

EE434

Biomedical Signal Processing

Lecture # 3

Instructor: M. Zübeyir Ünlü, PhD

zubeyirunlu@iyte.edu.tr

Digital Signal Processing

A Review - Part 2

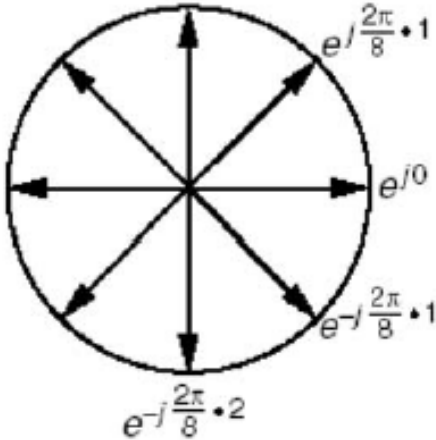
- **DTFT** is an important tool in digital signal processing, as it provides the spectral content of a discrete time signal
- However, the computed spectrum, $X(\omega)$ is a **continuous function** of ω , and therefore cannot be computed using a computer (the **freqz** function computes only an approximation of the **DTFT**, not the actual **DTFT**)
- We need a method to compute the spectral content of a discrete time signal and have a spectrum – actually a **discrete function** – so that it can be computed using a digital computer

- **A straightforward solution:** Simply sample the frequency variable ω of the DTFT in frequency domain in the $[0\ 2\pi]$ interval
- If we want, say N points in the frequency domain, then we divide ω in the $[0\ 2\pi]$ interval into N equal intervals
- Then the discrete values of ω are $0, 2\pi/N, 2\cdot 2\pi/N, 3\cdot 2\pi/N, \dots, (N-1)\cdot 2\pi/N$

- **Definition** - The simplest relation between a length- N sequence $x[n]$, defined for $0 \leq n \leq N-1$, and its DTFT $X(\omega)$ is obtained by uniformly sampling $X(\omega)$ on the ω -axis $0 \leq \omega \leq 2\pi$ at $\omega_k = 2\pi k/N$, $0 \leq k \leq N-1$
- From the definition of the DTFT we thus have
- **DFT analysis equation:**

$$X[k] = X(\omega) \Big|_{\omega=2\pi k/N} = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, \quad 0 \leq k \leq N-1$$

- DFT is **periodic** in both **time** and **frequency** domains!!!
 - Even though the original time domain sequence to be transformed is **not periodic**!
- There are several ways to explain this phenomenon. Mathematically , we can easily show that both the analysis and synthesis equations are periodic by N
- Now, understanding that **DFT is periodic in frequency domain** is straightforward: DFT is obtained by sampling DTFT at $2\pi/N$ intervals. Since DTFT was periodic with $2\pi \rightarrow$ DFT is periodic by N
- This can also be seen easily from the complex exponential wheel. Since there are only N vectors around a unit circle, the transform will repeat itself every N points



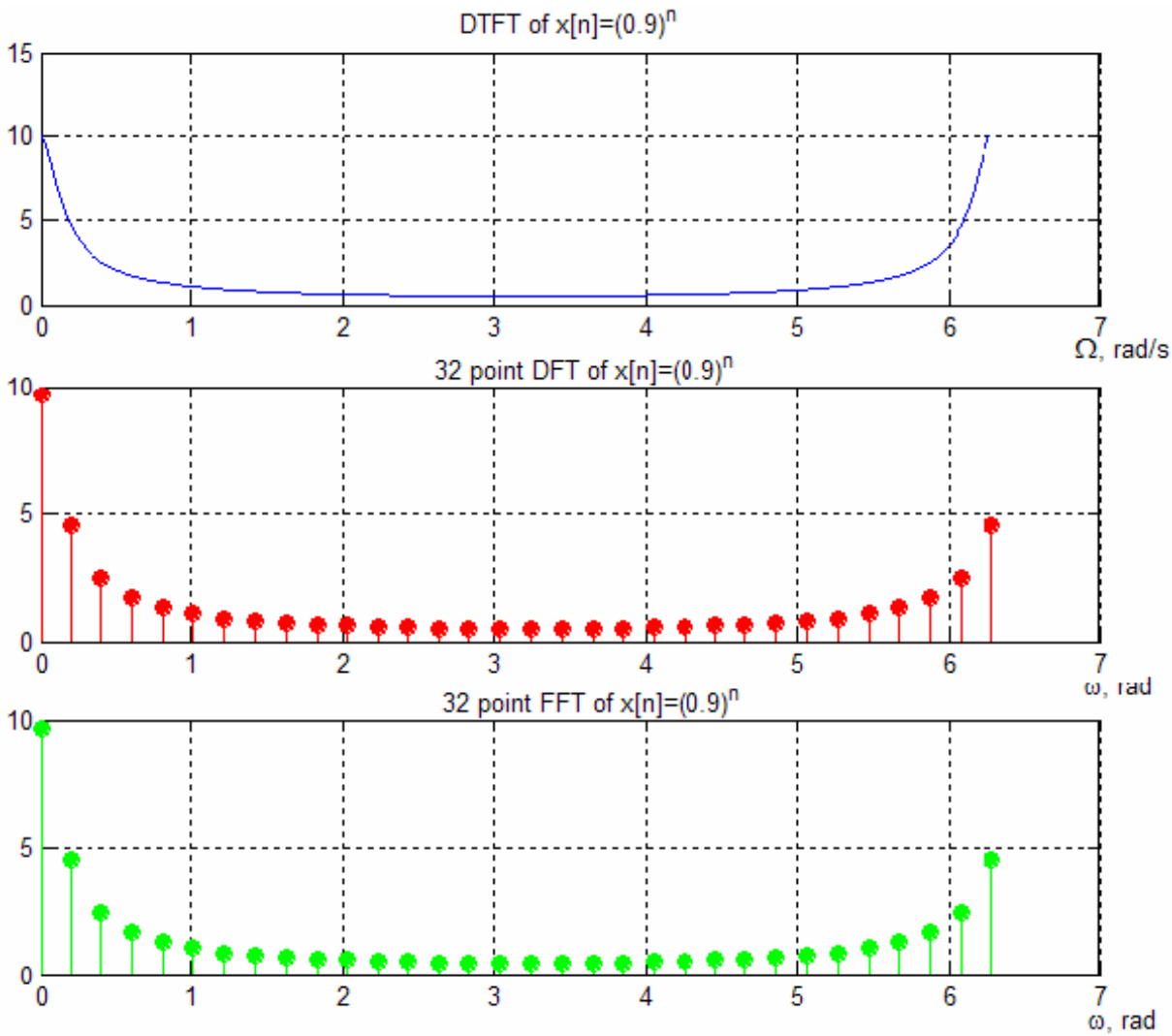
- But what does it mean that **DFT is also periodic in time domain**?
- Recall how we obtained the frequency spectrum of a sampled signal:
 - The spectrum of the sampled signal was identical to that of the continuous signal, except, with periodically replications of itself every 2π
 - That is, **sampling the signal in time domain, caused the frequency domain to be periodic with 2π**
- In obtaining DFT, we did the opposite: sample the frequency domain
 - From the **duality** of the Fourier transform, this corresponds to making the time domain periodic
 - However, the original signal was not periodic!
 - This is an artifact of the mathematical manipulations.

- Here is what is happening:
- When we sample the DTFT in frequency domain, the resulting “discrete spectrum” is not spectrum of the original discrete signal. Rather, the sampled spectrum is in fact the spectrum of a time domain signal that consists of periodically replicated versions of the original discrete signal
- Similar to sampling theorem, under rather mild conditions, we can reconstruct the DTFT, $X(\omega)$, from its DFT samples $X[k]$.

- In Matlab, the `fft()` computes **DFT** using a fast algorithm, called **Fast Fourier Transform (FFT)**
- `X = fft(x)` returns the **Discrete Fourier Transform (DFT)** of vector `X`, computed with a **Fast Fourier Transform (FFT)** algorithm
 - If `x` is a matrix, `fft` returns the Fourier transform of each column of the matrix. In this case, the length of `X` and the length of `x` are identical
 - `X = fft(x,N)` returns the N -point DFT. If the length of `x` is less than N , `x` is padded with trailing zeros to length N . If the length of `x` is greater than N , the sequence `x` is truncated
 - The N points returned by `fft` corresponds to frequencies in the $[0\ 2\pi]$ range, equally spaced with an interval of $2\pi/N$
 - Note that the N^{th} FFT point corresponds to 2π , which in turn corresponds to the sampling frequency
 - If `x[n]` is real, `X[k]` is symmetric. Using the `fftshift()` function shifts the center of symmetry so that the FFT is given in the $[-\pi\ \pi]$ interval, rather than $[0\ 2\pi]$
- `x=ifft(X,N)` returns the N -point **inverse discrete Fourier transform**

- Let's show that
 - DFT is indeed the sampled version of DTFT and
 - DFT and FFT produce identical results

```
n=0:31; k=0:31;  
x=0.9.^n;  
w=linspace(0, 2*pi, 512);  
K=linspace(0, 2*pi, 32);  
X1=1./(1-0.9*exp(-j*w)); % DTFT  
X2=(1-(0.9*exp(-j*(2*pi/32)*k)).^32)./  
(1-0.9*exp(-j*(2*pi/32)*k)); %DFT  
X=fft(x); %FFT  
subplot(311)  
plot(w, abs(X1)); grid  
subplot(312)  
stem(K, abs(X2), 'r', 'filled'); grid  
subplot(313)  
stem(K, abs(X), 'g', 'filled'); grid
```



- A generalization of the DTFT leads to the **z-transform** that may exist for many signals for which the DTFT does not
- DTFT is in fact a special case of the **z-transform**
- - ...just like the Fourier transform is a special case of _____(?)
- Furthermore, the use of the **z-transform** allows simple algebraic expressions to be used which greatly simplifies frequency domain analysis
- Digital filters are designed, expressed, applied and represented in terms of the **z-transform**
- For a given sequence $x[n]$, its **z-transform** $X(z)$ is defined as

$$X(z) = Z\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \quad x[n] \overset{Z}{\Leftrightarrow} X(z)$$

where z lies in the complex space, that is, $z = a + jb = re^{j\omega}$

- From the definition of the z -variable

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} = \sum_{n=-\infty}^{\infty} x[n](re^{j\omega})^{-n} = \sum_{n=-\infty}^{\infty} x[n]r^{-n}e^{-j\omega n}$$

- It follows that the DTFT is indeed a special case of the z -transform, specifically, z -transform reduces to DTFT for the special case of $r = 1$, that is, $|z|=1$, provided that the latter exists

$$X(\omega) = X(z)\Big|_{z=e^{j\omega}}$$

- The contour $|z|=1$ is a circle in the z -plane of unit radius \rightarrow the unit circle
- Hence, the DTFT is really the z -transform evaluated on the unit circle
- Just like the DTFT, z -transform too has its own, albeit less restrictive, convergence requirements, specifically, the infinite series $\sum_{n=-\infty}^{\infty} x[n]z^{-n}$ must converge
- For a given sequence, the set R of values of z for which its z -transform converges is called the **region of convergence (ROC)**

- Determine the **z-transform** and the corresponding **ROC** of the **causal** sequence $x[n]=\alpha^n u[n]$

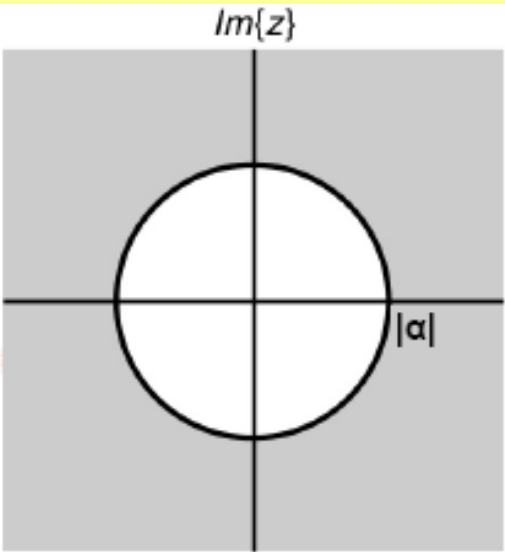
$$X(z) = \sum_{n=-\infty}^{\infty} \alpha^n u[n] z^{-n} = \sum_{n=0}^{\infty} \alpha^n z^{-n}$$

This power (geometric) series converges to

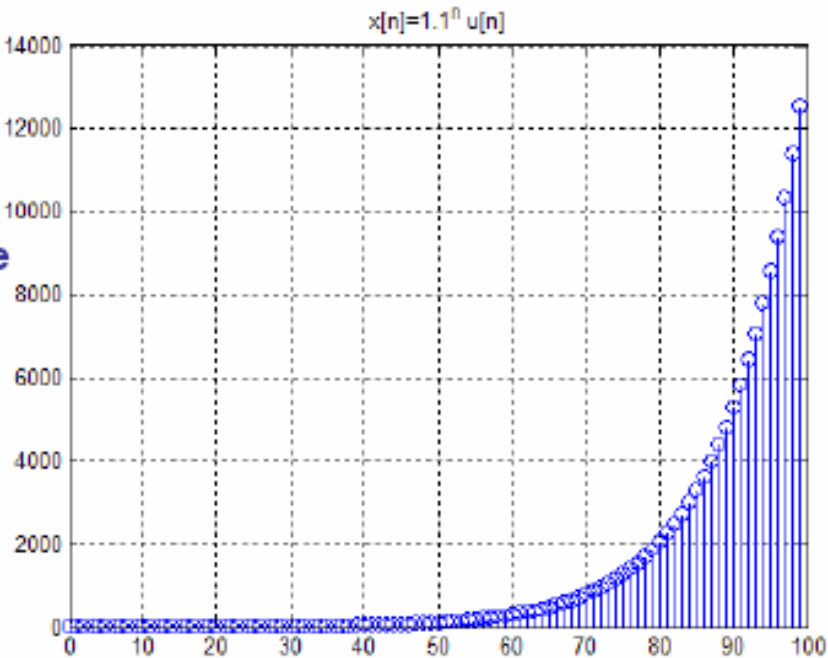
$$X(z) = \frac{1}{1 - \alpha z^{-1}}, \quad \text{for } |\alpha z^{-1}| < 1$$

$$= \frac{z}{z - \alpha}, \quad \text{for } \infty > |z| > |\alpha|$$

→ **ROC is the annular region** $|z| > |\alpha|$



- Note that this sequence does not have a DTFT if $|\alpha|>1$, however, it does have a z-transform!
- This is a right-sided sequence, which has an ROC that is outside of a circular area!



