Digital Signal Processing Lab

Laboratory report submitted for the partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Mohit Akhouri - 19ucc023

Course Coordinator
Dr. Divyang Rawal

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

September 2021

Contents

Chapter *9*

Experiment - 9

9.1 Aim of the Experiment

- Digital Filter Design

9.2 Software Used

- MATLAB
- Simulink

9.3 Theory

9.3.1 About Window function :

In signal processing and statistics, a window function (also known as an **apodization function** or **tapering function**) is a mathematical function that is zero-valued outside of some chosen interval, normally symmetric around the middle of the interval, usually near a maximum in the middle, and usually tapering away from the middle. Mathematically, when another function or waveform/data-sequence is ”multiplied” by a window function, the product is also zero-valued outside the interval: all that is left is the part where they overlap, the ”view through the window”. Equivalently, and in actual practice, the segment of data within the window is first isolated, and then only that data is multiplied by the window function values. Thus, tapering, not segmentation, is the main purpose of window functions.

In typical applications, the window functions used are non-negative, smooth, **bell-shaped curves**. Rectangle, triangle, and other functions can also be used. A more general definition of window functions does not require them to be identically zero outside an interval, as long as the product of the window multiplied by its argument is square integrable, and, more specifically, that the function goes sufficiently rapidly toward zero.

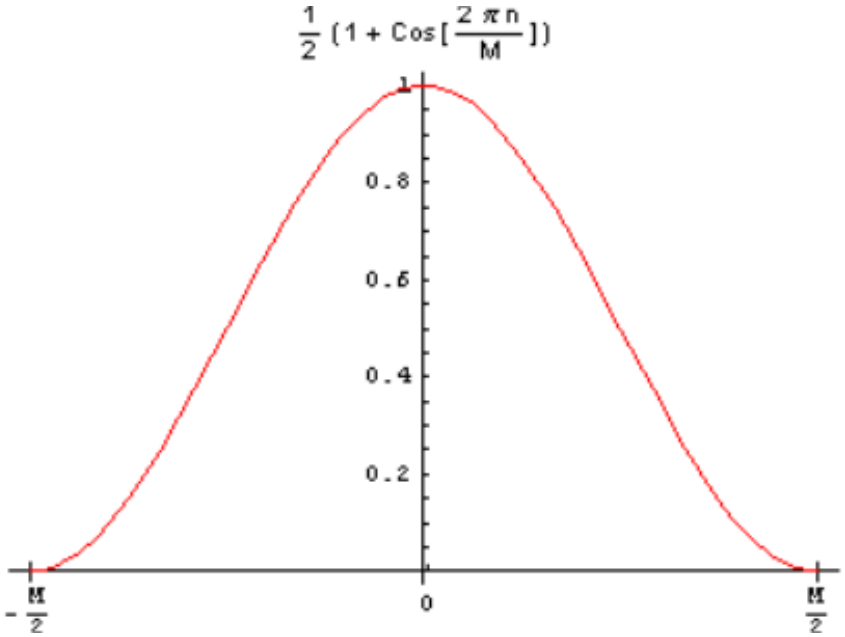

Figure 9.1 Hanning Window function

Windows are sometimes used in the **design of digital filters**, in particular to convert an "ideal" impulse response of infinite duration, such as a sinc function, to a finite impulse response (FIR) filter design. That is called the **window method**.

9.3.2 Types of Window functions :

9.3.2.1 About Rectangular Window :

The rectangular window (sometimes known as the **boxcar** or **Dirichlet window**) is the simplest window, equivalent to replacing all but N values of a data sequence by zeros, making it appear as though the waveform suddenly turns on and off. The window function $w(n)$ is given as :

$$w(n) = 1 \tag{9.1}$$

The rectangular window is the **1st order B-spline window** as well as the **0th power power-of-sine window**.

9.3.2.2 About B-spline Window :

B-spline windows can be obtained as k-fold convolutions of the rectangular window. They include the **rectangular window** itself (k = 1), the **Triangular window** (k = 2) and the **Parzen window** (k = 4). Alternative definitions sample the appropriate normalized B-spline basis functions instead of convolving discrete-time windows. A k^{th}-order B-spline basis function is a piece-wise polynomial function of degree k-1 that is obtained by **k-fold self-convolution** of the rectangular function.

9.3.2.3 About Hamming Window :

The Hamming window was proposed by **Richard W. Hamming**. That choice places a zero-crossing at frequency $5\pi/(N-1)$, which cancels the first sidelobe of the Hann window, giving it a height of about one-fifth that of the Hann window. The Hamming window is often called the **Hamming blip** when used for **pulse shaping**. The window function $w(n)$ is given as :

$$w(n) = 0.42 - 0.5 * cos\left(\frac{2\pi n}{M-1}\right) + 0.08 * cos\left(\frac{4\pi n}{M-1}\right) \tag{9.2}$$

9.3.2.4 About Blackman Window :

The Blackman window is a **taper** formed by using the first three terms of a summation of cosines. It was designed to have close to the minimal leakage possible. It is close to optimal, only slightly worse than a **Kaiser window**. The window function $w(n)$ is given as :

$$w(n) = 0.54 - 0.46 * cos\left(\frac{2\pi n}{M-1}\right) \tag{9.3}$$

9.3.3 About Notch Filter :

A notch filter is a type of **band-stop filter**, which is a filter that attenuates frequencies within a specific range while passing all other frequencies unaltered. For a notch filter, **this range of frequencies is very narrow**.

The range of frequencies that a band-stop filter attenuates is called the **stopband**. The narrow stopband in a notch filter makes the frequency response resemble a deep notch, which gives the filter its name. It also means that notch filters have a **high Q factor**, which is the ratio of center frequency to bandwidth.

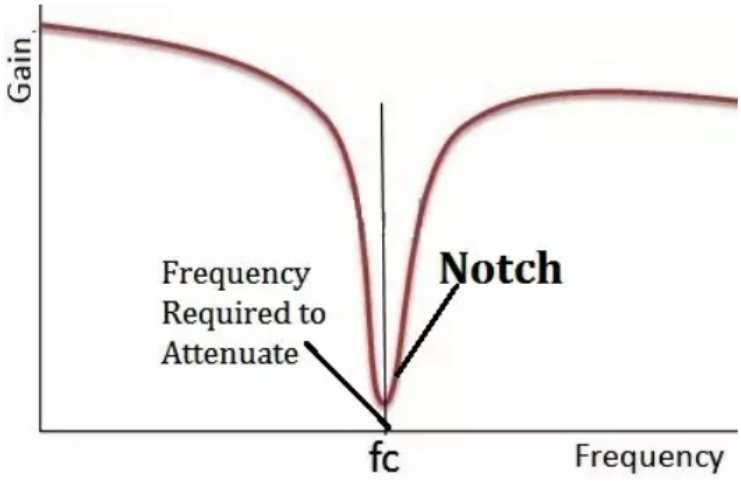

Figure 9.2 Frequency Response of Notch Filter

9.4 Code and results

9.4.1 Simulating various window based low pass FIR filter for different cutoff frequencies :

```
% 19ucc023
% Mohit Akhouri
% Experiment 9 - Observation 1a

% In this code , we will implement a low pass filter based on various
% windowing techniques - like rectangular , hamming and blackman. We
 design
% the low pass filter for various cutoff frequencies like pi/2,pi/4
 and
% pi/6. We plot the amplitude and phase response of the various low
 pass
% filter.

clc;
clear all;
close all;

M = 31; % delay in time domain

wc = pi/2; % cutoff frequency
hd = zeros(1,M); % to store the filter impulse response
w1 = zeros(1,M); % to store the window function for rectangular window
w2 = zeros(1,M); % to store the window function for hamming window
w3 = zeros(1,M); % to store the window function for blackman window

% main loop algorithm for calculation of window functions and filter
% impulse response for various cases
for n=0:M-1
    H = @(w) exp(-1j*w*(n-((M-1)/2))); % response of the low pass
 filter
    hd(n+1) = (integral(H,-wc,wc))/(2*pi); % filter impulse response
    w1(n+1) = 1; % window function for rectangular window
    w2(n+1) = 0.42 - 0.5*cos((2*pi*n)/(M-1)) + 0.08*cos((4*pi*n)/
(M-1)); % window function for hamming window
    w3(n+1) = 0.54 - 0.46*cos((2*pi*n)/(M-1)); % window function for
 blackman window
end

h1 = hd .* w1; % finite impulse response of rectangular window based
 low pass filter
h2 = hd .* w2; % finite impulse response of hamming window based low
 pass filter
h3 = hd .* w3; % finite impulse response of blackman window based low
 pass filter

% Plotting the magnitude and phase responses of various low pass
 filter
figure;
freqz(h1);
title('19ucc023 - Mohit Akhouri','Rectangular window for low pass
 filter for cutoff frequency = \pi/2');
grid on;
```

Figure 9.3 Part 1 of the Code for the observation 1(a)

```
figure;
freqz(h2);
title('19ucc023 - Mohit Akhouri','Hamming window for low pass filter
 for cutoff frequency = \pi/2');
grid on;

figure;
freqz(h3);
title('19ucc023 - Mohit Akhouri','Blackman window for low pass filter
 for cutoff frequency = \pi/2');
grid on;

wc = pi/4; % cutoff frequency
hd = zeros(1,M); % to store the filter impulse response
w1 = zeros(1,M); % to store the window function for rectangular window
w2 = zeros(1,M); % to store the window function for hamming window
w3 = zeros(1,M); % to store the window function for blackman window

% main loop algorithm for calculation of window functions and filter
% impulse response for various cases
for n=0:M-1
    H = @(w) exp(-1j*w*(n-((M-1)/2))); % response of the low pass
 filter
    hd(n+1) = (integral(H,-wc,wc))/(2*pi); % filter impulse response
    w1(n+1) = 1; % window function for rectangular window
    w2(n+1) = 0.42 - 0.5*cos((2*pi*n)/(M-1)) + 0.08*cos((4*pi*n)/
(M-1)); % window function for hamming window
    w3(n+1) = 0.54 - 0.46*cos((2*pi*n)/(M-1)); % window function for
 blackman window
end

h1 = hd .* w1; % finite impulse response of rectangular window based
 low pass filter
h2 = hd .* w2; % finite impulse response of hamming window based low
 pass filter
h3 = hd .* w3; % finite impulse response of blackman window based low
 pass filter

% Plotting the magnitude and phase responses of various low pass
 filter
figure;
freqz(h1);
title('19ucc023 - Mohit Akhouri','Rectangular window for low pass
 filter for cutoff frequency = \pi/4');
grid on;

figure;
freqz(h2);
title('19ucc023 - Mohit Akhouri','Hamming window for low pass filter
 for cutoff frequency = \pi/4');
grid on;

figure;
```

