometimes, even minimum)
  - The design of an FIR filter of order M may be accomplished by finding either the length-(M+1) impulse response samples of h[n] or the (N+1) samples of its frequency response $H(\omega)$ **(Chapter 10)**



This week in DSP

- IIR filter design
- Bilinear transformation for analog ←→ digital domains
- Analog IIR filter design
  - Butterworth approximation and Matlab implementation
  - Chebyshev approximation and Matlab implementation
- Spectral transformations
  - Analog – to – analog transformations
  - Digital – to – digital transformations

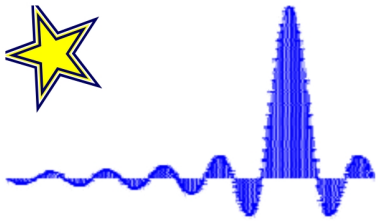


IIR Filter Design

- Major disadvantage of the FIR filter: Long filter lengths
  - IIR filters yield much shorter filters for the same specs ➔ computationally efficient.
  - Two potential concerns of IIR filters must be addressed, however: stability, linear phase
- Classic IIR filter design:
  1. Convert the digital filter specifications into an analog prototype lowpass filter specifications
  2. Determine the analog lowpass filter transfer function $|H_a(\Omega)|$ and corresponding $H_a(s)$
     - Butterworth / Chybshev / Elliptic / Bessel
  3. Transform $|H_a(s)|$ into the desired digital transfer function H(z)
     - Bilinear and inverse bilinear transformations for mapping s ←➔z-planes

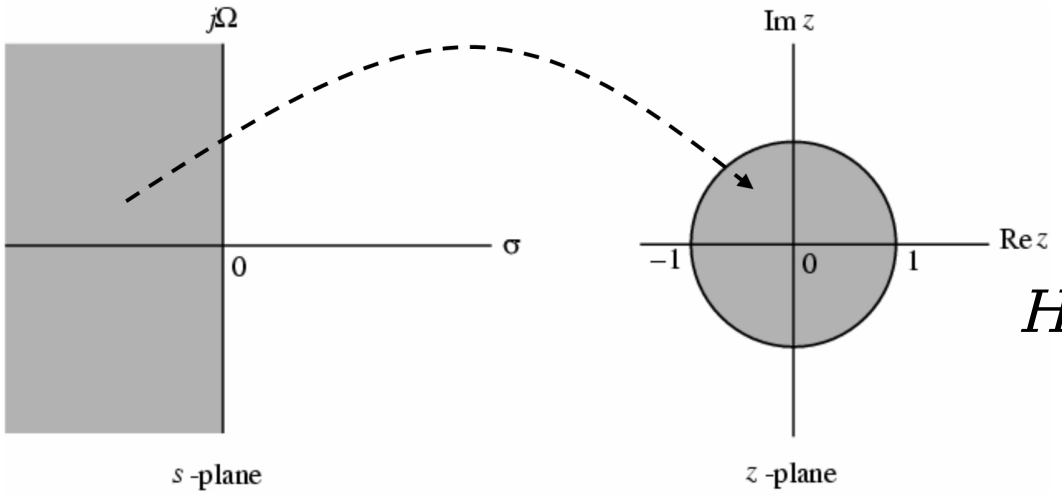
$$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

$$H(z) = H_a(s)\Big|_{s=\frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$



Analog ↔ Digital Frequency

- The parameter T_s often does not play a role in the design, and therefore $T_s=2$ is chosen for convenience. Then, we have



$$z=\frac{1+s}{1-s}$$

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

- The left half of the s-plane corresponds to inside the unit circle in the z-plane
- The jΩ axis corresponds to the unit circle
- The stability requirement of the analog filters carry to digital filters:
  - Analog: The poles of the filter frequency response must be on the left half plane
  - Digital: The poles of the filter frequency response must be inside the unit circle, i.e., the ROC must include the unit circle.

Effect of Bilinear Transformation

➲ Since, the frequency response is defined on the unit circle, $z=e^{j\omega}$ ←→ $s=j\Omega$

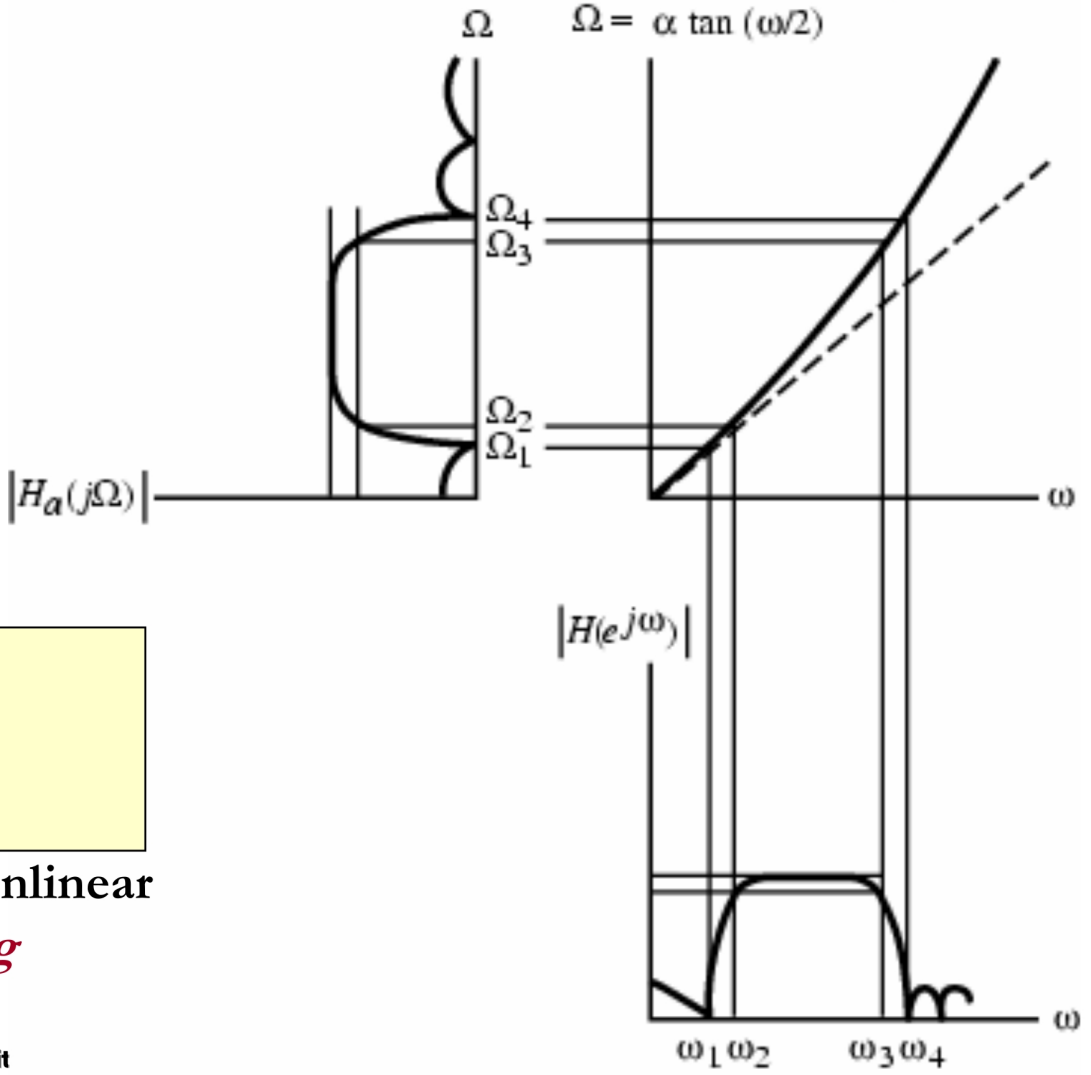
$$s=\frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

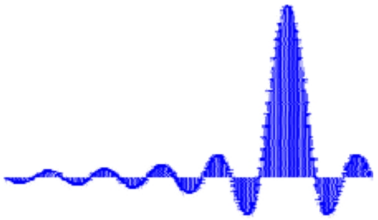
$$j\Omega=\frac{2}{T_s}\left(\frac{1-e^{-j\omega}}{1+e^{-j\omega}}\right)$$

$$\Omega=\frac{2}{T_s}\tan\left(\frac{\omega}{2}\right)$$

This mapping is (highly) nonlinear

➔ *Frequency warping*





Bilinear Transformation

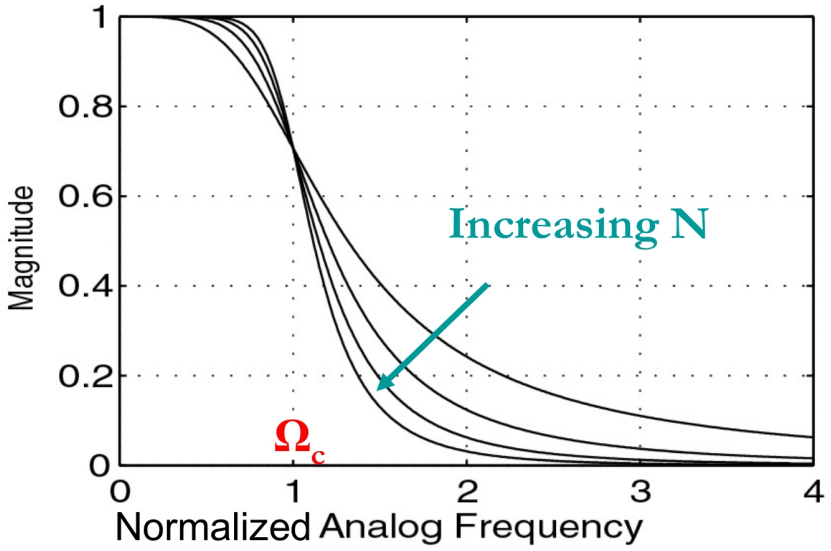
- Steps in the design of a digital filter -
  - Prewarp ω_p, ω_s to find their analog equivalents Ω_p, Ω_s → $\Omega_{p,s} = \frac{2}{T_S}\tan(\omega_{p,s}/2)$
  - Design the analog filter $H_a(s)$
  - Design the digital filter H(z) by applying bilinear transformation to $H_a(s)$ → $s = \frac{2}{T_S}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$
- How to design the analog filter?
  - Butterworth filter – maximally flat
  - Chebychev (type I and type II) filters – Equiripple in passband or stopband
  - Elliptic filter – Sharper transition band but nonlinear phase and nonequiripple
  - Bessel filter – Linear phase in passband, at the cost of wide transition band
- All of these filters are defined for lowpass characteristic
  - ***Spectral transformations*** are then used to convert lowpass to any one of highpass, bandpass or bandstop (on Wednesday).
- …and then there is direct design that sidesteps all of the above steps…

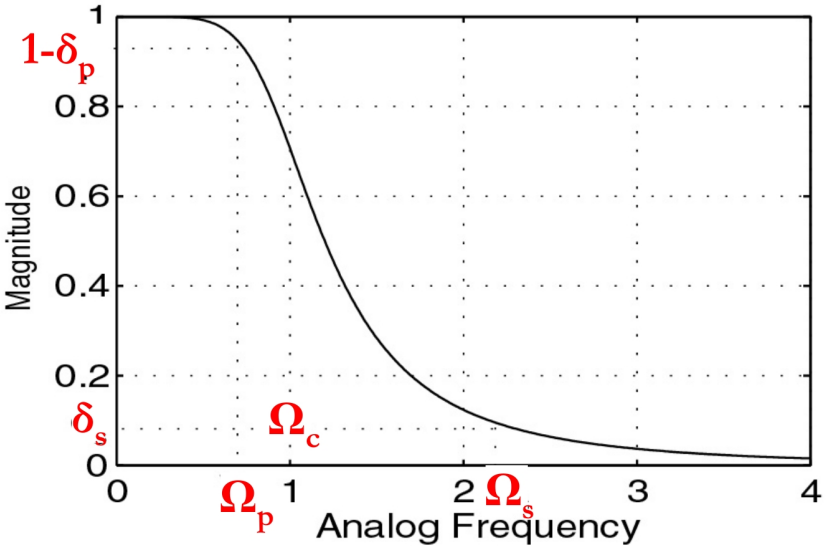
The Butterworth Approximation

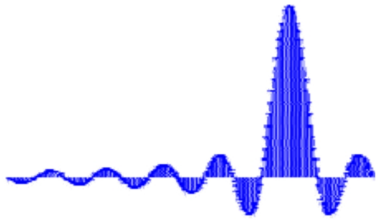
- The magnitude-square response of an N^{th} order analog lowpass Butterworth filter:

$$|H(\Omega)| = \sqrt{\frac{1}{1+(\Omega/\Omega_c)^{2N}}} \Leftrightarrow |H(\Omega)|^2 = \frac{1}{1+(\Omega/\Omega_c)^{2N}}$$