- Now consider the **anti-causal** sequence $y[n] = -\alpha^n u[-n-1]$

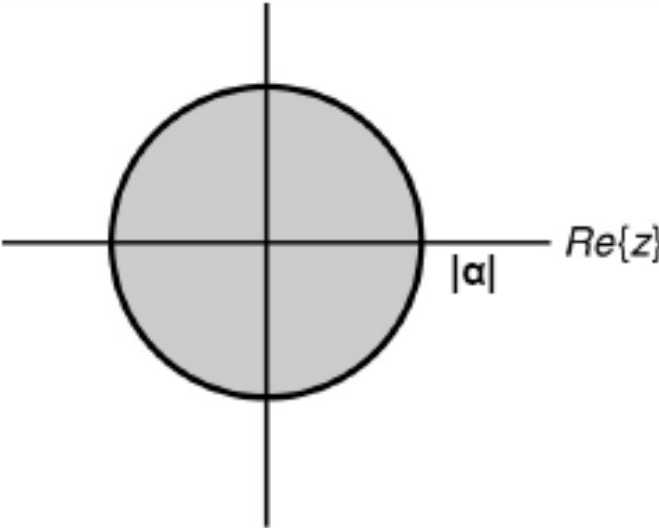
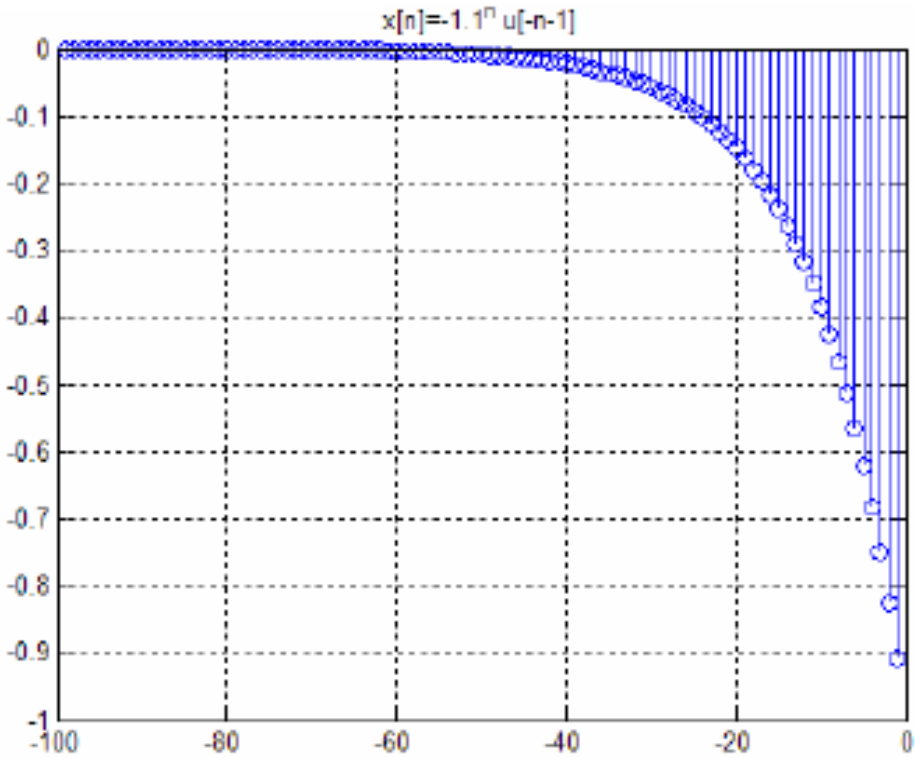
$$Y(z) = \sum_{n=-\infty}^{-1} -\alpha^n z^{-n} = -\sum_{m=1}^{\infty} \alpha^{-m} z^m = -\alpha^{-1} z \sum_{m=0}^{\infty} \alpha^{-m} z^m$$

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

$$= \frac{z}{z - \alpha}, \text{ for } |\alpha^{-1} z| < 1 \Rightarrow |z| < |\alpha|$$

ROC is the annular region $|z| < |\alpha|$

- The **z-transforms** of the two sequences $\alpha^n u[n]$ and $-\alpha^n u[-n-1]$ are identical even though the two parent sequences are different
- Only way a unique sequence can be associated with a **z-transform** is by specifying its **ROC**
- This is a left-sided sequence, which has an **ROC** that is inside of a circular area!



- If the two sequences $x[n]$ and $y[n]$ denote the impulse responses of a system (digital filter), then their **z -transforms** represent the **transfer functions** of these systems
- Both **transfer functions** have a **pole** at $z = \alpha$, which make the **transfer function** asymptotically approach to infinity at this value. Therefore, $z = \alpha$ is not included in either of the ROCs
- The circle with the radius of α is called the **pole circle**. A system may have many poles, and hence many pole circles
- For **right sided sequences**, the ROC extend **outside of the outermost pole circle**, whereas for **left sided sequences**, the ROC is the **inside of the innermost pole circle**
- For **two-sided sequences**, the ROC will be the **intersection of the two ROC areas corresponding to the left and right sides of the sequence**
- Since **DTFT** is the **z -transform** evaluated on the **unit circle**, that is for $z=e^{j\omega}$, **DTFT** of a sequence exists if and only if the ROC includes the unit circle!

- The **z-transforms** of LTI systems can be expressed as a ratio of two polynomials in z^{-1} , hence they are **rational transforms**
- Starting with the **Linear Constant Coefficient Difference Equation** representation of an LTI system:

$$\sum_{i=0}^N a_i y[n-i] = \sum_{j=0}^M b_j x[n-j], \quad a_0 = 1$$

$$y[n] + a_1 y[n-1] + a_2 y[n-2] + \cdots + a_N y[n-N] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M]$$



$$Y(z) + a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) + \cdots + a_N z^{-N} Y(z) = b_0 X(z) + b_1 z^{-1} X(z) + \cdots + b_M z^{-M} X(z)$$

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

MATLAB uses this representation for all digital filters / systems / transfer functions!!!

- A **rational z -transform** can be alternately written in factored form as

$$H(z) = \frac{b_0 \prod_{l=1}^M (1 - \xi_l z^{-1})}{a_0 \prod_{l=1}^N (1 - p_l z^{-1})} = z^{(N-M)} \frac{p_0 \prod_{l=1}^M (z - \xi_l)}{d_0 \prod_{l=1}^N (z - p_l)}$$

- At a root $z = \zeta_\ell$ of the **numerator polynomial** $H(\zeta_\ell) = 0$, and as a result, these values of z are known as the **zeros** of $H(z)$
- At a root $z = p_\ell$ of the **denominator polynomial** $H(p_\ell) \rightarrow \infty$, and as a result, these values of z are known as the **poles** of $H(z)$
 - Note that $H(z)$ has M finite **zeros** and N finite **poles**
 - If $N > M$ there are additional $N-M$ **zeros** at $z = 0$ (the origin in the z -plane)
 - If $N < M$ there are additional $M-N$ **poles** at $z = 0$
- Why is this important?
 - As we will see later, a digital filter is designed by placing appropriate number of **zeros** at the frequencies (z -values) to be suppressed, and **poles** at the frequencies to be amplified!

Recall that for a system to be **causal**, its impulse response must satisfy $h[n] = 0, n < 0$, that is for a **causal system**, the impulse response is **right sided**. Based on this, and our previous observations, we can make the following important conclusions:

- The **ROC of a causal system extends outside** of the outermost pole circle
- The **ROC of an anticausal system** (whose $h[n]$ is purely **left-sided**) **lies inside** of the innermost pole circle
- The **ROC of a noncausal system** (whose $h[n]$ **two-sided**) is **bounded** by two different pole circles
- For a system to be **stable**, its $h[n]$ must be absolutely summable → An LTI system is **stable**, if and only if the ROC of its transfer function $H(z)$ includes the unit circle!
- A **causal system's ROC** lies outside of a pole circle. If that system is also **stable**, its ROC must include unit circle → Then **a causal system is stable**, if and only if, **all poles are inside the unit circle**! Similarly, an **anticausal system is stable**, if and only if **its poles lie outside the unit circle**.
- An FIR filter is always **stable**, why?

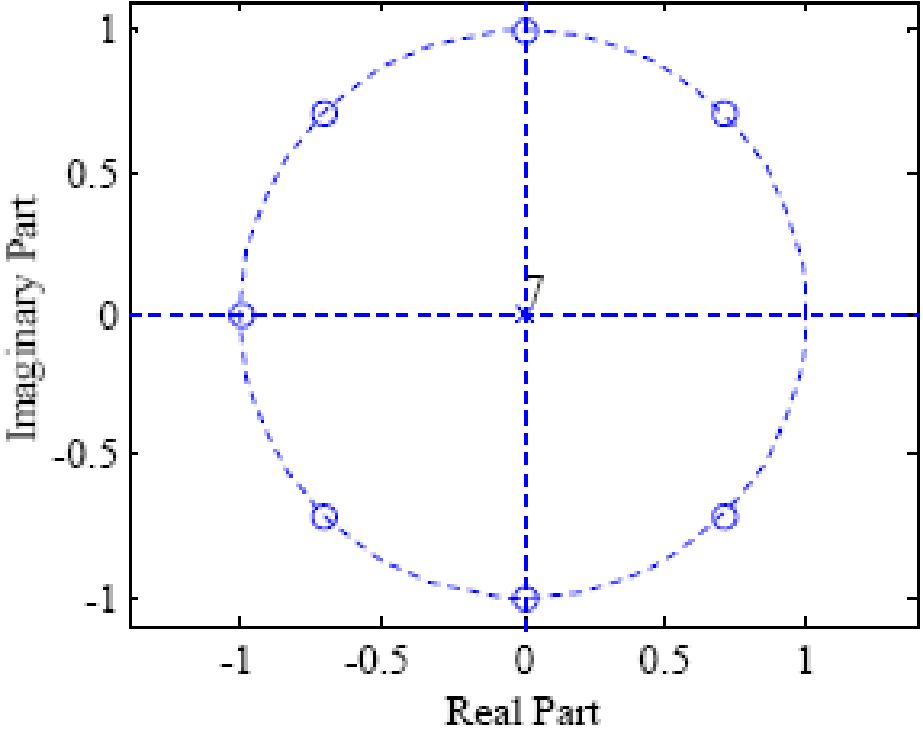
- Consider the M -point **moving-average FIR filter** with an impulse response

$$h[n] = \begin{cases} 1/M, & 0 \leq n \leq M-1 \\ 0, & \text{otherwise} \end{cases}$$

$$H(z) = \frac{1}{M} \sum_{n=0}^{M-1} z^{-n} = \frac{1 - z^{-M}}{M(1 - z^{-1})} = \frac{z^M - 1}{M[z^{M-1}(z - 1)]}$$

Observe the following:

- The transfer function has **M zeros on the unit circle** at $z = e^{j2\pi k/M}, 0 \leq k \leq M-1$
- There are **$M-1$ poles at $z = 0$** and a **single pole at $z = 1$**
- The pole at $z = 1$ exactly cancels the zero at $z = 1$
- The **ROC is the entire z -plane except $z = 0$**

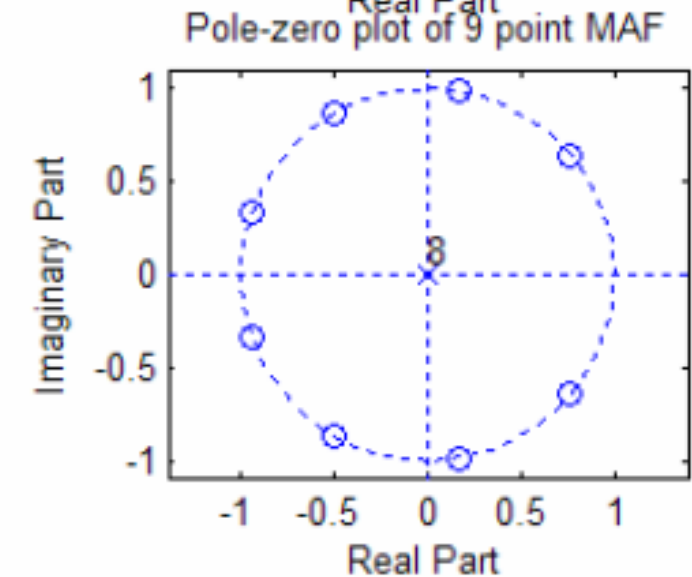
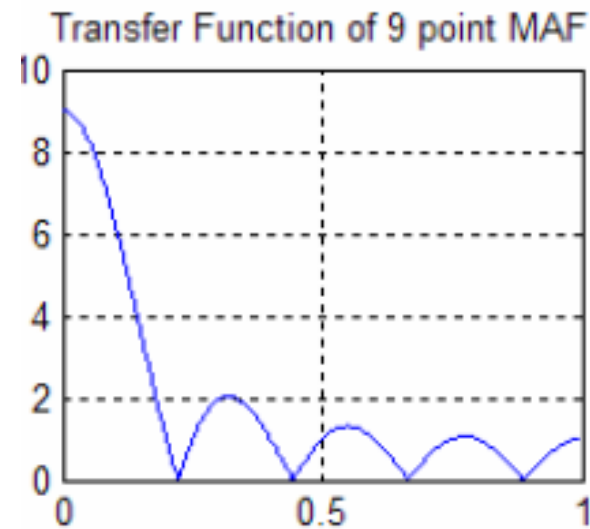
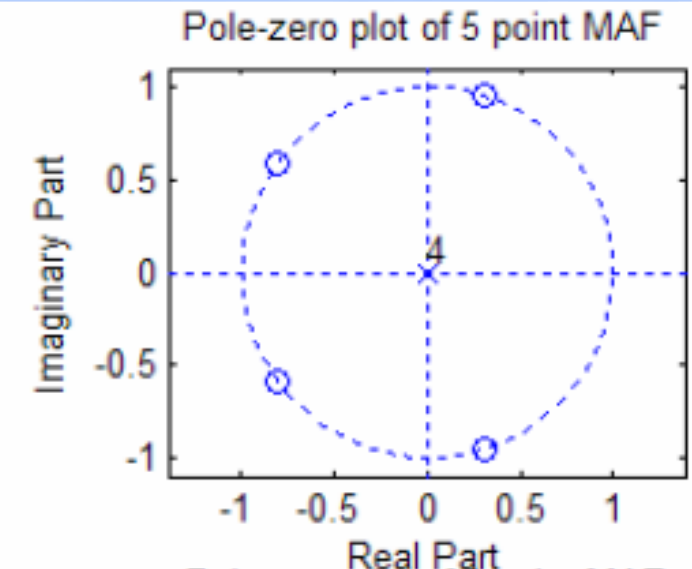
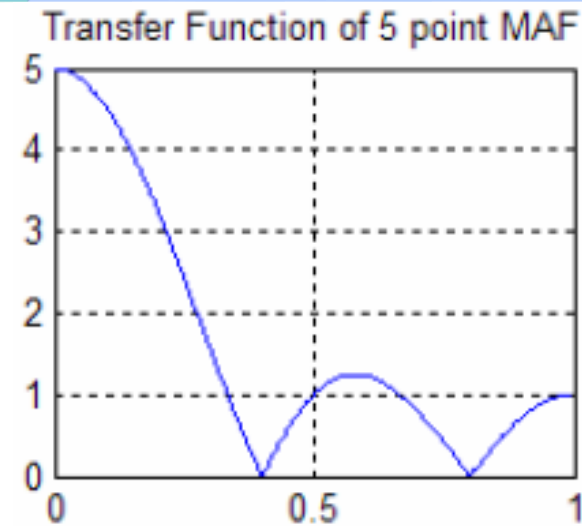


```
%h1=1/5* [1 1 1 1 1];
%h2=1/9 * [1 1 1 1 1 1 1 1 1];
```

```
b1=1/5*[1 1 1 1 1]; a1=1;
b2=1/9*[1 1 1 1 1 1 1 1 1]; a2=1;
```

```
[H1 w]=freqz(b1, 1, 512);
[H2 w]=freqz(b2, 1, 512);
```