Figure 9.4 Part 2 of the Code for the observation 1(a)

```
freqz(h3);
title('19ucc023 - Mohit Akhouri','Blackman window for low pass filter
 for cutoff frequency = \pi/4');
grid on;

wc = pi/6; % cutoff frequency
hd = zeros(1,M); % to store the filter impulse response
w1 = zeros(1,M); % to store the window function for rectangular window
w2 = zeros(1,M); % to store the window function for hamming window
w3 = zeros(1,M); % to store the window function for blackman window

% main loop algorithm for calculation of window functions and filter
% impulse response for various cases
for n=0:M-1
    H = @(w) exp(-1j*w*(n-((M-1)/2))); % response of the low pass
 filter
    hd(n+1) = (integral(H,-wc,wc))/(2*pi); % filter impulse response
    w1(n+1) = 1; % window function for rectangular window
    w2(n+1) = 0.42 - 0.5*cos((2*pi*n)/(M-1)) + 0.08*cos((4*pi*n)/
(M-1)); % window function for hamming window
    w3(n+1) = 0.54 - 0.46*cos((2*pi*n)/(M-1)); % window function for
 blackman window
end

h1 = hd .* w1; % finite impulse response of rectangular window based
 low pass filter
h2 = hd .* w2; % finite impulse response of hamming window based low
 pass filter
h3 = hd .* w3; % finite impulse response of blackman window based low
 pass filter

% Plotting the magnitude and phase responses of various low pass
 filter
figure;
freqz(h1);
title('19ucc023 - Mohit Akhouri','Rectangular window for low pass
 filter for cutoff frequency = \pi/6');
grid on;

figure;
freqz(h2);
title('19ucc023 - Mohit Akhouri','Hamming window for low pass filter
 for cutoff frequency = \pi/6');
grid on;

figure;
freqz(h3);
title('19ucc023 - Mohit Akhouri','Blackman window for low pass filter
 for cutoff frequency = \pi/6');
grid on;
```

Figure 9.5 Part 3 of the Code for the observation 1(a)

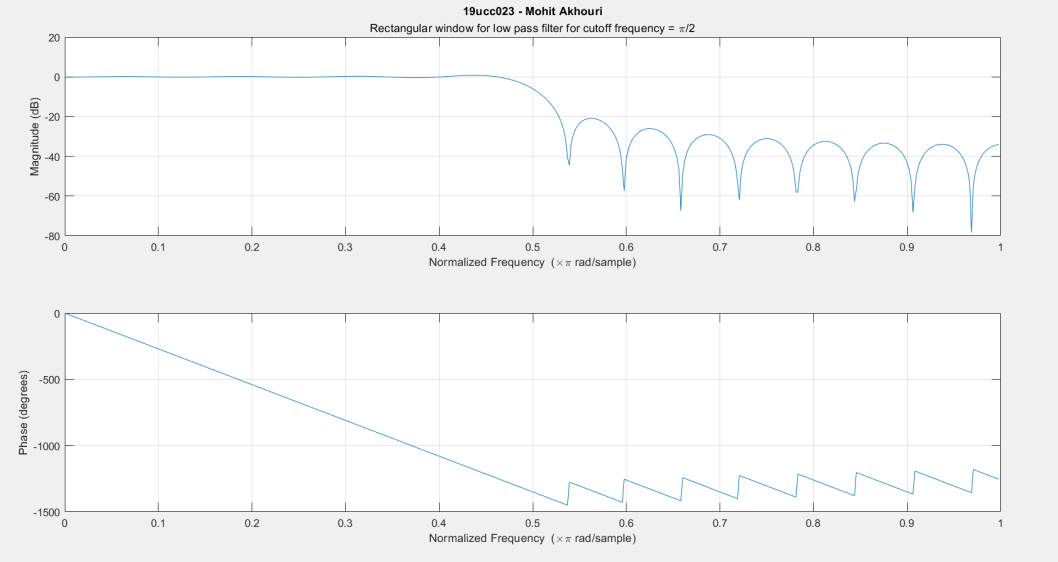

Figure 9.6 Plot of Rectangular window based Low pass filter for $w_c = \frac{\pi}{2}$

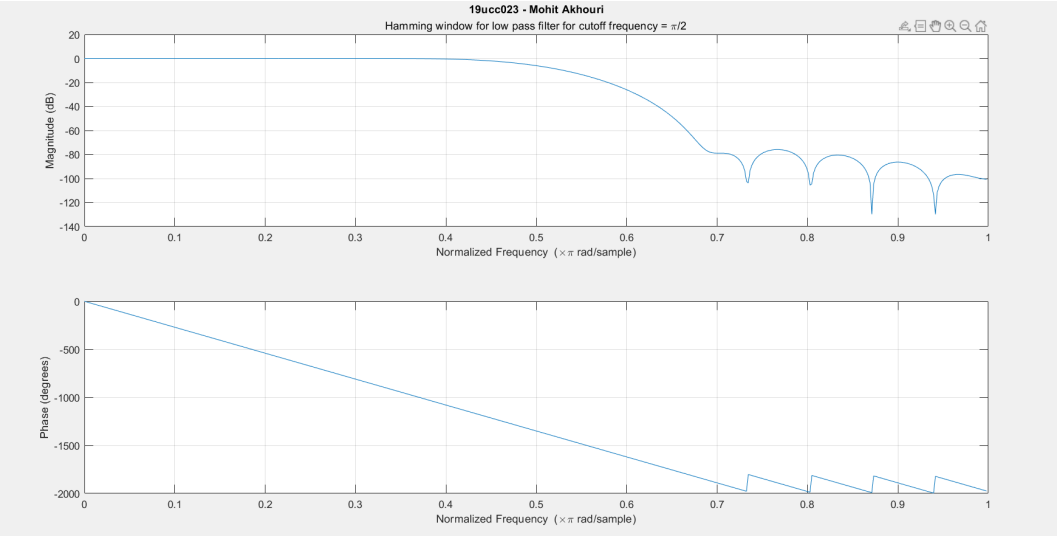

Figure 9.7 Plot of Hamming window based Low pass filter for $w_c = \frac{\pi}{2}$

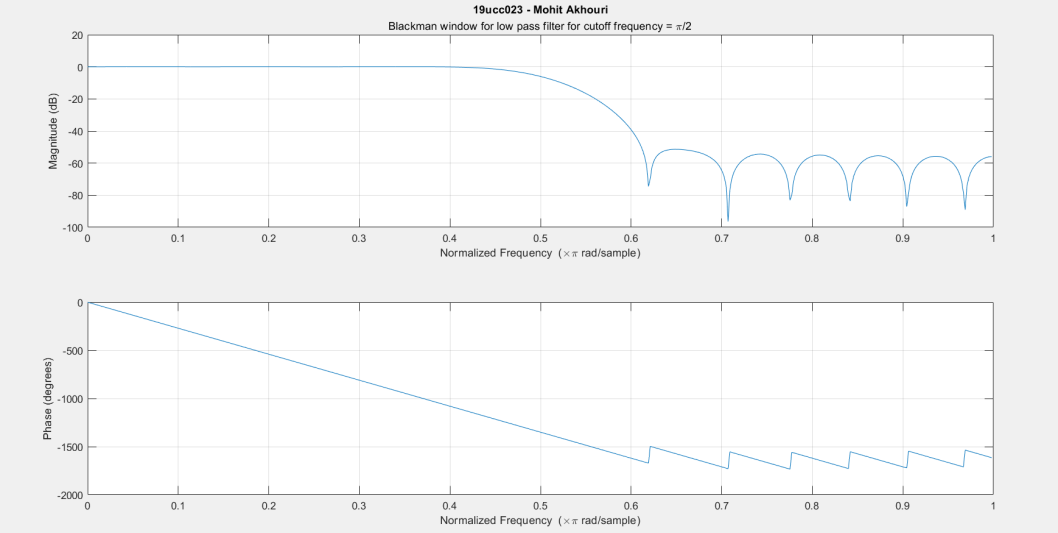

Figure 9.8 Plot of Blackman window based Low pass filter for $w_c = \frac{\pi}{2}$

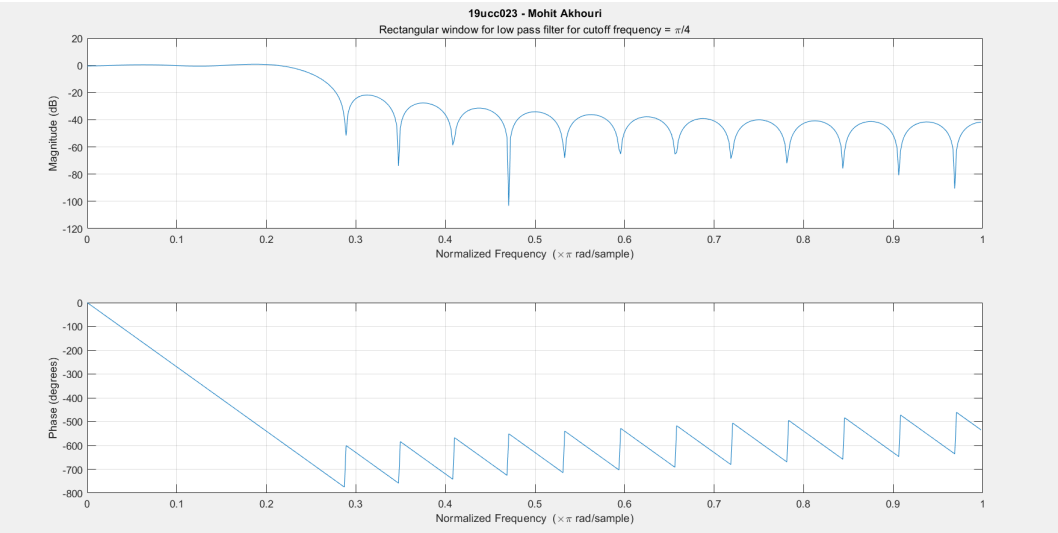

Figure 9.9 Plot of Rectangular window based Low pass filter for $w_c = \frac{\pi}{4}$

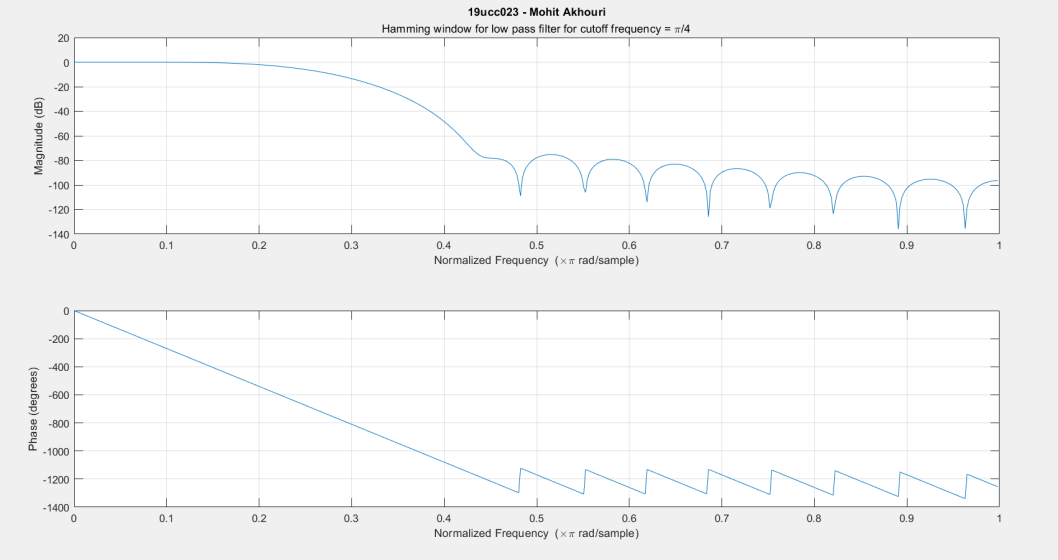

Figure 9.10 Plot of Hamming window based Low pass filter for $w_c = \frac{\pi}{4}$