  - Ω_c is the 3-dB cutoff frequency ($20\log|H(\Omega_c)|=-3$), N is the filter order
  - The most interesting property of this function is that the first 2N-2 derivatives of this function is zero at $\Omega=0$. ➔ The function is as flat as possible, without being a constant.
  - ***The Butterworth LPF is therefore said to have a maximally-flat magnitude at $\Omega=0$.***







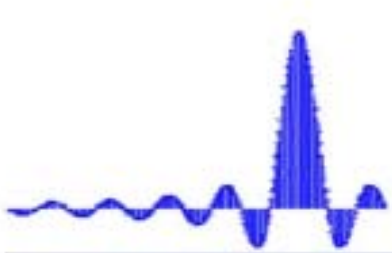
Selectivity & Discrimination

- Many design formulas for analog filters can be streamlined through two new parameters:
  - Selectivity Factor (Transition ratio), ***r***: $$r = \frac{\Omega_p}{\Omega_s}, \quad 0 < r < 1$$
    - Note that for an ideal filter, there is no transition band → $\Omega_p = \Omega_s$ → **r**=1
    - Selectivity is then a measure of how far the edge frequencies are from each other: the closer they are (the smaller the transition band) the higher the selectivity
  - Discrimination Factor, ***d***
    - For δ_p=0 or δ_s=0(no pass or stop band ripple) → d=0
      Hence, for an ideal filter, the discrimination factor is zero
    - Discrimination factor is then a measure of ripple in the filter characteristic. Less ripple → small discrimination

$$d = \left[\frac{\frac{1}{(1-\delta_p)^2} - 1}{\left(\frac{1}{\delta_s}\right)^2 - 1} \right]^{1/2}$$

  - If the specs are given in terms of ε and A, a variation of ***d*** is also given as

$$d' = \frac{\varepsilon}{\sqrt{A^2 - 1}}$$



Analog Filters

Analog LP filter design (ch.4.4): requirements set in terms of the discrimination parameter ***r*** and the discrimination factor ***d.***

Widely used approximations are (Ω_c is the 3dB cutoff frequency):

Butterworth:

$$\left|H_a(j\Omega)\right|^2 = \frac{1}{1+(\Omega/\Omega_c)^{2N}}$$

Type 1 Chebyshev:

$$\left|H_a(j\Omega)\right|^2 = \frac{1}{1+\varepsilon^2 T_N^2(\Omega/\Omega_c)}, \quad T_N(\Omega) = \begin{cases} \cos(N\cos^{-1}\Omega), & |\Omega| \le 1 \\ \cosh(N\cos^{-1}\Omega), & |\Omega| > 1 \end{cases}$$

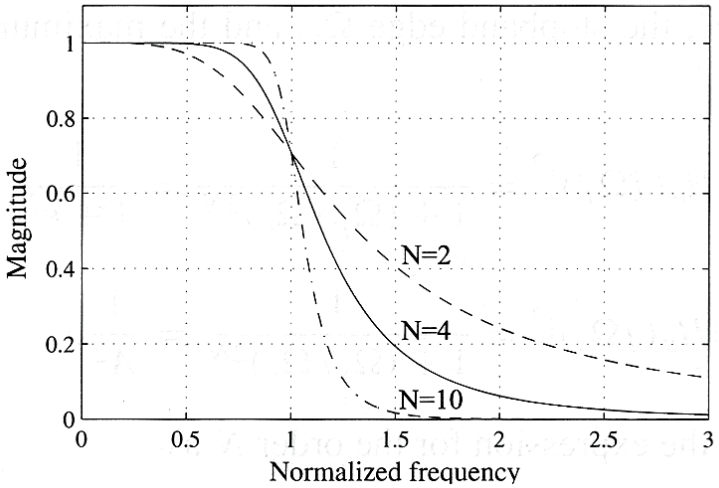
Elliptical:

$$\left|H_a(j\Omega)\right|^2 = \frac{1}{1+\varepsilon^2 R_N^2(\Omega/\Omega_c)}, \quad \text{with } R_N(1/\Omega) = 1/R_N(\Omega)$$

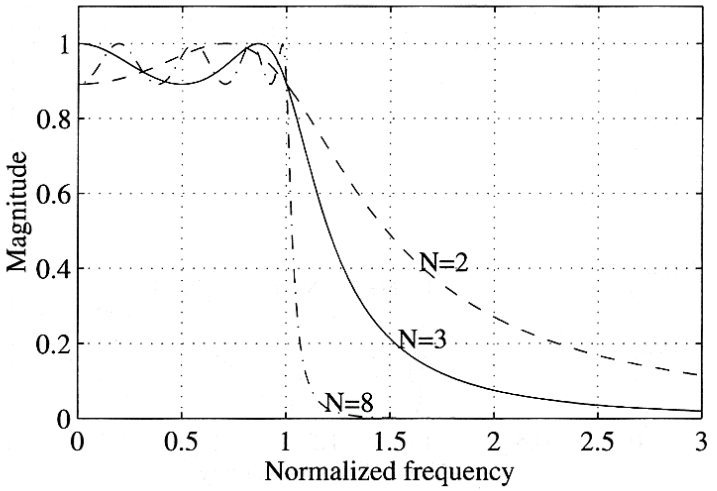
Bessel:

$$\left|H_a(s)\right|^2 = \frac{d_0}{B_N(s)}, \quad B_N(s) = (2N-1)B_{N-1}(s) + s^2 B_{N-2}(s)$$

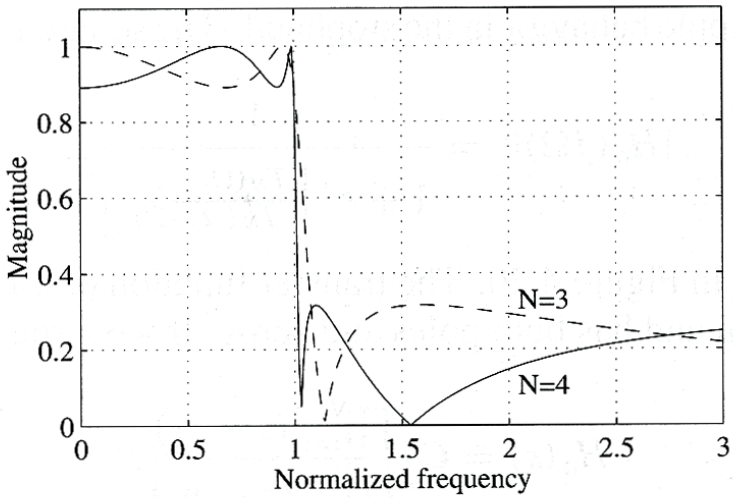
Analog Filters



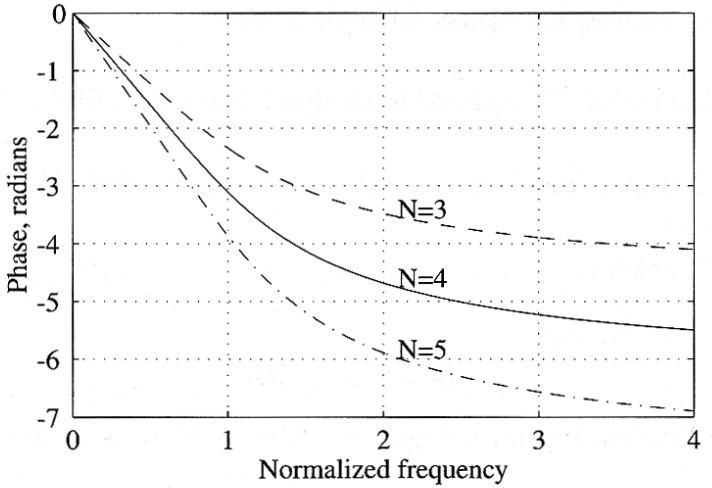
Butterworth



Type 1 Chebyshev



Elliptical



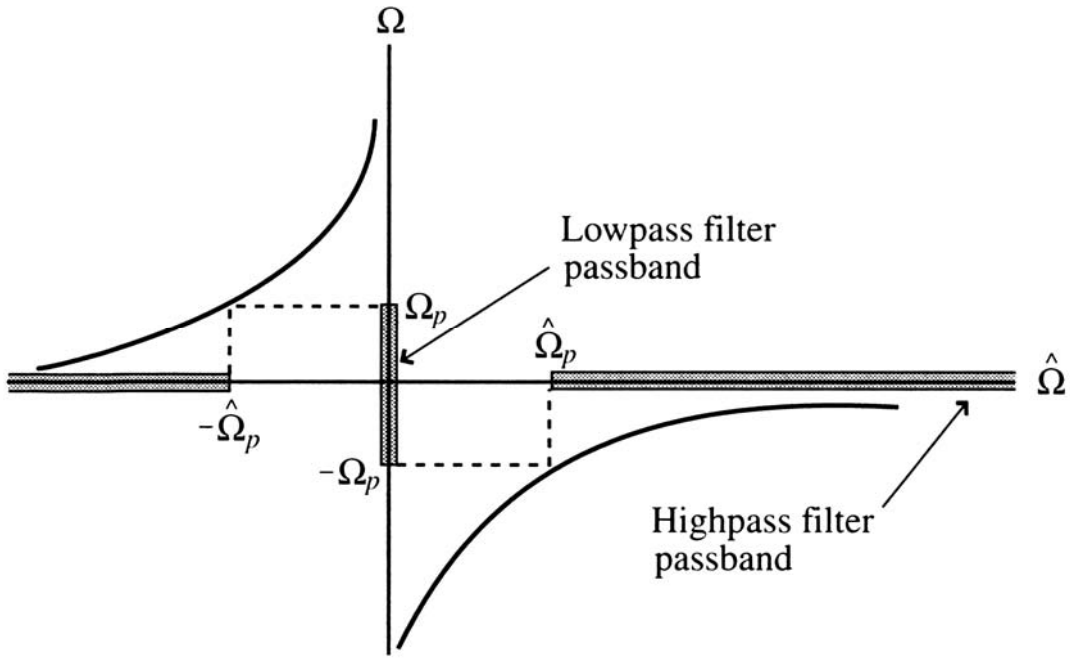
Bessel

Analog Filters

Spectral transformation in frequency to get from a LP filter to a HP filter

$$s = -\frac{\Omega_p \hat{\Omega}_p}{\hat{s}} \quad \Rightarrow \quad \Omega = -\frac{\Omega_p \hat{\Omega}_p}{\hat{\Omega}} \text{ on the imaginary axis}$$

Graph of the spectral mapping:

