National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Sir Hassan Khaliq Dated: 4/26/2023

Course/Section: BEE-12C Semester: 6th

EE-330 Digital Signal Processing

Lab#9 FIR filter design using Windowing

		PLO4-CLO4		PLO5-CLO5	PLO8-CLO6	PLO9-CLO7
Name	**Reg. No**	**Viva / Quiz / Lab Performance** 5 Marks	**Analysis of data in Lab Report** 5 Marks	**Modern Tool Usage** 5 Marks	**Ethics and Safety** 5 Marks	**Individual and Team Work** 5 Marks
Hassan Rizwan	**335753**					
Muhammad Ahmed Mohsin	**333060**					
Syeda Fatima Zahra	**334379**					

1 TABLE OF CONTENTS

FIR filter design using Windowing

2 LAB INSTRUCTIONS

- The students should perform and demonstrate each lab task separately for stepwise evaluation (please ensure that course instructor/lab engineer has signed each step after ascertaining its functional verification)
- Each group shall submit one lab report on LMS within 6 days after lab is conducted. Lab reports submitted via email will not be graded.
- Students are however encouraged to practice on their own in spare time for enhancing their skills.

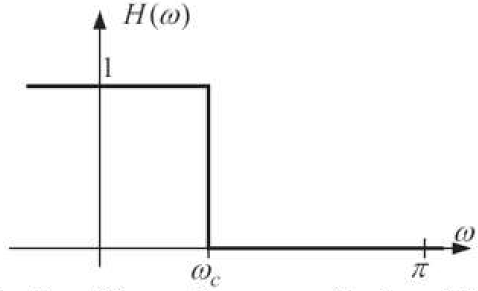
3 LAB REPORT INSTRUCTIONS

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

- Lab objectives
- MATLAB/C codes
- Results (graphs/tables) duly commented and discussed Conclusion

4 FIR WINDOW FILTER DESIGN WITH MATLAB

Ideal Low-pass filter specifications



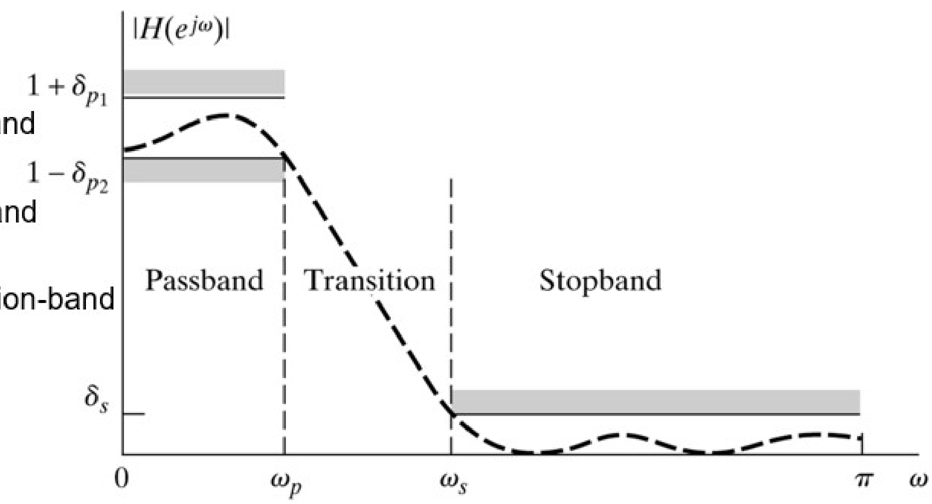
- For frequency selective filters, the magnitude of the desired response is usually specified in terms of the *tolerable*
  - pass-band distortion,
  - stop-band attenuation, and
  - width of transition band