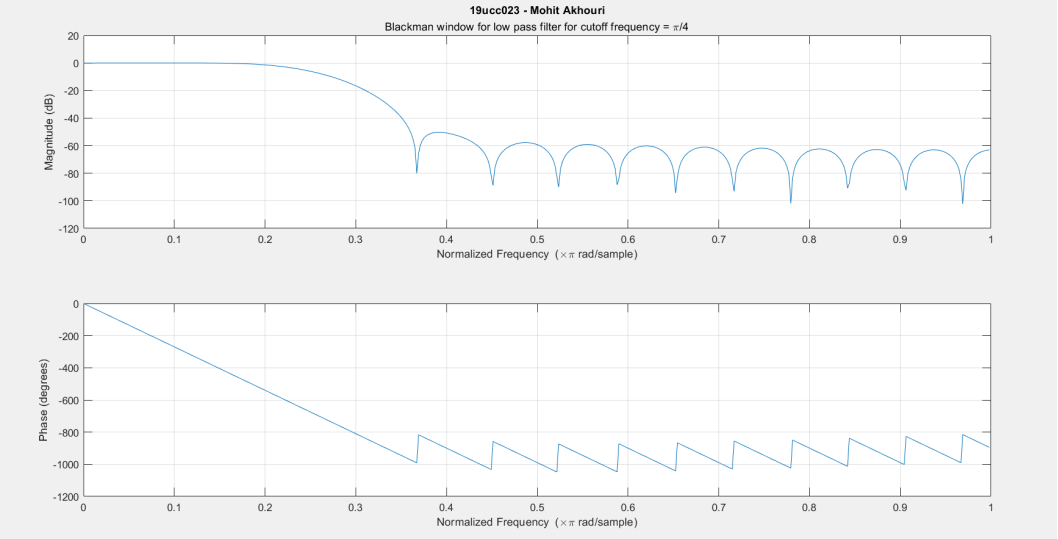

Figure 9.11 Plot of Blackman window based Low pass filter for $w_c = \frac{\pi}{4}$

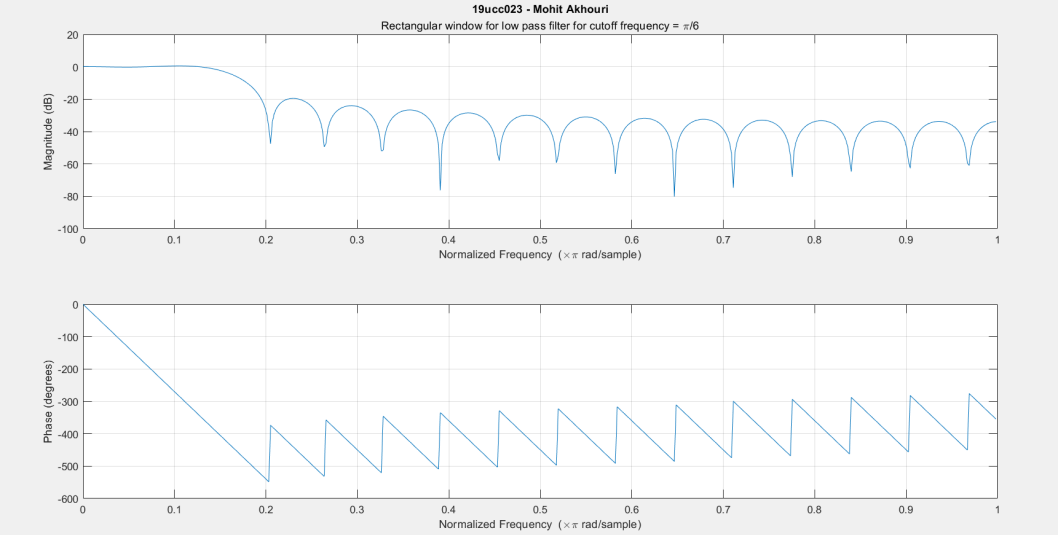

Figure 9.12 Plot of Rectangular window based Low pass filter for $w_c = \frac{\pi}{6}$

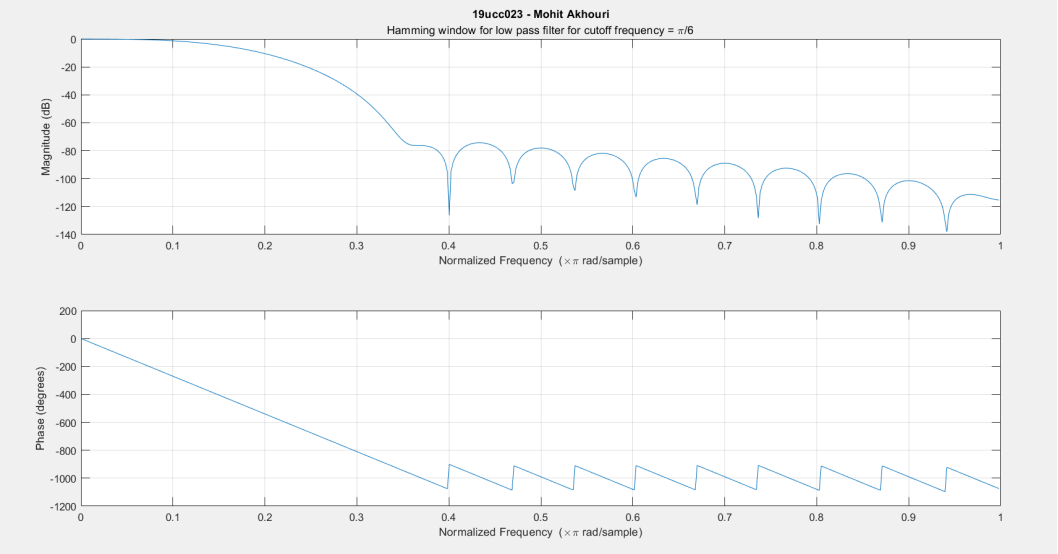

Figure 9.13 Plot of Hamming window based Low pass filter for $w_c = \frac{\pi}{6}$

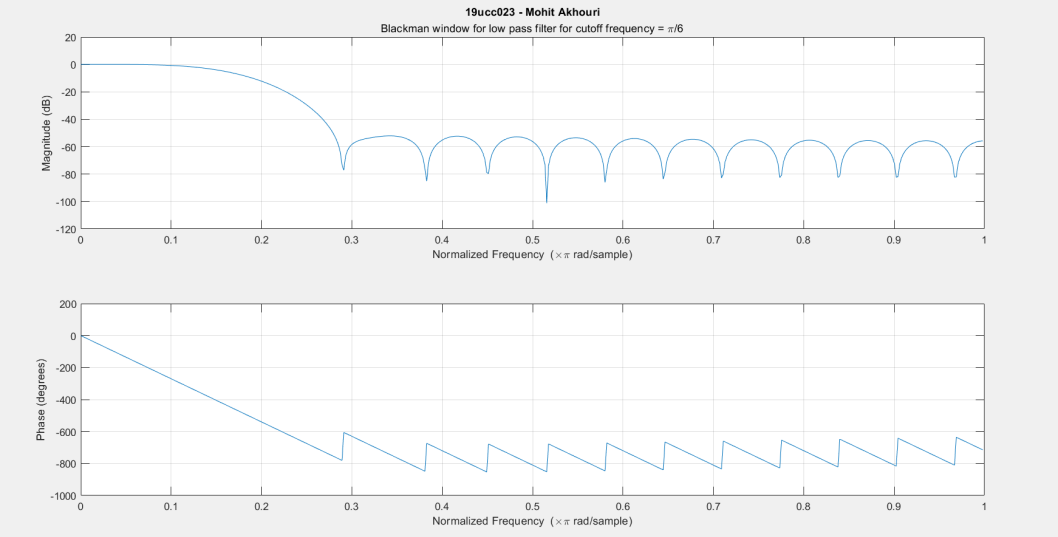

Figure 9.14 Plot of Blackman window based Low pass filter for $w_c = \frac{\pi}{6}$

9.4.2 Simulating various window based high pass FIR filter for different cutoff frequencies :

```
% 19ucc023
% Mohit Akhouri
% Experiment 9 - Observation 1b

% In this code , we will implement a high pass filter based on various
% windowing techniques - like rectangular , hamming and blackman. We
 design
% the high pass filter for various cutoff frequencies like pi/2,pi/4
 and
% pi/6. We plot the amplitude and phase response of the various high
 pass
% filter.

clc;
clear all;
close all;

M = 31; % delay in time domain

wc = pi/2; % cutoff frequency
hd = zeros(1,M); % to store the filter impulse response
w1 = zeros(1,M); % to store the window function for rectangular window
w2 = zeros(1,M); % to store the window function for hamming window
w3 = zeros(1,M); % to store the window function for blackman window

% main loop algorithm for calculation of window functions and filter
% impulse response for various cases
for n=0:M-1
    H = @(w) exp(-1j*w*(n-((M-1)/2))); % response of the high pass
 filter
    hd(n+1) = (integral(H,-pi,-wc) + integral(H,wc,pi))/(2*pi); %
 filter impulse response
    w1(n+1) = 1; % window function for rectangular window
    w2(n+1) = 0.42 - 0.5*cos((2*pi*n)/(M-1)) + 0.08*cos((4*pi*n)/
(M-1)); % window function for hamming window
    w3(n+1) = 0.54 - 0.46*cos((2*pi*n)/(M-1)); % window function for
 blackman window
end

h1 = hd .* w1; % finite impulse response of rectangular window based
 high pass filter
h2 = hd .* w2; % finite impulse response of hamming window based high
 pass filter
h3 = hd .* w3; % finite impulse response of blackman window based high
 pass filter

% Plotting the magnitude and phase responses of various high pass
 filter
figure;
freqz(h1);
title('19ucc023 - Mohit Akhouri','Rectangular window for high pass
 filter for cutoff frequency = \pi/2');
```