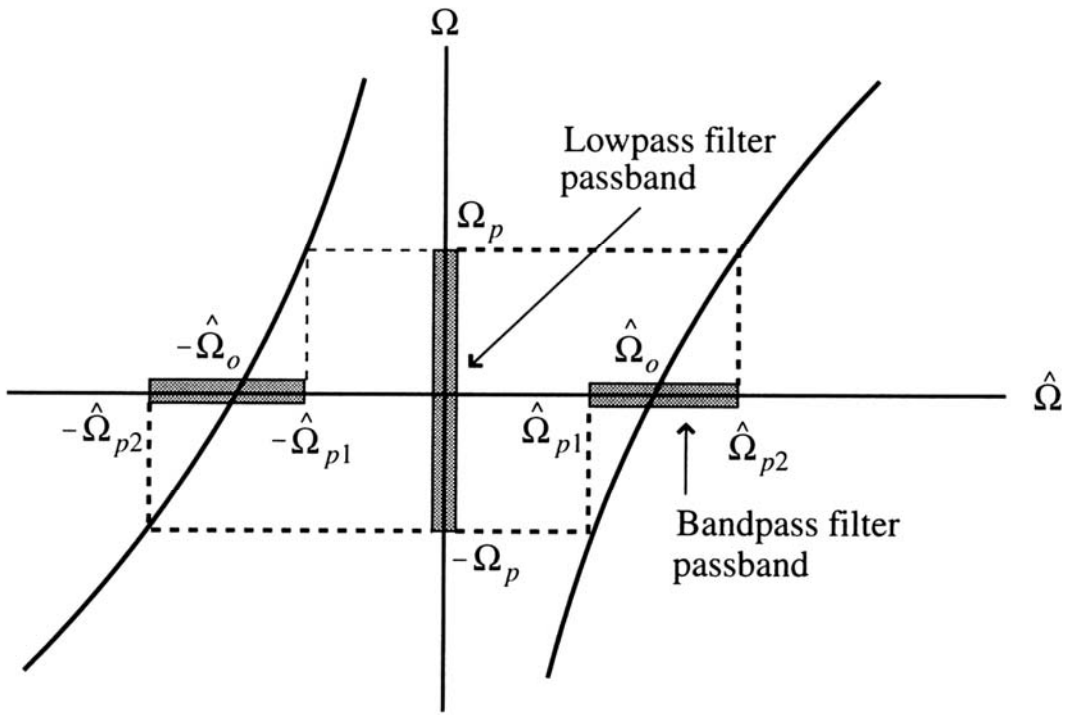


Analog Filters

Spectral transformation in frequency to get from a LP filter to a BP filter

$$s = \Omega_p \frac{\hat{s}^2 + \hat{\Omega}_0^2}{\hat{s}(\hat{\Omega}_{p2} - \hat{\Omega}_{p1})} \quad \Rightarrow \quad \Omega = -\Omega_p \frac{\hat{\Omega}_0^2 - \hat{\Omega}^2}{\hat{\Omega} B_w}$$

Graph of the spectral mapping:

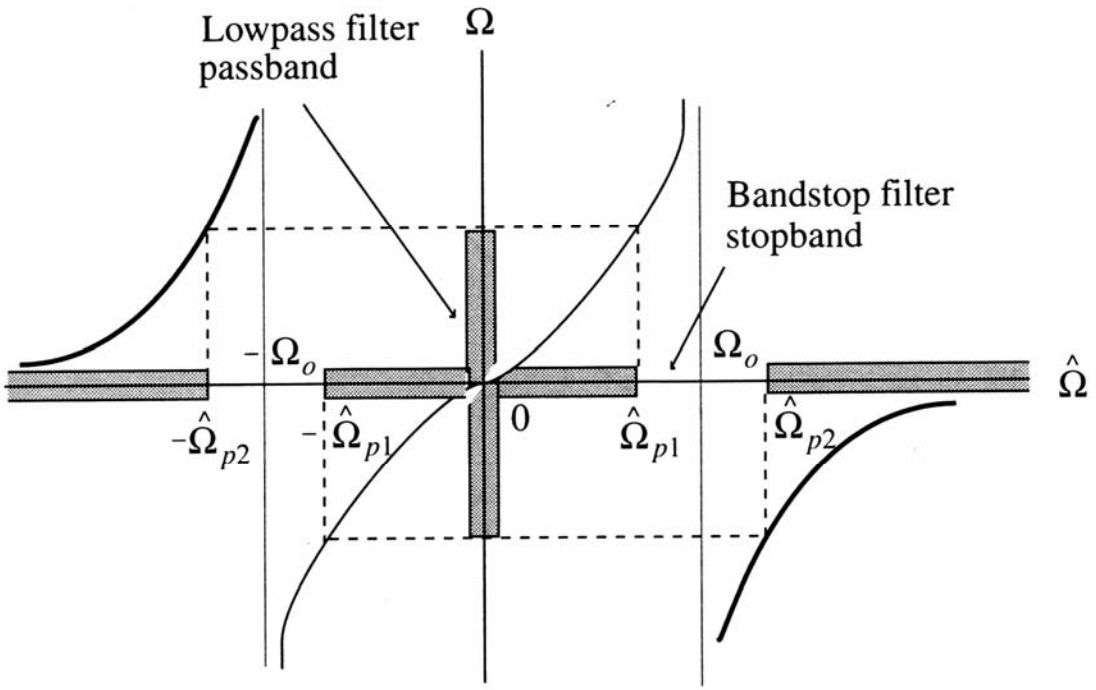


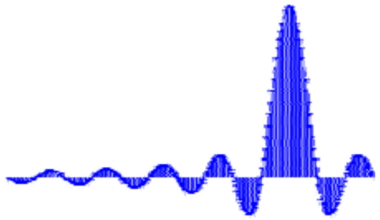
Analog Filters

Spectral transformation in frequency to get from a LP filter to a BS filter

$$s = \Omega_s \frac{\hat{s}(\hat{\Omega}_{s2} - \hat{\Omega}_{s1})}{\hat{s}^2 + \hat{\Omega}_0^2} \quad \Rightarrow \quad \Omega = -\Omega_s \frac{\hat{\Omega} B_w}{\hat{\Omega}_0^2 - \hat{\Omega}^2}$$

Graph of the spectral mapping:





Butterworth Filter Design

- Butterworth filter has two parameters to be determined: filter order N, and filter cutoff frequency Ω_c
  - To determine the filter order N:

$$\left|H(\Omega)^2\right|_{\Omega=\Omega_p} = \frac{1}{1+(\Omega_p/\Omega_c)^{2N}} = (1-\delta_p)^2$$

$$\left|H(\Omega)^2\right|_{\Omega=\Omega_s} = \frac{1}{1+(\Omega_s/\Omega_c)^{2N}} = (\delta_s)^2$$

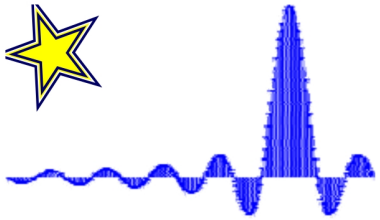
$$N = \frac{1}{2}\frac{\log\left(\frac{1}{(1-\delta_p)^2}-1\right)-\log\left(\frac{1}{\delta_s^2}-1\right)}{\log\left(\frac{\Omega_p}{\Omega_s}\right)} = \frac{\log(d)}{\log(r)}$$

  - Once N is determined, we find the value of $\Omega=\Omega_c$ for which $H(\Omega)$ drops 3 dB

$$\frac{1}{1+\left(\frac{\Omega_p}{\Omega_c}\right)^{2N}} = 1-\delta_p^{\,2}$$

  - Once $H(\Omega)$ is obtained, it is usually written in partial fraction form to obtain zeros and poles

$$H(s) = \frac{K}{(s-s_1)(s-s_2)\cdots(s-s_{2N-1})}$$



Designing a Digital LPF Using Butterworth Appr.

1. Prewarp ω_p, ω_s to find their analog equivalents Ω_p, Ω_s

$$\Omega_{p,s} = \frac{2}{T_s}\tan(\omega_{p,s}/2)$$

2. Design the analog filter: Determine N and Ω_c

   a) From δ_p, δ_s, Ω_p and Ω_s obtain the order of the filter N

$$N = \frac{1}{2}\frac{\log\left(\frac{1}{(1-\delta_p)^2}-1\right)-\log\left(\frac{1}{\delta_s^2}-1\right)}{\log\left(\frac{\Omega_p}{\Omega_s}\right)} = \frac{\log(d)}{\log(r)}$$

   Note that the order N must be integer, so the value obtained from this expression must be rounded up to exceed the specifications

   b) Use N, δ_p, and Ω_p to calculate the 3dB cutoff frequency Ω_c

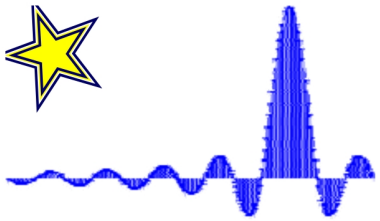
$$\frac{1}{1+\left(\frac{\Omega_p}{\Omega_c}\right)^{2N}} = 1-\delta_p^2$$

   c) Determine the corresponding H(s) and its poles
   If the filter order is large, manually computing H(s) is typically difficult, and usually done using filter design software (such as Matlab)

$$H(s) = \frac{K}{(s-s_1)(s-s_2)\cdots(s-s_{2N-1})}$$

3. Apply bilinear transformation to obtain H(z)

$$z = \frac{1+\frac{T_s}{2}s}{1-\frac{T_s}{2}s}$$



IN MATLAB

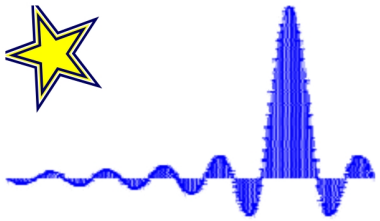
➲ Yes, you guessed it right, Matlab has several functions:

`buttord` **Butterworth filter order selection**.

[N, Wn] = buttord(Wp, Ws, Rp, Rs) returns the order N of the lowest order ***digital*** Butterworth filter that loses no more than **Rp** dB in the passband and has at least **Rs** dB of attenuation in the stopband. **Wp** and **Ws** are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample).

`buttord()` also returns **Wn**, the Butterworth natural frequency (or, the "3 dB frequency") to use with **`butter()`** to achieve the desired specs.

[N, Wn] = buttord(Wp, Ws, Rp, Rs, 's') does the computation for an ***analog filter***, in which case **Wp** and **Ws** are in radians/second. Note that for analog filter design **Wn** is not restricted to the [0 1] range. When **Rp** is chosen as 3 dB, the **Wn** in **`butter()`** is equal to **Wp** in **`buttord()`**.



IN MATLAB

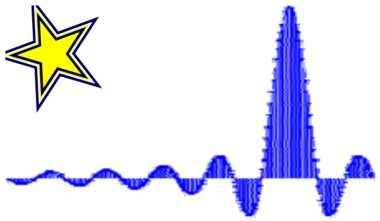
`butter()` Butterworth digital and analog filter design.

[B,A] = butter(N,Wn) designs an N^{th} order lowpass ***digital*** Butterworth filter and returns the filter coefficients in length N+1 vectors **B** (numerator) and **A** (denominator). The coefficients are listed in descending powers of z. The cutoff frequency **Wn** must be 0.0 < **Wn** < 1.0, with 1.0 corresponding to half the sample rate. If **Wn** is a two-element vector, **Wn** = [W1 W2], **`butter()`** returns an order 2N bandpass filter with passband W1 < W < W2.

[B,A] = butter(N,Wn,'high') designs a highpass filter.
[B,A] = butter(N,Wn,'stop') is a bandstop filter if **Wn** = [W1 W2].

When used with three left-hand arguments, as in **[Z,P,K] = butter(...)**, the zeros and poles are returned in length N column vectors Z and P, and the gain in scalar K.

butter(N,Wn,'s'), butter(N,Wn,'high','s') and **butter (N,Wn,'stop','s')** design ***analog*** Butterworth filters. In this case, **Wn** is in [rad/s] and it can be greater than 1.0.



bilinear() **Bilinear transformation with optional frequency prewarping.**

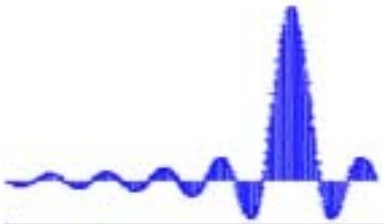
[Zd,Pd,Kd] = bilinear(Z,P,K,Fs) converts the s-domain transfer function specified by **Z**, **P**, and K to a z-transform discrete equivalent obtained from the bilinear transformation:

$$H(z) = H(s)\big|_{s = 2*Fs*(z-1)/(z+1)}$$

where column vectors Z and P specify the zeros and poles, scalar K specifies the gain, and Fs is the sampling frequency in Hz. If normalized T_s=2 is used, then Fs=0.5.