Parameters

$[0;\omega_p]$ = pass-band

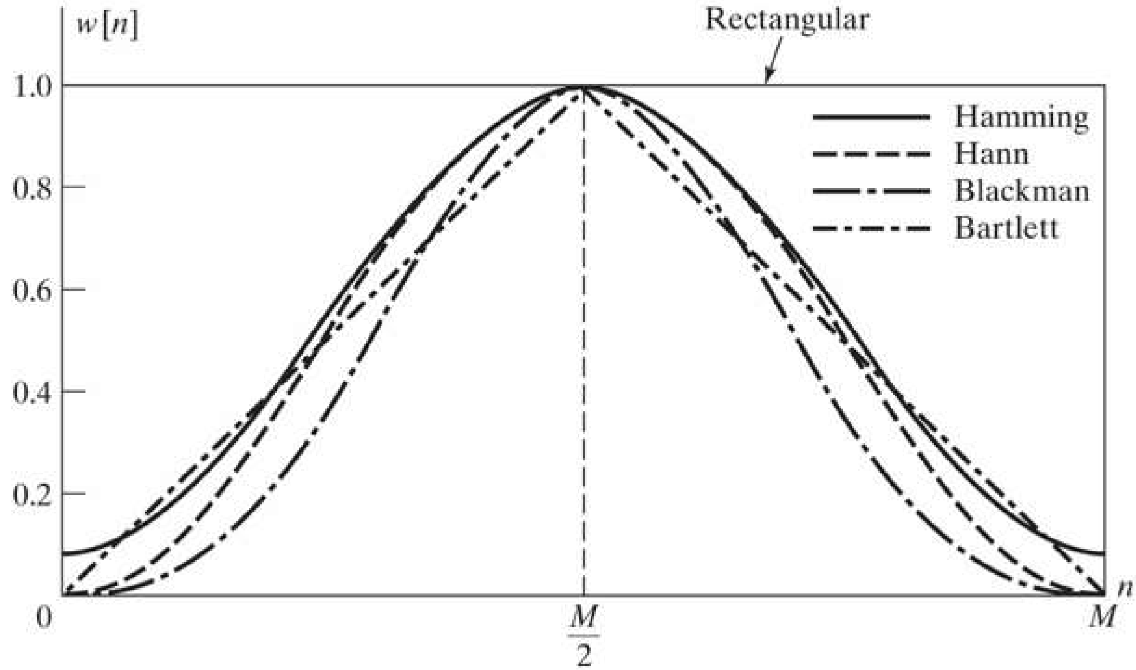
$[\omega_s;\pi]$ = stop-band

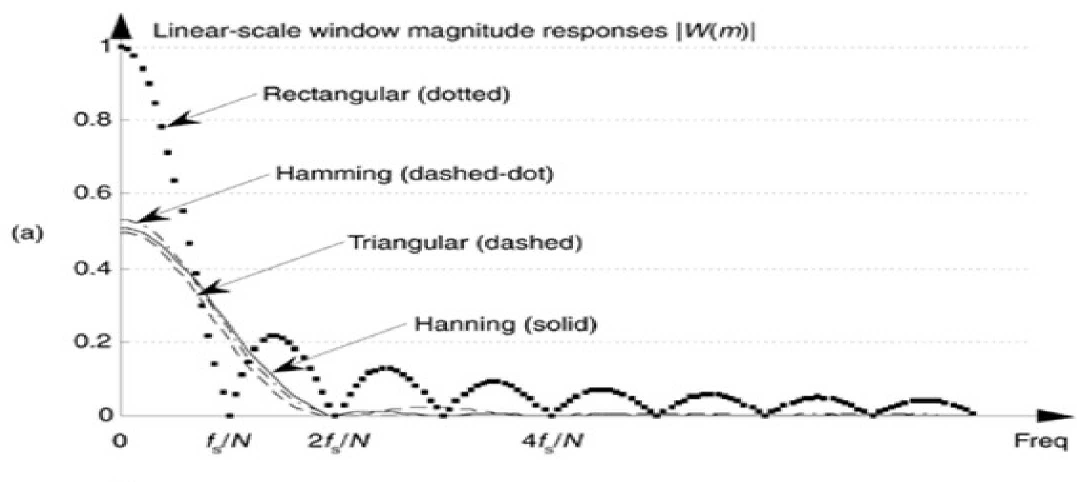
$[\omega_p;\omega_s]$ = transition-band



δ_{p1}; δ_{p2} = pass-band ripple; Often expressed in dB via $20\log_{10}(1+\delta_p)$

δ_s = stop-band ripple; Often expressed in dB via $20\log_{10}(\delta_s)$





5 LINEAR-PHASE FIR FILTER:

Shapes of impulse and frequency responses and locations of system function zeros of linear-phase FIR filters will be discussed. Let $h(n)$, $0 \le n \le M-1$ be the impulse response of length (or duration) *M*. Then the system function is

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

which has $(M - 1)$ poles at the origin $z = 0$ (trivial poles) and $(M - 1)$ zeros located anywhere in the z-plane. The frequency response function is

$$H\left(e^{jw}\right)=\sum_{n=0}^{M-1} h(n)e^{-jw}, \qquad -\pi \prec w \leq \pi$$