Figure 9.15 Part 1 of the Code for the observation 1(b)

```
grid on;

figure;
freqz(h2);
title('19ucc023 - Mohit Akhouri','Hamming window for high pass filter
 for cutoff frequency = \pi/2');
grid on;

figure;
freqz(h3);
title('19ucc023 - Mohit Akhouri','Blackman window for high pass filter
 for cutoff frequency = \pi/2');
grid on;

wc = pi/4; % cutoff frequency
hd = zeros(1,M); % to store the filter impulse response
w1 = zeros(1,M); % to store the window function for rectangular window
w2 = zeros(1,M); % to store the window function for hamming window
w3 = zeros(1,M); % to store the window function for blackman window

% main loop algorithm for calculation of window functions and filter
% impulse response for various cases
for n=0:M-1
    H = @(w) exp(-1j*w*(n-((M-1)/2))); % response of the high pass
 filter
    hd(n+1) = (integral(H,-pi,-wc) + integral(H,wc,pi))/(2*pi); %
 filter impulse response
    w1(n+1) = 1; % window function for rectangular window
    w2(n+1) = 0.42 - 0.5*cos((2*pi*n)/(M-1)) + 0.08*cos((4*pi*n)/
(M-1)); % window function for hamming window
    w3(n+1) = 0.54 - 0.46*cos((2*pi*n)/(M-1)); % window function for
 blackman window
end

h1 = hd .* w1; % finite impulse response of rectangular window based
 high pass filter
h2 = hd .* w2; % finite impulse response of hamming window based high
 pass filter
h3 = hd .* w3; % finite impulse response of blackman window based high
 pass filter

% Plotting the magnitude and phase responses of various high pass
 filter
figure;
freqz(h1);
title('19ucc023 - Mohit Akhouri','Rectangular window for high pass
 filter for cutoff frequency = \pi/4');
grid on;

figure;
freqz(h2);
title('19ucc023 - Mohit Akhouri','Hamming window for high pass filter
 for cutoff frequency = \pi/4');
grid on;
```

Figure 9.16 Part 2 of the Code for the observation 1(b)

```
figure;
freqz(h3);
title('19ucc023 - Mohit Akhouri','Blackman window for high pass filter
 for cutoff frequency = \pi/4');
grid on;

wc = pi/6; % cutoff frequency
hd = zeros(1,M); % to store the filter impulse response
w1 = zeros(1,M); % to store the window function for rectangular window
w2 = zeros(1,M); % to store the window function for hamming window
w3 = zeros(1,M); % to store the window function for blackman window

% main loop algorithm for calculation of window functions and filter
% impulse response for various cases
for n=0:M-1
    H = @(w) exp(-1j*w*(n-((M-1)/2))); % response of the high pass
 filter
    hd(n+1) = (integral(H,-pi,-wc) + integral(H,wc,pi))/(2*pi); %
 filter impulse response
    w1(n+1) = 1; % window function for rectangular window
    w2(n+1) = 0.42 - 0.5*cos((2*pi*n)/(M-1)) + 0.08*cos((4*pi*n)/
(M-1)); % window function for hamming window
    w3(n+1) = 0.54 - 0.46*cos((2*pi*n)/(M-1)); % window function for
 blackman window
end

h1 = hd .* w1; % finite impulse response of rectangular window based
 high pass filter
h2 = hd .* w2; % finite impulse response of hamming window based high
 pass filter
h3 = hd .* w3; % finite impulse response of blackman window based high
 pass filter

% Plotting the magnitude and phase responses of various high pass
 filter
figure;
freqz(h1);
title('19ucc023 - Mohit Akhouri','Rectangular window for high pass
 filter for cutoff frequency = \pi/6');
grid on;

figure;
freqz(h2);
title('19ucc023 - Mohit Akhouri','Hamming window for high pass filter
 for cutoff frequency = \pi/6');
grid on;

figure;
freqz(h3);
title('19ucc023 - Mohit Akhouri','Blackman window for high pass filter
 for cutoff frequency = \pi/6');
grid on;
```

Figure 9.17 Part 3 of the Code for the observation 1(b)

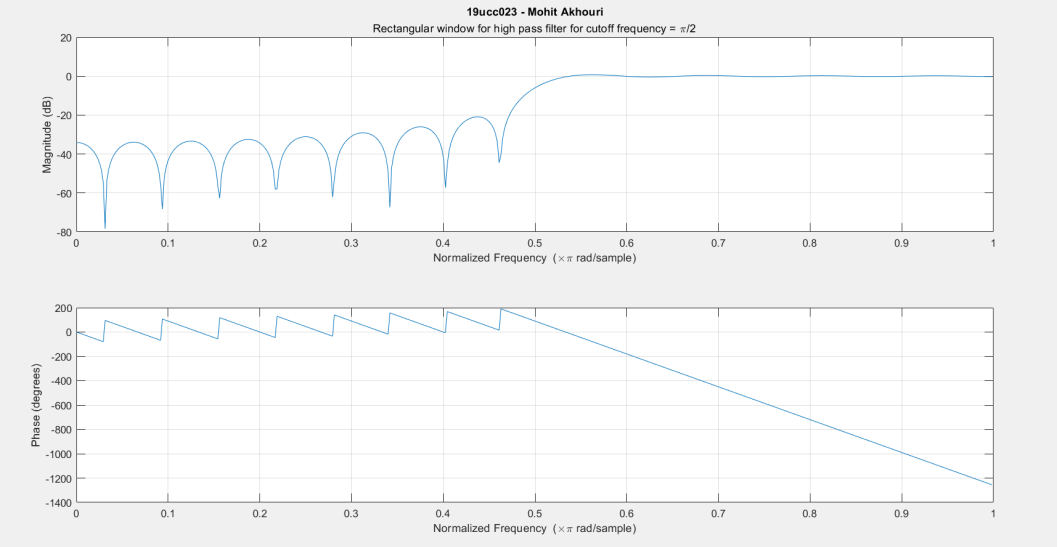

Figure 9.18 Plot of Rectangular window based High pass filter for $w_c = \frac{\pi}{2}$

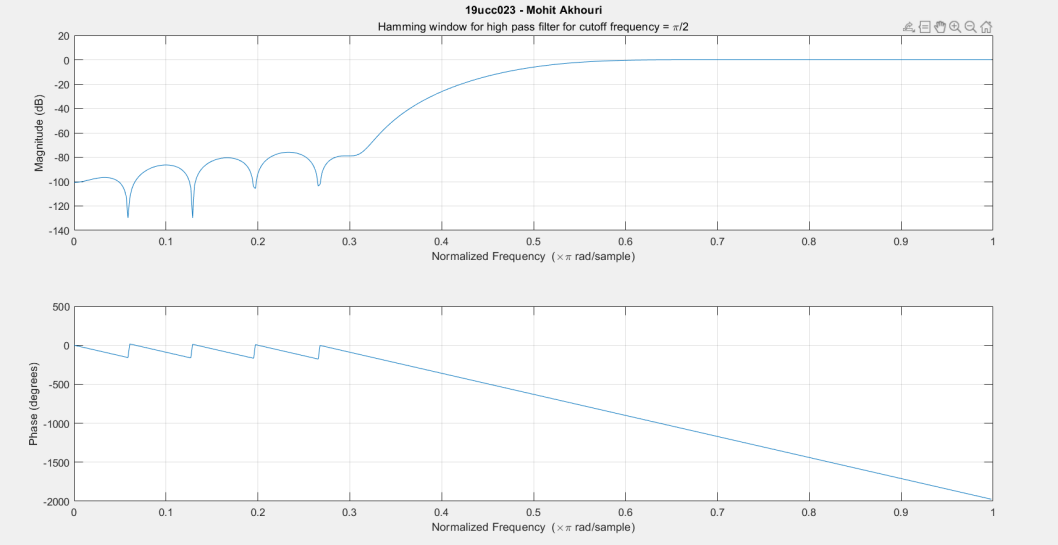

Figure 9.19 Plot of Hamming window based High pass filter for $w_c = \frac{\pi}{2}$

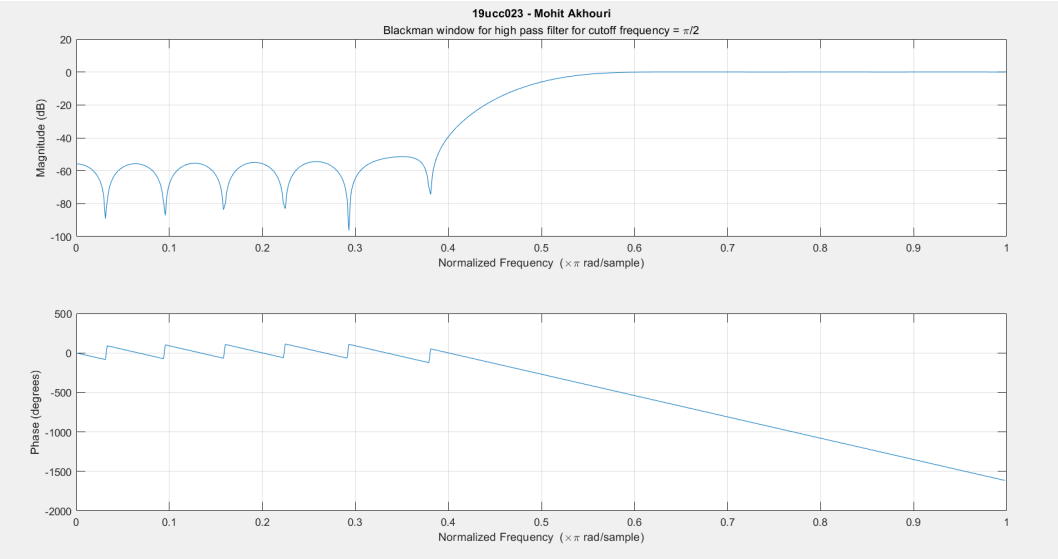

Figure 9.20 Plot of Blackman window based High pass filter for $w_c = \frac{\pi}{2}$

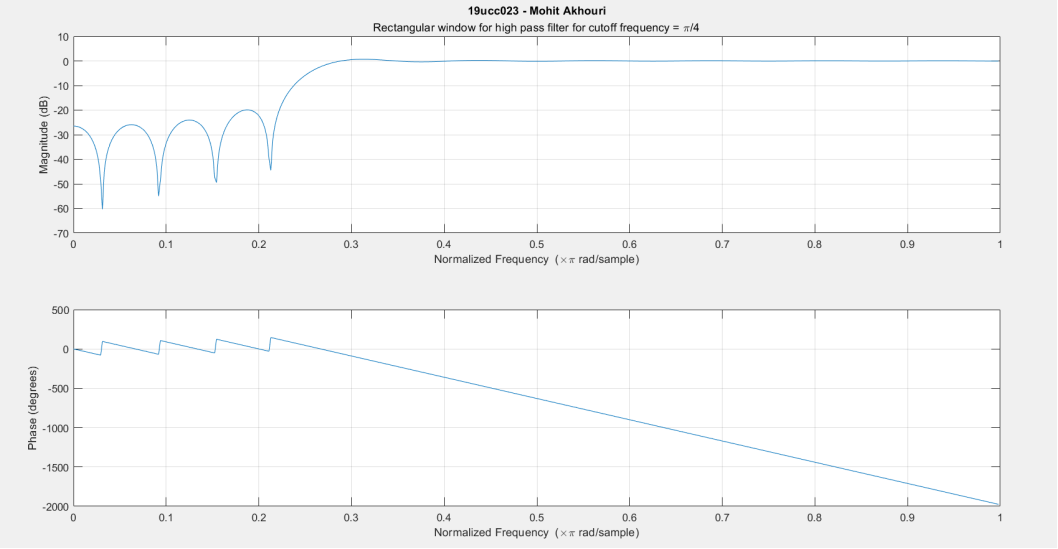

Figure 9.21 Plot of Rectangular window based High pass filter for $w_c = \frac{\pi}{4}$