[NUMd,DENd] = bilinear (NUM,DEN,Fs), where **NUM** and **DEN** are row vectors containing numerator and denominator transfer function coefficients, **NUM(s)/DEN(s),** in descending powers of s, transforms to z-transform coefficients **NUMd(z)/DENd(z).**

Each form of **bilinear()** accepts an optional additional input argument that specifies prewarping. For example, **[Zd,Pd,Kd] = bilinear (Z,P,K,Fs,Fp)** applies prewarping before the bilinear transformation so that the frequency responses before and after mapping match exactly at frequency point Fp (match point Fp is specified in Hz).



EXAMPLE

- Design an analog IIR lowpass filter using the Butterworth approximation that meets the following specs: $f_p = 2kHz$, $f_s = 3kHz$, $\alpha_p < 2dB$ $\alpha_s > 50dB$, $f_s = 10kHz$.
  - $\omega_p = 0.4\pi$ rad, $\omega_s = 0.6\pi$ rad, $\delta_p = 0.2057$, $\delta_s = 0.0032$ ➔ $\Omega_p = 0.7265$ rad/s, $\Omega_s = 1.3764$ rad/s

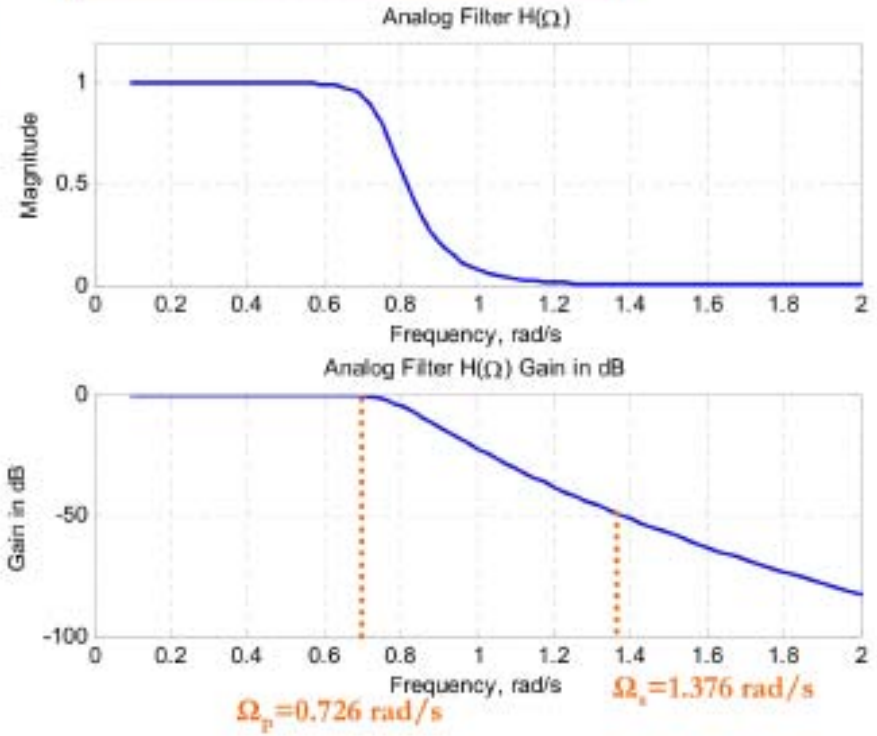
$$z = \frac{1 + \frac{T_s}{2}s}{1 - \frac{T_s}{2}s}$$

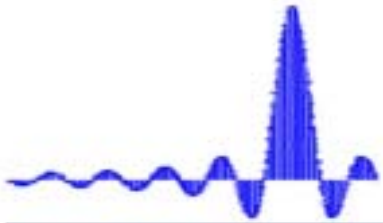
(take $T_s = 2$ for simplified normalization)

$$\alpha_p = -20\log_{10}(1-\delta_p) \Rightarrow \delta_p = 1 - 10^{(-\alpha_p/20)}$$

$$\alpha_s = -20\log_{10}(\delta_s) \Rightarrow \delta_s = 10^{(-\alpha_s/20)}$$

```
[N Wn]=buttord(Wp, Ws, Rp, Rs)
[b a]=butter(N, Wn, 's');
[H W]=freqs(b,a); %Note freqs() for analog
subplot(211)
plot(W, abs(H));
grid
axis([0 2 0 1.2])
xlabel('Frequency, rad/s')
ylabel('Magnitude')
title('Analog Filter H(\Omega)')
subplot(212)
plot(W, 20*log10(abs(H)));
axis([0 2  0 1])
grid
xlabel('Frequency, rad/s')
title('Analog Filter H(\Omega) Gain in dB')
ylabel('Gain in dB')
axis([0 2 -100 0])
```





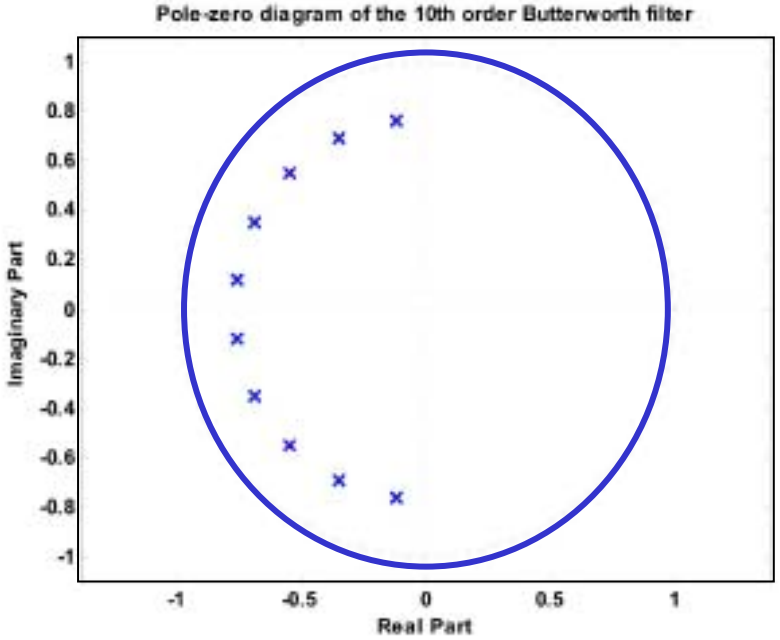
EXAMPLE

➲ Let's take a closer look at the filter designed:

```
[z, p, k]=tf2zpk(b, a);
zplane(b, a)
```

What do we notice?

- No zeros!
- All poles are on the left half plane, and inside the unit circle!

