****

**NATIONAL INSTITUTE OF TECHNOLOGY, CALICUT**

**DIGITAL SIGNAL PROCESSING**

**LAB-5**

**FIR FILTER DESIGN USING WINDOWING**

**Submitted by**

**B. Veda Sree -B150734EC**

**B. Sravani – B150783EC**

**C. Vamsi Narayana Reddy-B150530EC**

**Ch. Lakshmi Priyanka – B150837EC**

**AIM:**

To implement the design FIR filters with the given specifications in frequency domain using Windowing in MATLAB.

**THEORY:**

**FIR Filters:**

A Finite impulse response filter is a filter whose impulse response is of finite duration. This is in contrast to IIR filters whose impulse response is of infinite duration.

**FIR vs IIR FILTERS:**

FIR Filters have both advantages as well as disadvantages when compared to IIR filters.

FIR filters have following primary advantages:

1.The symmetry in impulse response ensures linear phase response of FIR filters. Hence, FIR filters are widely used in applications with linear phase requirement.

2.They are always stable. All the poles of FIR filters are at origin in z-plane (inside the unit circle).

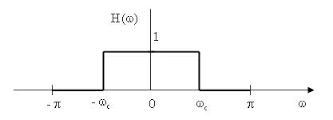
3.They can be realized efficiently in hardware.

The primary disadvantage of FIR filters is that they often require a much higher filter order than IIR filters to achieve a given level of performance. Correspondingly, delay of these filters is much higher than that of an equal performance IIR filter.

**FIR FILTER DESIGN USING WINDOWING:**

Consider an ideal low pass filter with impulse response H(w) and cut-off frequency wc

H(w) = 



The impulse response of this ideal low pass filter is



* The impulse response is a sinc function which is infinite and non-causal. Hence, it cannot be implemented practically. So, we shift the impulse response and truncate it by applying a suitable window. By truncating it we allow certain approximation error.
* If h\_ideal[n] is the impulse response of ideal filter and w[n] is the impulse response of window, then h\_designed[n] = h\_ideal[n]. w[n]
* Since time domain multiplication implies frequency domain convolution, the frequency response of designed filter will be convolution of ideal frequency response with the frequency response of window. Hence the designed filter will have some ripples in pass band as well as stop band.
* Different windows like Rectangular window, Bartlett window, Hann window, Hamming window etc., can be used for windowing. The choice of window is based on the stop band attenuation or other specifications.

**Procedure followed for FIR filter design using windowing:**

1.Choose appropriate window based on stop band attenuation.

2.Estimate the order of the filter N by equating transition band width of the desired filter to Main lobe width of the window chosen.

3. Obtain the cutoff frequency of ideal filter from given specifications and hence the ideal filter impulse response.

4.Multiply the ideal response with a suitable window and obtain its Frequency response.

**PROBLEM:**

1. **Design a Low pass filter with the following specifications: Compute the order of the filter based on the transition width. Choose the window based on the stop band attenuation.**

Pass band edge frequency = 1000Hz; Stop band edge frequency = 1500Hz

Minimum stop band attenuation = 50 dB;

Maximum pass band attenuation = 0.9 dB

Sampling Frequency = 8000Hz

Plot the impulse response, magnitude spectrum and phase spectrum.

**MATLAB CODE:**

function [hw]=lpf()

clear all

clc

fpass=1000;

fstop=1500;

fs=8000;

d1=0.9;

d2=50;

w=(fpass+fstop)\*2\*pi/(2\*fs);

tb=(fstop-fpass)\*2\*pi/fs;

N=(8\*pi)/tb;

h=zeros(1,N);

for n=1:N

h(n)=sin(w\*(n-ceil(N/2)))/(pi\*(n-ceil(N/2)));

end

h(ceil(N/2))=w/pi;

figure

stem(h)

xlabel('n');

ylabel('h(n)');

title('LPF impulse response without windowing');

fvtool(h);

hold on;

title('LPF without windowing');

for x=1:N

w(x)=0.54-0.46\*(1-cos((2\*pi\*x/(N-1))));

end

figure

stem(w);

title('Hamming window');

xlabel('n');

ylabel('w(n)');

hw=(h.\*w);

figure

stem(hw)

xlabel('n');

ylabel('h(n)');

title('Impulse response after windowing');

fvtool(hw);

xlabel('n');

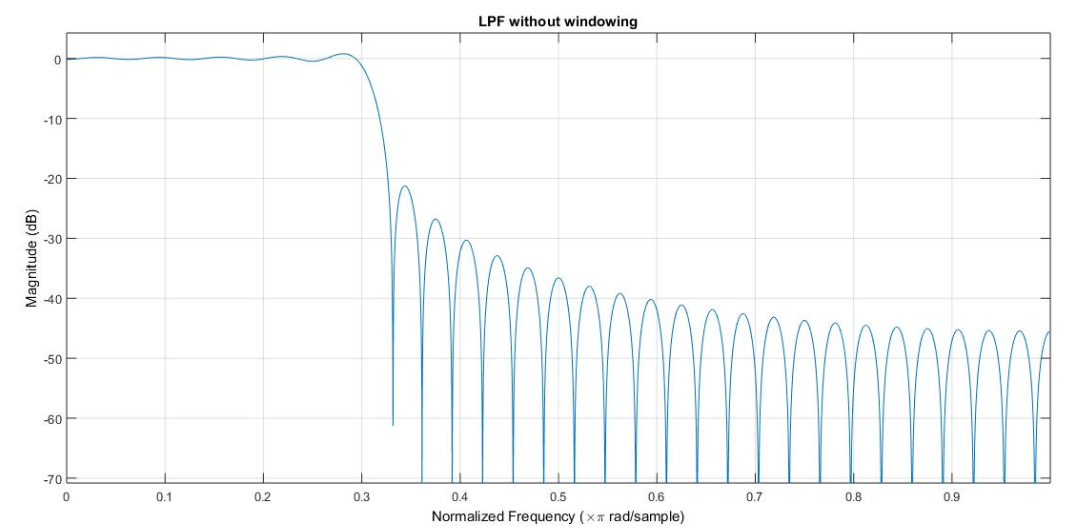
ylabel('h(n)');

title('LPF after windowing');