6 WINDOW DESIGN TECHNIQUES

Summary of commonly used window function characteristics

Window Name	Transition Approximate	Width Δw Exact Values	Min. Stopband Attenuation
Rectangular	$\frac{4\pi}{M}$	$\frac{1.8\pi}{M}$	21 dB
Bartlett	$\frac{8\pi}{M}$	$\frac{6.1\pi}{M}$	25 dB
Hanning	$\frac{8\pi}{M}$	$\frac{6.2\pi}{M}$	44 dB
Hamming	$\frac{8\pi}{M}$	$\frac{6.6\pi}{M}$	53 dB
Blackman	$\frac{12\pi}{M}$	$\frac{11\pi}{M}$	74 dB

MATLAB provides several routines to implement window functions discussed above table.

- w = boxcar(n) returns the n-point rectangular window in array w.
- w = triang(n) returns the n-point Bartlett (Triangular) window in array w.
- w = hanning(n) returns the n-point symmetric Hanning window in a column vectorin array w .
- w = hamming(n) returns the n-point symmetric Hamming window in a column vector in array w.
- w = blackman(n) returns the n-point symmetric Blackman window in a column vector in array w.

- w = kaiser(n,beta) returns the beta-valued n-point Kaiser window in array w.

Using these routines, we can use MATLAB to design FIR filters based on the window technique, which also requires an ideal lowpass impulse response $h_d(n)$ as shown below.

```
function hd=ideal_lp(wc,M);
% Ideal LowPass filter computation
% --------------------------------
% [hd] = ideal_lp(wc,M);
% hd = ideal impulse response between 0 to M-1
% wc = cutoff frequency in radians
% M = length of the ideal filter
%
alpha = (M-1)/2;
n = [0:1:(M-1)];
m = n - alpha +eps;     % add smallest number to avoid divided by zero
hd = sin(wc*m)./(pi*m);
```

To display the frequency domain plots of digital filters, MATLAB provides the **freqz** routine. Using this routing, we can developed a modified version, called freqz_m, which returns the magnitude response in absolute as well as dB scale, the phase response, and the group delay response as shown below.

```
function [db,mag,pha,grd,w] = freqz_m(b,a)
% Modified version of freqz subroutine
% ------------------------------------
% [db,mag,pha,grd,w] = freqz_m(b,a)
% db = relative magnitude in dB computed over 0 to pi radians
% mag = absolute magnitude computed over 0 to pi radians
% pha = Phase response in radians over 0 to pi radians
% grd = Group delay over 0 to pi radians
%     w = 501 frequency samples between 0 to pi radians
%     b = numerator polynomial of H(z)    (for FIR: b=h)
```

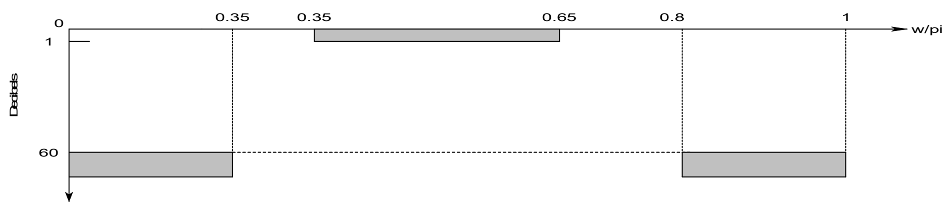
```
%      a = denominator polynomial of H(z)     (for FIR: a=[1])
%
[H,w] = freqz(b,a,1000,'whole') ;
    H = (H(1:1:501))';       w = (w(1:1:501))';
  mag = abs(H);
db = 20*log10((mag+eps)/max(mag));
pha = angle(H);
grd = grpdelay(b,a,w);
```

7 LAB TASK 1:

Design the following digital bandpass filter.

$$\begin{aligned}
&\textit{lower stopband edge}: \quad w_{1s} = 0.2\pi, \quad A_s = 60dB \\
&\textit{lower passband edge}: \quad w_{1p} = 0.35\pi \quad R_p = 1dB \\
&\textit{upper passband edge}: \quad w_{2p} = 0.65\pi, \quad R_p = 1dB \\
&\textit{upper stopband edge}: \quad w_{2s} = 0.8\pi \quad A_s = 60dB
\end{aligned}$$

These quantities are shown in the following figure.