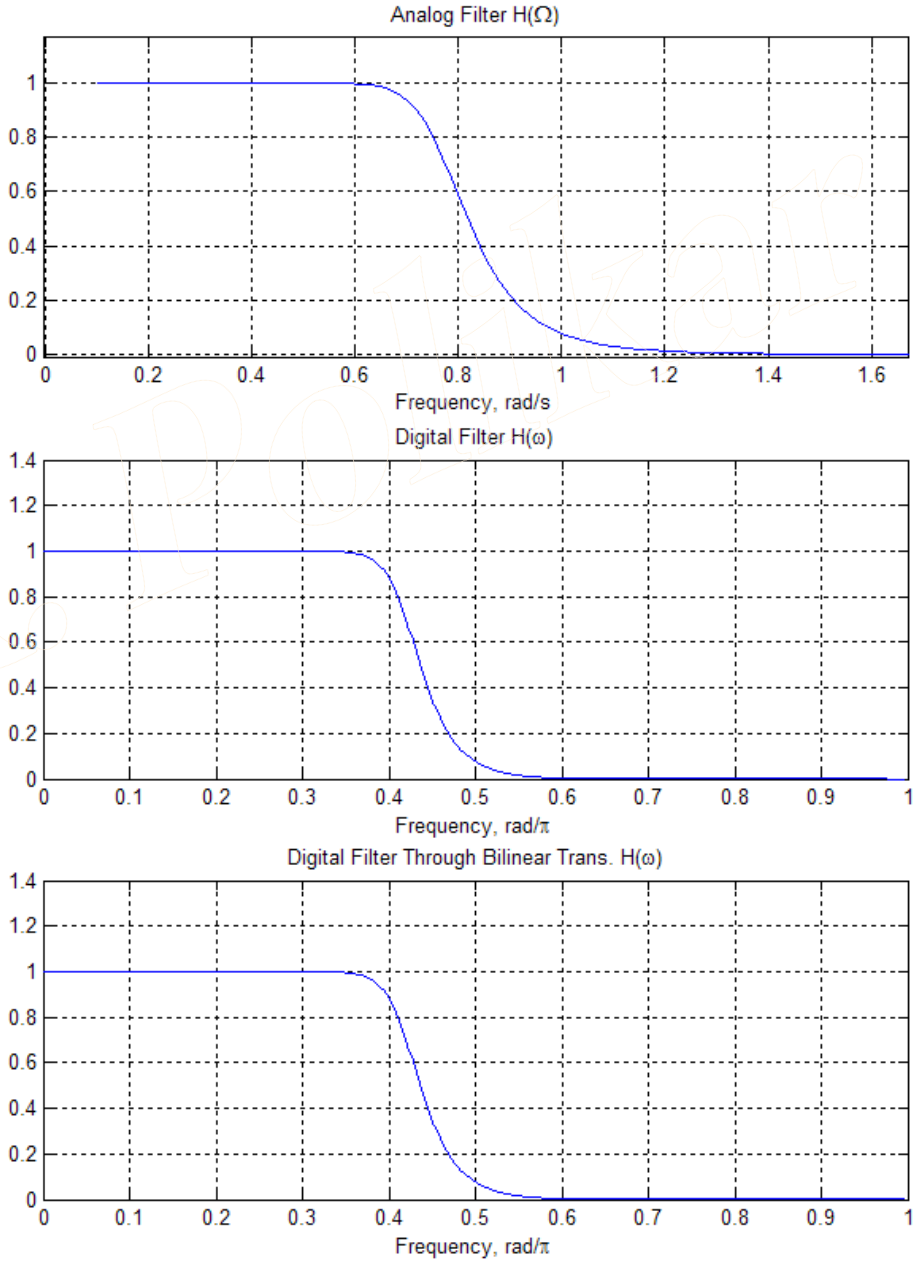


Rowan University

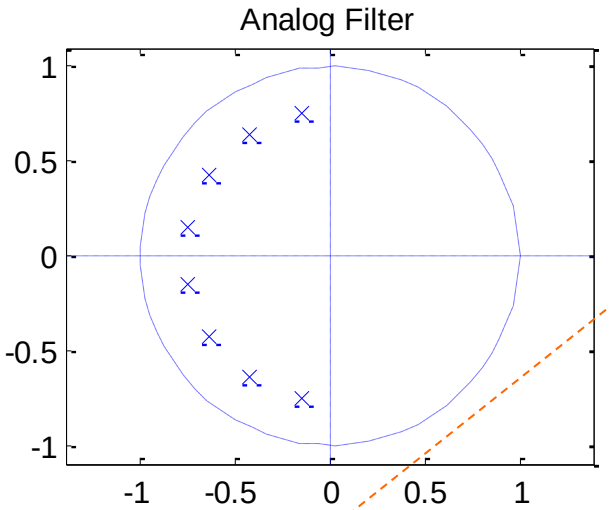
butter_demo.m

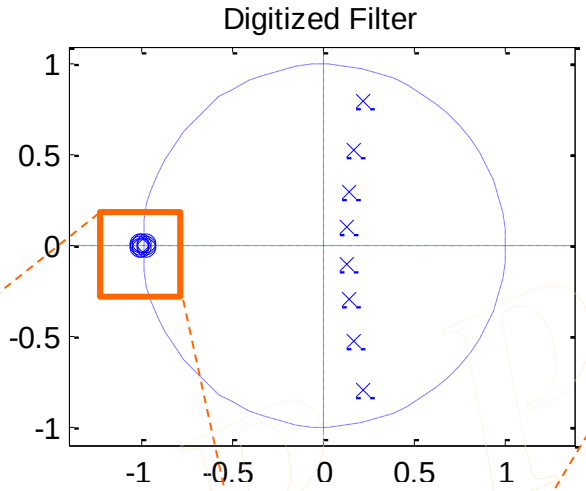
EXAMPLE

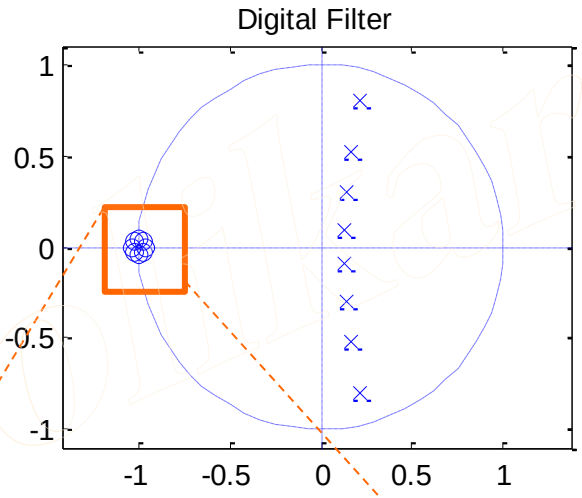
```
[N Wn]=buttord(0.7264, 1.3764, 2, 50, 's')
[b a]=butter(N, Wn, 's');
[H W]=freqs(b,a); % analog frequency response
subplot(311); plot(W, abs(H)); grid
xlabel('Frequency, rad/s')
title('Analog Filter H(\Omega)')

%If we were to design this completely in digital domain

[Nd, Wnd]=buttord(0.4, 0.6, 2, 50);
[bd ad]=butter(Nd, Wnd);
[HD WD]=freqz(bd,ad); % digital frequency response
subplot(312); plot(WD/pi, abs(HD)); grid
xlabel('Frequency, rad/\pi')
title('Digital Filter H(\omega)')

% To convert analog into digital
[bdd add]=bilinear(b, a, 0.5); %Note that we took Fs=0.5
[HDD WDD]=freqz(bdd,add);
subplot(313); plot(WDD/pi, abs(HDD)); grid
xlabel('Frequency, rad/\pi')
title('Digital Filter Through Bilinear Trans. H(\omega)')
```

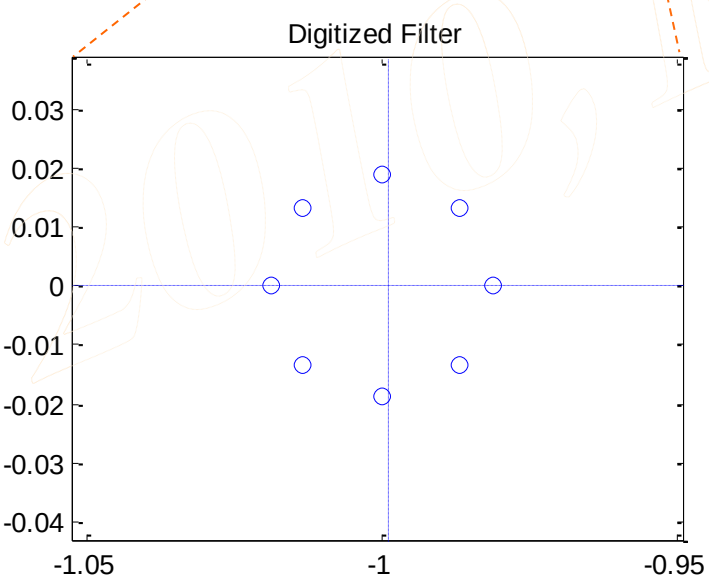


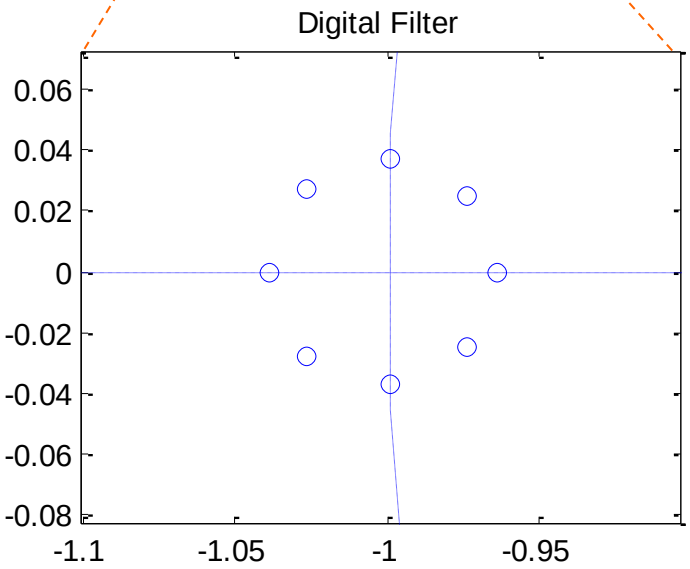
Pole-Zero Plots







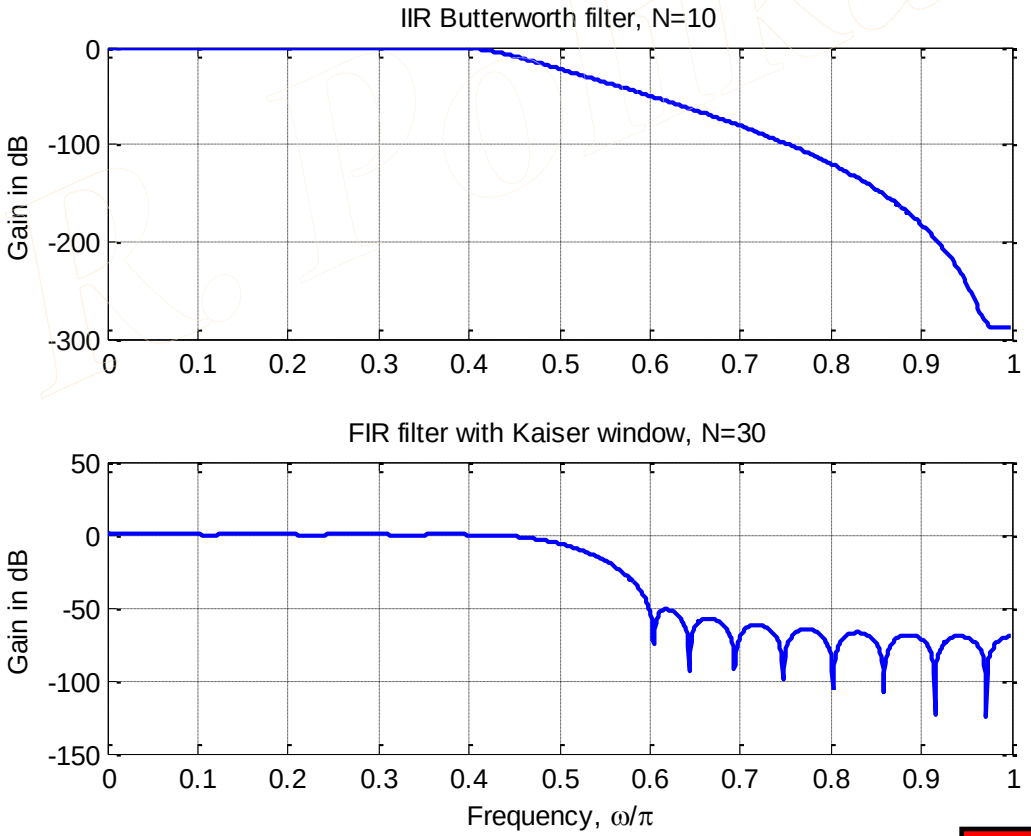




Compare to FIR filter

- Let's design an FIR filter with the exact same specs, using a Kaiser window

```
[N,Wn,beta,type] = kaiserord([0.4 0.6], [1 0], [0.2057 0.0032], 2)
b = fir1(N, Wn, type, kaiser(N+1,beta));
[H w]=freqz(b, 1, 1024);
plot(w/pi, 20*log10(abs(H)));
grid
xlabel('Frequency, \omega/\pi')
title(' Magnitude response of the filter with desired specs')
```



Matlab returns:

N=30,
Wn=0.5,
beta=4.522,
type=low

CHEBYSHEV FILTER

- The (almost) flat passband and stopband characteristics of Butterworth filter come at the cost of ***wide transition band***.
- Two types of Chebyshev filters:
  - Type I has equiripple in the passband, and monotonic behavior in the stopband
  - Type II has equiripple in the stopband, and monotonic behavior in the passband

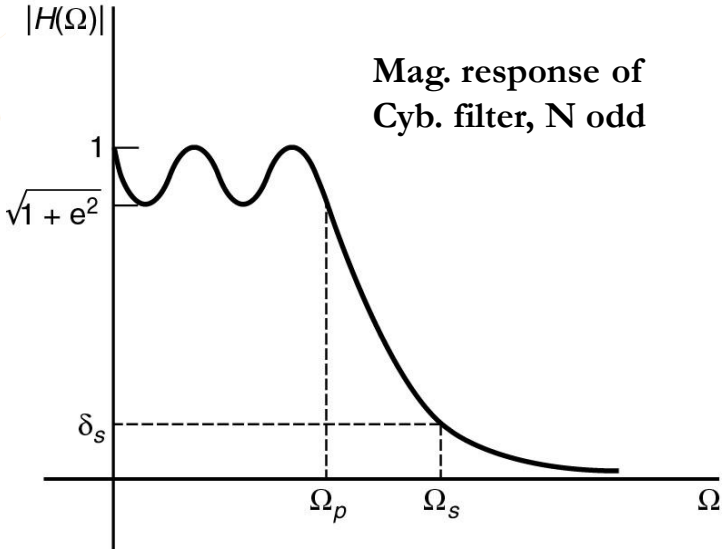
TYPE I

$$|H(\Omega)| = \frac{1}{\sqrt{1+\varepsilon^2 C_N^2\left(\Omega/\Omega_p\right)}}$$

where $C_N(\Omega)$ is the ***Chebyshev polynomial*** of order N:

$$C_N(\Omega) = \begin{cases} \cos\left(N\cos^{-1}\Omega\right), & |\Omega| \le 1 \\ \cosh\left(N\cosh^{-1}\Omega\right), & |\Omega| > 1 \end{cases}$$

ε is a user defined parameter that controls ripple amount.



How to Design a Chebyshev Filter

- Designing a Chebyshev filter requires that the appropriate filter order and cutoff frequency be determined so that the filter will satisfy the specs.
  - Given the four parameters: passband and stopband edge frequencies (Ω_p, Ω_s) , and passband and stopband ripples (δ_p , δ_s) , determine the filter order:

$$\varepsilon = \sqrt{\frac{1}{(1-\delta_p)^2} - 1}$$

$$N = \frac{\cosh^{-1}\left(\frac{1}{\varepsilon}\sqrt{\frac{1}{\delta_s^2} - 1}\right)}{\cosh^{-1}(\Omega_s / \Omega_p)}$$

  - No close form formula exists for computing Ω_c. Therefore, for Type I, take $\Omega_c = \Omega_p$
  - Compute the Chebyshev polynomial Messy...!
  - Determine the poles of the filter ➔ Obtain the analog filter Really messy, in fact, downright ugly!
  - Apply bilinear transformation to obtain the digital filter

Matlab to the rescue!

In Matlab

cheby1() Chebyshev Type I digital **and** analog filter design.
cheby2() Chebyshev Type II digital **and** analog filter design.

[B,A] = **cheby1**(N,R,Wn) designs an N^{th} order **type I** lowpass digital Chebyshev filter with R decibels of peak-to-peak ripple in the passband. **cheby1** returns the filter coefficients in length N+1 vectors **B** (numerator) and **A** (denominator). The cutoff frequency **Wn** must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate. Use R=0.5 as a starting point, if you are unsure about choosing R.

[B,A] = **cheby2**(N,R,Wn) designs an N^{th} order **type II** lowpass digital Chebyshev filter with the stopband ripple R decibels down and stopband edge frequency Wn. **cheby2()** returns the filter coefficients in length N+1 vectors **B** (numerator) and **A** (denominator). The cutoff frequency **Wn** must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate. Use R = 20 as a starting point, if you are unsure about choosing R.