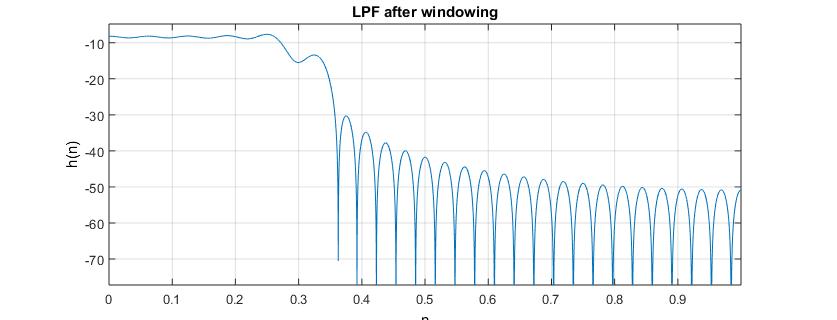
**PLOTS:**

**Magnitude response:**

1. **LPF without windowing**

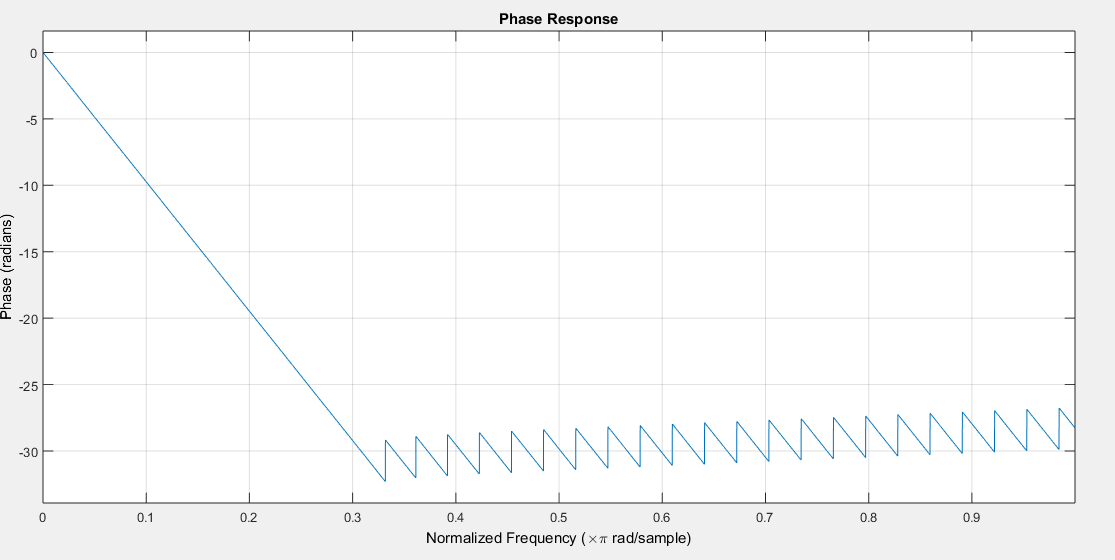


1. **LPF windowing**

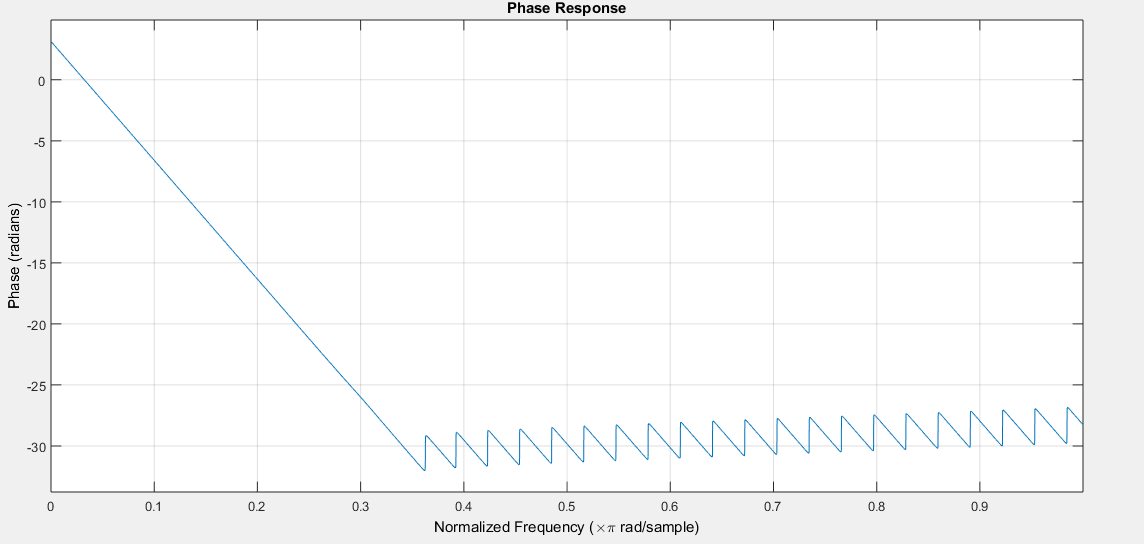


**Phase response:**

1. **Phase response without windowing**

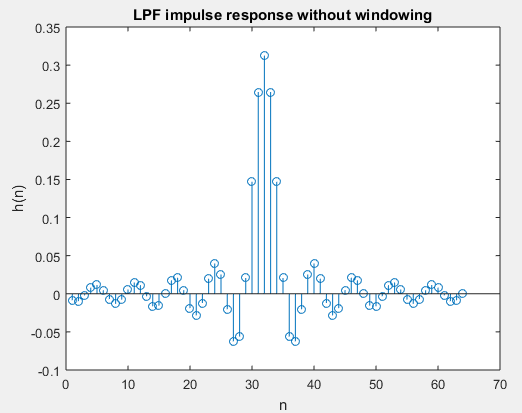
****

1. **phase response with windowing**

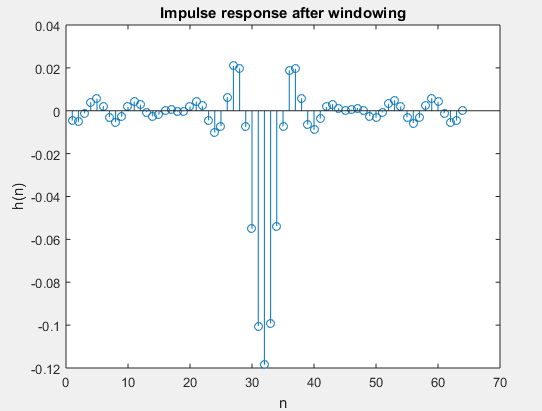
****

**Impulse response:**

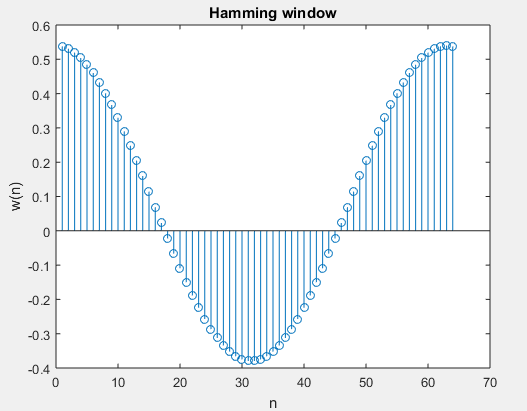
1. **Impulse response without windowing**



1. **Impulse response with windowing**

****

**Hamming Window:**

****

**INFERENCES:**

* According to the specifications given, a minimum stop band attenuation of 50 dB is given. Hence Hamming window is preferred in this case.
* Length of the filter 'N' is calculated by equating the Transition Bandwidth of require filter with the main lobe width of Hamming window.

Transition Bandwidth = ws-wp . Main lobe width of Hamming window = 8π/M; where M=N-1, order of the filter. Given pass band frequency=fp=1000Hz; fs=1500Hz.

Then normalized frequency is obtained by w=(f\*2π)/fs; where fs=sampling frequency. The normalized pass band and stop band frequencies are 0.25π and 0.375π. The cut-off frequency of ideal low pass filter is chosen as wc= (wp+ws)/2 hence, wc= 0.3125π.

