National Institute of Technology, Calicut

Department of Electronics and Communication Engineering

EC3093D - Digital Signal Processing Lab

Experiment No. 5: FIR Filter design (Tool – MATLAB)

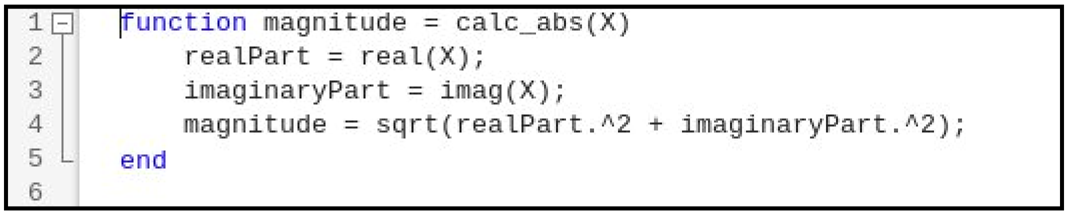
Submitted by : Group A-03

Group Members

Adwayith K S	B210664EC
Adhyuth Narayan	B210650EC
Aditya Tuppad	B210038EC

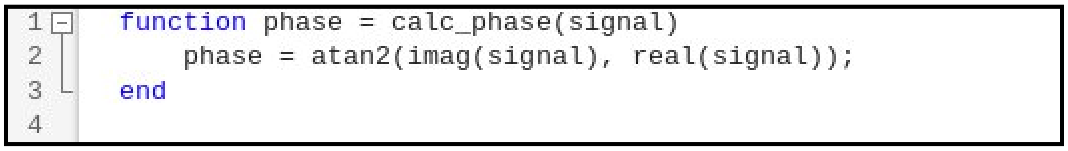
The self-defined function alternatives for the inbuild functions which were used throughout the experiment are given below :

- Function for calculating magnitude :



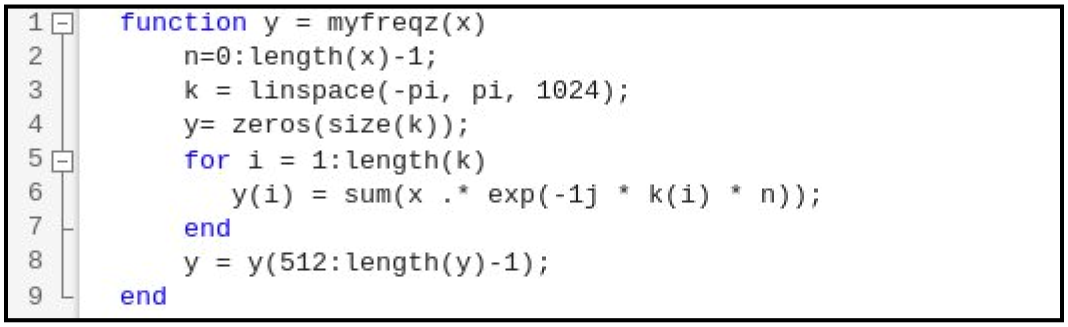
```
function magnitude = calc_abs(X)
    realPart = real(X);
    imaginaryPart = imag(X);
    magnitude = sqrt(realPart.^2 + imaginaryPart.^2);
end
```

- Function for calculating phase :



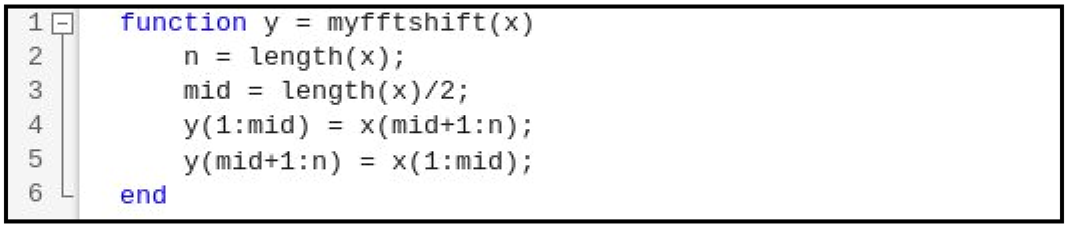
```
function phase = calc_phase(signal)
    phase = atan2(imag(signal), real(signal));
end
```

- Alternative function for inbuild function freqz :