If **Wn** is a two-element vector, **Wn** = [W1 W2], **cheby1** returns an order 2N bandpass filter with passband W1 < W < W2. [**B,A**] = **cheby1** (N,R,**Wn**,'high') designs a highpass filter.
[B,A] = **cheby1**(N,R,**Wn**,'stop') is a bandstop filter if Wn = [W1 W2].

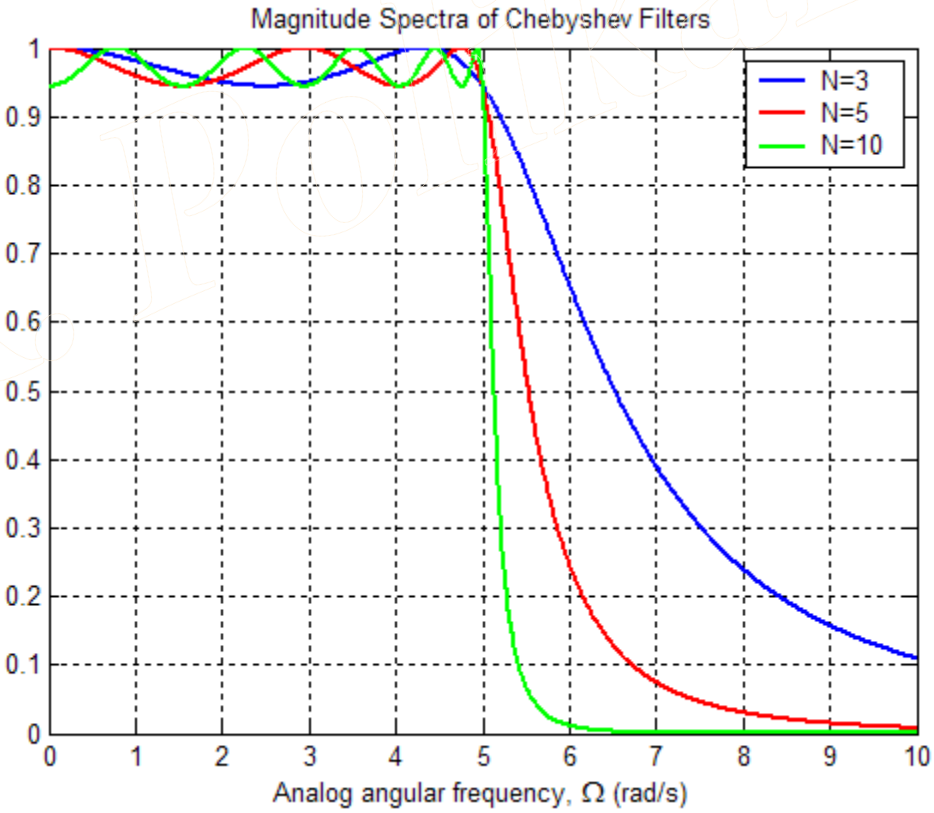
When used with three left-hand arguments, as in [Z,P,K] = **cheby1(...)**, the zeros and poles are returned in length N column vectors **Z** and **P**, and the gain in scalar K.

cheby1(N,R,Wn,'s'), **cheby1**(N,R,Wn,'high','s') **cheby1**,'stop','s') design analog Chebyshev Type I filters. In this case, **Wn** is in [rad/s] and it can be greater than 1.0

Effect of Filter Order

Let's create 3 Chebyshev filters of different orders with a cutoff frequency of 5 rad/s

```
w=linspace(0, 10, 1024); %Create analog frequency base
[b1 a1]=cheby1(3, 0.5, 5, 's');
[b3 a2]=cheby1(5, 0.5, 5, 's');
[b2 a3]=cheby1(10, 0.5, 5, 's');

%Obtain analog frequency responses at "w"
H1=freqs(b1, a1, w);
H2=freqs(b2, a2, w);
H3=freqs(b3, a3, w);

%Normalize magnitude to plot together
H1=abs(H1)/ max(abs(H1));
H2=abs(H2)/ max(abs(H2));
H3=abs(H3)/ max(abs(H3));

plot(w, abs(H1));
grid; hold on
plot(w, abs(H2), 'r')
plot(w, abs(H3), 'g')
legend('N=3', 'N=5', 'N=10')
title('Magnitude Spectra of Chebyshev Filters');
xlabel('Analog angular frequency, \Omega (rad/s)')
```



In Matlab

`cheb1ord()` Chebyshev Type I filter order selection.
`cheb2ord()` Chebyshev Type II filter order selection.

[N, Wn] = **`cheb1ord`**(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Chebyshev Type I filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to π rad/sample.). For example,

Lowpass: Wp = .1, Ws = .2 Highpass: Wp = .2, Ws = .1 **(note the reversal of freq.)**
Bandpass: Wp = [.2 .7], Ws = [.1 .8] Bandstop: Wp = [.1 .8], Ws = [.2 .7]

`cheb1ord()` also returns Wn, the Chebyshev natural frequency to use with **`cheby1()`** to achieve the specifications.

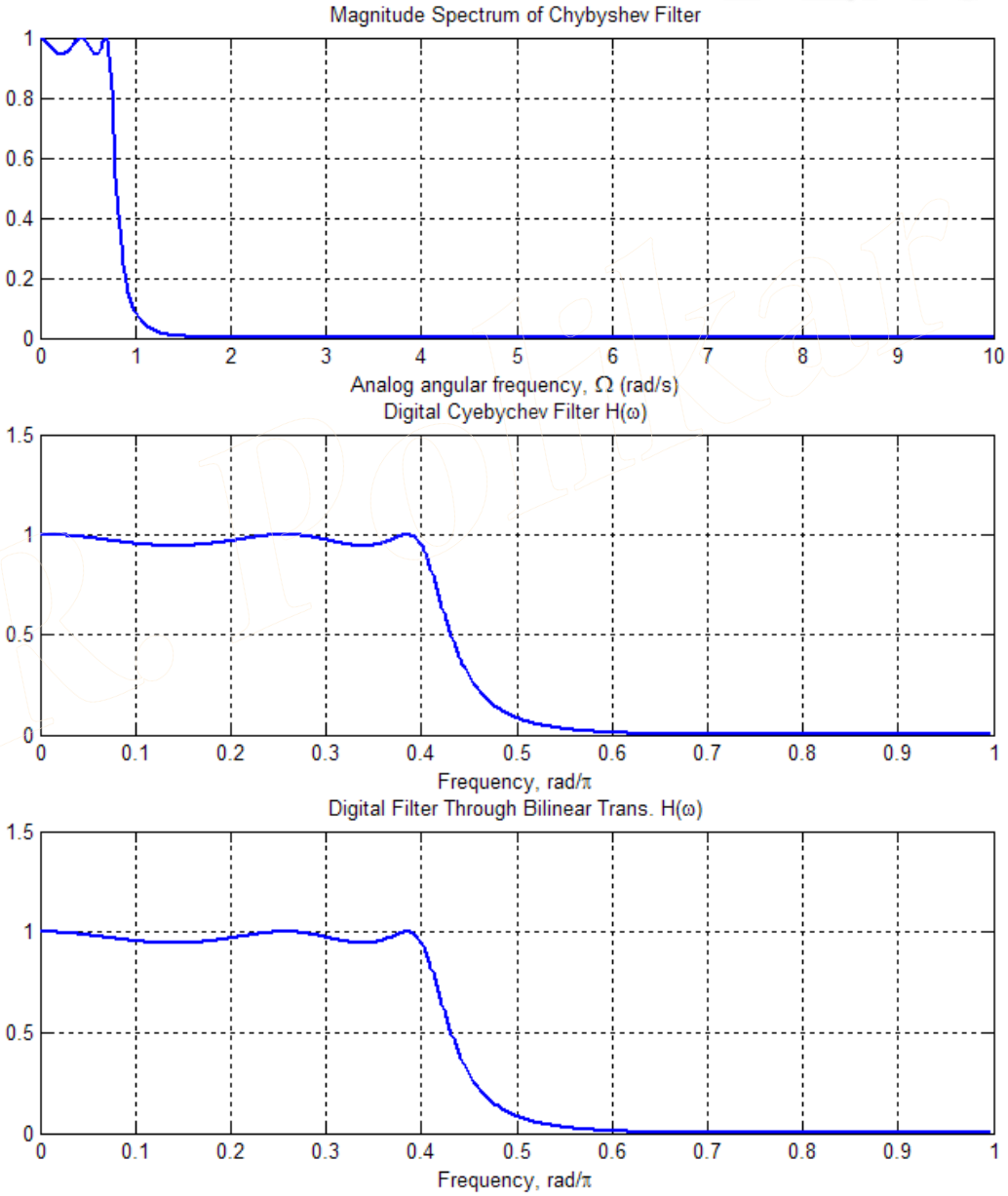
[N, Wn] = **`cheb1ord`**(Wp, Ws, Rp, Rs, 's') does the computation for an **analog filter**, in which case Wp and Ws are in radians/second.

If designing an analog filter, use `bilinear()` to convert the analog filter coeff. to digital filter coeff.

chebyshev_demo2.m

DEMO

```
%Provide wp, ws, rp, rs,   Prewarp, with Ts=2)
Wp=tan(wp/2);
Ws=tan(ws/2);

[N Wn]=cheb1ord(Wp, Ws, rp, rs, 's');
[b a]=cheby1(N, rp, Wn, 's');
[H w]=freqs(b, a);
subplot(311); plot(w, abs(H)); grid

%Design this completely in digital domain

[Nd, Wnd]=cheb1ord(wp/pi, ws/pi, rp, rs);
[bd ad]=cheby1(Nd, rp, Wnd);
[HD WD]=freqz(bd,ad);
subplot(312);  plot(WD/pi, abs(HD)); grid

% To convert analog into digital
[bdd add]=bilinear(b, a, 0.5); %Note Fs=0.5
[HDD WDD]=freqz(bdd,add);
subplot(313); plot(WDD/pi, abs(HDD)); grid
```



Elliptic Filters (Cauer Filter)

- Provides much sharper transition band at the expense of equiripple in both bands, and nonlinear behavior in the passband.

$$|H(\Omega)| = \frac{1}{\sqrt{1+\varepsilon^2 U_N^2(\Omega/\Omega_c)}}$$

  - Where ε again controls the ripple amount and U_N is some cryptic function (Jacobian elliptic function of order N)

$$U\ \Omega = \int_{\theta=0}^{\Omega} \frac{d\theta}{\sqrt{1-N\sin^2\theta}}$$

**(if you must insist….
see if you can pull N out of this!)**

The functions **`ellipord()`** and **`ellip()`** determine the elliptic filter parameters, and design the elliptic filter, respectively. The syntax and usage are similar to those of Chebyshev.

[N, Wn] = `ellipord`(Wp, Ws, Rp, Rs, 's');

[b, a] = `ellip`(N, Rp, Rs, Wn, 's');

BESSEL FILTER

- Yet another cryptically designed filter, used only in most desperate situations.
  - Advantage: Has (approximately) linear phase in the passband, though there is no guarantee that the linearity will be preserved when bilinear transform is applied.
  - Disadvantage: Horrendously wide transition band (no free lunch!)

In Matlab

[z, p, k]=**besselap**(N) returns the poles and gain of an order N Bessel ***normalized*** analog lowpass filter prototype (Ω_c=1) . $N \leq 25$. The function returns the poles in the length N column vector p and the gain in scalar k. z is an empty matrix because there are no zeros.

Analog Bessel filters are characterized by a group delay that is maximally flat at zero frequency and almost constant throughout the passband. The group delay at zero frequency is

$$\tau_g(\Omega_c) = -\left.\frac{d\theta(\Omega)}{d\Omega}\right|_{\Omega=\Omega_c} = \left(\frac{(2N)!}{2^N N!}\right)^{1/N}$$

Bessel Filters

[B,A] = **besself**(N,Wn) designs an N^{th} order lowpass analog Bessel filter and returns the filter coefficients in length N+1 vectors **B** and **A**. The cut-off frequency Wn must be greater than 0. The group delay will then be approximately constant up to this frequency. The larger the filter order, the better the filter's group delay will approximate a constant up to the frequency Wn.