* It can be observed from the magnitude plot that the obtained edge frequencies are close to the required specifications.
* From the phase plot, it can be observed that phase of the filter is linear only till the pass band frequency of the filter.
* The impulse response is even symmetrical about the middle point which ensures linear phase of FIR filters.

1. **Design a high pass filter with the following specifications:**

**Pass band edge frequency = 1500Hz; Stop band edge frequency = 1000Hz**

**Minimum stop band attenuation = 50dB; Maximum pass band attenuation = 0.9 dB. Sampling Frequency = 8000Hz**

**Plot the impulse response, magnitude spectrum and phase spectrum.**

**MATLAB CODE:**

fpass=1500;

fstop=1000;

fs=8000;

d1=0.9;

d2=50;

w=(fpass+fstop)\*2\*pi/(2\*fs);

tb=(fpass-fstop)\*2\*pi/fs;

N=(8\*pi)/tb;

h=zeros(1,N);

for n=1:N

h(n)=-sin(w\*(n-ceil(N/2)))/(pi\*(n-ceil(N/2)));

end

h(ceil(N/2))=1-(w/pi);

figure

stem(h)

xlabel('n');

ylabel('h(n)');

title('HPF impulse response without windowing');

fvtool(h);

hold on;

title('HPF without windowing');

for x=1:N

w(x)=0.54-0.46\*(1-cos((2\*pi\*x/(N-1))));

end

figure

stem(w);

title('Hamming window');

xlabel('n');

ylabel('w(n)');

hw=(h.\*w);

figure

stem(hw)

xlabel('n');

ylabel('h(n)');

title('Impulse response after windowing');

fvtool(hw);

xlabel('n');