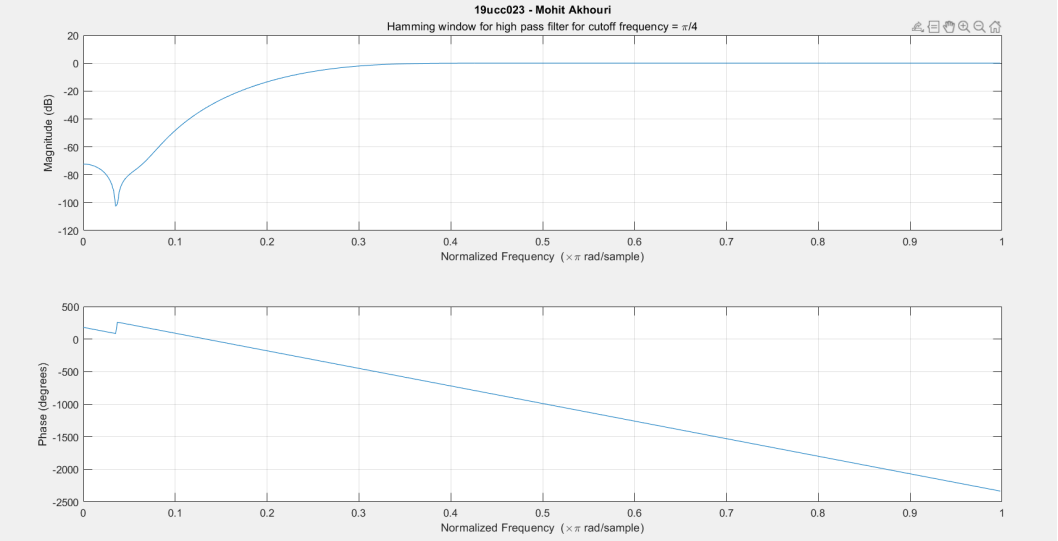

Figure 9.22 Plot of Hamming window based High pass filter for $w_c = \frac{\pi}{4}$

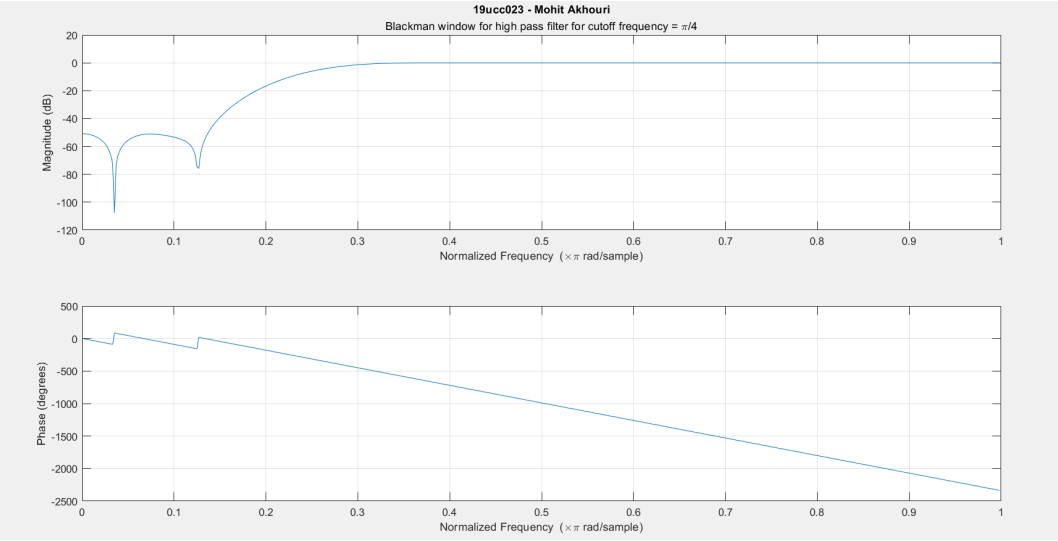

Figure 9.23 Plot of Blackman window based High pass filter for $w_c = \frac{\pi}{4}$

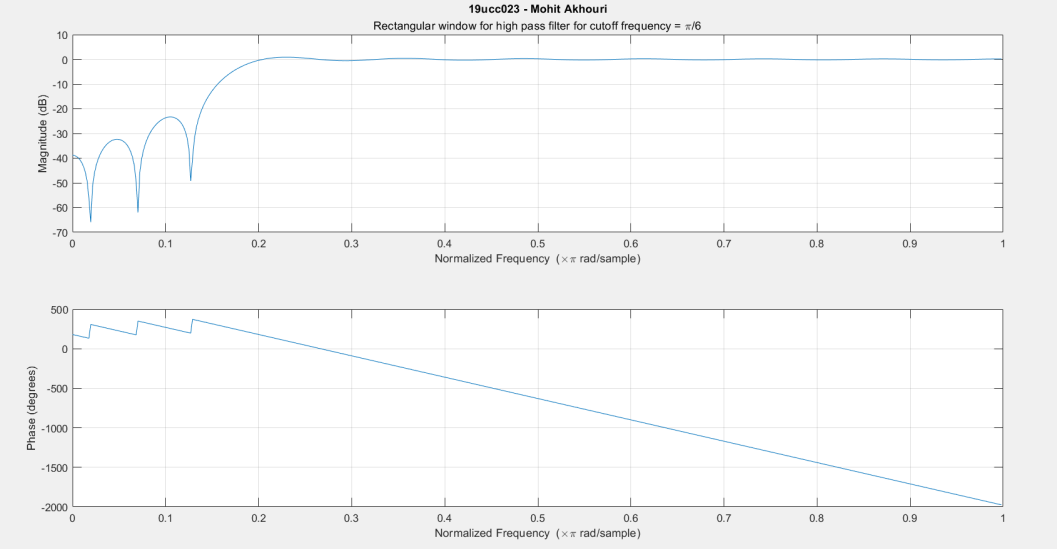

Figure 9.24 Plot of Rectangular window based High pass filter for $w_c = \frac{\pi}{6}$

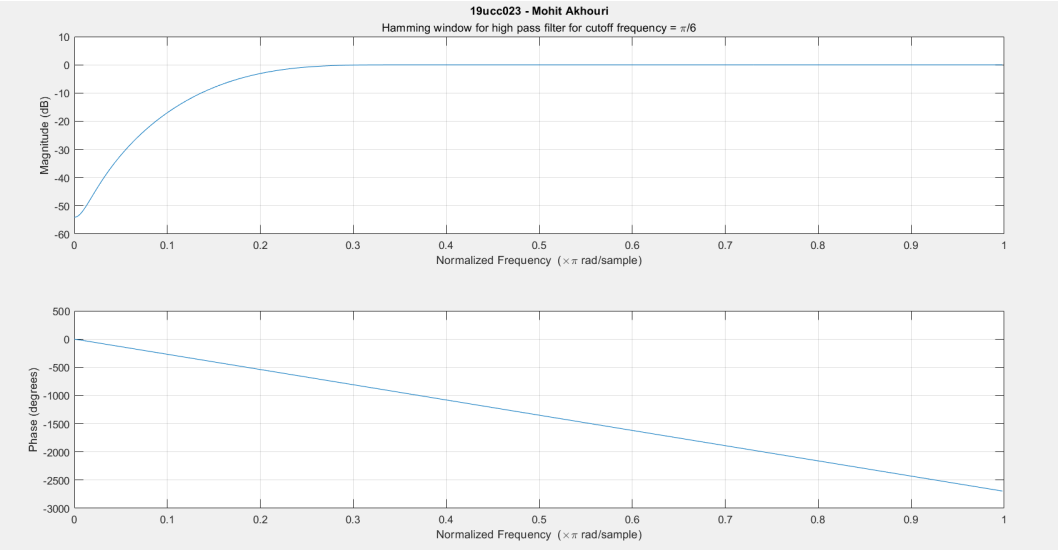

Figure 9.25 Plot of Hamming window based High pass filter for $w_c = \frac{\pi}{6}$

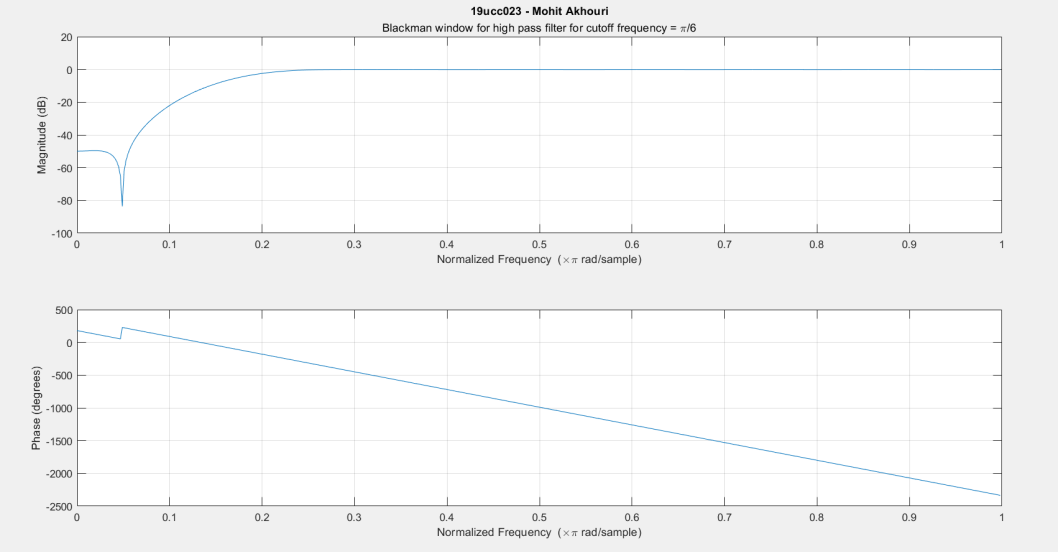

Figure 9.26 Plot of Blackman window based High pass filter for $w_c = \frac{\pi}{6}$

9.4.3 Simulating various window based band pass FIR filter for different cutoff frequencies :

```
% 19ucc023
% Mohit Akhouri
% Experiment 9 - Observation 1c

% In this code , we will implement a band pass filter based on various
% windowing techniques - like rectangular , hamming and blackman. We
 design
% band pass filter for cutoff frequencies - wc(low) = pi/4 and
 wc(high) =
% pi/2. We also plot the magnitude and phase spectrum of the designed
% filter.

clc;
clear all;
close all;

M = 31; % delay in time domain

wc_low = pi/4; % low frequency cutoff for band pass filter
wc_high = pi/2; % high frequency cutoff for band pass filter

hd = zeros(1,M); % to store the filter impulse response
w1 = zeros(1,M); % to store the window function for rectangular window
w2 = zeros(1,M); % to store the window function for hamming window
w3 = zeros(1,M); % to store the window function for blackman window

% main loop algorithm for calculation of window functions and filter
% impulse response for various cases
for n=0:M-1
    H = @(w) exp(-1j*w*(n-((M-1)/2))); % response of the band pass
 filter
    hd(n+1) = (integral(H,-wc_high,-wc_low) +
 integral(H,wc_low,wc_high))/(2*pi); % filter impulse response
    w1(n+1) = 1; % window function for rectangular window
    w2(n+1) = 0.42 - 0.5*cos((2*pi*n)/(M-1)) + 0.08*cos((4*pi*n)/
(M-1)); % window function for hamming window
    w3(n+1) = 0.54 - 0.46*cos((2*pi*n)/(M-1)); % window function for
 blackman window
end

h1 = hd .* w1; % finite impulse response of rectangular window based
 band pass filter
h2 = hd .* w2; % finite impulse response of hamming window based band
 pass filter
h3 = hd .* w3; % finite impulse response of blackman window based band
 pass filter

% Plotting the magnitude and phase responses of various band pass
 filter
figure;
freqz(h1);
```