Follow the steps:

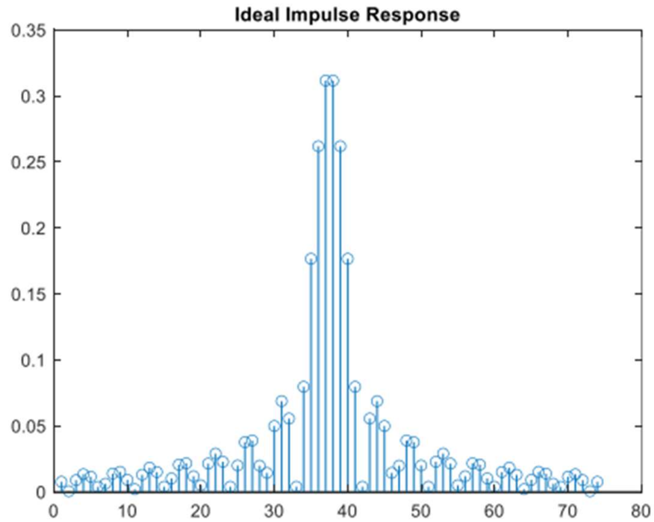
- **Find transition width**
- **Calculate M**
- **Create ideal band pass filter with two ideal low pass filter(for ideal lowpass use `hd=ideal_lp(wc,M) code given Above`)**
- **Implement appropriate window (use the table)**
- **Multiply window coefficient with your ideal band pass filter**
- **Plot the ideal impulse response**
- **Plot actual impulse response**
- **Plot frequency response**

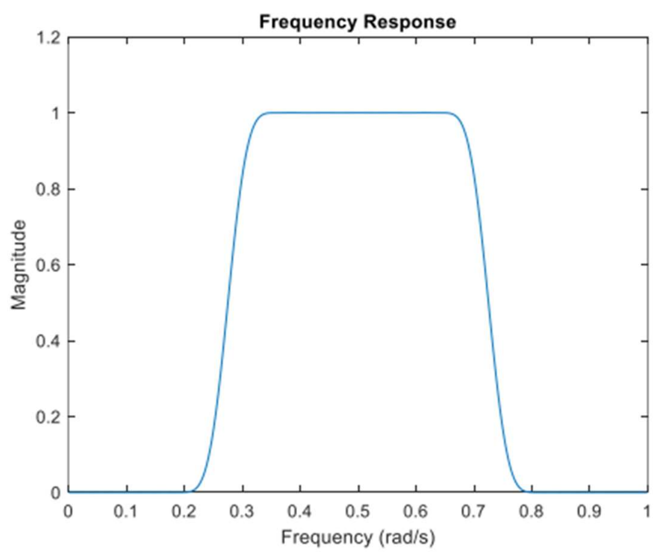
CODE:

```
% Define the parameters

wp1 = 0.35*pi;
ws1 = 0.2*pi;
wp2 = 0.65*pi;
ws2 = 0.8*pi;
tranband = wp1 - ws1;
% Calculate the filter length
M = ceil((11*pi) / tranband);
% Calculate the cutoff frequencies
wc1 = (ws1+wp1)/2;
wc2 = (ws2+wp2)/2;
% Create the ideal low-pass filters
hd1 = ideal_lp(wc1, M);
hd2 = ideal_lp(wc2, M);
% Create the bandpass filter
hd = hd2 - hd1;
% Apply the Kaiser window
wn = kaiser(M, 8);
h = wn .* hd;
% Plot the ideal impulse response
figure
stem(abs(hd))
title("Ideal Impulse Response")
% Plot the actual impulse response
figure
stem(abs(h))
title("Actual Impulse Response")
% Plot the frequency response
figure
[H, w] = freqz(h, 1, 1024);
plot(w/pi, abs(H))
title("Frequency Response")
xlabel("Frequency (rad/s)")
ylabel("Magnitude")
```