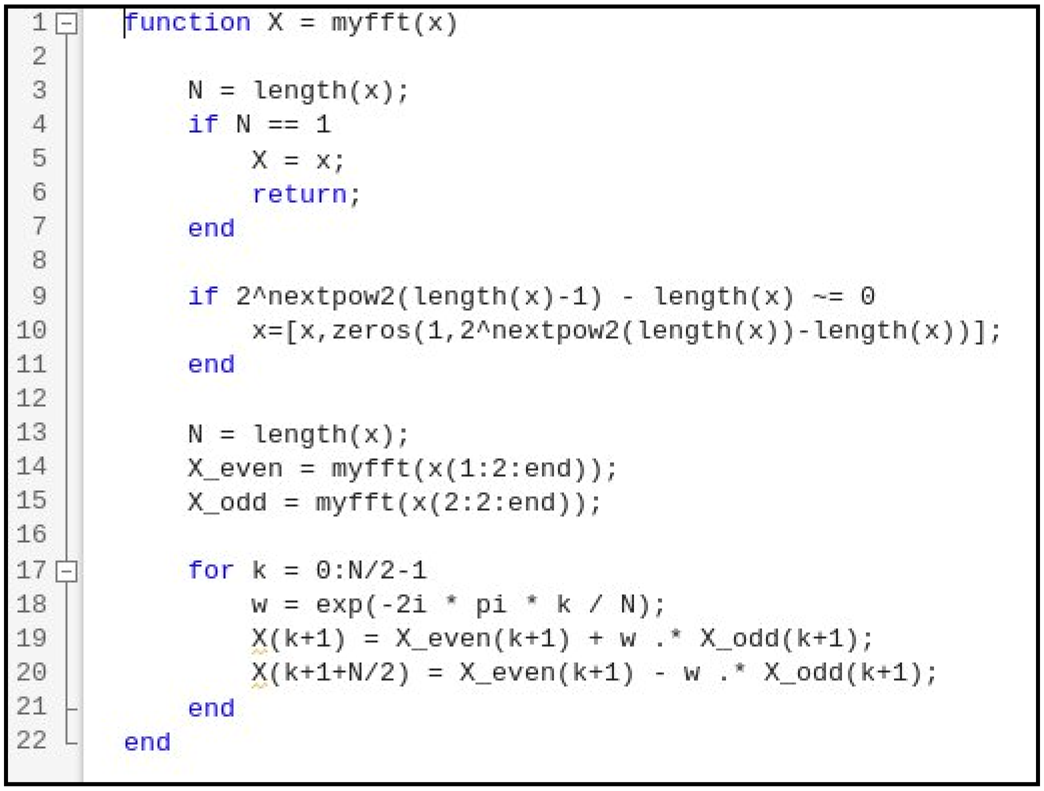
```
function y = myfreqz(x)
    n=0:length(x)-1;
    k = linspace(-pi, pi, 1024);
    y= zeros(size(k));
    for i = 1:length(k)
       y(i) = sum(x .* exp(-1j * k(i) * n));
    end
    y = y(512:length(y)-1);
end
```

- Alternative function for fftshift :



```
function y = myfftshift(x)
    n = length(x);
    mid = length(x)/2;
    y(1:mid) = x(mid+1:n);
    y(mid+1:n) = x(1:mid);
end
```

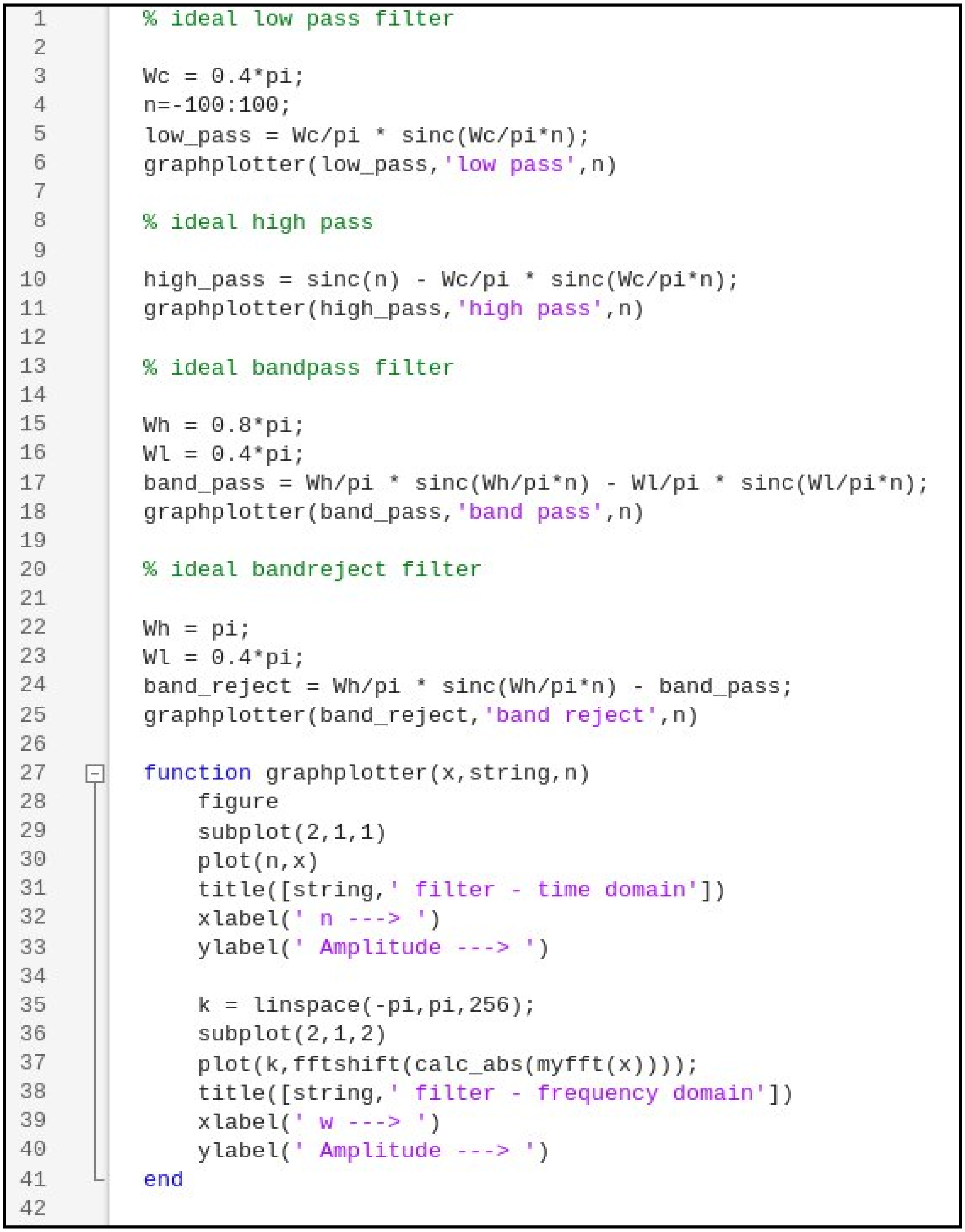
- Function to calculate FFT :



```
function X = myfft(x)

    N = length(x);
    if N == 1
        X = x;
        return;
    end

    if 2^nextpow2(length(x)-1) - length(x) ~= 0
        x=[x,zeros(1,2^nextpow2(length(x))-length(x))];
    end

    N = length(x);
    X_even = myfft(x(1:2:end));
    X_odd = myfft(x(2:2:end));

    for k = 0:N/2-1
        w = exp(-2i * pi * k / N);
        X(k+1) = X_even(k+1) + w .* X_odd(k+1);
        X(k+1+N/2) = X_even(k+1) - w .* X_odd(k+1);
    end
end
```