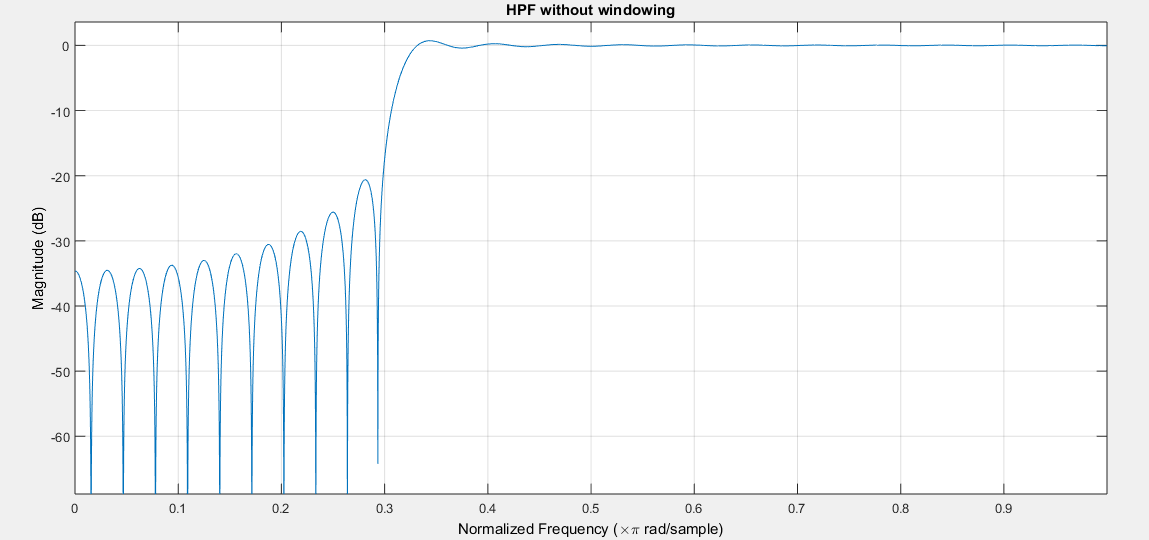
ylabel('h(n)');

title('HPF after windowing');