Note that Bessel filters are typically designed as lowpass, though matlab allows you to design highbass, bandpass using the a simialr syntax as with other filters.

When used with three left-hand arguments, as in [Z,P,K] = **besself**(...), the zeros and poles are returned in length N column vectors Z and P, and the gain in scalar K.

Also note that lowpass Bessel filters have a monotonically decreasing magnitude response, as do lowpass Butterworth filters. Compared to the Butterworth, Chebyshev, and elliptic filters, **the Bessel filter has the slowest rolloff and requires the highest order** to meet an attenuation specification.

Spectral Transformations

- But what about other types of filters?
- Spectral transformations: Process for converting lowpass filters into highpass, bandpass or bandstop filters
  - In fact, spectral transformations can be used to convert a LPF into another LPF with a different cutoff frequency.
  - The transformation can be done in either analog or discrete domain.
- Here is how to ***"design a non-lowpass filter by designing a lowpass filter"***
  - Prewarp the digital filter edge frequencies into analog frequencies
  - Transform the filter specs into LPF specs using spectral transformation
  - Design analog LPF
  - Obtain the corresponding desired digital filter by either
    1. Convert the analog LPF to digital LPF using bilinear transformation and use the digital – to – digital spectral transformation to obtain the non-LPF characteristic
    2. Convert the analog LPF to the desired analog non-LPF using the analog – to – analog spectral transformation, followed by the bilinear transformation to obtain the digital equivalent. This is the approach taken by Matlab.

Analog –to– Analog Transformations

LP→LP

$$s = \frac{\Omega_p}{\hat{\Omega}_p^{LP}}\hat{s} \Rightarrow \Omega = \frac{\Omega_p}{\hat{\Omega}_p^{LP}}\hat{\Omega}$$

LP→HP

$$s = \frac{\Omega_p \hat{\Omega}_p^{HP}}{\hat{s}} \Rightarrow \Omega = -\frac{\Omega_p \hat{\Omega}_p^{HP}}{\hat{\Omega}}$$

LP→BP

$$s = \Omega_p \frac{\hat{s}^2 + \hat{\Omega}_{p1}^{BP}\hat{\Omega}_{p2}^{BP}}{\hat{s}\left(\hat{\Omega}_{p2}^{BP} - \hat{\Omega}_{p1}^{BP}\right)} \Rightarrow \Omega = -\Omega_p \frac{\overbrace{\hat{\Omega}_{p1}^{BP}\hat{\Omega}_{p2}^{BP}}^{\hat{\Omega}_0^2} - \hat{\Omega}^2}{\hat{\Omega}\underbrace{\left(\hat{\Omega}_{p2}^{BP} - \hat{\Omega}_{p1}^{BP}\right)}_{BW}}$$

LP→BS

$$s = \Omega_s \frac{\hat{s}\left(\hat{\Omega}_{s2}^{BS} - \hat{\Omega}_{s1}^{BS}\right)}{\hat{s}^2 + \hat{\Omega}_{s1}^{BS}\hat{\Omega}_{s2}^{BS}} \Rightarrow \Omega = \Omega_s \frac{\overbrace{\hat{\Omega}\left(\hat{\Omega}_{s2}^{BS} - \hat{\Omega}_{s1}^{BS}\right)}^{BW}}{\underbrace{\hat{\Omega}_{s1}^{BS}\hat{\Omega}_{s2}^{BS}}_{\hat{\Omega}_0^2} - \hat{\Omega}^2}$$

Ω / $\hat{\Omega}$: Prototype / desired frequency

Ω_p : Passband edge-frequency of prototype analog LPF, typically normalized to 1 rad/s.

$\hat{\Omega}_p^{LP}$: Passband edge-frequency of desired analog LPF

$\hat{\Omega}_p^{HP}$: Passband edge-frequency of desired analog HPF

$\hat{\Omega}_{p1}^{BP}$: Lower passband edge-frequency of desired analog BPF

$\hat{\Omega}_{p2}^{BP}$: Upper passband edge-frequency of desired analog BPF

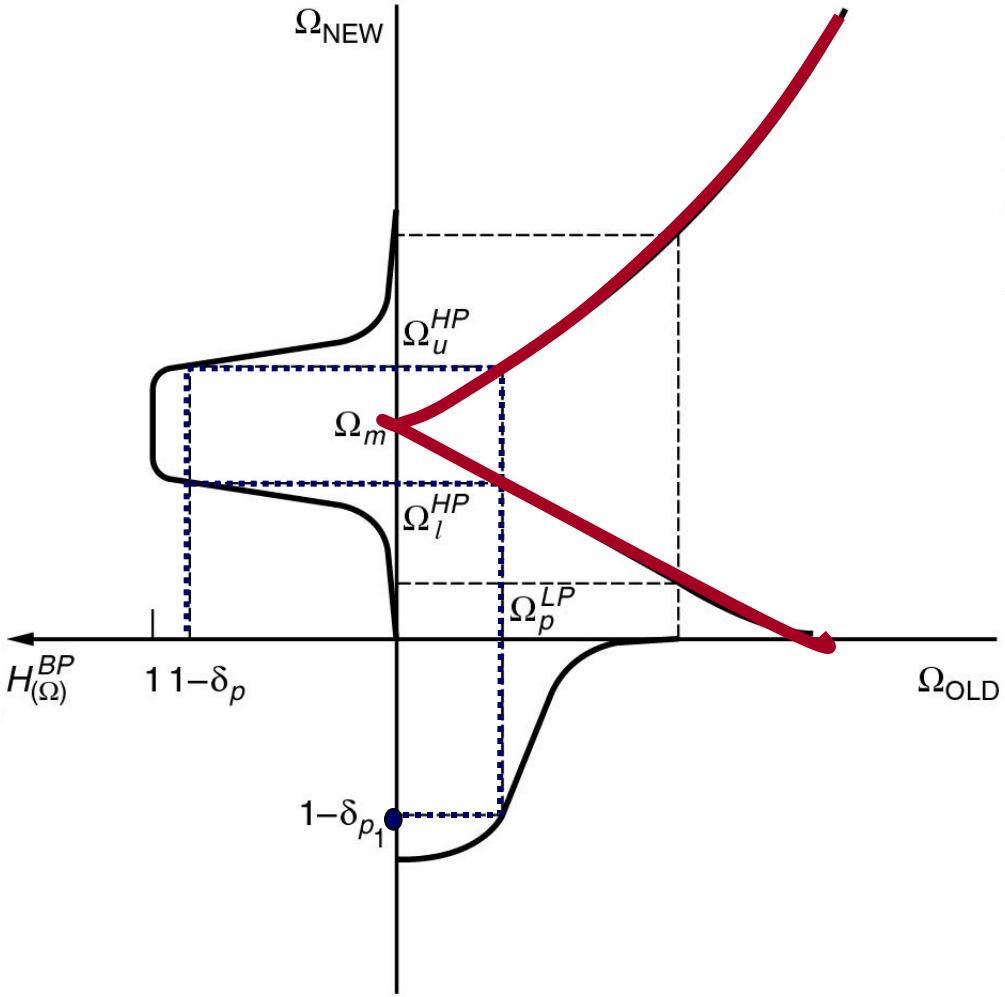
$\hat{\Omega}_{s1}^{BS}$: Lower passband edge-frequency of desired analog BSF

$\hat{\Omega}_{s2}^{BS}$: Upper passband edge-frequency of desired analog BSF

$\hat{\Omega}_0$: Center frequency of the desired f.

BW: Bandwidth of the desired filter

Analog –to – Analog Transformations



EXAMPLE

- Design a type I Chebyshev digital IIR highpass filter with the following specifications: F_p=700 Hz, F_{sp}=500 Hz, α_p=1dB, α_s=32dB, F_T=2kHz
  - First compute normalized angular digital frequencies:

$$\omega_s = \frac{2\pi F_{sp}}{F_T} = \frac{2\pi \times 500}{2000} = 0.5\pi \qquad \omega_p = \frac{2\pi F_p}{F_T} = \frac{2\pi \times 700}{2000} = 0.7\pi$$

  - Prewarp these frequencies:

$$\hat{\Omega}_p^{HP} = \tan\left(\omega_p / 2\right) = 1.9626105 \qquad \hat{\Omega}_s^{HP} = \tan\left(\omega_s / 2\right) = 1.0$$

  - For the prototype lowpass filter pick Ω_p=1 rad/s
  - Use the spectral transformation to convert these into lowpass equivalents:

$$s = \frac{\Omega_p \Omega_p^{HP}}{\hat{s}} \Rightarrow \Omega = -\frac{\Omega_p \hat{\Omega}_p^{HP}}{\hat{\Omega}} \quad (s = j\Omega) \qquad \Omega_s = -\frac{\Omega_p \Omega_p^{HP}}{\Omega_s^{HP}} = \frac{1 \cdot 1.962105}{1} = 1.962105$$

  - Then the analog prototype LPF specs are: Ω_p=1, Ω_s=1.962105, α_p=1dB, α_s=32dB

Digital – to – Digital transformations

- Digital – to – digital transformations are obtained by replacing every z^{-1} in H(z) with a mapping function $F(z^{-1})$
- In digital – to – digital transformations, the following conditions must be satisfied:
  1. The mapping must transform the LP prototype into the desired type of filter (duh!)
  2. Inside of the unit circle must be mapped to itself (so that stability is insured – part 1)
  3. The unit circle must also be mapped to itself (so that the stability is insured – part 2).
- The transformations given on the right satisfy all of these requirements

Filter type	Spectral transformation	Design parameters
Lowpass	$z^{-1} = \frac{\hat{z}^{-1} - \alpha}{1 - \alpha \hat{z}^{-1}}$	$\alpha = \frac{\sin\left(\frac{\omega_c - \hat{\omega}_c}{2}\right)}{\sin\left(\frac{\omega_c + \hat{\omega}_c}{2}\right)}$ $\hat{\omega}_c$ = desired cutoff frequency
Highpass	$z^{-1} = -\frac{\hat{z}^{-1} + \alpha}{1 + \alpha \hat{z}^{-1}}$	$\alpha = -\frac{\cos\left(\frac{\omega_c + \hat{\omega}_c}{2}\right)}{\cos\left(\frac{\omega_c - \hat{\omega}_c}{2}\right)}$ $\hat{\omega}_c$ = desired cutoff frequency
Bandpass	$z^{-1} = -\frac{\hat{z}^{-2} - \frac{2\alpha\beta}{\beta+1}\hat{z}^{-1} + \frac{\beta-1}{\beta+1}}{\frac{\beta-1}{\beta+1}\hat{z}^{-2} - \frac{2\alpha\beta}{\beta+1}\hat{z}^{-1} + 1}$	$\alpha = \frac{\cos\left(\frac{\hat{\omega}_{c2} + \hat{\omega}_{c1}}{2}\right)}{\cos\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)}$ $\beta = \cot\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)\tan\left(\frac{\omega_c}{2}\right)$ $\hat{\omega}_{c2}, \hat{\omega}_{c1}$ = desired upper and lower cutoff frequencies
Bandstop	$z^{-1} = \frac{\hat{z}^{-2} - \frac{2\alpha}{1+\beta}\hat{z}^{-1} + \frac{1-\beta}{1+\beta}}{\frac{1-\beta}{1+\beta}\hat{z}^{-2} - \frac{2\alpha}{1+\beta}\hat{z}^{-1} + 1}$	$\alpha = \frac{\cos\left(\frac{\hat{\omega}_{c2} + \hat{\omega}_{c1}}{2}\right)}{\cos\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)}$ $\beta = \tan\left(\frac{\hat{\omega}_{c2} - \hat{\omega}_{c1}}{2}\right)\tan\left(\frac{\omega_c}{2}\right)$ $\hat{\omega}_{c2}, \hat{\omega}_{c1}$ = desired upper and lower cutoff frequencies

NEEDLESS TO SAY…

- For any application of practical complexity, these transformations cannot be done by hand. Digital filter design software, such as Matlab, is typically used for the transformations. These transformations are all for analog transformation and must be followed by bilinear transformation for digital filter design.