1. Compute the impulse response of the following filters and plot the filter functions.
    a) Ideal Low pass filter
    b) Ideal High pass filter
    c) Ideal Band pass filter
    d) Ideal Band reject filter

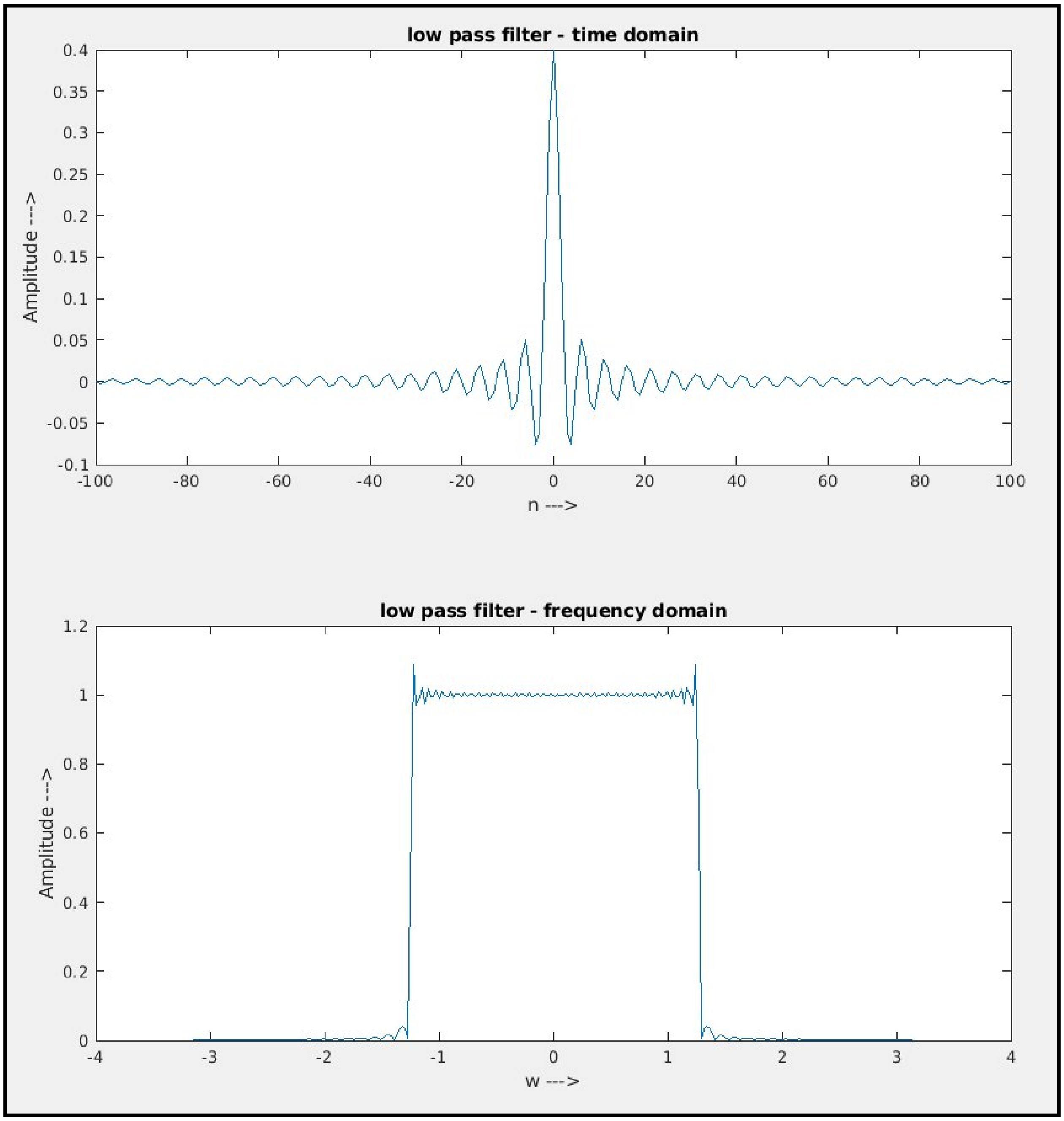
Code :



```matlab
% ideal low pass filter

Wc = 0.4*pi;
n=-100:100;
low_pass = Wc/pi * sinc(Wc/pi*n);
graphplotter(low_pass,'low pass',n)

% ideal high pass

high_pass = sinc(n) - Wc/pi * sinc(Wc/pi*n);
graphplotter(high_pass,'high pass',n)

% ideal bandpass filter

Wh = 0.8*pi;
Wl = 0.4*pi;
band_pass = Wh/pi * sinc(Wh/pi*n) - Wl/pi * sinc(Wl/pi*n);
graphplotter(band_pass,'band pass',n)

% ideal bandreject filter

Wh = pi;
Wl = 0.4*pi;
band_reject = Wh/pi * sinc(Wh/pi*n) - band_pass;
graphplotter(band_reject,'band reject',n)

function graphplotter(x,string,n)
    figure
    subplot(2,1,1)
    plot(n,x)
    title([string,' filter - time domain'])
    xlabel(' n ---> ')
    ylabel(' Amplitude ---> ')

    k = linspace(-pi,pi,256);
    subplot(2,1,2)
    plot(k,fftshift(calc_abs(myfft(x))));
    title([string,' filter - frequency domain'])
    xlabel(' w ---> ')
    ylabel(' Amplitude ---> ')
end