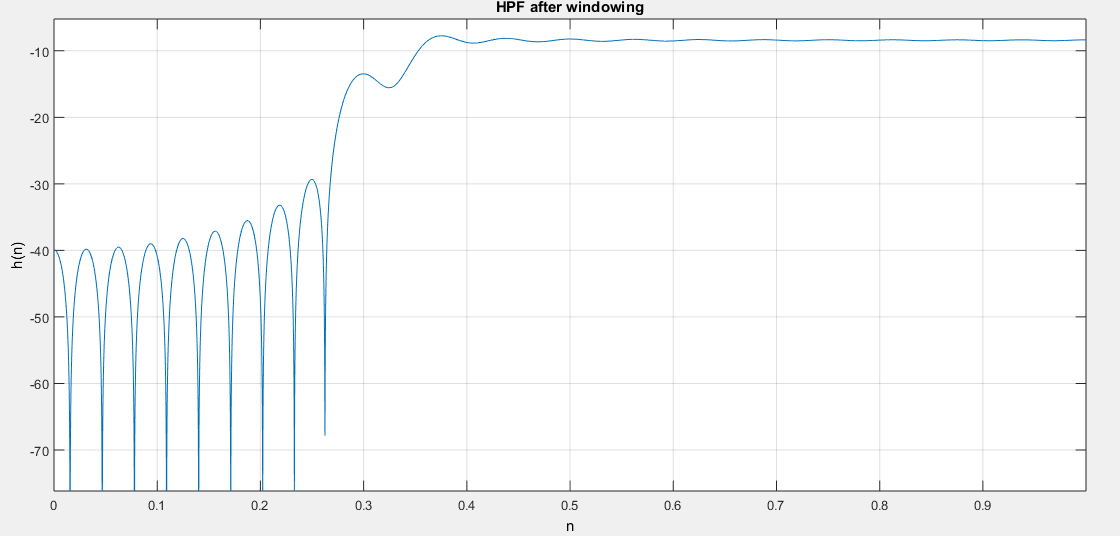
**PLOTS:**

**Magnitude response:**

1. **HPF without windowing**

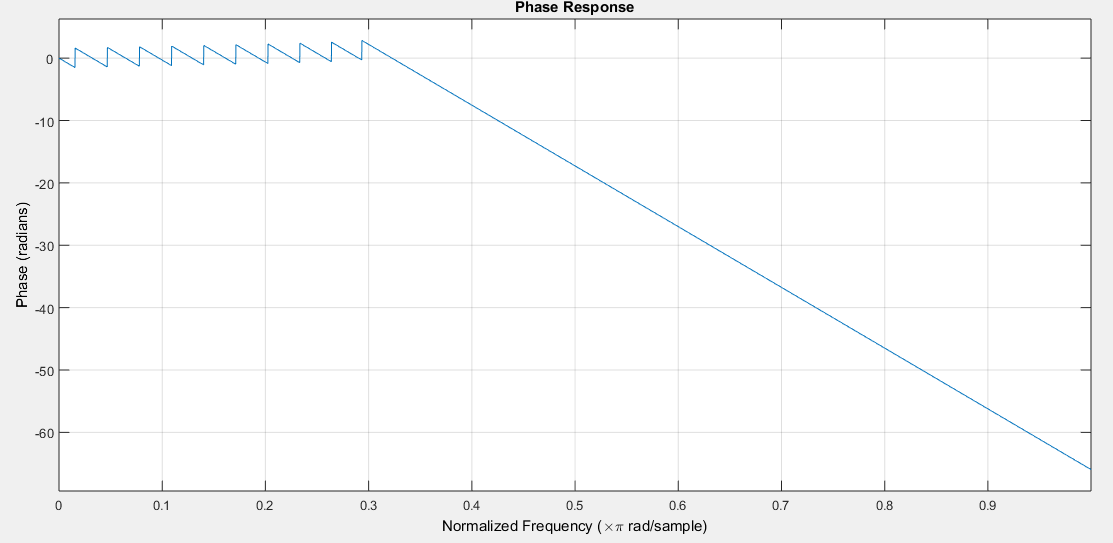


1. **HPF windowing**

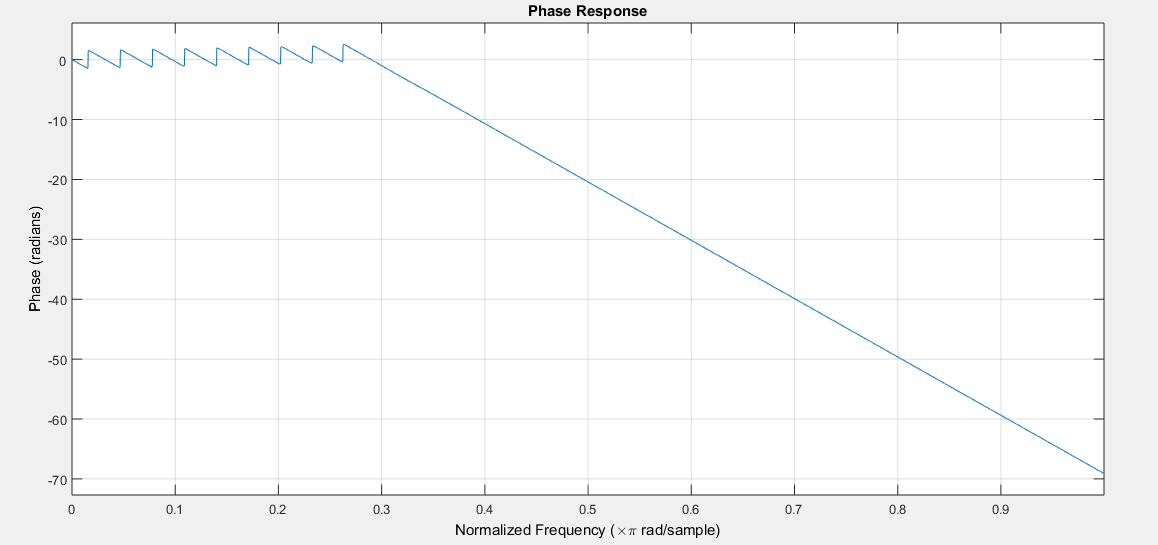
****

**Phase response:**

1. **Phase response without windowing**

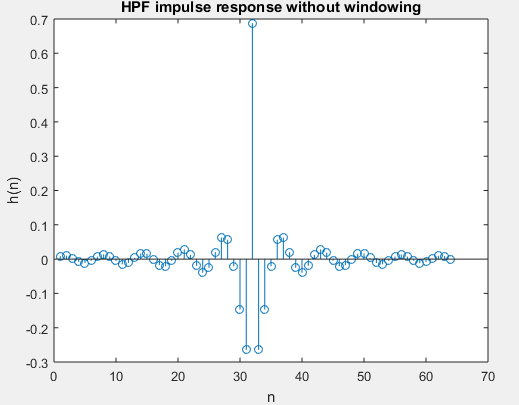


1. **Phase response with withdrawing**

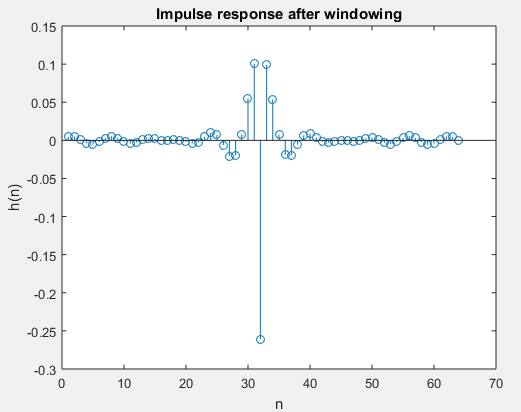
****

**Impulse response:**

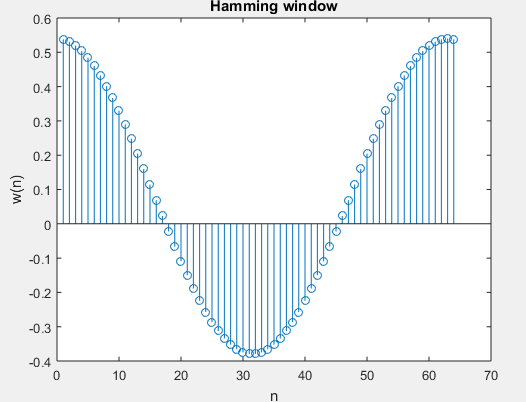
1. **Impulse response without windowing**



1. **Impulse response with windowing**

****

**Hamming:**

****

**INFERENCES:**

* The specifications of high pass filter are fs=1000Hz ; fp=1500Hz
* The normalized frequencies are ws=0.25π and wp=0.375π and cut-off frequency of ideal high pass filter is chosen as wc=0.3125π

It can be observed that the obtained edge frequencies are close to the required edge frequencies.

* Phase response is linear only within the pass band of the filter.
* Frequency response of high pass filter can be obtained as

Hhighpass=1-Hlowpass

Hence in time domain , hhpf[n]=δ[n-n0] - hlpf[n] where n0=M/2 ; M is the order of the filter.

* The impulse response of high pass filter can be observed to be exact negative of that of impulse response of low pass filter except at n=M/2.
* The impulse response is even symmetrical about middle point.