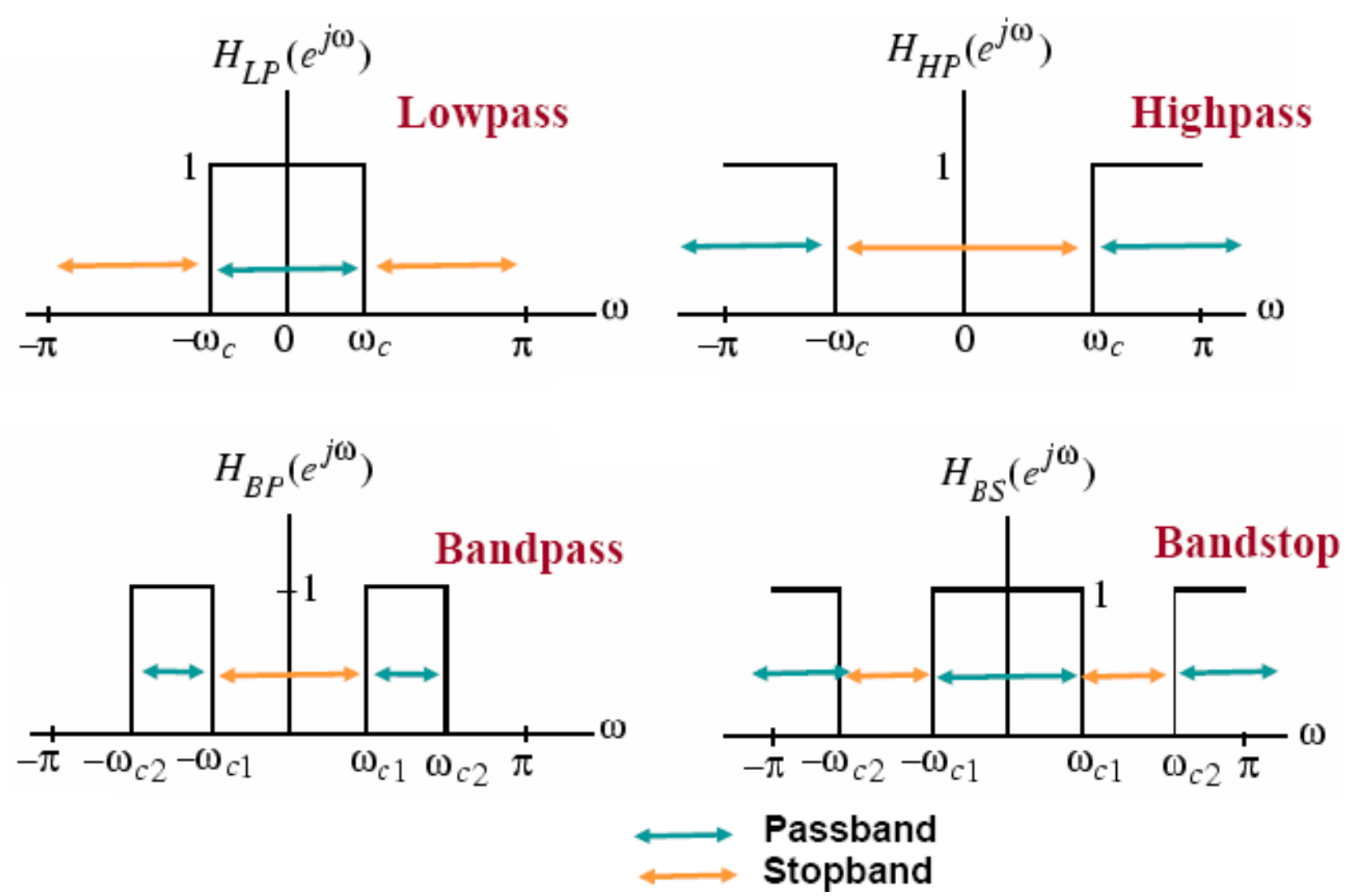
```
subplot(221)
plot(w/pi, abs(H1)); grid
title('Transfer Function of 5 point MAF')
subplot(222)
zplane(b1,a1);
title('Pole-zero plot of 5 point MAF')
subplot(223)
plot(w/pi, abs(H2)); grid
title('Transfer Function of 9 point MAF')
subplot(224)
zplane(b2,a2);
title('Pole-zero plot of 9 point MAF')
```



Observe the effects of zeros and the poles !!!

- An **ideal filter** is a digital filter designed to pass signal components of certain frequencies without distortion, which therefore has a frequency response equal to 1 at these frequencies, and has a frequency response equal to 0 at all other frequencies
- The range of frequencies where the frequency response takes the value of **one** is called the **passband**
- The range of frequencies where the frequency response takes the value of **zero** is called the **stopband**
- The **transition frequency** from a passband to stopband region is called the **cutoff frequency**
- Note that an **ideal filter cannot be realized**. Why?

- The frequency responses of four common ideal filters in the $[-\pi \pi]$ range are



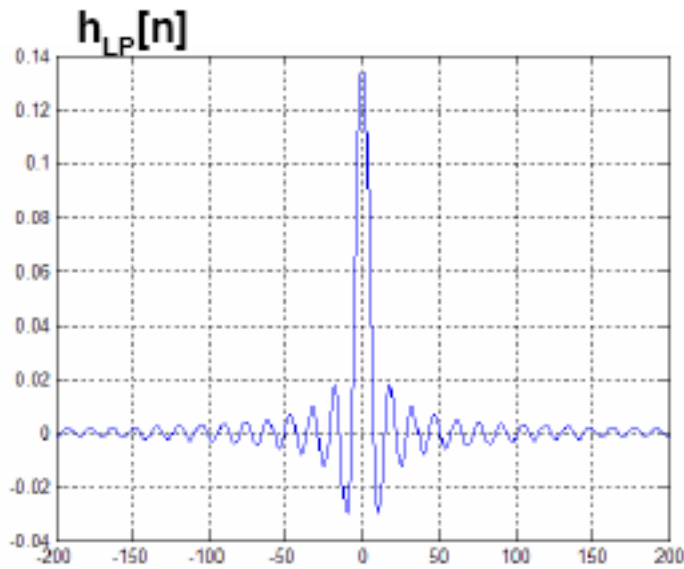
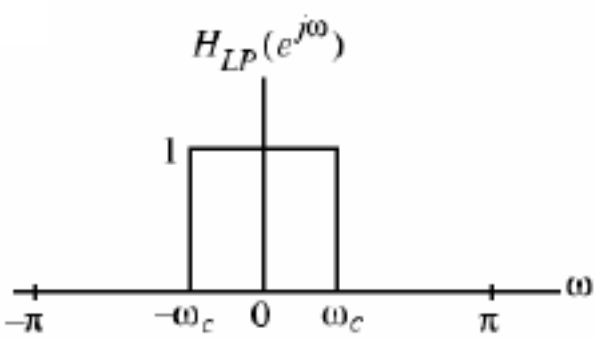
- Recall that the DTFT of a rectangular pulse is a **sinc function**

$$x[n] = \text{rect}_M[n] = \begin{cases} 1, & -M \leq n \leq M \\ 0, & \text{otherwise} \end{cases} \quad \Leftrightarrow \quad \sum_{n=-M}^M e^{-j\omega n} = \frac{\sin(M + 1/2)\omega}{\sin(\omega/2)}, \quad \omega \neq 0$$

- From the *duality theorem*, the inverse DTFT of a **rectangular pulse** is also a **sinc function**. Since the **ideal (lowpass) filter** is of **rectangular shape**, its **impulse response** must be of **sinc**

$$H_{LP}(\omega) = \begin{cases} 1, & 0 \leq |\omega| \leq \omega_c \\ 0, & \omega_c \leq |\omega| \leq \pi \end{cases}$$

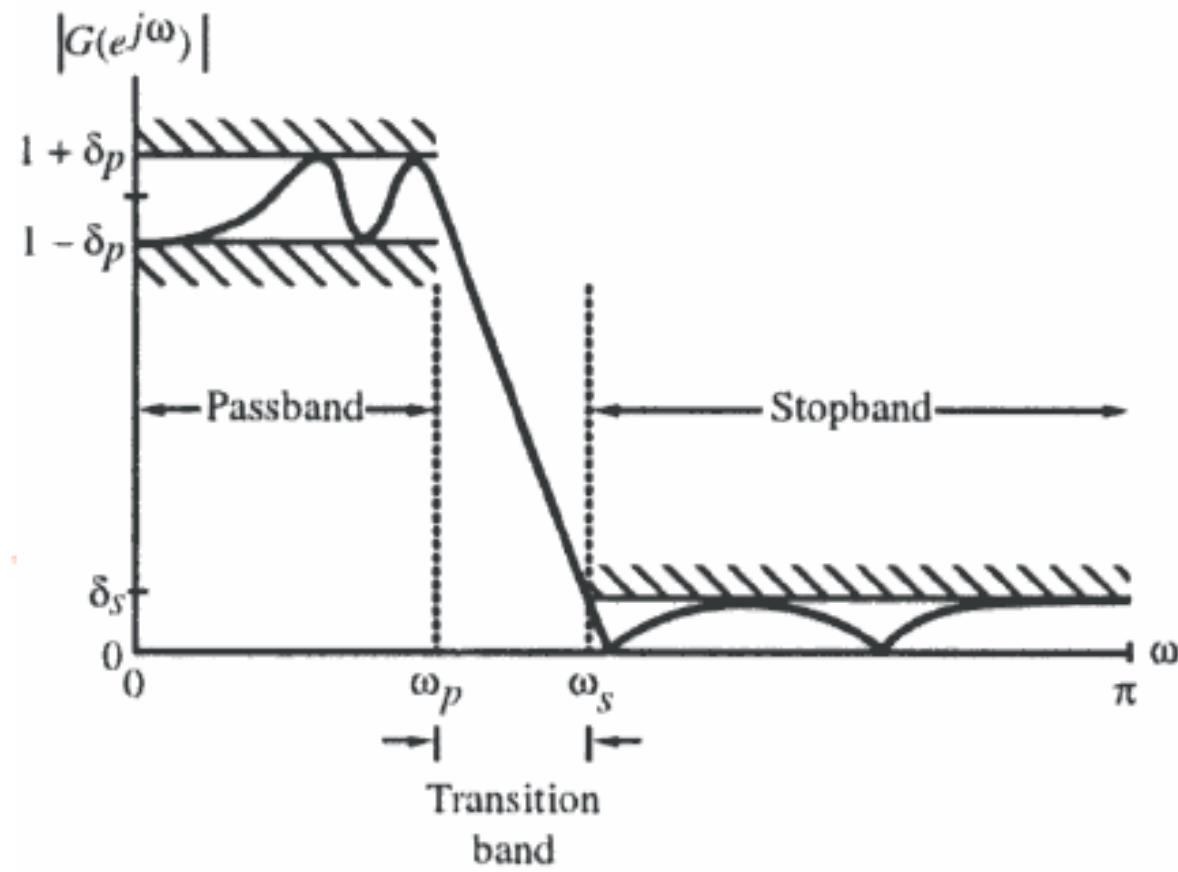
$$h_{LP}[n] = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} 1 \cdot e^{j\omega n} d\omega$$
$$= \frac{\sin(\omega_c n)}{\pi n}, \quad -\infty < n < \infty$$



- We note the following about the **impulse response of an ideal filter**
 - $h_{LP}[n]$ is **not absolutely summable**
 - The corresponding transfer function is therefore **not BIBO stable**
 - $h_{LP}[n]$ is **not causal**, and is of **doubly infinite length**
 - The remaining three ideal filters are also characterized by **doubly infinite, noncausal impulse responses** and also are **not absolutely summable**
- Thus, the **ideal filters** with the ideal **brick wall** frequency responses **cannot be realized with finite dimensional LTI filter**

In order to develop a **stable** and **realizable** filter transfer function

- The ideal frequency response specifications are relaxed by including a **transition band** between the passband and the stopband
- This permits the magnitude response to **decay slowly** from its maximum value in the passband to the zero value in the stopband
- Moreover, the magnitude response is allowed to **vary by a small amount** both in the passband and the stopband
- Typical magnitude response specifications of a lowpass filter therefore looks like →



- So far we have seen transfer functions characterized primarily according to their
 - **Impulse response length** (FIR / IIR)
 - **Magnitude spectrum characteristics** (LPF, HPF, BPF, BSF)
- A third classification of a transfer function is with respect to its phase characteristics
 - **Zero phase**
 - **Linear phase**
 - **Generalized linear phase**
 - **Non-linear phase**
- Recall that the phase spectrum tells us _____
- In many applications, it is necessary that the digital filter designed does not distort the phase of the input signal components with frequencies in the passband

- A frequency selective system (filter) with frequency response

$$H(\omega) = |H(\omega)| \cdot \angle H(\omega) = |H(\omega)| \cdot e^{j\theta(\omega)}$$

changes the **amplitude** of all frequencies in the signal by a factor of $|H(\omega)|$, and adds a **phase** of $\theta(\omega)$ to all frequencies

- Note that both the **amplitude change** and the **phase delay** are functions of ω
- The phase $\theta(\omega)$ is in terms of radians, but can be expressed in terms of time, which is called the **phase delay**. The **phase delay** at a particular frequency ω_0 is given as

$$\tau_p(\omega_o) = -\frac{\theta(\omega_o)}{\omega_o}$$

Compare this to the phase delay in cont. time

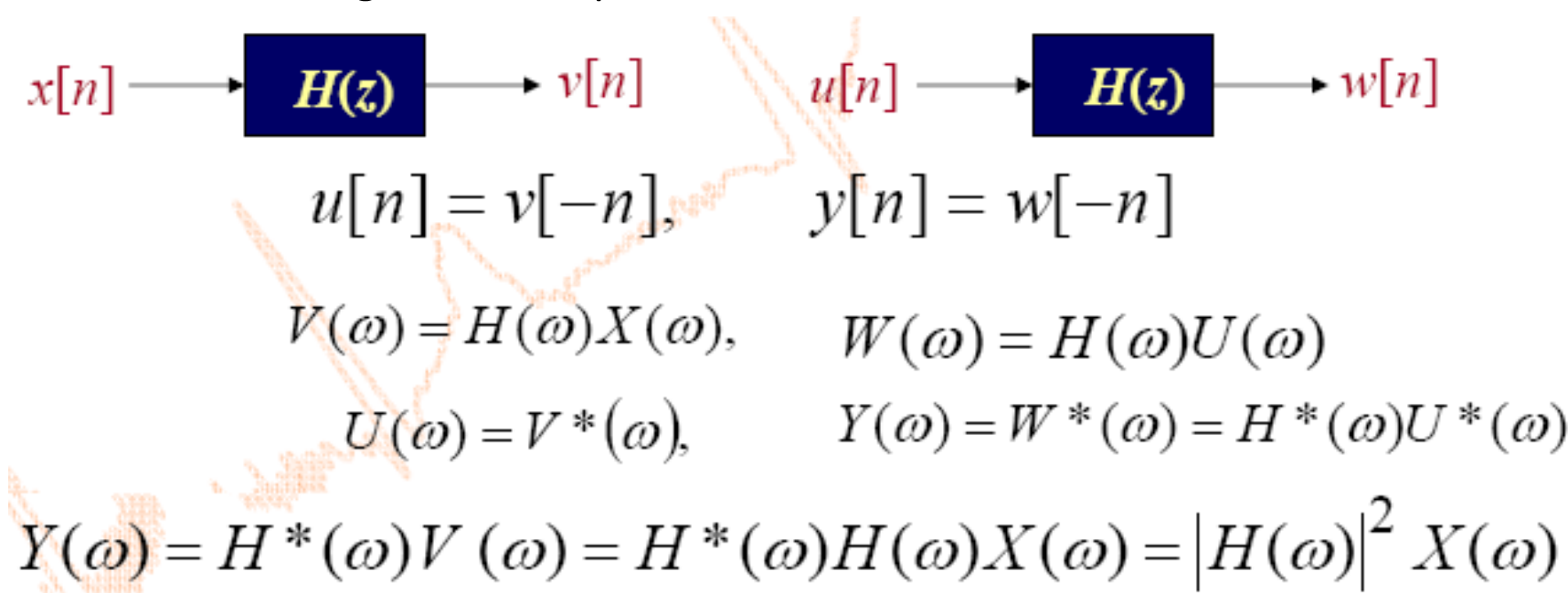
$$t_\theta = \frac{\theta}{2\pi f} = \frac{\theta}{\omega}$$

- If an input system consists of many frequency components (which most practical signals do), then we can also define **group delay**, the **phase shift** by which the envelope of the signal shifts. This can also be considered as the **average phase delay** – in seconds – of the filter as a function of frequency, given by

$$\tau_g(\omega_c) = - \left. \frac{d\theta(\omega)}{d\omega} \right|_{\omega=\omega_c}$$


- One way to avoid any **phase distortion** is to make sure the frequency response of the filter does not delay any of the spectral components. Such a transfer function is said to have a **zero – phase** characteristic
- A **zero – phase transfer function** has no phase component, that is, the spectrum is purely real (no imaginary component) and non-negative
- However, it is **NOT possible** to design a **causal digital filter with a zero phase**.
Why?
- **Hint:** What do we need in the impulse response to ensure that the frequency response is real and non-negative?