```

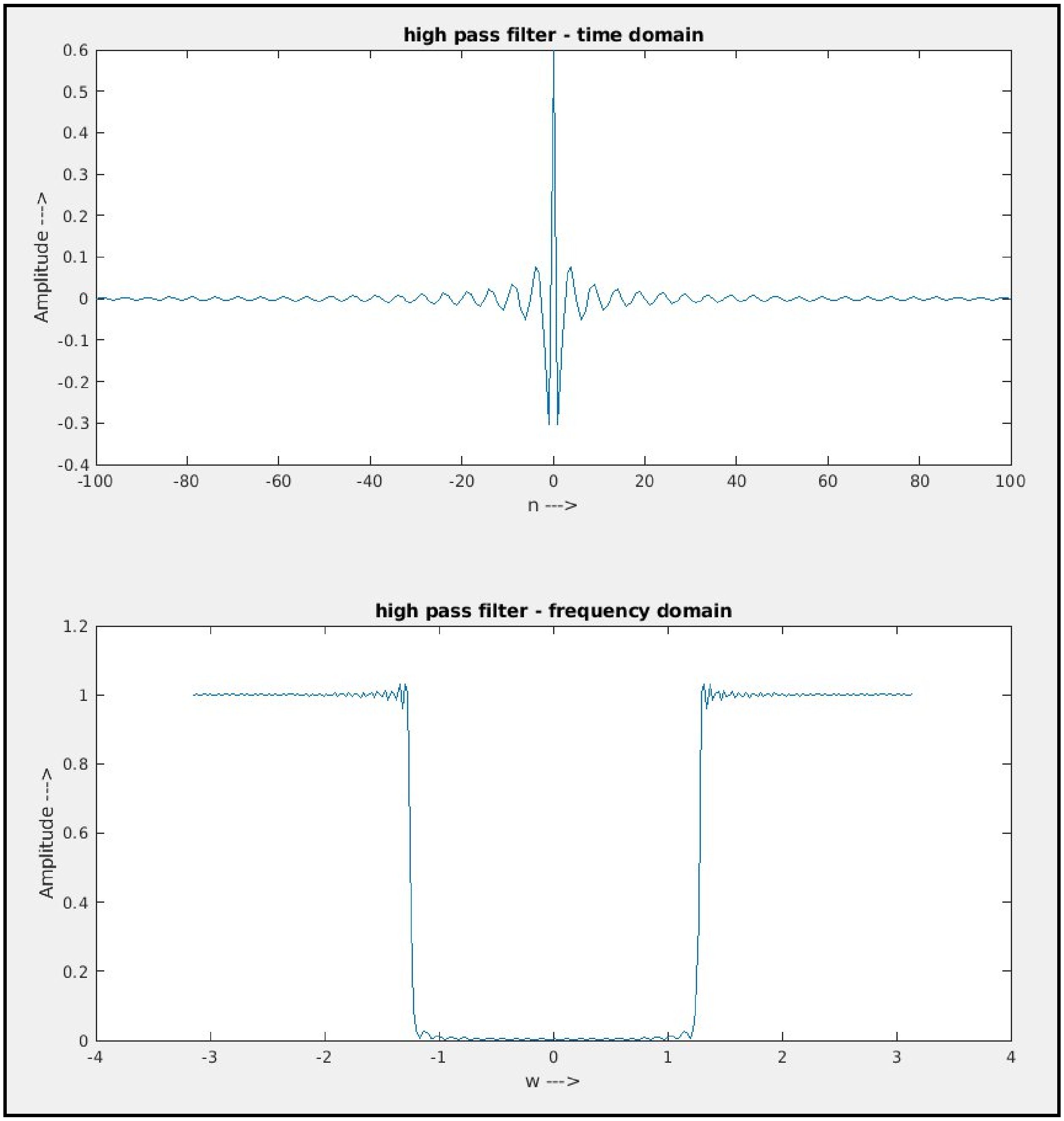
Output :