1. **Design a filter with the following specifications:**

****

**Plot the impulse response, magnitude spectrum and phase spectrum.**

**MATLAB CODE:**

clear all

clc

f1pass=0.4\*pi;

f1stop=0.15\*pi;

f2pass=0.6\*pi;

f2stop=0.8\*pi;

d1=1;

d2=60;

w=f2pass-f1pass;

tb=min((f1pass-f1stop), (f2stop-f2pass));

N=12\*pi/tb;

M=ceil(N/2);

h=zeros(1,N);

for n=1:N

h(n)=(sin(f2pass\*(n-M))/(pi\*(n-M)))-(sin(f1pass\*(n-M))/(pi\*(n-M)));

end

h(M)=w/pi;

figure

stem(h);

xlabel('n');

ylabel('h(n)');

title('BPF Impulse response without windowing');

fvtool(h);

xlabel('n');

ylabel('h(n)');

title('BPF without windowing');

for n=1:N

w1(n)=0.42-(0.5\*cos((2\*pi\*n)/N-1))+(0.08\*cos((4\*pi\*n)/N-1));

end

hw=(h.\*w1);

figure

stem(hw);

xlabel('n');

ylabel('h(n)');

title('Impulse response after windowing');

fvtool(hw);

xlabel('n');

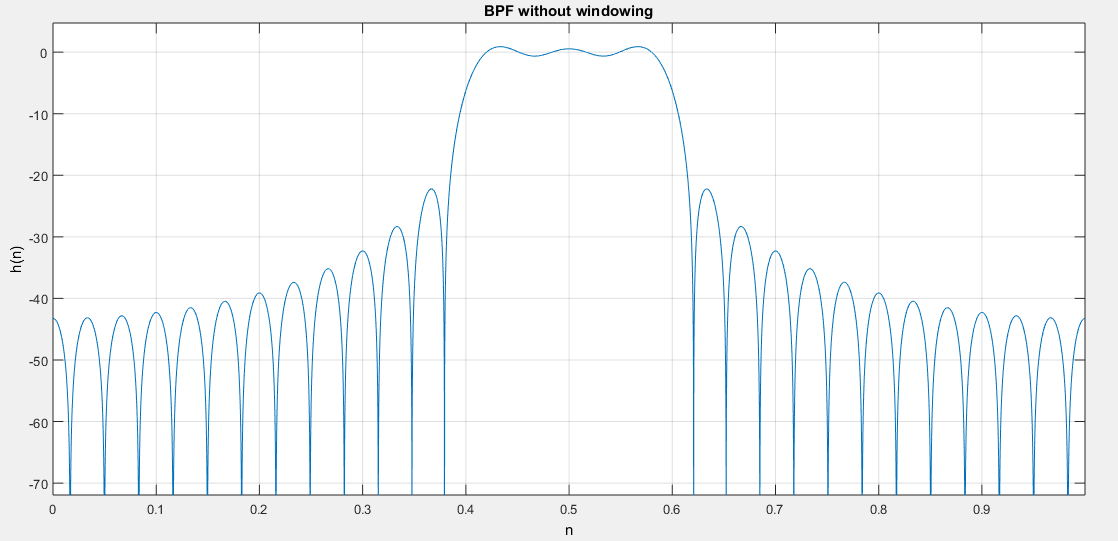
ylabel('h(n)');

title('BPF after windowing');