- Now, for **non-real-time processing** of **real-valued input signals of finite length**, **zero-phase filtering** can be implemented by **relaxing the causality requirement**
- A **zero-phase filtering** scheme can be obtained by the following procedure:
 - Process the input data (finite length) with a **causal real-coefficient filter** $H(z)$.
 - Time reverse the output of this filter and process by the same filter.
 - Time reverse once again the output of the second filter



- Note that a **zero-phase filter** cannot be implemented for real-time applications. Why?
- For a **causal transfer function** with a **nonzero phase response**, the phase distortion can be avoided by ensuring that the transfer function has (preferably) a unity magnitude and a **linear-phase characteristic** in the frequency band of interest

$$H(\omega) = e^{-j\alpha\omega}$$



$$|H(\omega)| = 1 \quad \angle H(\omega) = \theta(\omega) = -\alpha\omega$$

- Note that this phase characteristic is linear for all ω in $[0 \ 2\pi]$.
- Recall that the **phase delay** at any given frequency ω_0 was
- If we have **linear phase**, that is, $\theta(\omega) = -\alpha\omega$, then the total delay at any frequency ω_0 is $\tau_0 = -\theta(\omega_0)/\omega_0 = -\alpha\omega_0/\omega_0 = \alpha$
- Note that this is identical to the **group delay** $d\theta(\omega)/d\omega$ evaluated at ω_0

$$\tau_p(\omega_o) = -\frac{\theta(\omega_o)}{\omega_o}$$

$$\tau_g(\omega_0) = -\left.\frac{d\theta(\omega)}{d\omega}\right|_{\omega=\omega_0}$$

- If the phase spectrum is **linear**, then the phase delay is independent of the frequency, and it is the same constant α for all frequencies
- In other words, all frequencies are delayed by α seconds, or equivalently, the entire signal is delayed by α seconds
 - Since the entire signal is delayed by a constant amount, there is no distortion!
- If the filter does not have linear phase, then different frequency components are delayed by different amounts, causing significant distortion

- It is **typically impossible to design a linear phase IIR filter**, however, designing FIR filters with precise linear phase is very easy:

$$H(z) = \sum_{n=0}^N h[n] z^{-n} = h[0] + h[1]z^{-1} + h[2]z^{-2} + \dots + h[M]z^{-M}$$

- Consider a causal FIR filter of length $M+1$ (order M)
- This transfer function has linear phase, if its impulse response $h[n]$ is either

symmetric:

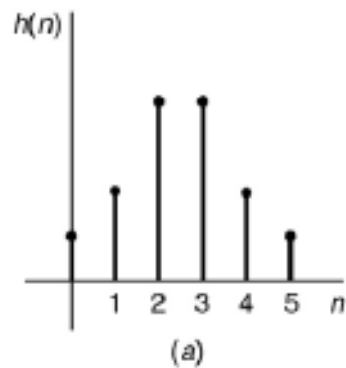
$$h[n] = h[M - n], \quad 0 \leq n \leq M$$

or **anti-symmetric:**

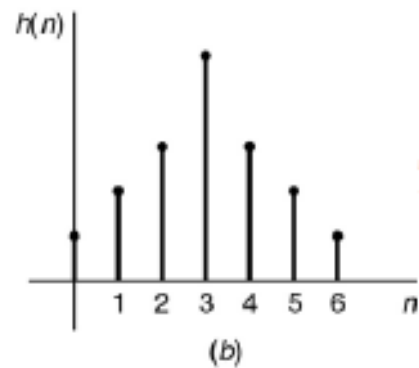
$$h[n] = -h[M - n], \quad 0 \leq n \leq M$$

- There are four possible scenarios: filter length even or odd, and impulse response is either symmetric or antisymmetric

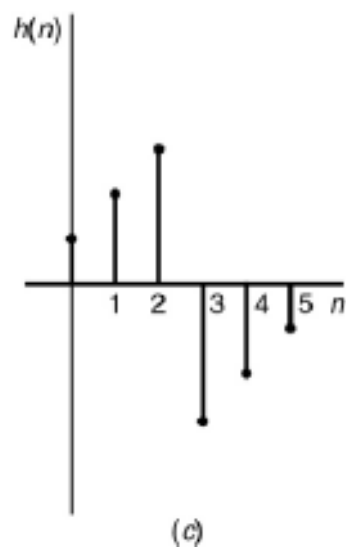
FIR II: even length, symmetric



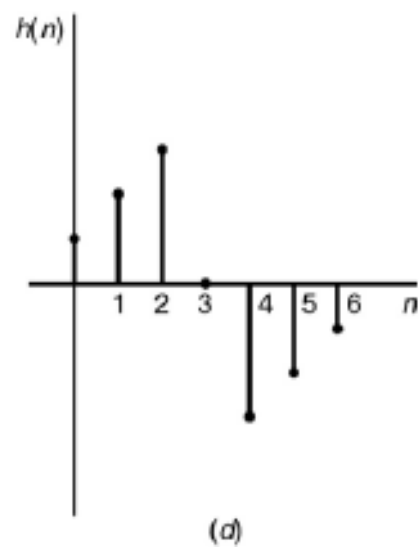
FIR I: odd length, symmetric



FIR IV: even length, antisymmetric



FIR III: odd length, antisymmetric



Note for this case that $h[M/2]=0$

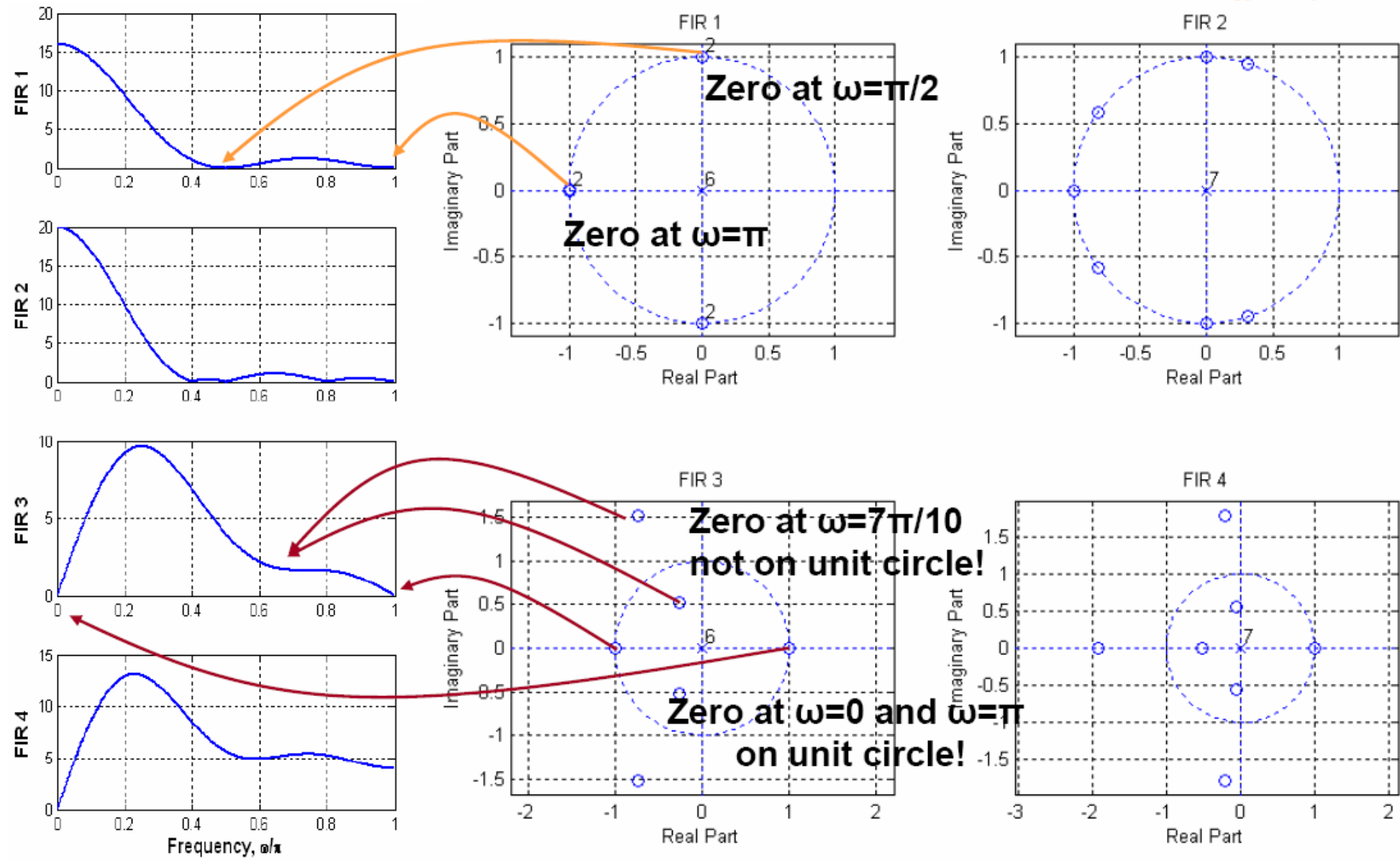
```
h1=[1 2 3 4 3 2 1]; % FIR 1
h2=[1 2 3 4 4 3 2 1]; % FIR 2
h3=[-1 -2 -3 0 3 2 1]; %FIR 3
h4=[-1 -2 -3 -4 4 3 2 1] ; % FIR 4

[H1 w]=freqz(h1, 1, 512); [H2 w]=freqz(h2, 1, 512);
[H3 w]=freqz(h3, 1, 512); [H4 w]=freqz(h4, 1, 512);

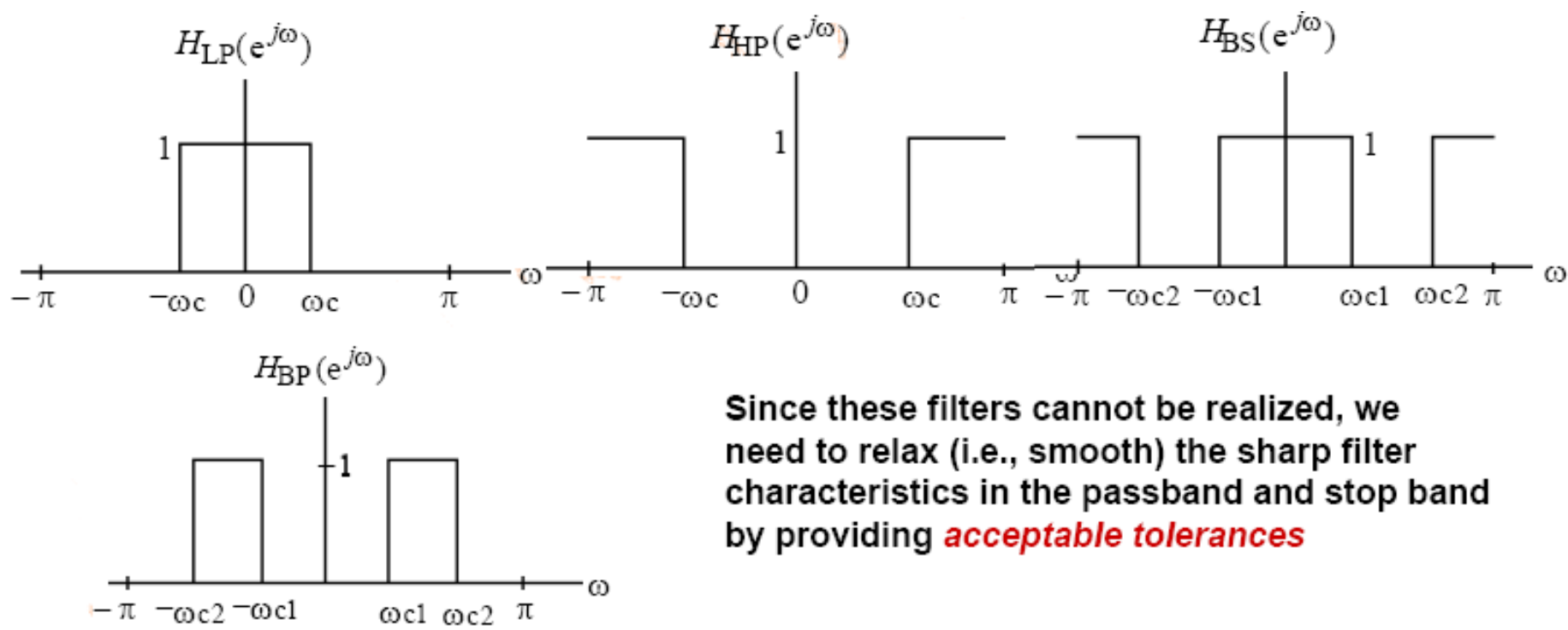
% Plot the magnitude and phase responses
% in angular frequency from 0 to pi
subplot(421); plot(w/pi, abs(H1));grid; ylabel('FIR 1')
subplot(422); plot(w/pi, unwrap(angle(H1)));grid;
subplot(423); plot(w/pi, abs(H2));grid; ylabel('FIR 2')
subplot(424); plot(w/pi, unwrap(angle(H2)));grid;
subplot(425); plot(w/pi, abs(H3));grid; ylabel('FIR 3')
subplot(426); plot(w/pi, unwrap(angle(H3)));grid;
subplot(427); plot(w/pi, abs(H4));grid
xlabel('Frequency, \omega/\pi'); ylabel('FIR 4')
subplot(428); plot(w/pi, unwrap(angle(H4)));grid
xlabel('Frequency, \omega/\pi')
```

```
%Plot the zero - pole plots
figure
subplot(221)
zplane(h1,1); grid
title('FIR 1')
subplot(222)
zplane(h2,1);grid
title('FIR 2')
subplot(223)
zplane(h3,1);grid
title('FIR 3')
subplot(224)
zplane(h4,1);grid
title('FIR 4')

lin_phase_demo2.m
```