(i) Ideal low pass filter



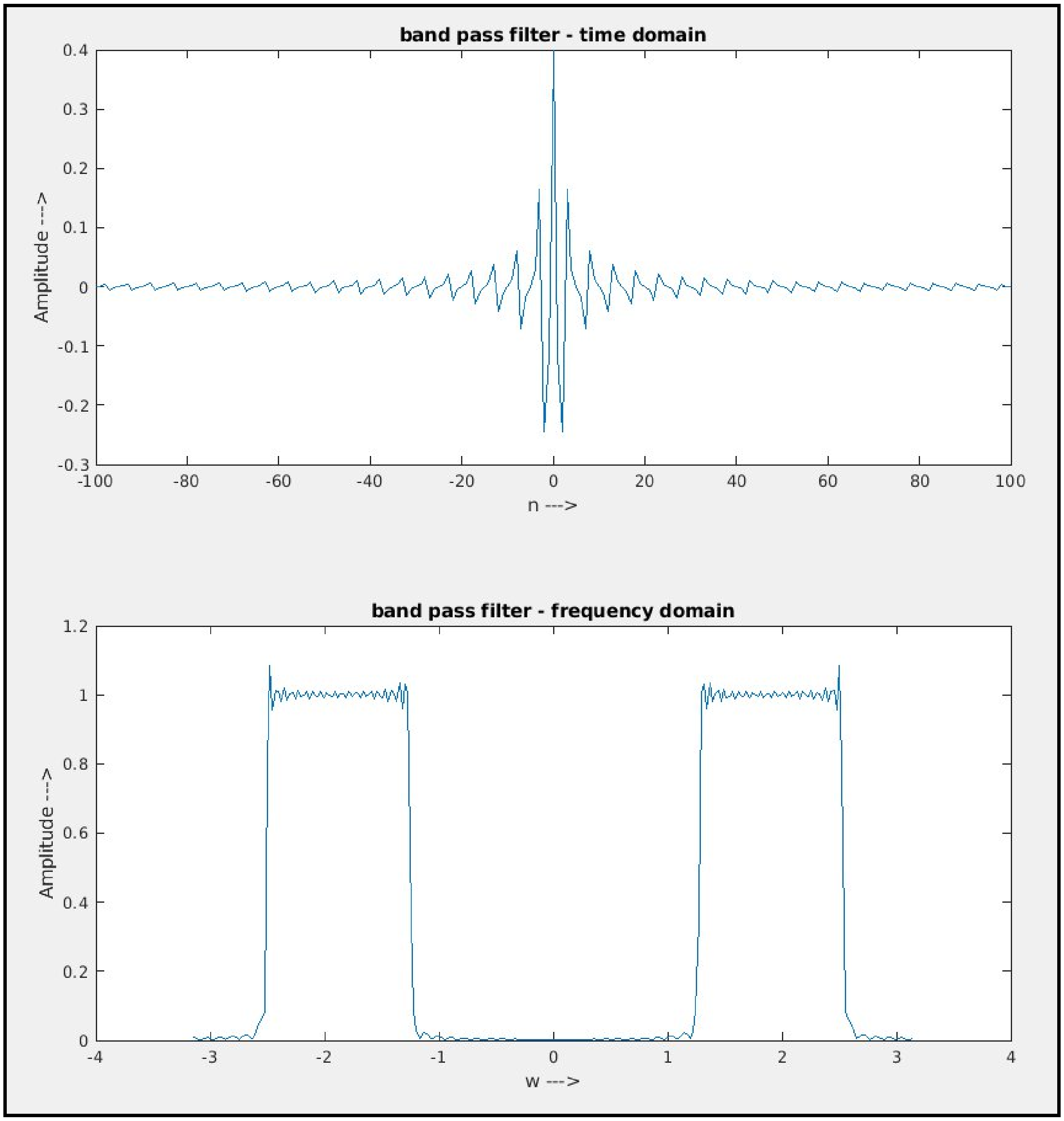
- The time domain representation of a low pass filter is a sinc function.
- In the frequency domain we can see that there are ripples in the edges, it is because the sinc function here is of finite length and not infinite as it is in the ideal case.