`lp2lp()` Lowpass to lowpass analog filter transformation.
[NUMT,DENT] = **`lp2lp`**(NUM,DEN,Wo) transforms the lowpass filter prototype NUM(s)/DEN(s) with unity cutoff frequency of 1 rad/sec to a lowpass filter with cutoff frequency Wo (rad/sec).

`lp2hp()` Lowpass to highpass analog filter transformation.
[NUMT,DENT] = **`lp2hp`**(NUM,DEN,Wo) transforms the lowpass filter prototype NUM(s)/DEN(s) with unity cutoff frequency to a highpass filter with cutoff frequency Wo.

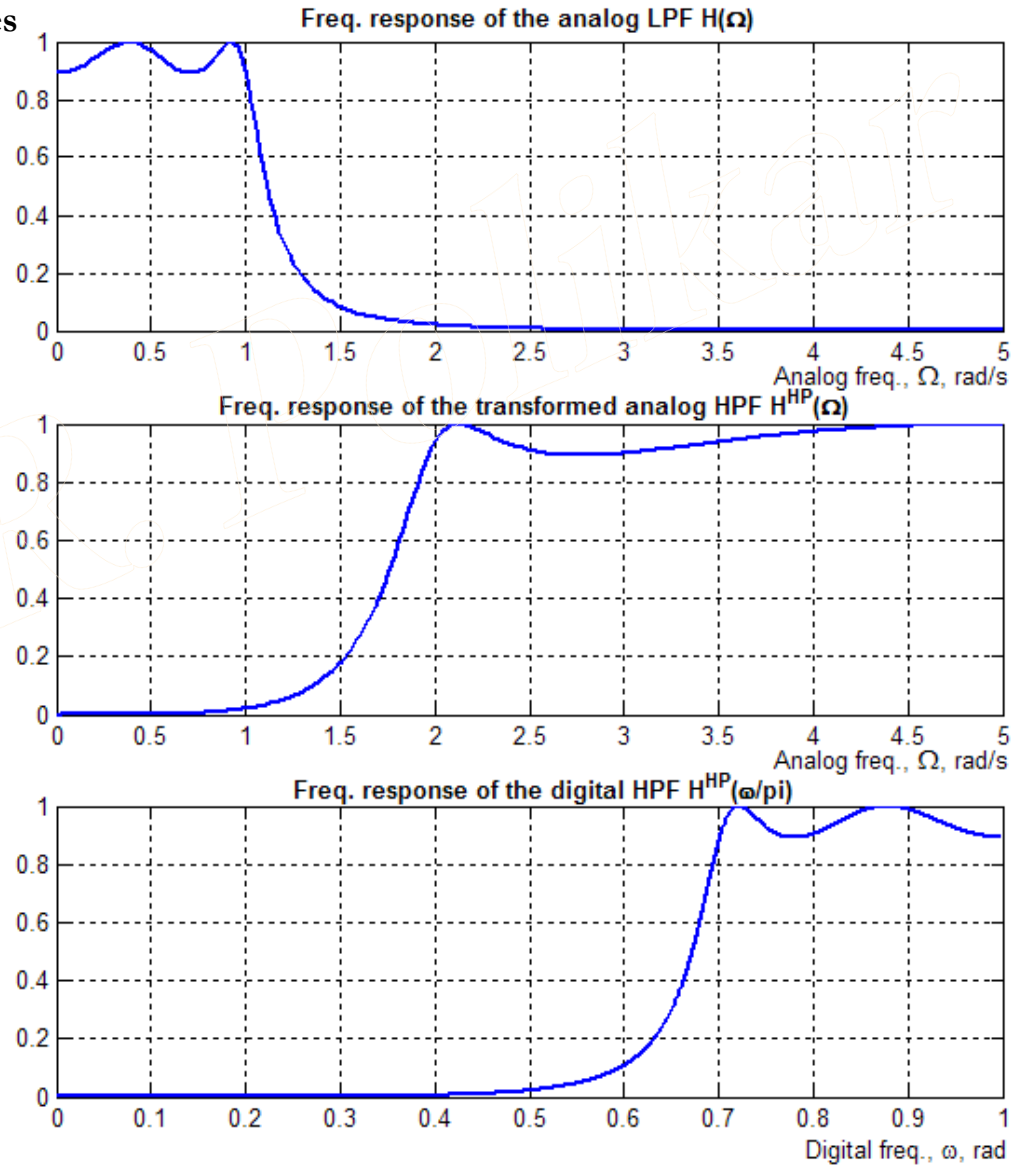
`lp2bp()` Lowpass to bandpass analog filter transformation.
[NUMT,DENT] = **`lp2bp`**(NUM,DEN,Wo,Bw) transforms the lowpass filter prototype NUM(s)/DEN(s) with unity cutoff frequency to a bandpass filter with center frequency Wo and bandwidth Bw.

`lp2bs()` Lowpass to bandstop analog filter transformation.
[NUMT,DENT] = **`lp2bs`**(NUM,DEN,Wo,Bw) transforms the lowpass filter prototype NUM(s)/DEN(s) with unity cutoff frequency to a bandstop filter with center frequency Wo and bandwidth Bw.

spect_trans_demo.m

Example Cont.

```
%Omega_p=tan(wp/2);  %convert to analog frequencies
%Omega_s=tan(ws/2);
%Wp=1;   Ws=Wp*Omega_p/Omega_s;

[N, Wn] = cheb1ord(1, 1.9626105, 1, 32, 's');
[B, A] = cheby1(N, rp, Wn, 's');

WC=linspace(0, 5, 200); subplot(311)
[HC]=freqs(B,A, WC);
plot(WC, abs(HC)); grid
title('Freq. response of the analog LPF H(\Omega)');
xlabel('Analog freq., \Omega, rad/s')

[BT, AT] = lp2hp(B, A, 1.9626105);
[HT]=freqs(BT,AT, WC);
subplot(312); plot(WC, abs(HT)); grid
title('Freq. response of the transformed analog….')
xlabel('Analog freq., \Omega, rad/s')

w=linspace(0, pi, 512);  %512 is default size for freqz
[num, den] = bilinear(BT, AT, 0.5);
subplot(313); [H w]=freqz(num, den);
plot(w/pi, abs(H)); grid
title('Freq. response of the digital HPF
H^{HP}(\omega/pi)');
xlabel('Digital freq., \omega, rad')
```



Direct Design

- The described techniques are the classic analog and digital filter design techniques that have been traditionally used
- Several numeric and recursive (iterative) optimization techniques are also available that allow us to design IIR filters directly by approximating the desired frequency response with an appropriate transfer function
- A definite advantage of such optimization techniques is that they are no longer restricted to standard filter types, such as the lowpass, highpass, bandpass or bandstop
  - Arbitrary filter shapes can be easily designed.
  - One example is the **`yulewalk()`** function in matlab, which is the IIR counterpart of the **`remez()`** and **`fir2()`** functions used for FIR filter design.

Direct IIR Design

`yulewalk()` Recursive filter design using a least-squares method.

[B,A] = **`yulewalk`**(N,F,M) finds the N^{th} order recursive filter coefficients **B** and **A** such that the filter:

$$\frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + ... + b(n)z^{-(n-1)}}{1 + a(1)z^{-1} + ... + a(n)z^{-(n-1)}}$$

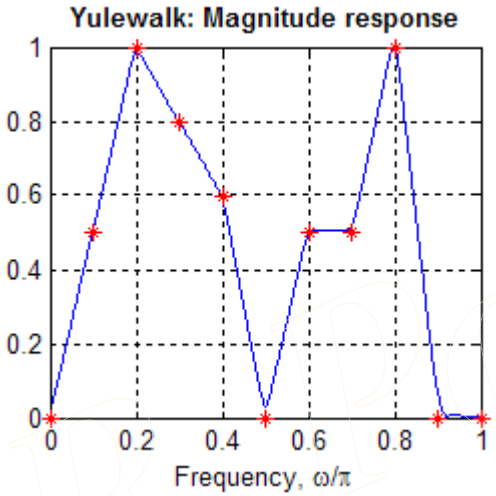
matches the magnitude frequency response given by vectors **F** and **M**. Vectors **F** and **M** specify the frequency and magnitude breakpoints for the filter such that **`plot`**(**F**,**M**) would show a plot of the desired frequency response. The frequencies in **F** must be between 0.0 and 1.0,with 1.0 corresponding to half the sample rate. They must be in increasing order and start with 0.0 and end with 1.0.

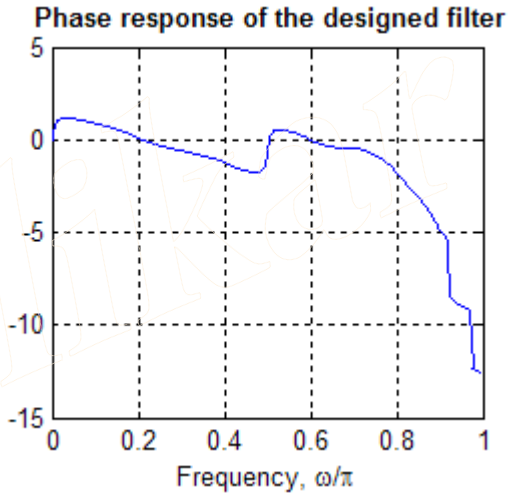
The filter order *N* usually needs to be determined by trial and error.

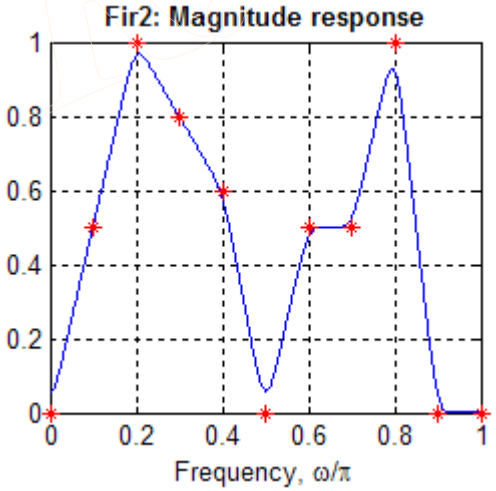
yulewalk_demo.m

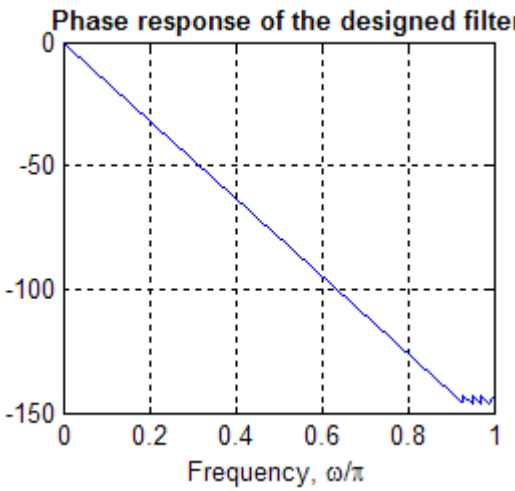
YULEWALK VS. FIR2
IIR VS. FIR

```
freq=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]
amp=[0 0.5 1 0.8 0.6 0 0.5 0.5 1 0 0]
[b1 a1]=yulewalk(25,  freq, amp);
b2=fir2(100, freq, amp, 1024);
subplot(221)
[H1 w]=freqz(b1, a1, 1024);
plot(w/pi, abs(H1)); hold on
plot(freq, amp, 'r*'); grid
xlabel('Frequency, \omega/\pi')
title(' Yulewalk: Magnitude response ')
subplot(222)
plot(w/pi, unwrap(angle(H1))); grid
xlabel('Frequency, \omega/\pi')
title(' Phase response of the designed filter')

subplot(223)
[H2 w]=freqz(b2, 1, 1024);
plot(w/pi, abs(H2)); hold on
plot(freq, amp, 'r*'); grid
xlabel('Frequency, \omega/\pi')
title(' Fir2: Magnitude response')
subplot(224)
plot(w/pi, unwrap(angle(H2))); grid
xlabel('Frequency, \omega/\pi')
title(' Phase response of the designed filter')
```









Compare filter orders and phase responses !!!

IIR vs. FIR