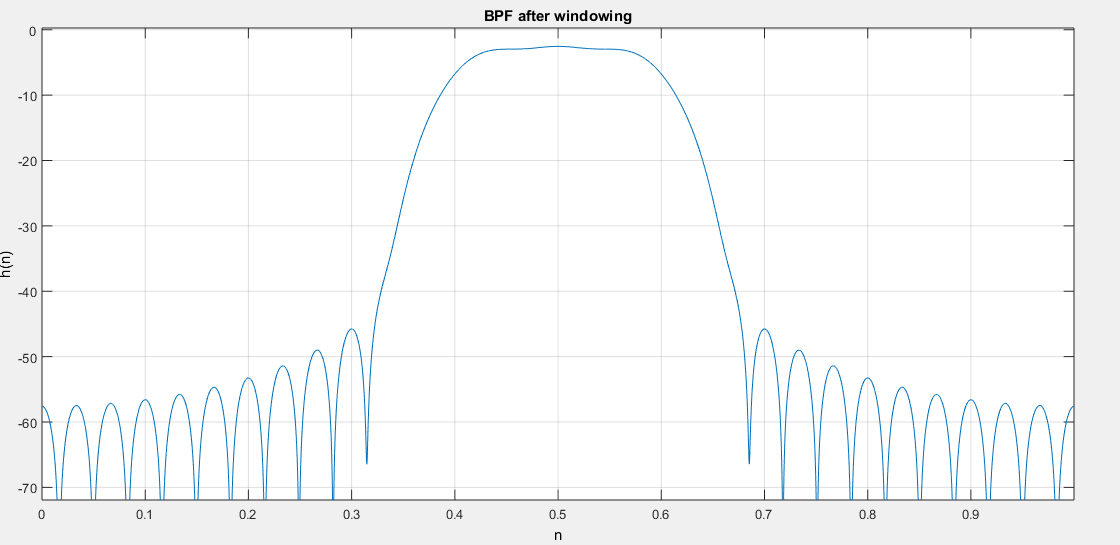
**PLOTS:**

**Magnitude response:**

1. **BPF without windowing**

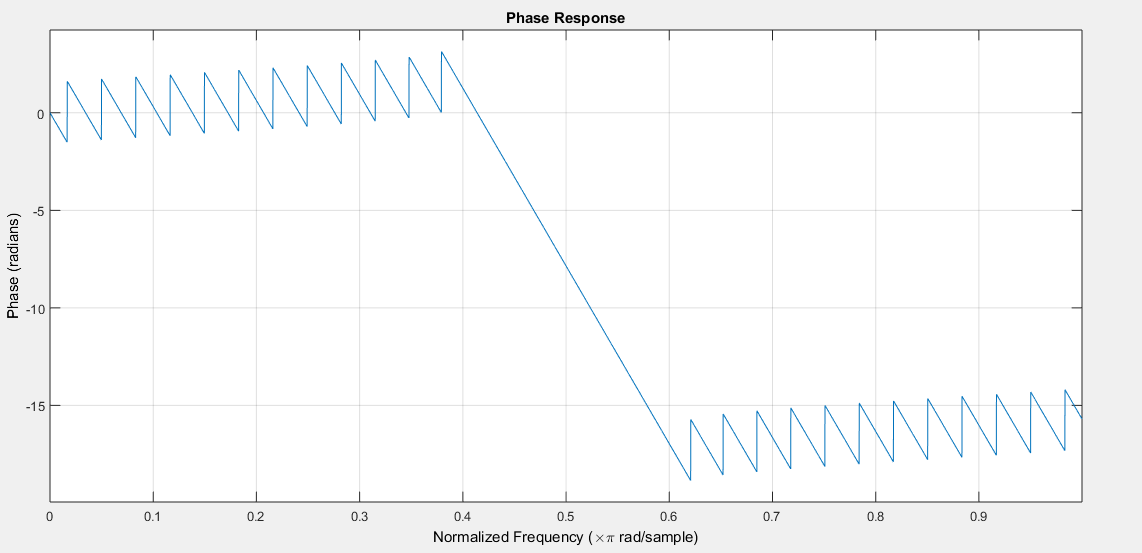
****

1. **BPF with windowing**

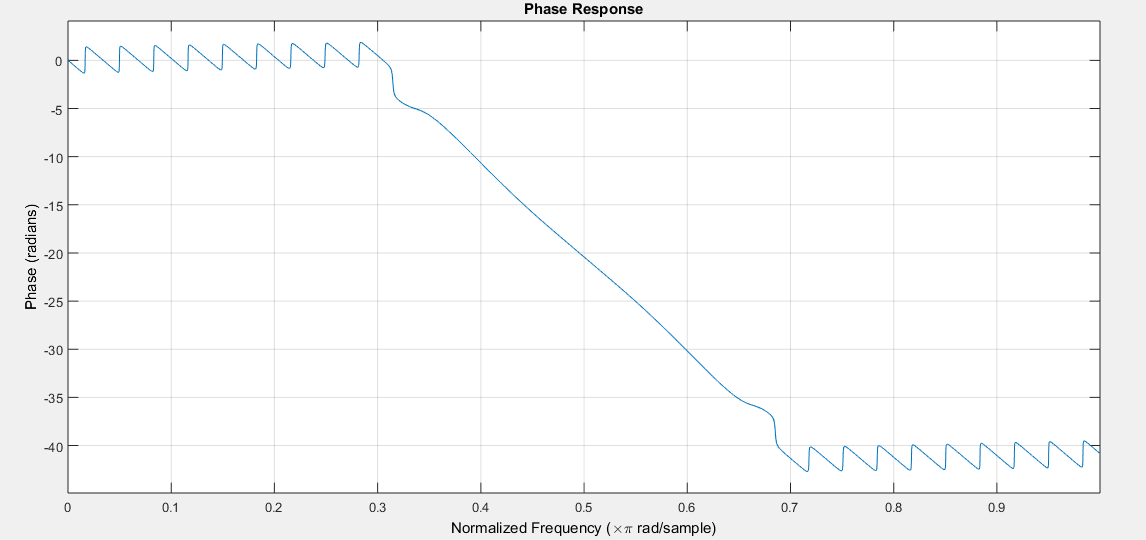
****

**Phase response:**

1. **Phase response without windowing**

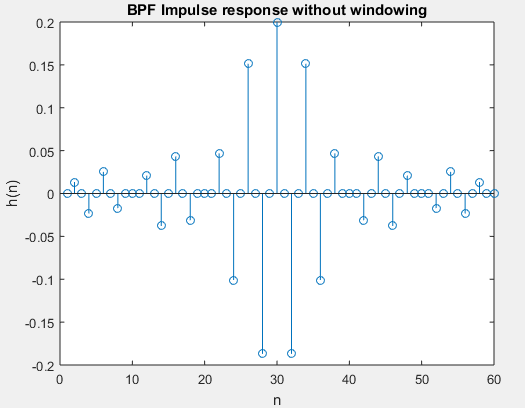
****

1. **Phase response with windowing**

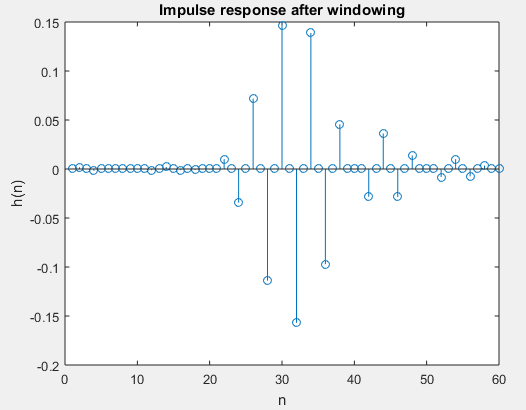
****

**Impulse response:**

1. **Impulse response without windowing**



1. **Impulse response after windowing**



1. **Generate a composite signal by adding sinusoids of different frequencies (100Hz, 200Hz, 500Hz, 2000Hz, and 4000Hz). Compute the o/p of the filters designed in Qns 1 – 3 with composite signal as i/p and verify the design.**

**MATLAB CODE:**

clear all

clc

t=[0:1:8200];

fun=sin(2\*pi\*100\*t/8200)+sin(2\*pi\*200\*t/8200)+sin(2\*pi\*500\*t/8200)+sin(2\*pi\*2000\*t/8200)+sin(2\*pi\*4000\*t/8200);

[hlpf]=lpf();

[hhpf]=hpf();