(ii) Ideal high pass filter



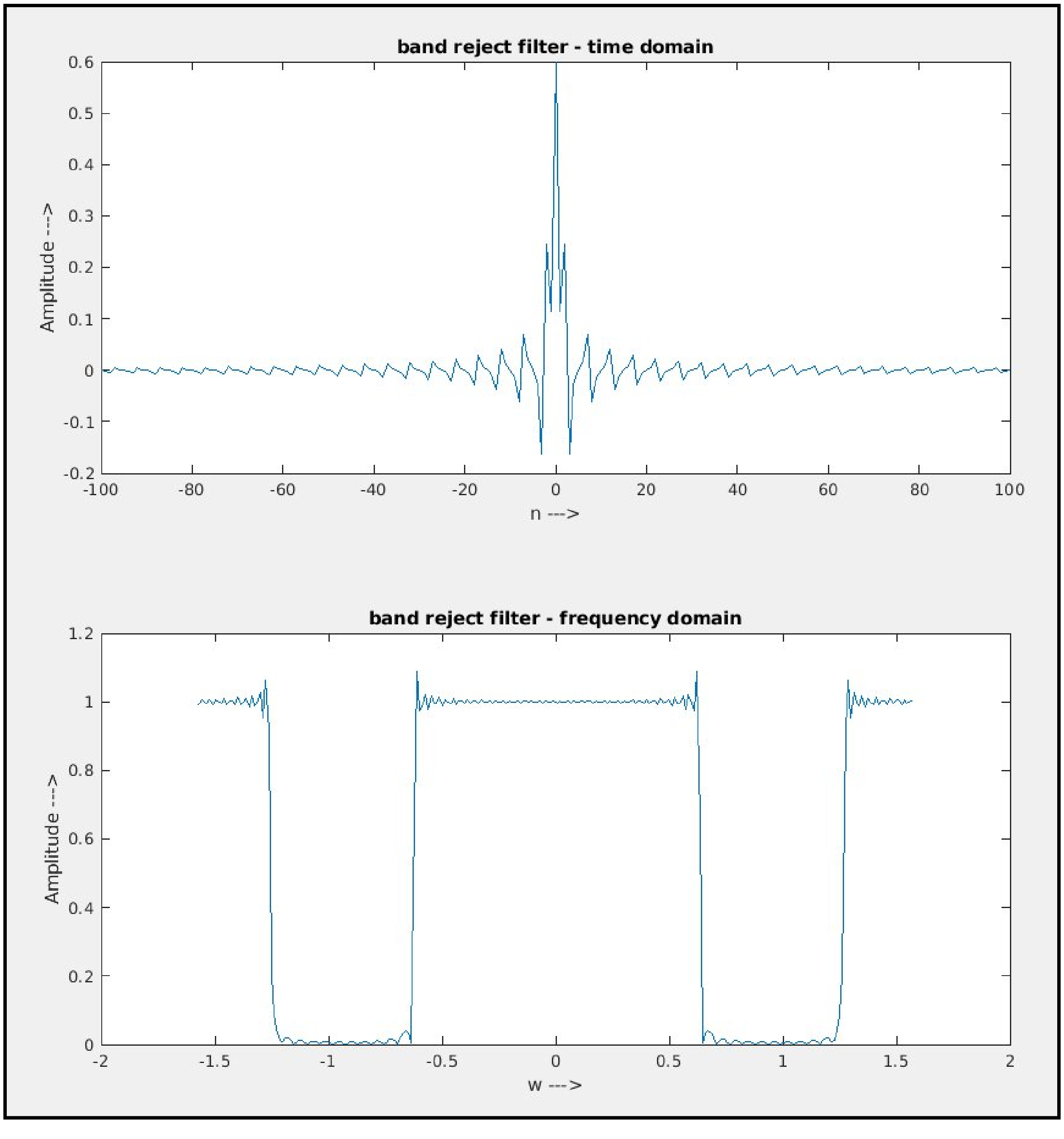
- Ideal high pass filter is a filter which only passes a frequency greater that a given cutoff frequency, in this case the given cutoff frequency is 0.4π = 1.2566.
- Here also there are ripples, this phenomenon is called gibbs phenomenon.

(iii) Ideal band-pass filter :



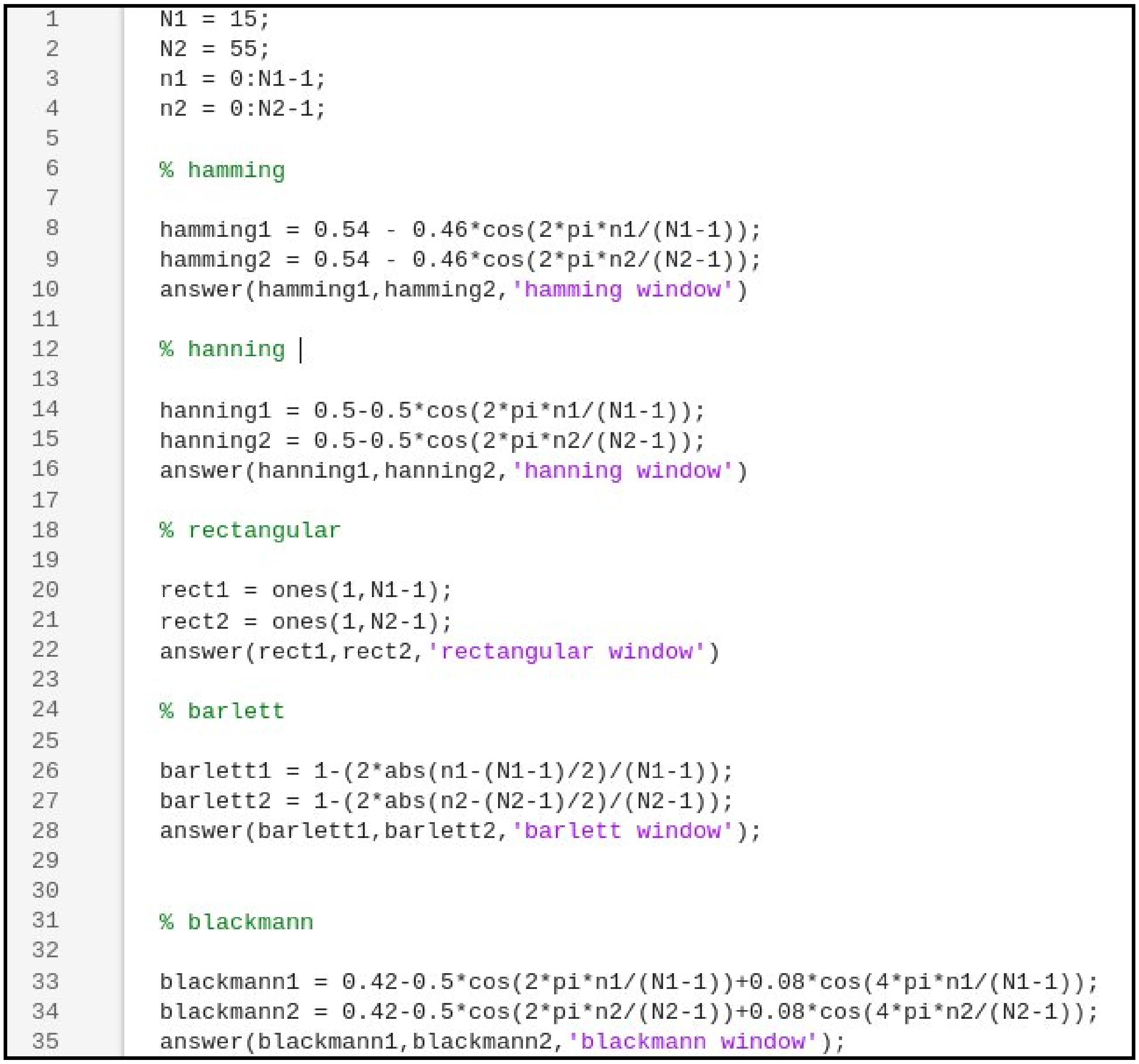
- Ideal band-pass filter is a filter which passes the frequency is a certain frequency range only, this frequency range is specified by the designer, here the frequency range is from 0.4π to 0.8π which is from 1.256 to 2.513.