- **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**

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

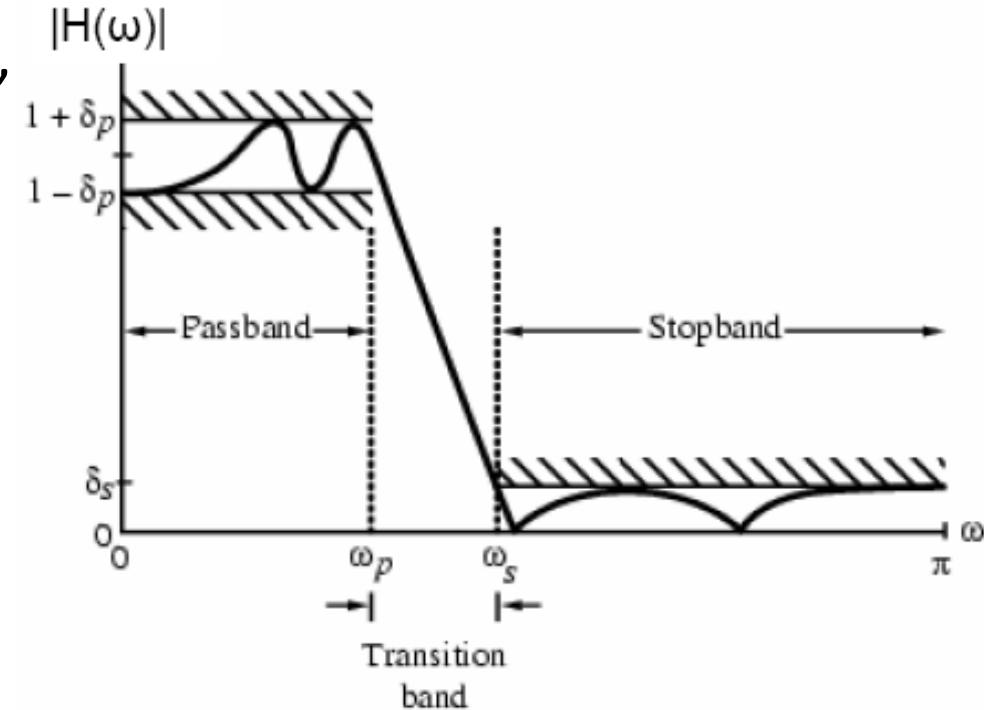
$$1 - \delta_p \leq |H(\omega)| \leq 1 + \delta_p, \quad |\omega| \leq \omega_p$$

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

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

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

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

with **peak pass** and **minimum stopband ripple**

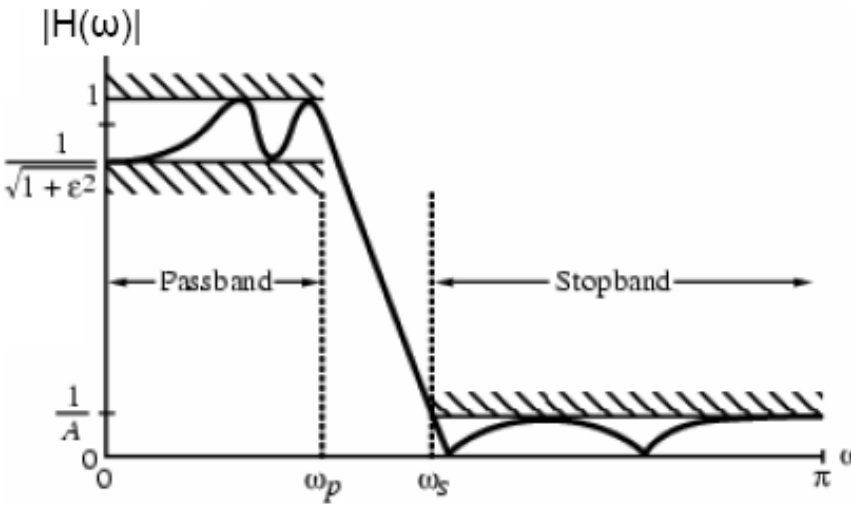
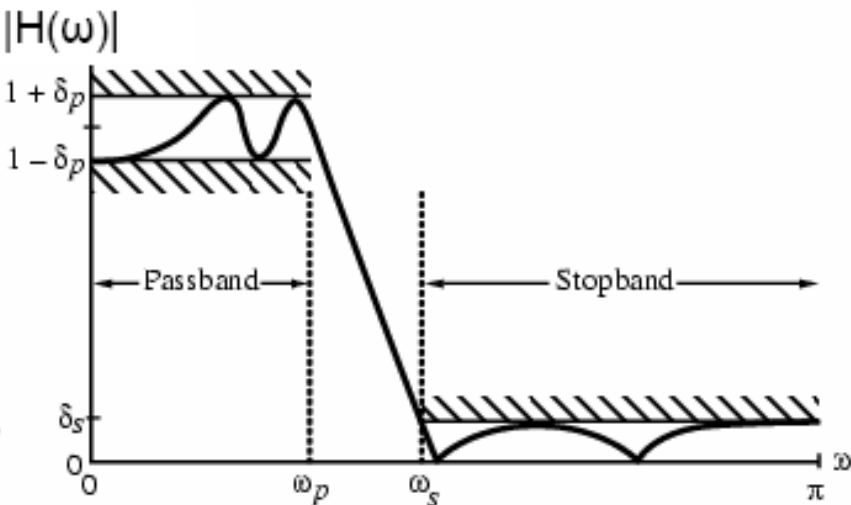
$$\alpha_p = -20 \log_{10}(1 - \delta_p)$$

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

- 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**

$$\frac{1}{A}$$

$$1/\sqrt{1 + \epsilon^2}$$



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 the unit circle, ROC includes unit circle, $h[n]$ right-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**: $h[n] = \pm h[M-n]$
 - 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

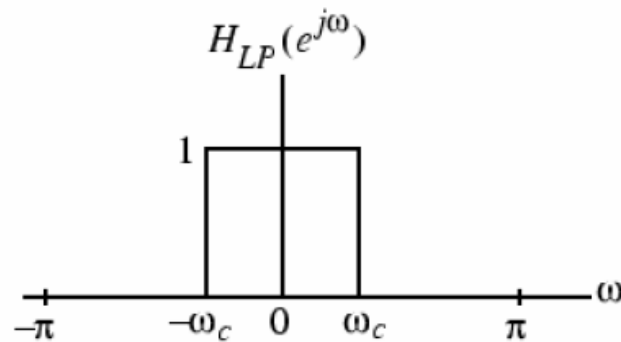
- Several advantages to both FIR and IIR type filters
- **Advantages of FIR filters** (**disadvantages of IIR filters**):
 - Can be designed with **exact linear phase**
 - Filter structure **always stable** with quantized coefficients
 - The filter startup **transients have finite duration**
- **Disadvantages of FIR filters** (**advantages 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...?**

- **IIR filter design:**
 1. **Convert** the **digital filter specifications** into an **analog prototype lowpass filter specifications**
 2. **Determine** the **analog lowpass filter transfer function** $|H(\Omega)|$
 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 $H(\omega)$

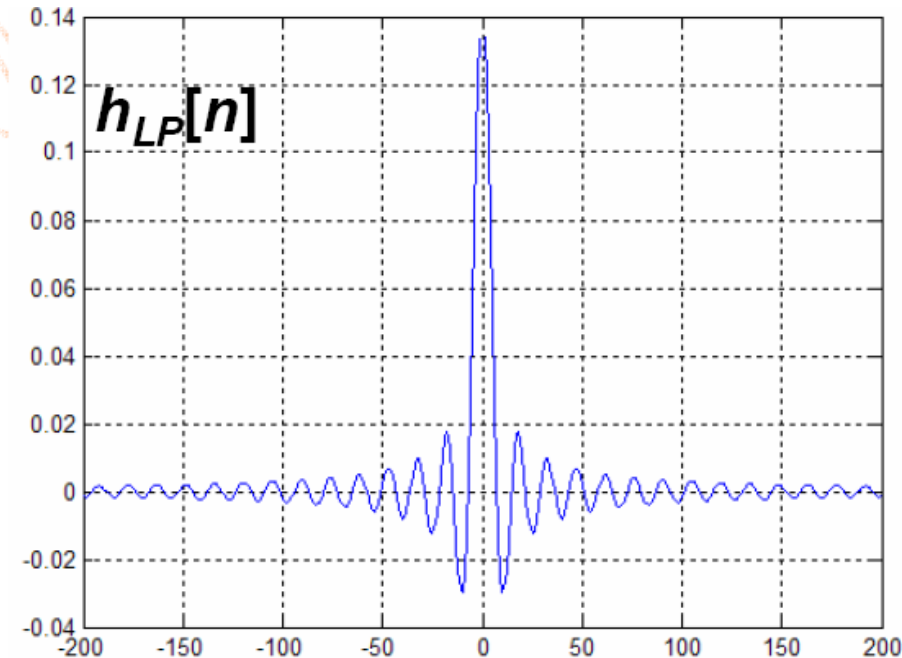
- Let's start with the **ideal lowpass filter**
- We know that there are **two problems** with this filter: **infinitely long**, and it is **non-causal**

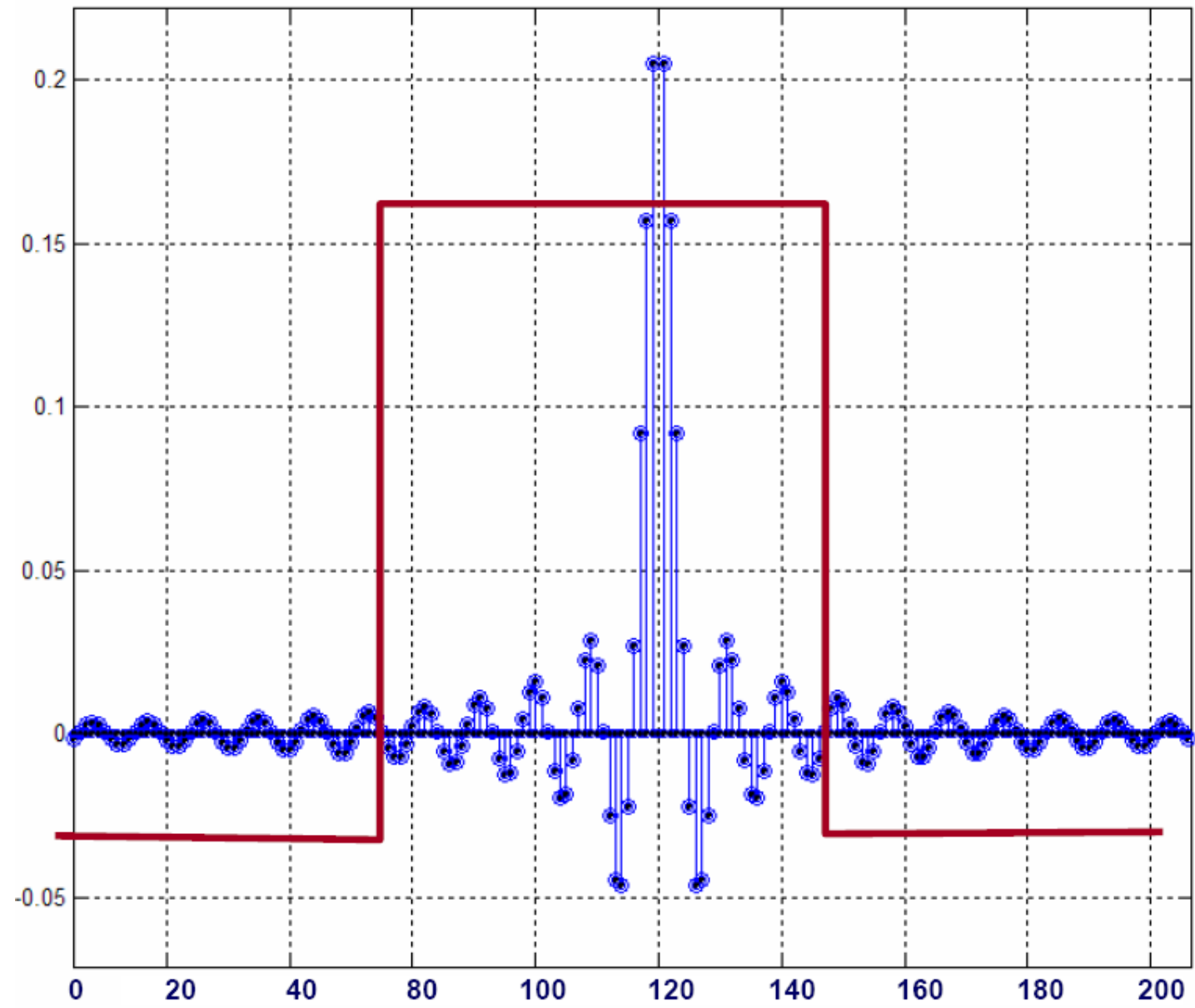
$$H_{LP}(\omega) = \begin{cases} 1, & 0 \leq |\omega| \leq \omega_c \\ 0, & \omega_c \leq |\omega| \leq \pi \end{cases} \longleftrightarrow h_{LP}[n] = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} 1 \cdot e^{j\omega n} d\omega$$

$$= \frac{\sin(\omega_c n)}{\pi n}, \quad -\infty < n < \infty$$



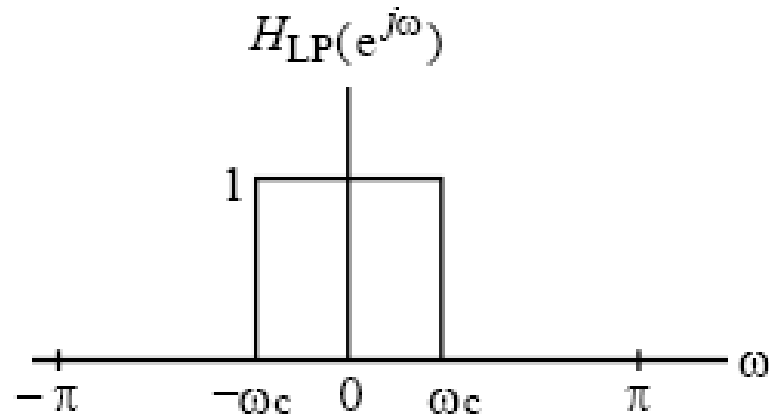
How can we overcome these two problems?