[hbpf]=bpf();

y1=conv(hlpf,fun);

y2=conv(hhpf,fun);

y3=conv(hbpf,fun);

figure

stem(abs(fft(fun)));

title('DFT of the composite signal');

figure

stem(abs(fft(y1)));

title('DFT of the composite signal passed through LPF');

figure

stem(abs(fft(y2)));

title('DFT of the composite signal passed through HPF');

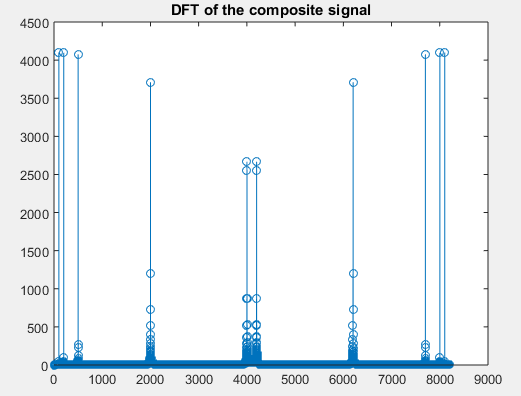
figure

stem(abs(fft(y3)));

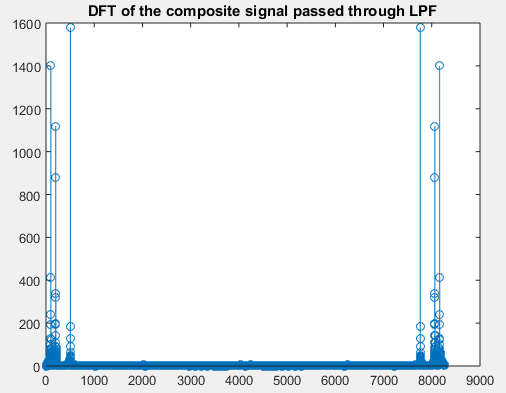
title('DFT of the composite signal passed through BPF');

**PLOTS:**

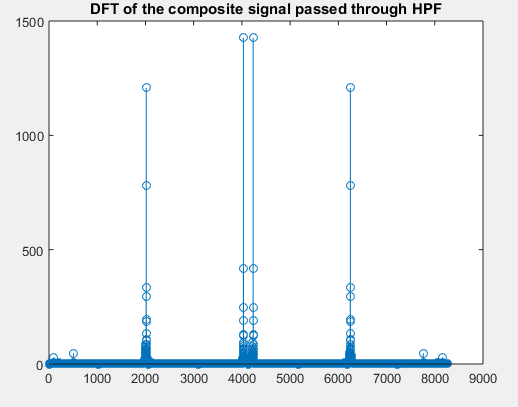
**Composite signal frequency response:**



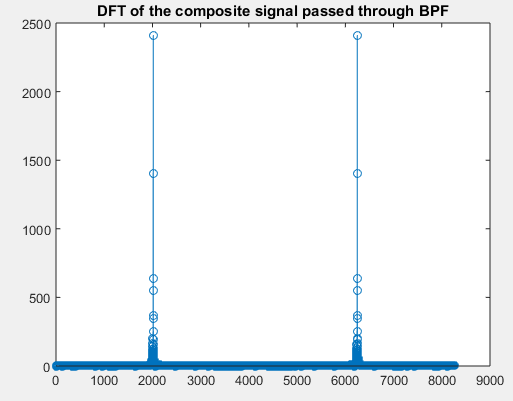
**Frequency response after passing through low pass filter:**

****

**Frequency response after passing through high pass filter:**



**Frequency response after passing through band pass filter:**



**INFERENCES:**

* The composite signal with sinusoidal frequencies 100Hz, 200Hz, 500Hz, 2000Hz, and 4000Hz is generated. The highest frequency present in the signal is 4000Hz. Hence, it should be sampled at sampling frequency which is greater than 8000Hz. Hence, sampling frequency is selected as fs=8200 Hz.
* The composite signal frequency response is symmetrical about 4100 and it has impulses at frequencies corresponding to sinusoidal frequencies present in it.
* After the signal is passed through low pass filter, the output has only frequencies at 100Hz,200 Hz and 500Hz , since only these frequencies fall in the pass band of the filter.(low pass filter is designed for pass band frequency=1000Hz and stop band frequency = 1500 Hz.)
* The output of high pass filter has only frequencies at 2000Hz and 4000Hz , which are in the pass band.(high pass filter is designed for stop band frequency = 1000 Hz and pass band frequency= 1500Hz)
* The Band pass filter is designed for normalized pass band frequencies wp1=0.4π and wp2=0.6π . The corresponding analog frequencies are

fp1=(wp1\*fs)/2π where fs=8200Hz.

fp1 =1640Hz and fp2=2460Hz.

* Only 2000Hz falls in the pass band of the designed band pass filter. Hence, only the frequency corresponding to 2000Hz will be shown up in the output.

1. **Familiarize various MATLAB functions for the design of FIR filters.**

* **fir1(n, window):**

b= fir1(n, Wn) - This function in MATLAB uses a Hamming window to design an nth order low pass, band pass or multi band FIR filter with linear phase. The filter type depends on number of elements of Wn.

* **fir2(n, f, m):**

b=fir2(n, f, m) - This function in MATLAB returns an nth order FIR filter with frequency-magnitude characteristics specified in the vectors f and m. The function linearly interpolates the desired frequency response into a dense grid and then uses inverse Fourier transform and hamming window to obtain the filter coefficients.

* In addition to these functions different window functions can be obtained directly using in-built MATLAB functions.

Eg: Hamming(N), Blackman(N), Hann(N), Bartlett(N) etc.

**RESULT:**

Designed and implemented different types of filters with the given specifications and compared the results with the desired responses. Also got familiarized with different tools in MATLAB used for filter design.