(iv) Ideal band-reject filter :

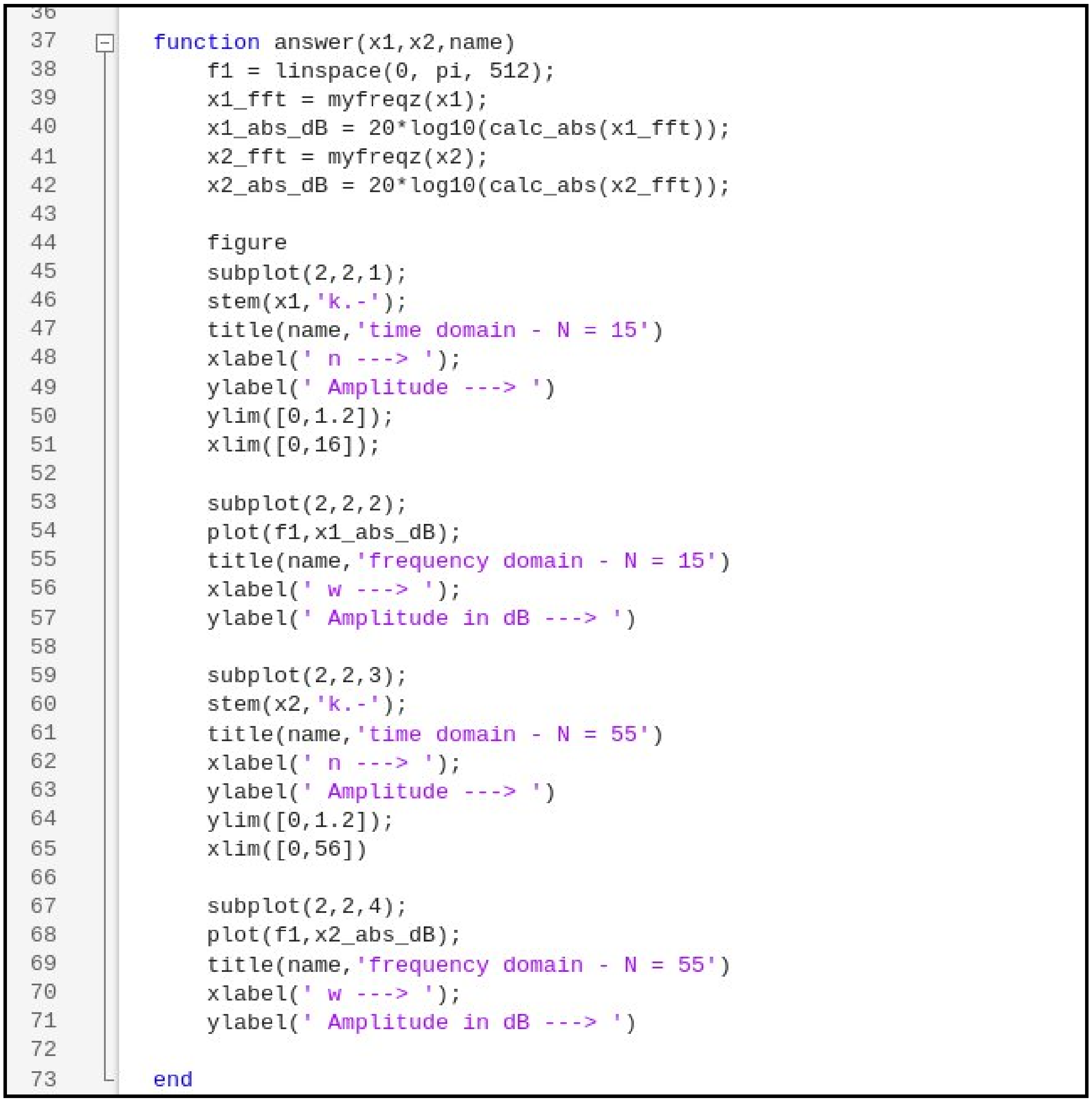


- Ideal band reject filter is a filter which passes all frequencies but rejects the frequencies in some range.
- Here the rejection range is specified as 0.4π to 0.8π which is from 1.256 to 2.513.