Figure 9.27 Part 1 of the Code for the observation 1(c)

```
title('19ucc023 - Mohit Akhouri','Rectangular window for band pass
 filter for cutoff frequency as w_{c}(low) = \pi/4 and w_{c}(high) =
 \pi/2');
grid on;

figure;
freqz(h2);
title('19ucc023 - Mohit Akhouri','Hamming window for band pass filter
 for cutoff frequency as w_{c}(low) = \pi/4 and w_{c}(high) = \pi/2');
grid on;

figure;
freqz(h3);
title('19ucc023 - Mohit Akhouri','Blackman window for band pass filter
 for cutoff frequency as w_{c}(low) = \pi/4 and w_{c}(high) = \pi/2');
grid on;
```

Published with MATLAB® R2020b

Figure 9.28 Part 2 of the Code for the observation 1(c)

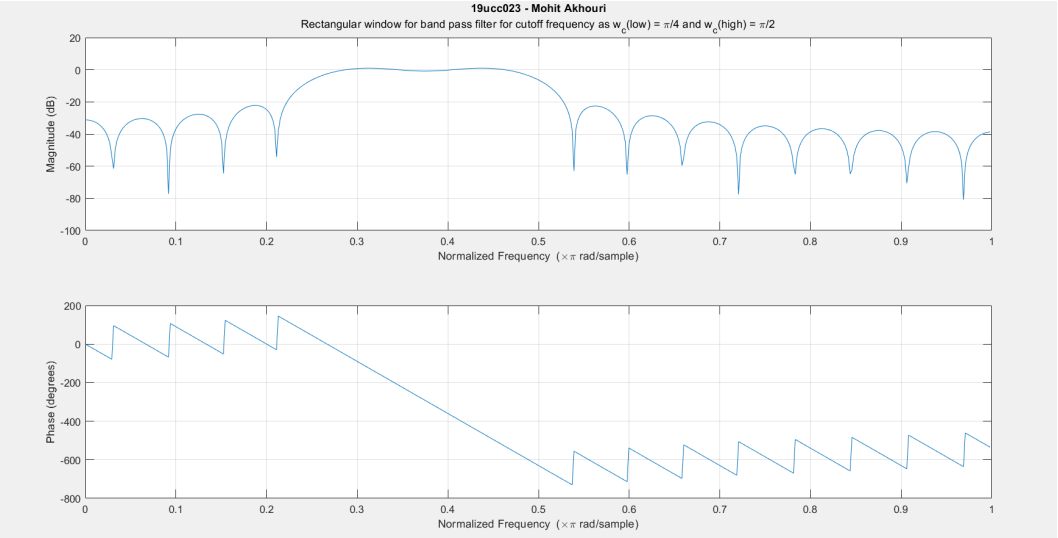

Figure 9.29 Plot of Rectangular window based Band pass filter for w_c(low) = $\frac{\pi}{4}$ and w_c(high) = $\frac{\pi}{2}$

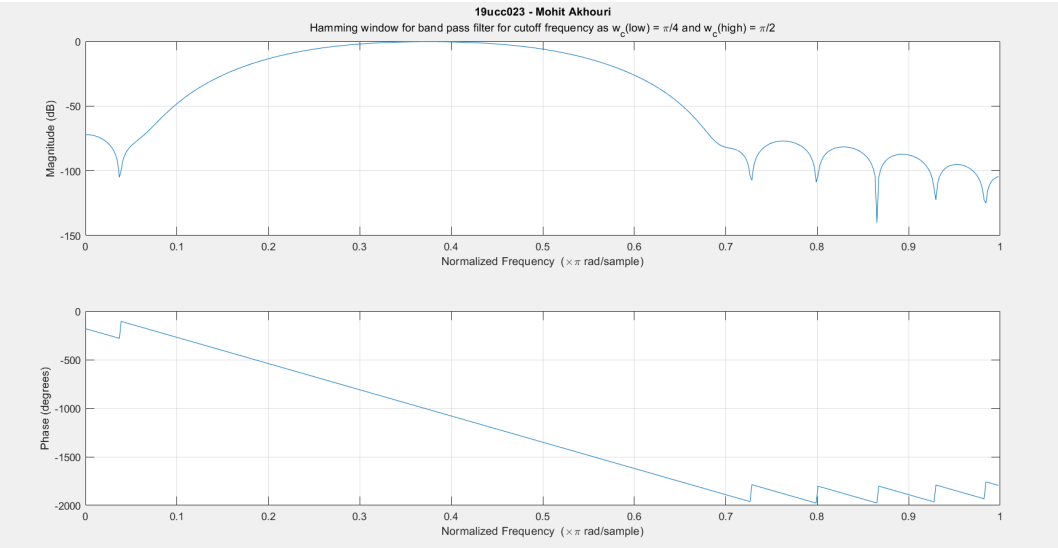

Figure 9.30 Plot of Hamming window based Band pass filter for w_c(low) = $\frac{\pi}{4}$ and w_c(high) = $\frac{\pi}{2}$

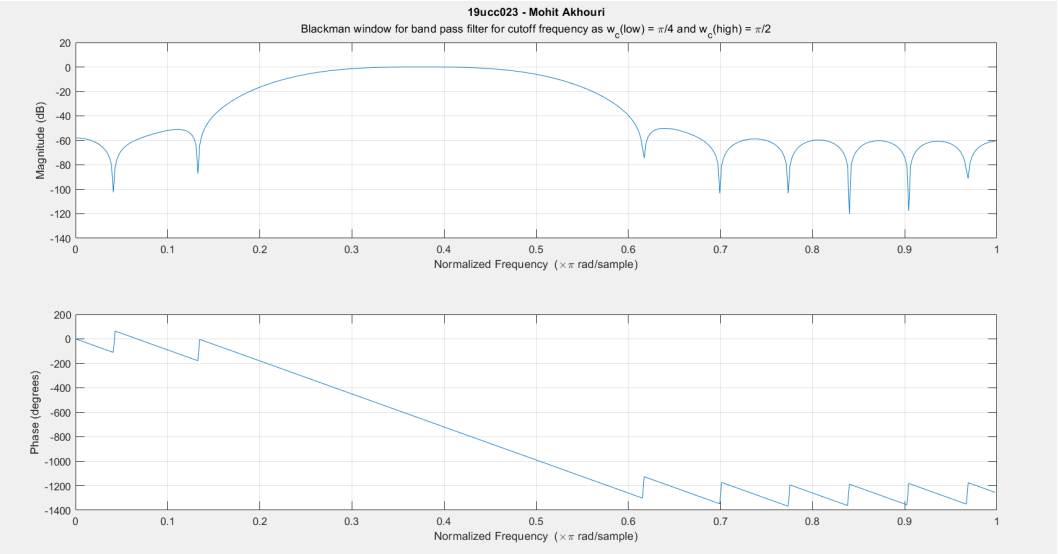

Figure 9.31 Plot of Blackman window based Band pass filter for w_c(low) = $\frac{\pi}{4}$ and w_c(high) = $\frac{\pi}{2}$

9.4.4 Removing the hissing sound using ideal low pass filter :

```
% 19ucc023
% Mohit Akhouri
% Experiment 9 - Observation 2

% This code will take the input of an audio signal with hissing
 component
% frequency greater than pi/2. Next it will design a low pass filter
 with
% frequency spectrum as : value '1' for all samples except for 66th
 and
% 192th samples , i.e. we design a IDEAL LOW PASS FILTER

% Finally we will divide the audio file into chunks of 1x256 and find
 the
% DFT for each of the chunk. Let it be X(w) . Finally we multiply the
% filter impulse response H(w) with X(w) to get the smoothened version
 of
% audio signal with hissing sound removed.

[x,fs] = audioread('inputwithhissgreaterthanpiby2.wav'); % reading an
 audio file
x = x'; % Taking the transpose to convert from Nx1 to 1xN for easy
 multiplication later on
x_length = length(x); % Finding length of audio signal x[n]

% plot of the original audio signal x[n] + unwanted components (hiss
 sound)
figure;
plot(x);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of original audio signal x[n] +
 hissing frequency components');
grid on;

samples = 256; % variable to store the sample number for partition of
 audio file
hw = ones(1,256); % ideal low pass filter impulse response

% making the 66th and 192th sample number = 0
hw(66) = 0;
hw(192) = 0;

% plot of the impulse response of the IDEAL LOW PASS FILTER
figure;
plot(hw,'Linewidth',1.5);
xlabel('frequency (radians/sec) ->');
ylabel('H(\omega) ->');
title('19ucc023 - Mohit Akhouri','Impulse response H(\omega) of an
 IDEAL LOW PASS FILTER for removing hissing frequency components');
grid on;
```

Figure 9.32 Part 1 of the Code for the observation 2

```
% main loop algorithm for dividing the audio file into chunks and
% individually applying fft on each chunk and multiplying with filter
% impulse response to get the smoothened version of audio signal
for i=1:samples:x_length
    x_sampled = x(i:i+255); % taking 256 samples chunk
    x_sampled_fft = fft(x_sampled,256); % taking DFT of the 256
 samples
    yw = x_sampled_fft .* hw; % filtering process takes place
    x(i:i+255) = ifft(yw); % rewriting the original audio file with
 smoothened version of audio
end

x = x'; % taking transpose back again to convert to original form
 ( Nx1 )

% Plot of the smoothened version of audio signal x[n]
figure;
plot(x);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','Plot of smoothened audio signal
 after removing the hissing frequency components');
grid on;

sound(x,fs); % listening to the smoothened version of the audio signal
 after passing through IDEAL LOW PASS FILTER
audiowrite('Exp9_obs_2_low_pass_filter_output.wav',x,fs); % Writing
 the final audio signal to an audio file
```