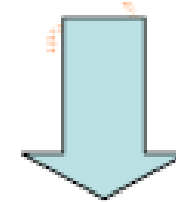


This is the basic, straightforward approach to **FIR filter design**:

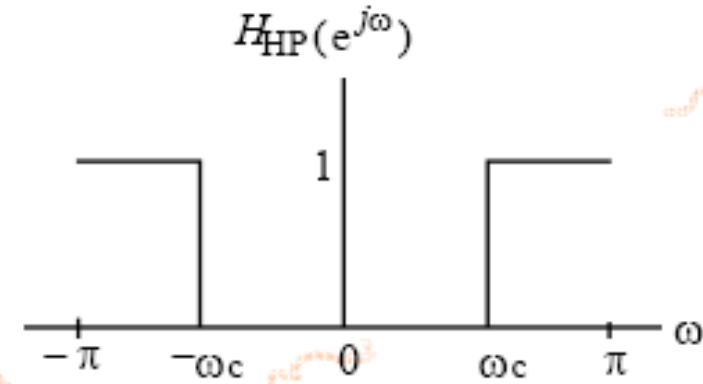
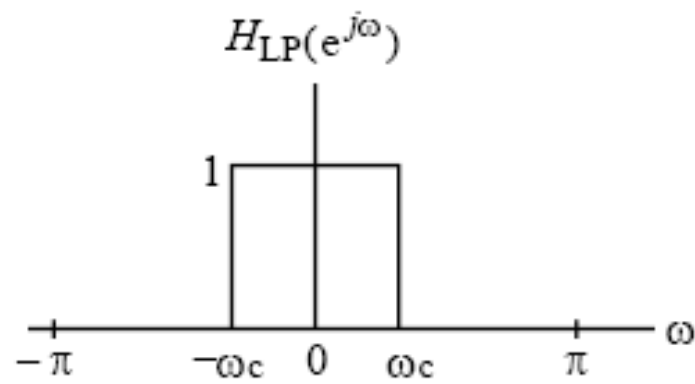
- Start with an ideal filter that meets the design criteria, say a filter $H(\omega)$
- Take the inverse DTFT of this $H(\omega)$ to obtain $h[n]$
 - This $h[n]$ will be double infinitely long, and non-causal → unrealizable
- Truncate using a window, say a rectangle, so that $M+1$ coefficients of $h[n]$ are retained, and all the others are discarded
 - We now have a finite length (order M) filter, $h_t[n]$, however, it is still non-causal
- Shift the truncated $h[n]$ to the right (i.e., delay) by $M/2$ samples, so that the first sample now occurs at $n=0$
 - The resulting impulse response, $h_t[n-M/2]$ is a causal, stable, **FIR filter**, which has an almost identical magnitude response and a phase factor of $e^{-jM/2}$ compared to the original filter, due to delay introduced



$$h_{LP}[n] = \frac{\sin(\omega_c n)}{\pi n}, \quad -\infty < n < \infty$$

Unrealizable!**Realizable!**

$$h_{LP}[n] = \begin{cases} \frac{\sin(\omega_c (n - M/2))}{\pi (n - M/2)}, & 0 \leq n \leq M, \quad n \neq \frac{M}{2} \\ \frac{\omega_c}{\pi}, & n = \frac{M}{2} \end{cases}$$



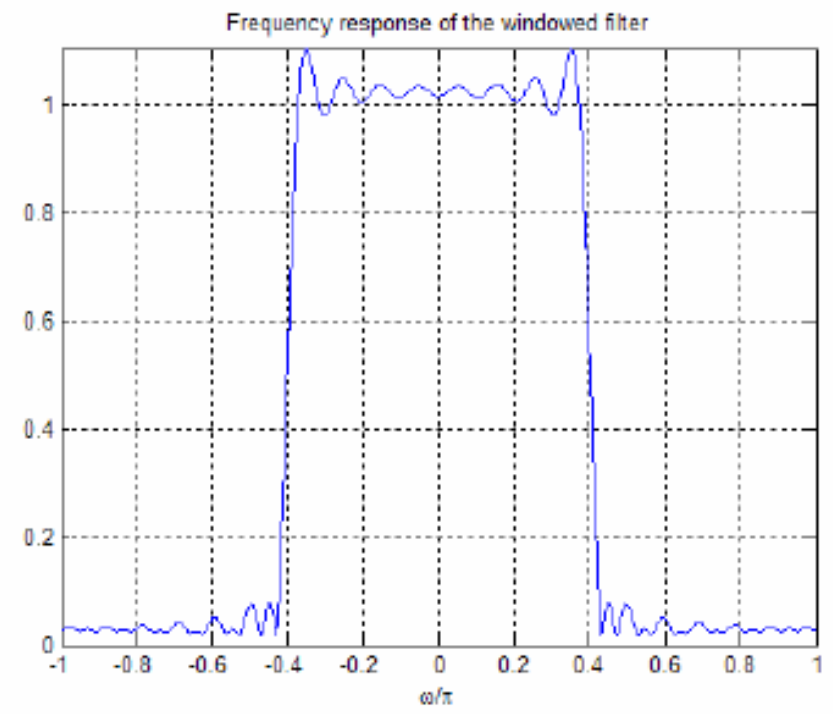
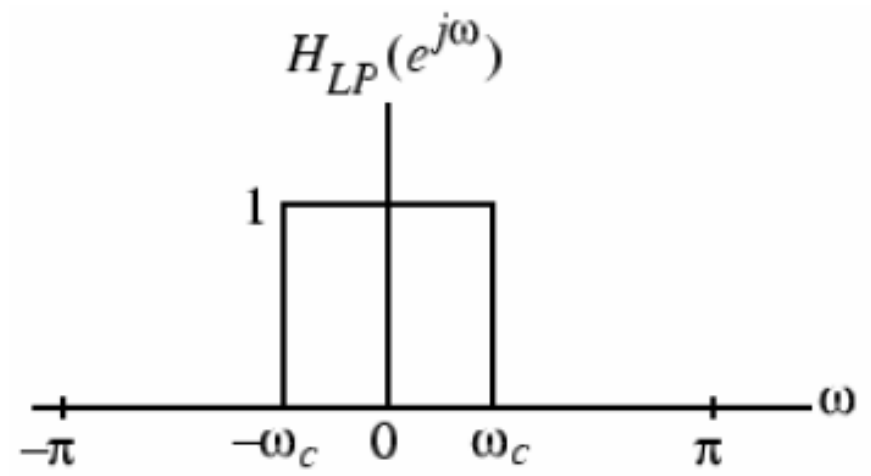
$$H_{HP}(\omega) = 1 - H_{LP}(\omega)$$



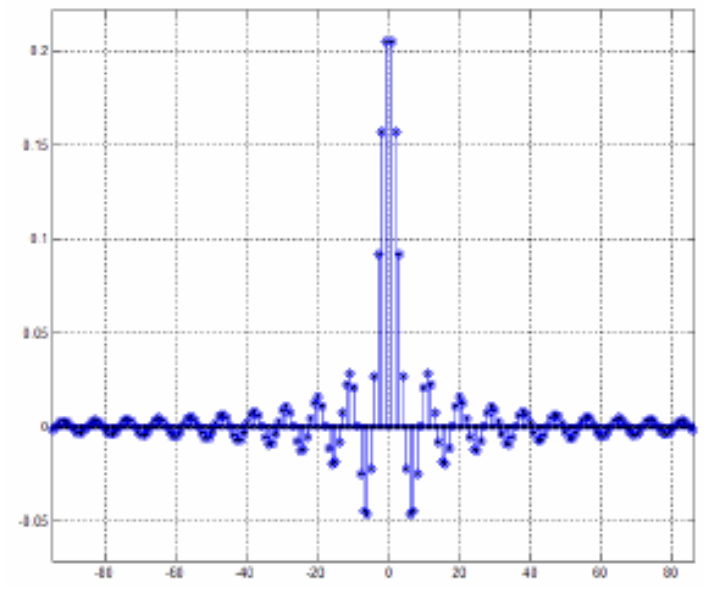
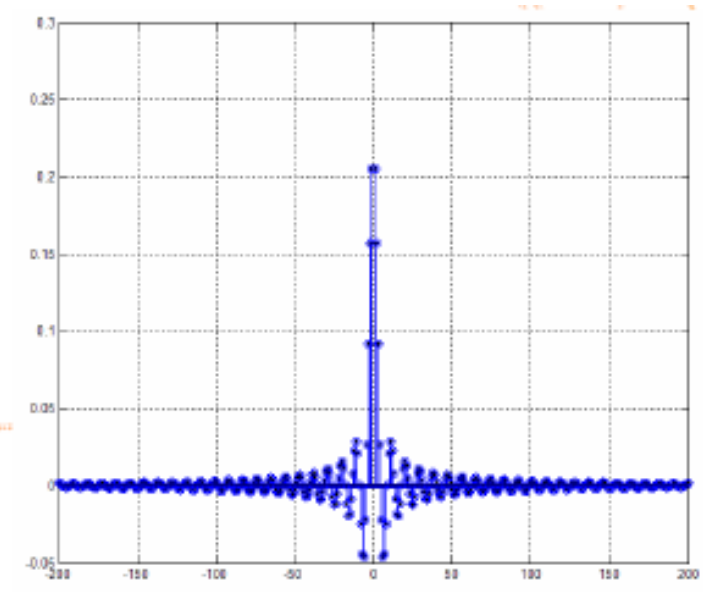
$$h_{HP}[n] = \delta[n] - h_{LP}[n]$$

$$h_{LP}[n] = \begin{cases} \frac{\sin(\omega_c(n - M/2))}{\pi(n - M/2)}, & 0 < n < M, \quad n \neq \frac{M}{2} \\ \frac{\omega_c}{\pi}, & n = \frac{M}{2} \end{cases}$$

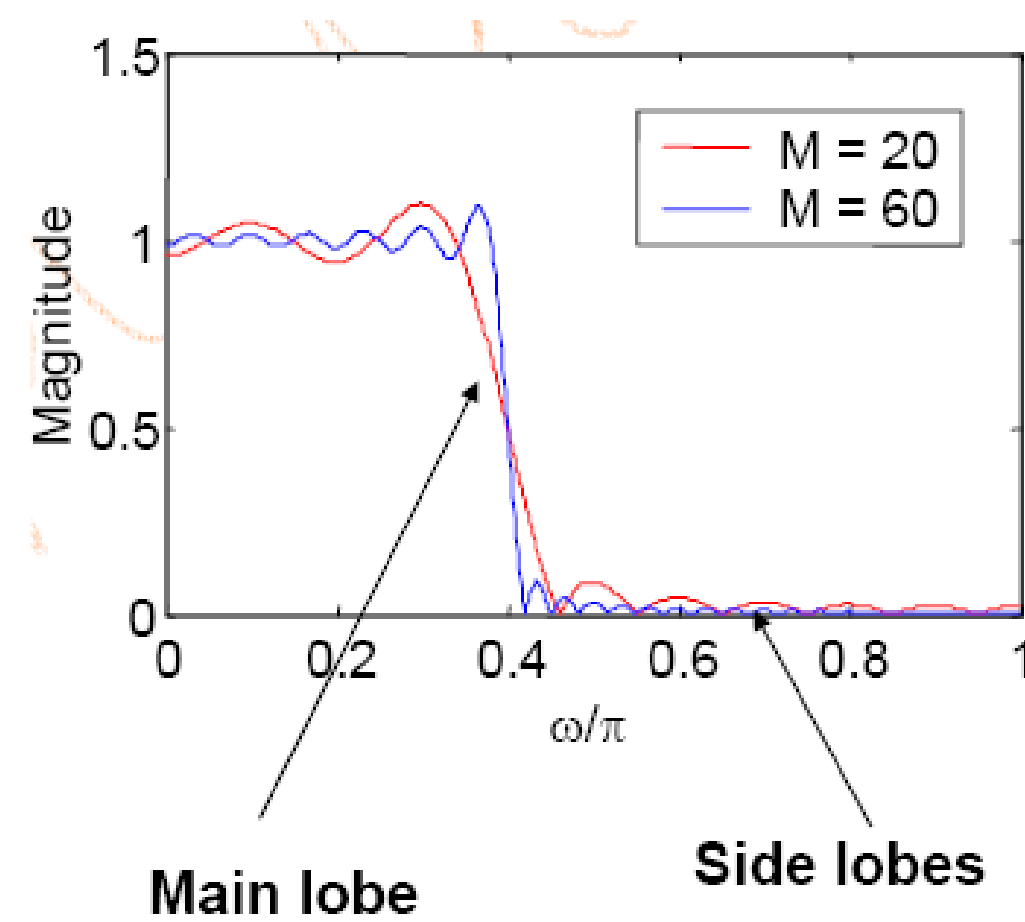
$$h_{HP}[n] = \begin{cases} -\frac{\sin(\omega_c(n - M/2))}{\pi(n - M/2)}, & 0 < n < M, \quad n \neq \frac{M}{2} \\ 1 - \frac{\omega_c}{\pi}, & n = \frac{M}{2} \end{cases}$$

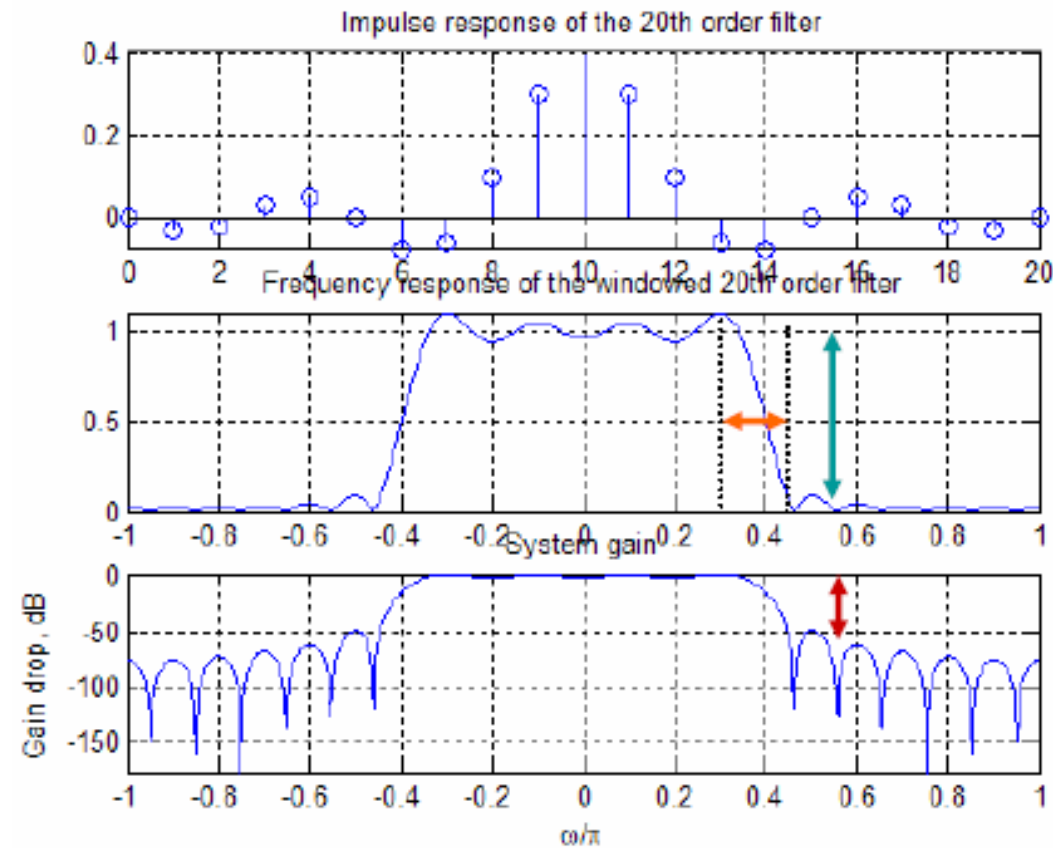


What happened...?