7.1 OUTPUT:





8 LAB TASK 2

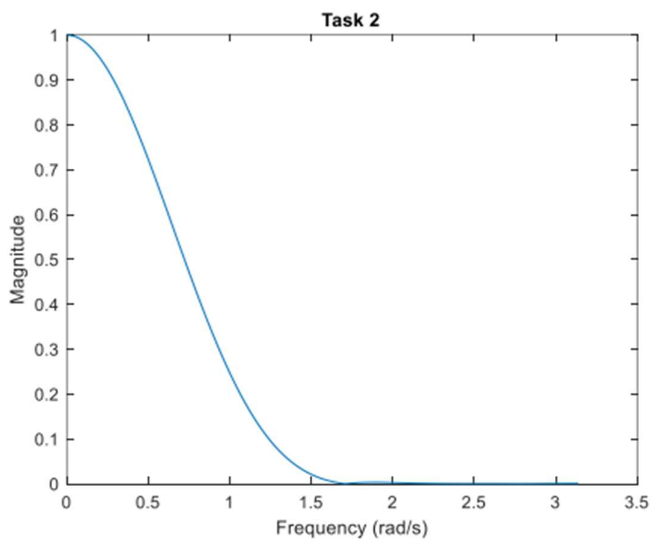
Design a lowpass FIR filter with a passband cutoff frequency of 1 kHz, a stopband edge at 4.3 kHz, and a sampling frequency of 10 kHz. We will use a Hamming window. The transition width is the difference of the stopband edge and the passband edge. Thus, the normalized transition width ∆f is the number of coefficients can then be estimated as

We must now determine the normalized cutoff frequency for MATLAB. Remember this is to be normalized to $F_s/2$. Here our cutoff frequency is 1 kHz .

8.1 Code:

```
% Define the parameters
fp = 1000;
fst = 4300;
Fs = 10000;
trans = fst - fp;
% Calculate the transition width
del_w = (2*pi*trans)/Fs;
% Calculate the filter length
M = ceil(6.6*pi/del_w);
% Create the filter
b = fir1(M, 1000/5000, 'kaiser', 8);
% Calculate the frequency response
[H, w] = freqz(b, 1, 1024);
% Plot the frequency response
figure
plot(w, abs(H))
xlabel("Frequency (rad/s)")
ylabel("Magnitude")
title("Task 2")
```

8.2 Output:



9 Lab Task 3:

Design Bandpass Filter with following specification using fir1 function.

Fs = 48 kHz

Passband Cutoff Frequencies = 8 kHz & 16 kHz

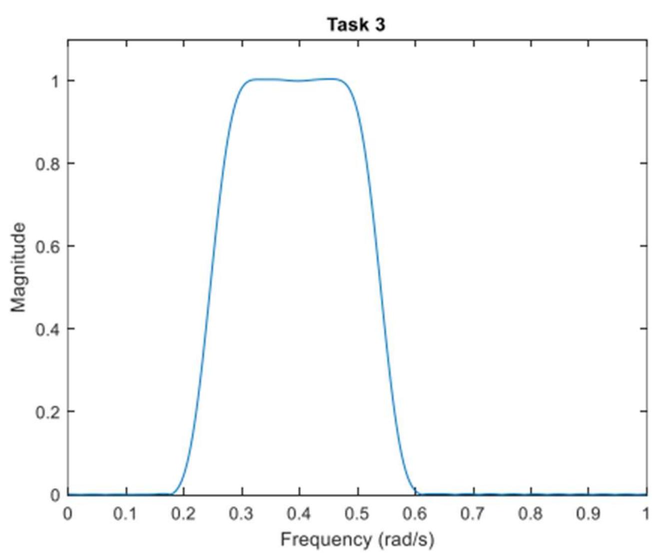
Stopband Edge Frequencies = 7 kHz & 17 kHz

Hamming Window

9.1 CODE:

```
% Define the parameters
fp1 = 8000;
fs1 = 7000;
fp2 = 16000;
fs2 = 17000;
Fs = 48000;
trans = fp1 - fs1;
fc1 = (fp1 + fs1)/2;
fc2 = (fp2 + fs2)/2;
% Calculate the transition width
del_w = (2*pi*trans)/Fs;
% Calculate the filter length
M = ceil(6.6*pi/del_w);
% Create the filter
b = fir1(M, [fc1/Fs/2, fc2/Fs/2], 'kaiser', 8);
% Calculate the frequency response
[H, w] = freqz(b, 1, 1024);
% Plot the frequency response
figure
plot(w, abs(H))
xlabel("Frequency (rad/s)")
ylabel("Magnitude")
```

9.2 OUTPUT:



10 CONCLUSION:

In this lab, we designed FIR filters using the windowing method. We found that windowing is a simple and effective method for designing FIR filters, and that it can be used to design linear phase filters. The main advantage of windowing is that it is easy to implement. The main disadvantage of windowing is that it can introduce some distortion into the frequency response. This distortion can be minimized by using a window function with a good stopband attenuation. In general, windowing is a good choice for designing FIR filters when simplicity and speed are important considerations. However, if high performance is required, other methods of designing FIR filters, such as least squares design, may be a better choice.