Published with MATLAB® R2020b

Figure 9.33 Part 2 of the Code for the observation 2

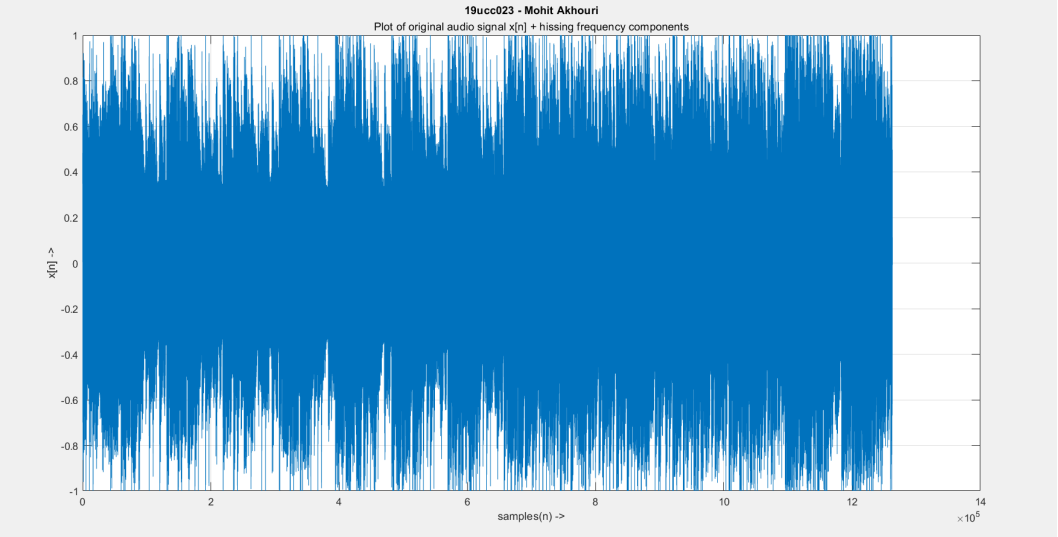

Figure 9.34 Plot of Original Audio Signal + hissing sound

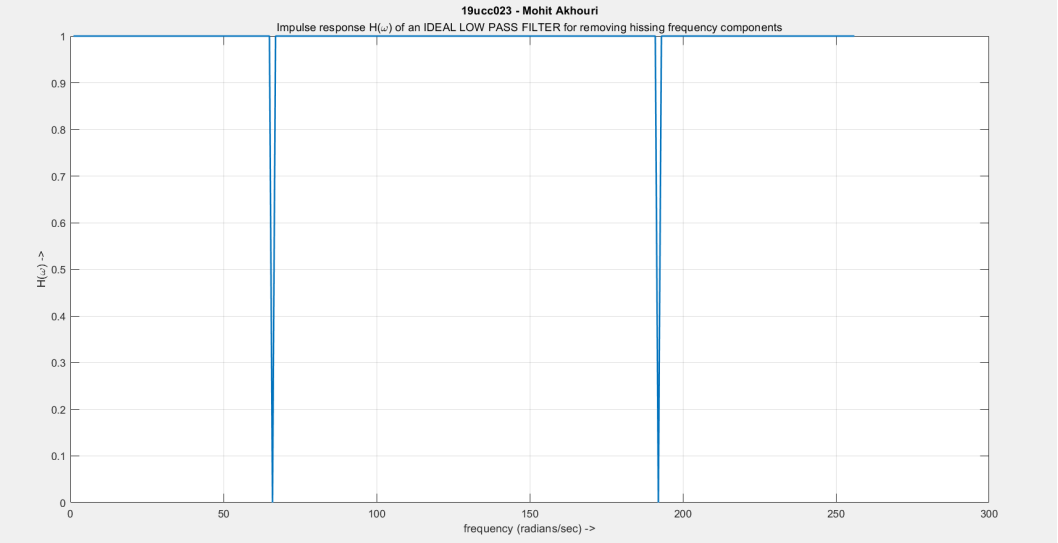

Figure 9.35 Plot of Frequency response of Ideal Low Pass Filter

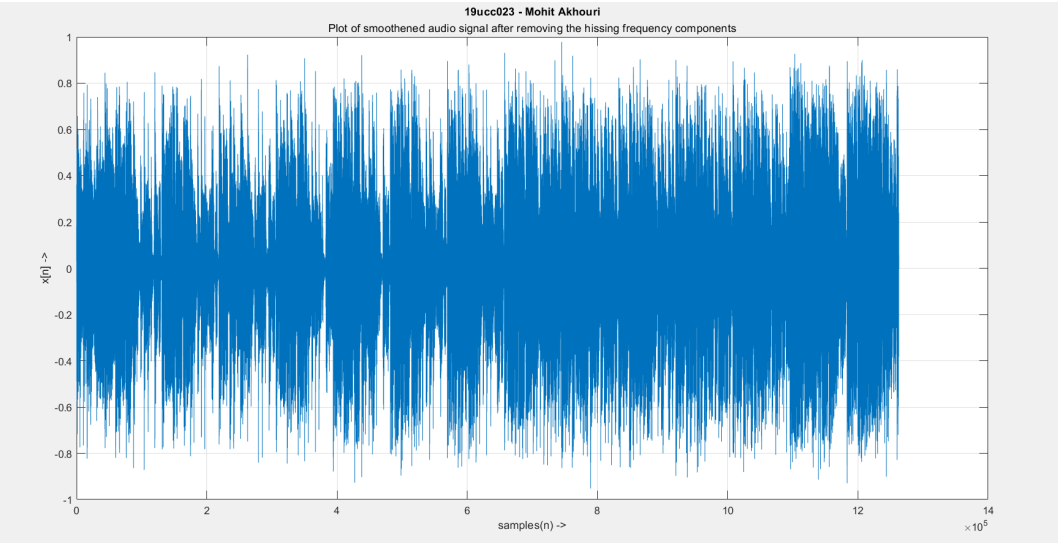

Figure 9.36 Plot of Smoothened audio signal after removing hissing sound

9.4.5 Removing the hissing sound using notch filter :

```
% 19ucc023
% Mohit Akhouri
% Experiment 9 - Observation 3

% This code will read an audio signal 'inputwithhissmadelaptop.wav'
 and
% sample it using 32 Khz. We will take the first 8 samples of this
 audio
% signal and try to locate the hissing frequencies ( approximately at
 3rd
% and 7th samples ).

% Next we will design a NOTCH FILTER and convolve it with distorted
 signal.
% What we finally get is a smoothened version of the audio signal with
% hissing removed and some distortion still left.

clc;
clear all;
close all;

[x,fs] = audioread('inputwithhissmadelaptop.wav'); % reading an audio
 file

samples_x = x(1:8); % Taking first 8 samples of audio signal x[n]
spectrum_sample = abs(fftshift(fft(samples_x))); % Finding the
 frequency spectrum of the first 8 samples for locating hissing
 frequencies

% Plot of the frequency spectrum of the first 8 samples
figure;
stem(spectrum_sample,'Linewidth',1.5);
xlabel('frequency (radians/sec) ->');
ylabel('X(\omega) ->');
title('19ucc023 - Mohit Akhouri','Frequency Spectrum of the first
 8 samples of input x[n] - hissing frequencies at 3^{rd} and 7^{th}
 samples');
grid on;

hn = [1 -2*cos(pi/2) 1]; % Filter impulse response for NOTCH FILTER
x = conv(x,hn); % convolution of distorted signal x[n] and impulse
 response of NOTCH FILTER h[n]

% Plots of input signal x[n] for first 8 samples , impulse response
 h[n]
% and convolved signal y[n].
figure;
subplot(3,1,1);
stem(samples_x,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('Plot of the first 8 samples of input signal x[n]');
```

Figure 9.37 Part 1 of the Code for the observation 3

```
grid on;
subplot(3,1,2);
stem(hn,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('h[n] ->');
title('impulse response h[n] of the NOTCH FILTER used for
 convolution');
grid on;
subplot(3,1,3);
stem(x(1:8),'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('Plot of the first 8 samples of input signal x[n] after
 convolution with NOTCH FILTER');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% Plot of the smoothened version of audio signal after removing the
 hissing
% frequencies
figure;
plot(x);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('19ucc023 - Mohit Akhouri','smoothened version of the audio
 signal x[n] after passing through NOTCH FILTER');
grid on;

sound(x,fs); % listening to the smoothened version of the audio signal
 after passing through NOTCH FILTER
audiowrite('Exp9_obs_3_notch_filter_output.wav',x,fs); % Writing the
 final audio signal to an audio file
```

Published with MATLAB® R2020b

Figure 9.38 Part 2 of the Code for the observation 3

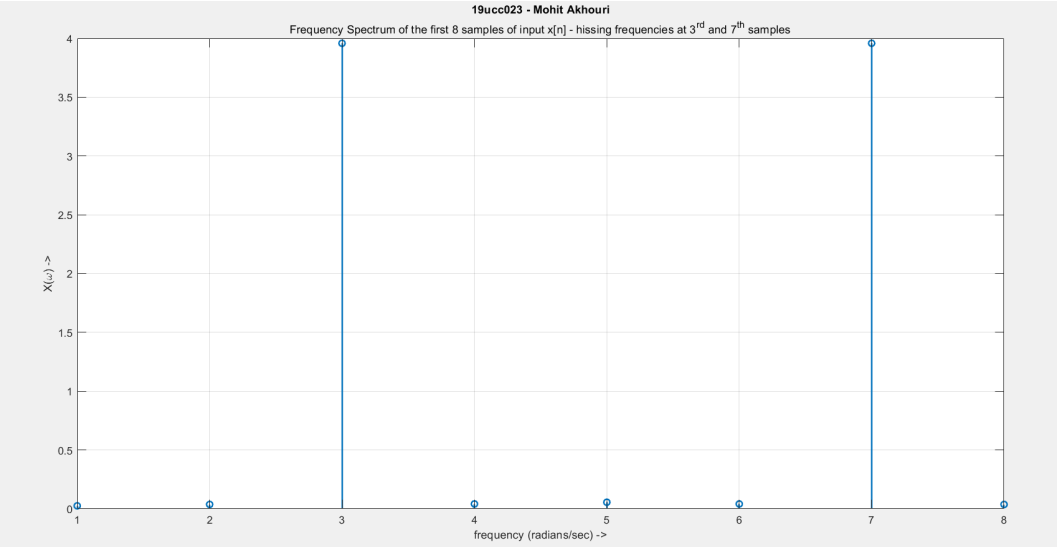

Figure 9.39 Frequency spectrum of the first 8 samples of input audio signal x[n]