- Truncating the impulse response of an **ideal filter** to obtain a **realizable filter**, creates oscillatory behavior in the frequency domain
 - The **Gibbs Phenomenon**
- We observe the following:
 - As $M \uparrow$, the number of ripples \uparrow however, ripple widths \downarrow
 - The height of the largest ripples remain constant, regardless of the filter length
 - As $M \uparrow$, the height of all other ripples \downarrow
 - The main lobe gets narrower as $M \uparrow$, that is, the drop-off becomes sharper
 - Similar oscillatory behavior can be seen in all types of truncated filters

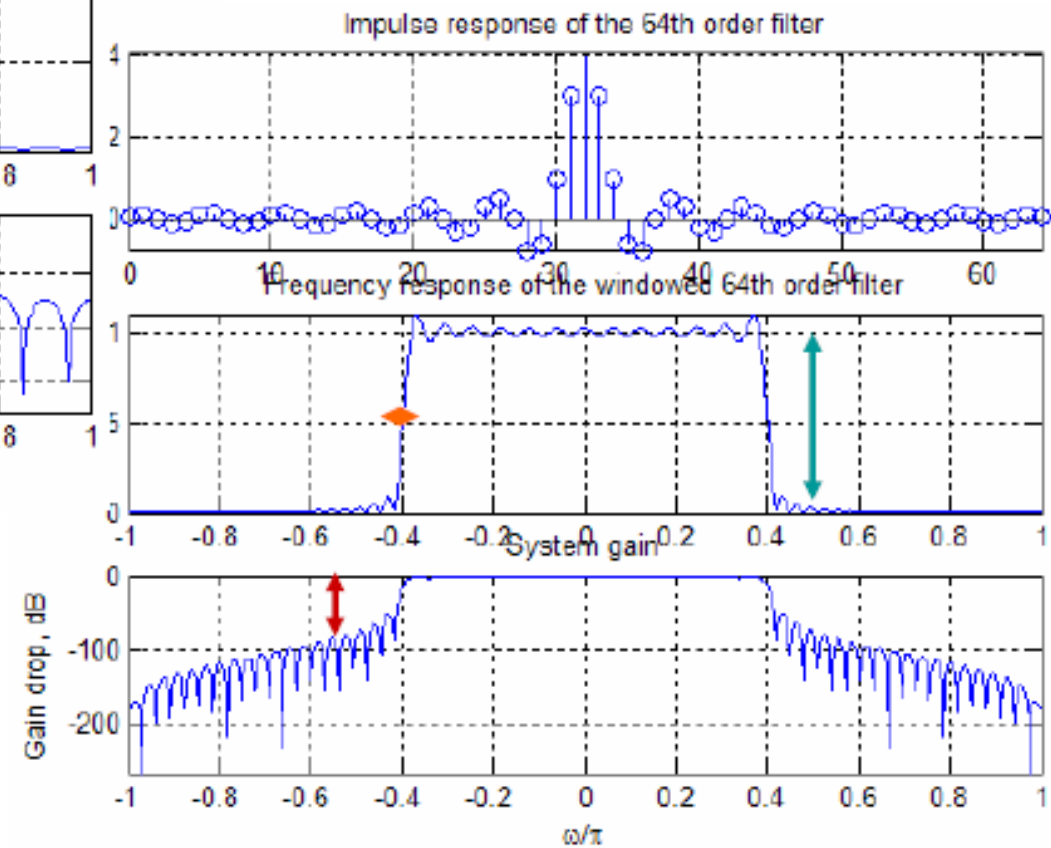




Transition band / main lobe width

Stopband attenuation

In dB



Here's what we want:

- Quick drop off → Narrow transition band
 - **Narrow main lobe**
 - **Increased stopband attenuation**
- Reduce the height of the side-lobe which causes the ripples
- Reduce Gibb's phenomenon (ringing effects, all ripples)
- Minimize the order of the filter
- Gibb's phenomenon can be reduced (but not eliminated) by using a **smoother window** that gently tapers off zero, rather than the brick wall behavior of the rectangular filter
 - Several window functions are available, which usually trade-off main-lobe width and stopband attenuation
 - **Rectangular window has the narrowest main-lobe width, but poor side-lobe attenuation**
 - **Tapered window causes the height of the sidelobes to diminish, with a corresponding increase in the main lobe width resulting in a wider transition at the cutoff frequency**

1. Rectangular

$$w(n) = \begin{cases} 1, & 0 \leq n \leq M \\ 0, & \text{otherwise} \end{cases}$$

2. Bartlett

$$w(n) = 1 - \frac{2 \left| n - \frac{M}{2} \right|}{M}$$

3. Blackman

$$w(n) = 0.42 - 0.5 \cos \frac{2\pi n}{M} + 0.08 \cos \frac{4\pi n}{M}$$

$0 < n < M$
(length $M+1$)

In your text:

$-M < n < M$

(length $2M+1$)

4. Hamming

$$w(n) = 0.54 - 0.46 \cos \frac{2\pi n}{M}$$

5. Hanning

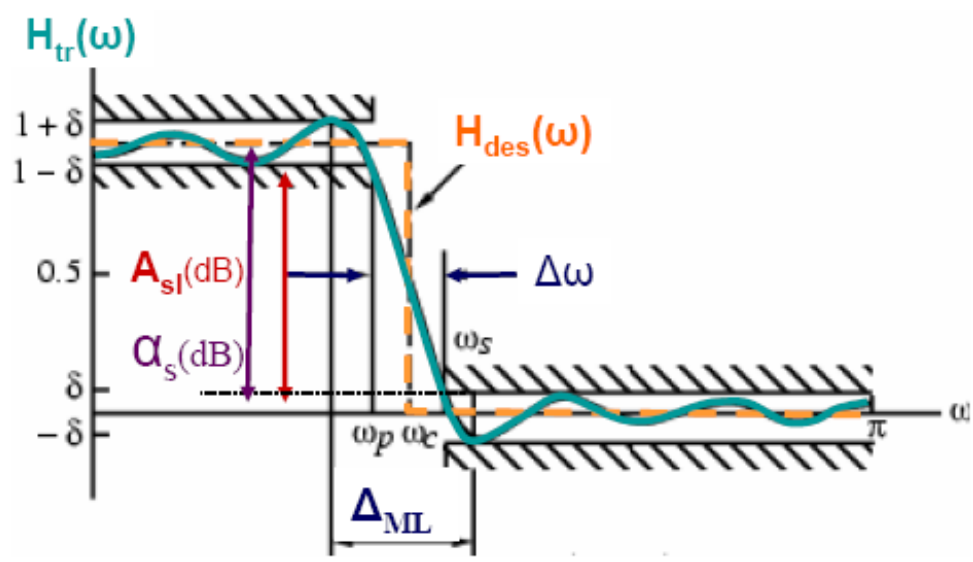
$$w(n) = 0.5 - 0.5 \cos \frac{2\pi n}{M}$$

6. Kaiser

$$I_0 \left\{ \beta \sqrt{1 - \left(\frac{n - M/2}{M/2} \right)^2} \right\}$$

All windows shown so far are **fixed window** functions

- Magnitude spectrum of each window characterized by a main lobe centered at $\omega = 0$ followed by a series of sidelobes with decreasing amplitudes
- Parameters predicting the performance of a window in filter design are:
 - **Main lobe width (Δ_{ML})** : the distance b/w nearest zero-crossings on both sides or transition bandwidth ($\Delta\omega = \omega_s - \omega_p$)
 - **Relative sidelobe level (A_{sl})**: difference in dB between the amp. of the largest sidelobe and the main lobe (or sidelobe attenuation (α_s))
- For a given window, both parameters all completely determined once the filter order M is set



Windows	Main lobewidth	Sidelob attenuation (dB)	Min. stopband attenuation	Transition bandwidth $\Delta\omega$
Rectangular	$4\pi/M$	-13	20.9	$0.92\pi/(M/2)$
Bartlett	$8\pi/M$	-27	See book	See book
Hanning	$8\pi/M$	-32	43.9	$3.11\pi/(M/2)$
Hamming	$8\pi/M$	-43	54.5	$3.32\pi/(M/2)$
Blackman	$12\pi/M$	-58	75.3	$5.56\pi/(M/2)$

How to design:

- Set $\omega_C = (\omega_p + \omega_s) / 2$
- Choose window type based on the specified sidelobe attenuation (A_{sl}) or minimum stopband attenuation (α_s)
- Choose M according to the transition band width ($\Delta\omega$) and/or mainlobe width (Δ_{ML}). Note that this is the only parameter that can be adjusted for fixed window functions. Once a window type and M is selected, so are A_{sl} , α_s , and Δ_{ML}
 - Ripple amplitudes cannot be custom designed
- Adjustable windows have a parameter that can be varied to trade-off between main-lobe width and side-lobe attenuation

- The most popular adjustable window

$$w[n] = \frac{I_0\left\{\beta \sqrt{1 - \left(\frac{n - M/2}{M/2}\right)^2}\right\}}{I_0(\beta)}, \quad 0 \leq n \leq M$$

where β is an adjustable parameter to trade-off between the **main lobe width** and **sidelobe attenuation**, and $I_0\{x\}$ is the **modified zeroth-order Bessel function** of the first kind:

$$I_0(x) = 1 + \sum_{k=1}^{\infty} \left[\frac{(x/2)^k}{k!} \right]^2$$

- In practice, this infinite series can be computed for a finite number of terms for a desired accuracy. In general, 20 terms is adequate

$$I_0(x) \cong 1 + \sum_{k=1}^{20} \left[\frac{(x/2)^k}{k!} \right]^2$$

Given the following:

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

1. Minimum ripple in dB:

$$\alpha_s = -20 \log_{10}(\delta_s) \text{ or } -20 \log_{10}(\min\{\delta_s, \delta_p\})$$

2. Normalized transition bandwidth:

$$\Delta\omega = \omega_s - \omega_p$$

3. Window parameters:

$$\beta = \begin{cases} 0.1102(\alpha_s - 8.7), & \alpha_s > 50 \text{ dB} \\ 0.5842(\alpha_s - 21)^{0.4} + 0.07886(\alpha_s - 21), & 21 \leq \alpha_s \leq 50 \text{ dB} \\ 0, & \alpha_s \leq 21 \text{ dB} \end{cases}$$

4. Filter length, $M+1$:

$$M + 1 = \begin{cases} \frac{\alpha_s - 7.95}{2.285\Delta\omega} + 1, & \alpha_s > 21 \\ \frac{5.79}{\Delta\omega}, & \alpha_s < 21 \end{cases}$$

5. Determine the corresponding Kaiser window

$$w[n] = \frac{I_0\left\{\beta\sqrt{1-\left(\frac{n-M/2}{M/2}\right)^2}\right\}}{I_0(\beta)}, \quad 0 \leq n \leq M$$

6. Obtain the filter by multiplying the ideal filter $h_I[n]$ with $w[n]$

Note: Design specs for Kaiser window in some books are different. This one, while may seem more complicated is actually easier to follow

Design an **FIR filter** with the following characteristics:

$\hookrightarrow \omega_p=0.3\pi \ \omega_s= 0.5\pi, \delta_s= \delta_p=0.01 \rightarrow \alpha=40\text{dB}, \Delta\omega=0.2\pi$

$$\beta = 0.5842(\alpha_s - 21)^{0.4} + 0.07886(\alpha_s - 21), \ 21 \leq \alpha_s < 50$$

$$\beta = 0.5842(19)^{0.4} + 0.07886 \times 19 = 3.3953$$

$$M + 1 = \frac{32.05}{2.285(0.2\pi)} + 1 = 23.2886 \approx 24$$

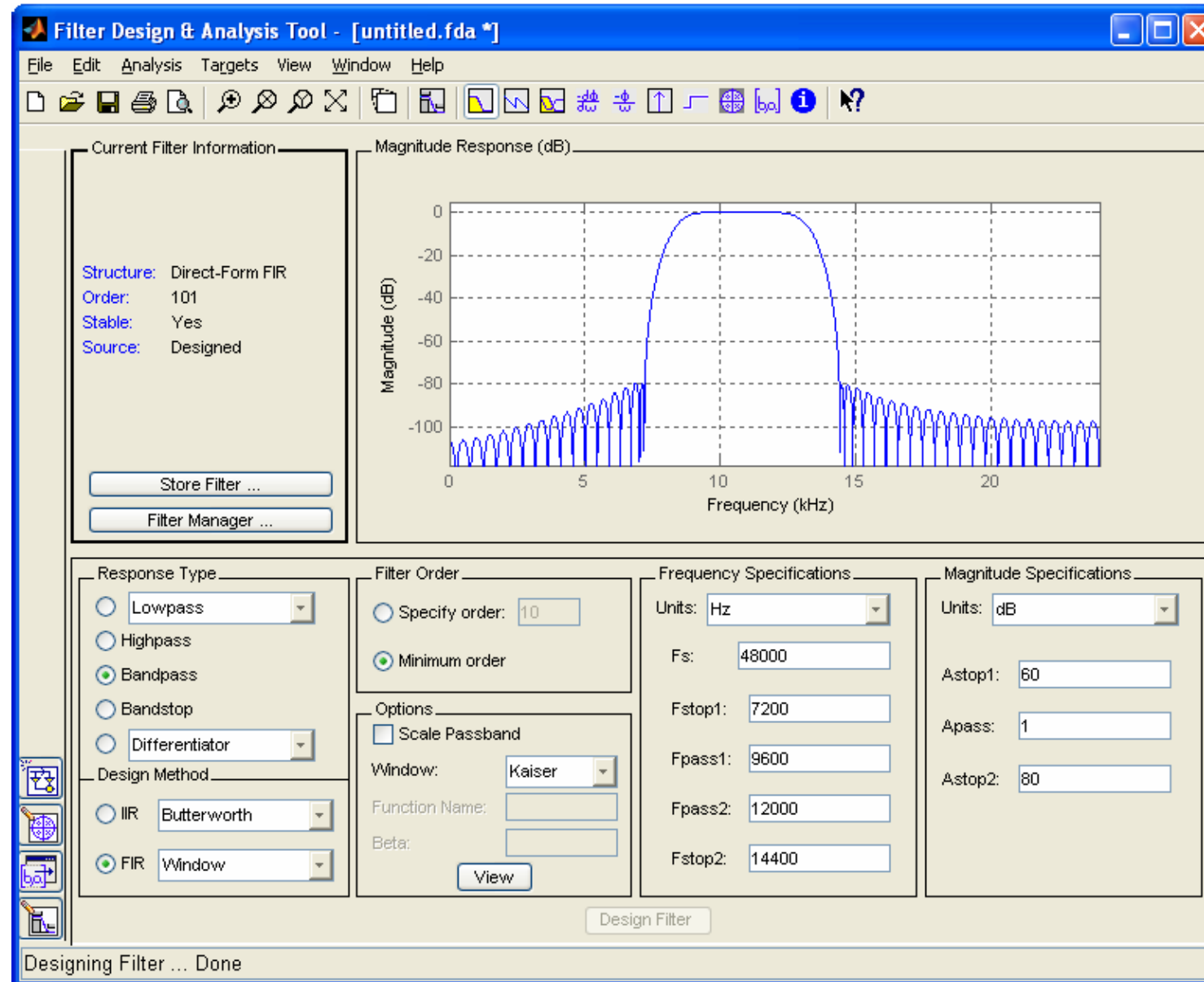
$$w[n] = \text{kaiser}(M + 1, \beta) \text{ (from matlab)}$$

$$h_{LP}[n] = \begin{cases} \frac{\sin(\omega_c(n - M/2))}{\pi(n - M/2)}, & 0 < n < M, \ n \neq \frac{M}{2} \\ \frac{\omega_c}{\pi}, & n = \frac{M}{2} \end{cases} \quad \rightarrow \quad h_{LP}[n] = \begin{cases} \frac{\sin(0.4\pi(n - 12))}{\pi(n - 12)}, & 0 < n < 23, \ n \neq 12 \\ 0.4, & n = 12 \end{cases}$$

$$h_t[n] = h_{LP}[n] \cdot w[n], \quad -12 \leq n \leq 12$$

- Depending on your specs, determine what kind of **window** you would like to use
 - For all window types, except Kaiser, once you choose the window, the only other parameter to choose is filter length M
 - For Kaiser window, determine M and b , based on the specs using the given expressions
- Compute the **window coefficients** $w[n]$ for the chosen window
- Compute **filter coefficients (taps)**
 - Determine the **ideal impulse response** $h_I[n]$ from the given equations for the type of **magnitude response** you need (lowpass, highpass, etc.)
 - Multiply **window** and **ideal filter coefficients** to obtain the **realizable filter coefficients** (also called **taps** or **weights**): $h[n] = h_I[n] \cdot w[n]$
- Convolve your signal with the **new filter coefficients** $y[n] = x[n] * h[n]$

- The following functions create N -point windows for the corresponding functions:
 - `rectwin(N)`
 - `bartlett(N)`
 - `hamming(N)`
 - `kaiser(N, beta)`
 - `hanning(N)`
 - `blackman(N)`
- Try this: `h=hamming(40); [H w]=freqz(h,1, 1024); plot(w, abs(H))`
 - Compare for various windows. Also plot gain in dB
- The function `window` **window design and analysis** tool provides a GUI to custom design several window functions
- The function `fdatool` **filter design and analysis tool** provides a GUI to custom design several types of filters from the given specs
- The function `sptool` **signal processing tool**, provides a GUI to custom design, view and apply to custom created signals. It also provides a GUI for spectral analysis



`fir1` function designs an FIR filter using the window method

`b = fir1(N,Wn)` designs an N^{th} order **lowpass** FIR digital filter and returns the filter coefficients in length $N+1$ vector `b`. The cut-off frequency `Wn` must be between $0 < Wn < 1.0$, with 1.0 corresponding to half the sample rate. The filter `b` is real and has linear phase. The normalized gain of the filter at `Wn` is -6 dB.

`b = fir1(N,Wn,'high')` designs an N^{th} order **highpass filter**. You can also use `b = fir1(N,Wn,'low')` to design a **lowpass filter**.

If `Wn` is a two-element vector, `Wn = [W1 W2]`, `FIR1` returns an order N **bandpass filter** with passband $W1 < W < W2$. You can also specify `b = fir1(N,Wn,'bandpass')`. If `Wn = [W1 W2]`, `b = fir1(N,Wn,'stop')` will design a **bandstop filter**.

If `Wn` is a multi-element vector, `Wn = [W1 W2 W3 W4 W5 ... WN]`, `FIR1` returns an order N **multiband filter** with bands $0 < W < W1$, $W1 < W < W2$, ..., $WN < W < 1$.

`b = fir1(N,Wn,'DC-1')` makes the first band a passband.

`b = fir1(N,Wn,'DC-0')` makes the first band a stopband.

`b = fir1(N,Wn,WIN)` designs an N^{th} order FIR filter using the $N+1$ length vector `WIN` to window the impulse response. **If empty or omitted, FIR1 uses a Hamming window of length $N+1$.** For a complete list of available windows, see the help for the `WINDOW` function. If using a Kaiser window, use the following `b = fir1(N,Wn,kaiser(n+1,beta))`

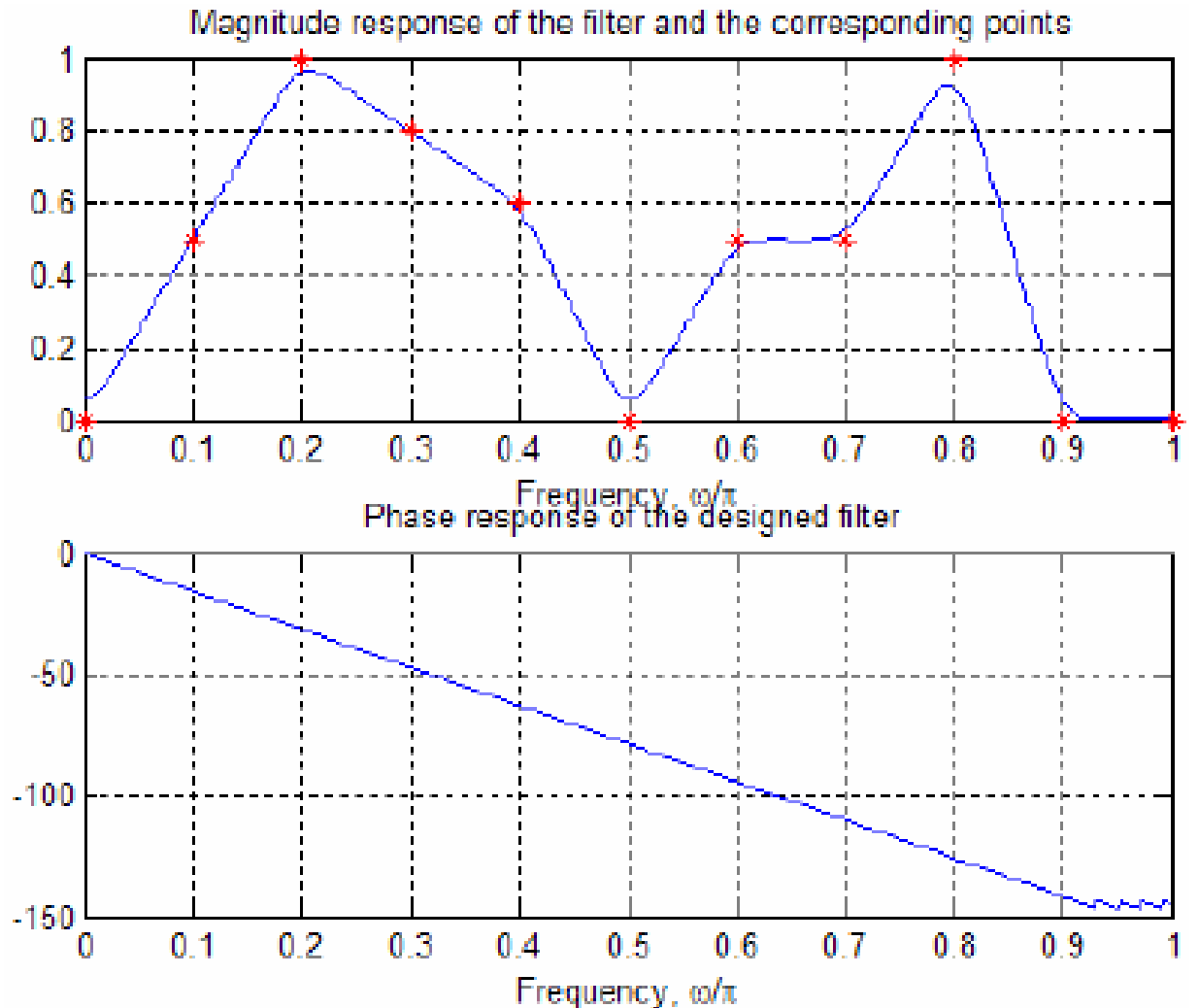
Here is the complete procedure:

- **Obtain the specs:** Cutoff frequency, passband and stopband edge frequencies, allowed maximum ripple, filter order
 - Note that you do not need filter order for **Kaiser** (to be determined), and you do not need the edge frequencies and ripple amount for the others
- Design the filter using the `fir1()` command. Default window is **Hamming**. For all window types – except **Kaiser** – provide filter order N and normalized cutoff frequency (between 0 and 1)
 - For **Kaiser** window, determine the **beta** and M manually from the given equations before providing them to the `fir1` command with **kaiser** as filter type
 - You can also use `kaiserord()` to estimate the filter order from
 - This gives you the “b” coefficients, or in other words, the impulse response $h[n]$
- Use this filter with the `filter()` command as `y=filter(b,a,x)`, where b are the coefficients obtained from `fir1()`, a=1 since this is an FIR filter, x is the signal to be filtered, and y is the filtered signal
- OR, use `sptool` for the entire design cycle

```
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]
```

```
b=fir2(100, freq, amp, 1024);
```

```
subplot(211)
[H w]=freqz(b, 1, 1024);
plot(w/pi, abs(H)); hold on
plot(freq, amp, 'r*')
grid
xlabel('Frequency, \omega/\pi')
title(' Magnitude response of the filter and the
corresponding points')
subplot(212)
plot(w/pi, unwrap(angle(H)));
xlabel('Frequency, \omega/\pi')
title(' Phase response of the designed filter')
grid
```



You will need to determine the filter order by trial and error. You may need higher orders if your specified points require a sharp transition

- Bilinear transformation for analog \leftrightarrow digital domains
- Analog IIR filter design
 - Butterworth approximation and Matlab implementation
 - Chebyshev approximation and Matlab implementation
 - Elliptic filters
 - Bessel filters
- Traditionally, these filters are first designed in the analog domain, and then converted into digital domain using bilinear transformation, but today they can be designed directly in the digital domain
- Spectral transformations
 - Analog – to – analog transformations
 - Digital – to – digital transformations
- Direct IIR filter design

Matlab has several functions:

`buttord` This function is used for **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 R_p dB in the passband and has at least R_s dB of attenuation in the stopband. W_p and W_s are the **passband and stopband edge frequencies**, normalized from 0 to 1 (where 1 corresponds to π radians/sample)

`buttord()` also returns W_n , 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 W_p and W_s are in radians/second. Note that for analog filter design W_n is not restricted to the $[0\ 1]$ range. When R_p is chosen as 3 dB, the W_n in `butter()` is equal to W_p in `buttord()`

`butter()` Butterworth digital and analog filter design

`[B,A] = butter(N,Wn)` designs an Nth 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**:

$H(z) = H(s)|_{s = 2*F_s*(z-1)/(z+1)}$ where column vectors Z and P specify the zeros and poles, scalar K specifies the gain, and F_s is the sampling frequency in Hz. If normalized $T_s=2$ is used, then $F_s=0.5$.

`[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)

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

`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]

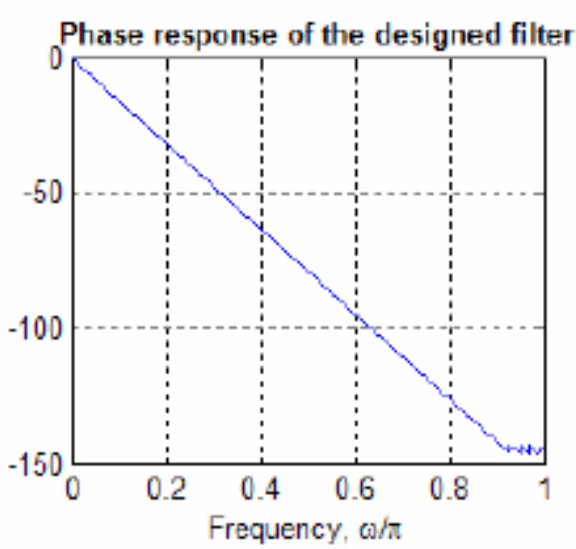
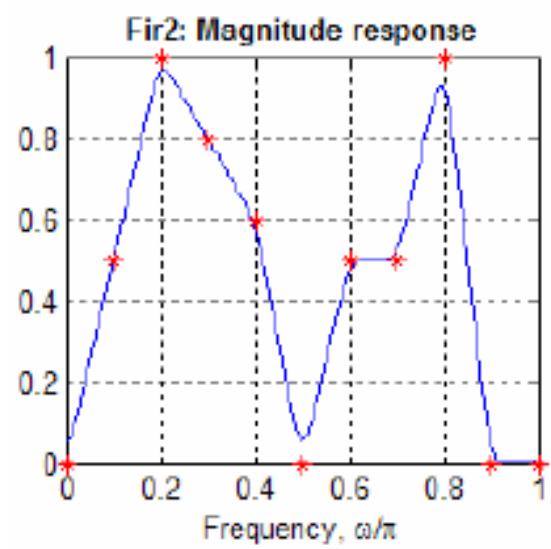
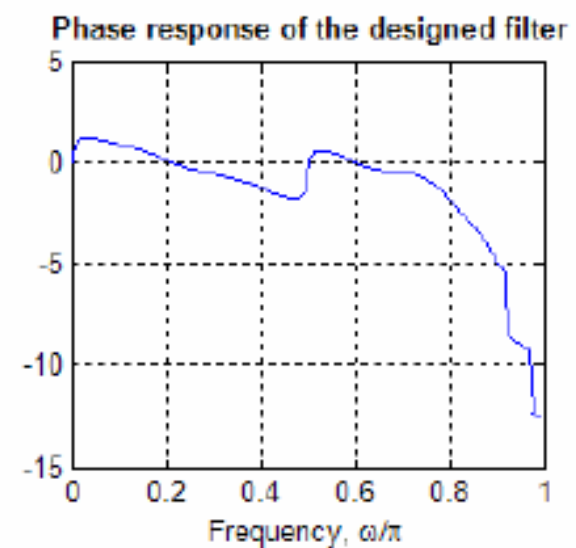
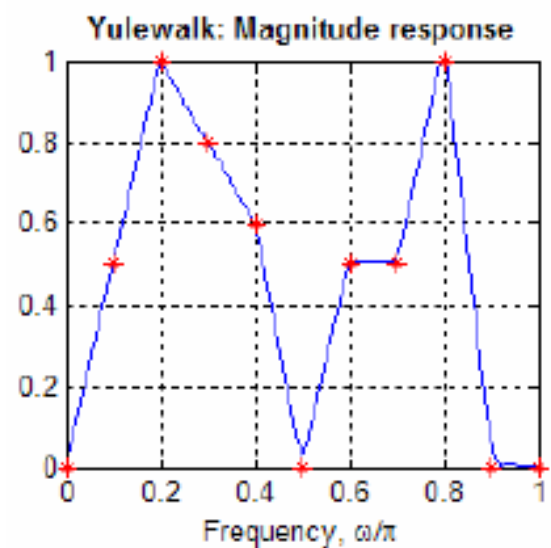
`cheb1ord()` also returns Wn, the Chebyshev natural frequency to use with `cheby1()` to achieve the specifications.

`[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.

- 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

```
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);
[H2 w]=freqz(b2, 1, 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)
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 !!!