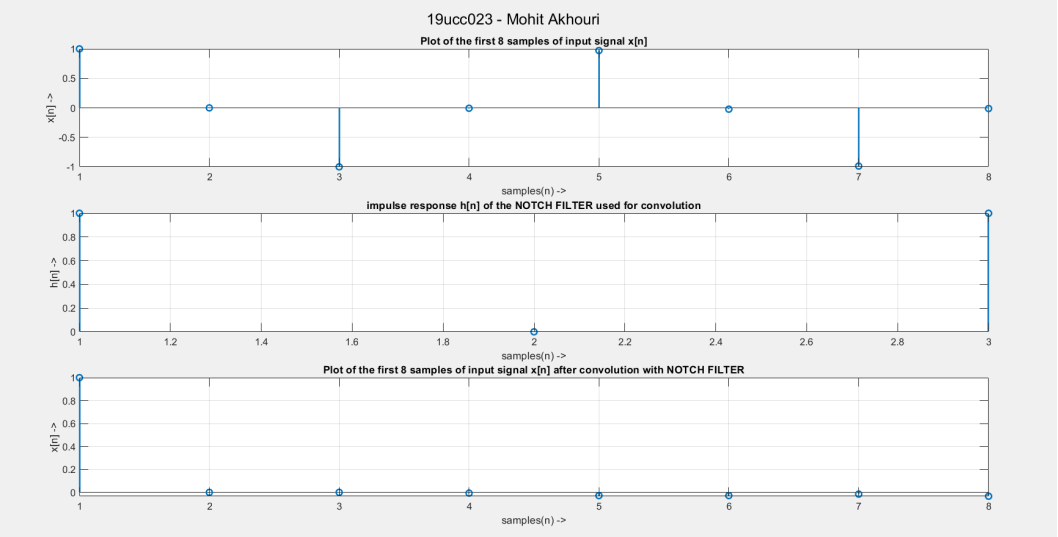

Figure 9.40 Plots for the filtering process via convolution with notch filter

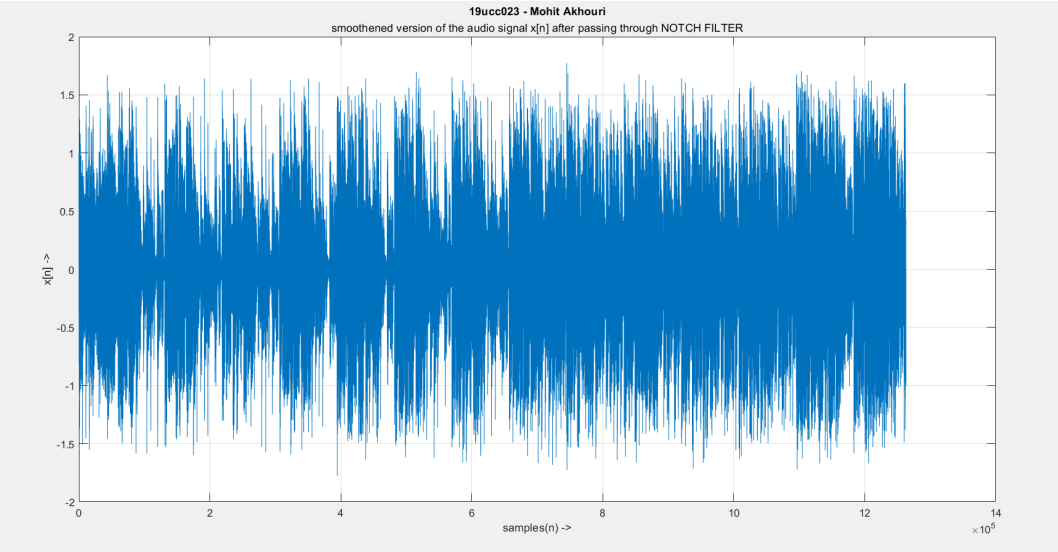

Figure 9.41 Plot of Smoothened audio signal after removing hissing sound

9.4.6 Ideal Low pass filter design and Notch filter design in Simulink :

```
% 19ucc023
% Mohit Akhouri
% Experiment 9 - Observation 4

% This code will call the simulink model and perform the filtering
% operations on the given audio signal with the help of IDEAL LOW PASS
% FILTER and NOTCH FILTER

sim('Simulink_Observation_5'); % calling the simulink model
fs = 32000; % Sampling frequency of audio signal

% Uncomment the below lines if you want to hear the audio signals
% sound(out.input_lowpass_filter.data,fs);
% sound(out.output_lowpass_filter.data,fs);

% Plots of input audio signal to LOWPASS FILTER and corresponding
 output
% obtained via SIMULINK MODEL
figure;
subplot(2,1,1);
plot(abs(fftshift(fft(out.input_lowpass_filter.data))));
xlabel('frequency (radians/sec) ->');
ylabel('X(\omega) ->');
title('Frequency spectrum of original input audio signal with hiss
 frequency greater than \pi/2');
grid on;
subplot(2,1,2);
plot(abs(fftshift(fft(out.output_lowpass_filter.data))));
xlabel('frequency (radians/sec) ->');
ylabel('Y(\omega) ->');
title('Frequency spectrum of smoothened output audio signal after
 passing through LOW PASS FILTER');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% Uncomment the below lines if you want to hear the audio signals
% sound(out.input_notch_filter.data,fs);
% sound(out.output_notch_filter.data,fs);

% Plots of input audio signal to NOTCH FILTER and corresponding output
% obtained via SIMULINK MODEL
figure;
subplot(2,1,1);
plot(abs(fftshift(fft(out.input_notch_filter.data))));
xlabel('frequency (radians/sec) ->');
ylabel('X(\omega) ->');
title('Frequency spectrum of original input audio signal with hiss
 frequency present at 3^{rd} and 7^{th} samples');
grid on;
subplot(2,1,2);
plot(abs(fftshift(fft(out.output_notch_filter.data))));
xlabel('frequency (radians/sec) ->');
```

Figure 9.42 Part 1 of the Code for the observation 4

```
ylabel('Y(\omega) ->');
title('Frequency spectrum of smoothened output audio signal after
 passing through NOTCH FILTER');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Published with MATLAB® R2020b

Figure 9.43 Part 2 of the Code for the observation 4

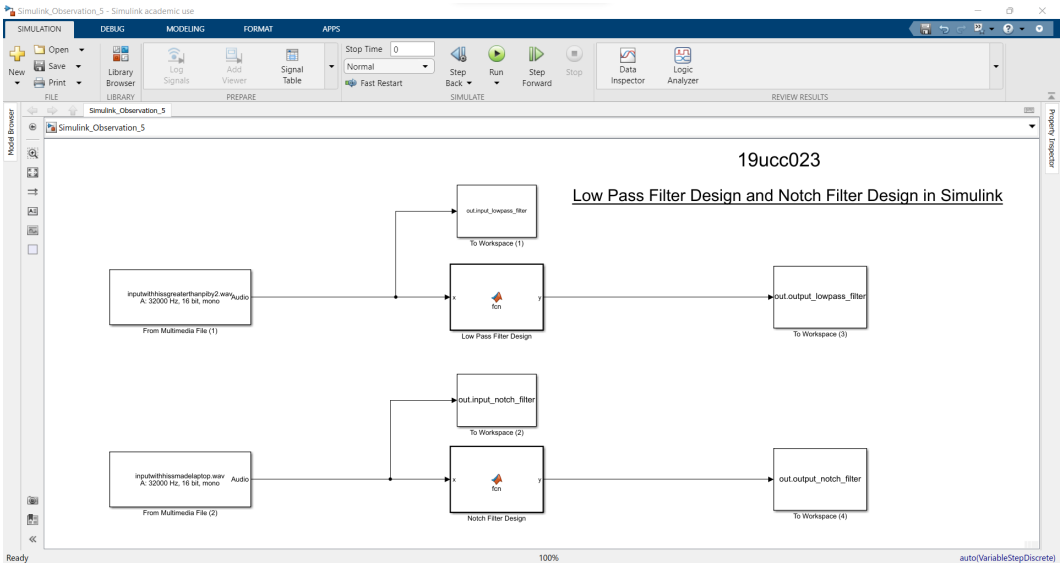

Figure 9.44 Simulink Model used for Filter Design

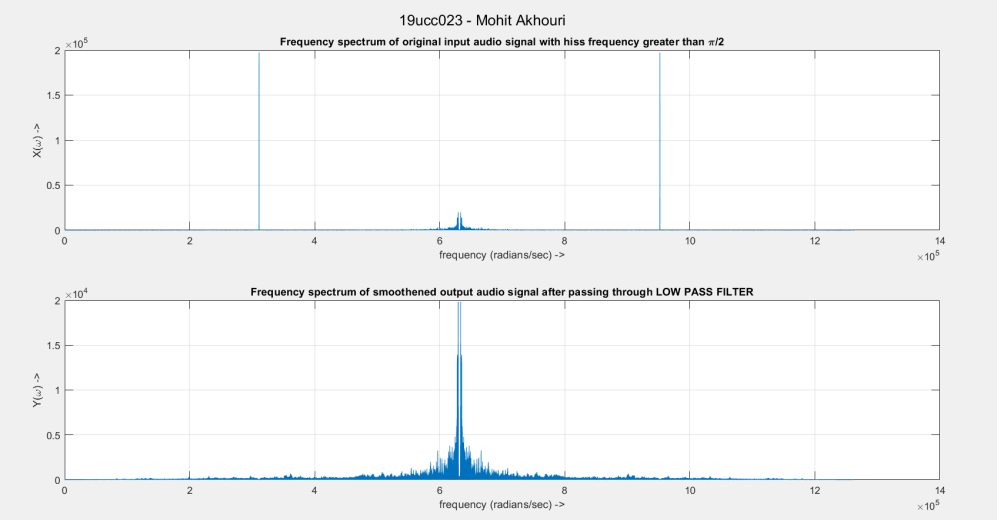

Figure 9.45 Frequency spectrums of original and smoothened audio signal for ideal low pass filter

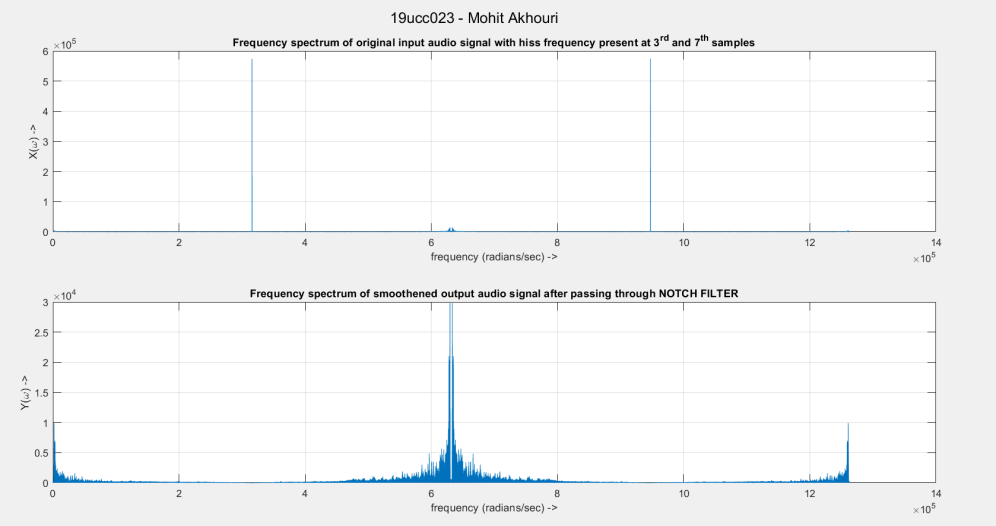

Figure 9.46 Frequency spectrums of original and smoothened audio signal for notch filter

9.5 Conclusion

In this experiment , we learnt the concepts of **Windowing based filter design** , **Low pass filter design** and **Notch filter design** of Digital Signal Processing. We learnt about different types of Window functions like - **rectangular window**,**hamming window** and **Blackman window**. We learnt how to design high,low and band pass filters using **window method**. We also learnt about the concept of **hissing sound** in audio signal and how to remove it. We design two types of filters - **Ideal low pass filter** and **Notch filter** for removal of hissing sound. We learnt about new MATLAB functions like **integral**, **audioread**, **sound** and **audiowrite**. We designed different types of filters for different cutoff frequencies like - $\pi/2$, $\pi/4$ and $\pi/6$. We also implemented the filter design concept in Simulink and compared the results obtained via MATLAB coding.