2. Find the Main Lobe Width (MLW) and Side Lobe Attenuation (SLA) of the following window functions used in the design of FIR filters. You can compute the window function w[n] and the magnitude spectrum W(ω) to find the required parameters. Plot the window function w[n] and its spectrum W(ω) for N=15 and N=55, where N is the length of the window and give appropriate comments based on your results.
   a. Rectangular Window
   b. Bartlett (Triangular) Window
   c. Hanning Window
   d. Hamming Window
   e. Blackman Window



```
N1 = 15;
N2 = 55;
n1 = 0:N1-1;
n2 = 0:N2-1;

% hamming

hamming1 = 0.54 - 0.46*cos(2*pi*n1/(N1-1));
hamming2 = 0.54 - 0.46*cos(2*pi*n2/(N2-1));
answer(hamming1,hamming2,'hamming window')

% hanning

hanning1 = 0.5-0.5*cos(2*pi*n1/(N1-1));
hanning2 = 0.5-0.5*cos(2*pi*n2/(N2-1));
answer(hanning1,hanning2,'hanning window')

% rectangular

rect1 = ones(1,N1-1);
rect2 = ones(1,N2-1);
answer(rect1,rect2,'rectangular window')

% barlett

barlett1 = 1-(2*abs(n1-(N1-1)/2)/(N1-1));
barlett2 = 1-(2*abs(n2-(N2-1)/2)/(N2-1));
answer(barlett1,barlett2,'barlett window');


% blackmann

blackmann1 = 0.42-0.5*cos(2*pi*n1/(N1-1))+0.08*cos(4*pi*n1/(N1-1));
blackmann2 = 0.42-0.5*cos(2*pi*n2/(N2-1))+0.08*cos(4*pi*n2/(N2-1));
answer(blackmann1,blackmann2,'blackmann window');
```

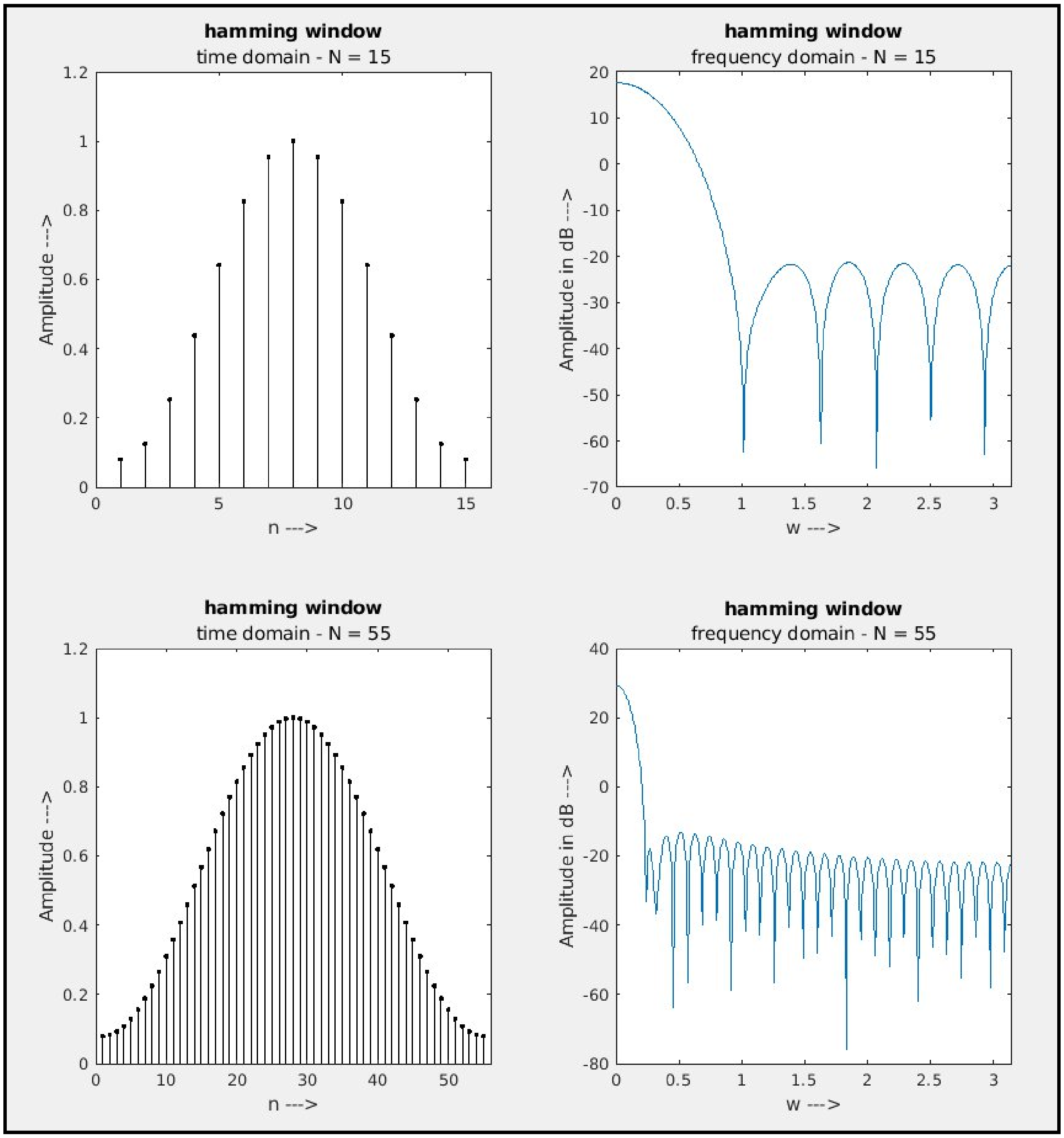


```
function answer(x1,x2,name)
    f1 = linspace(0, pi, 512);
    x1_fft = myfreqz(x1);
    x1_abs_dB = 20*log10(calc_abs(x1_fft));
    x2_fft = myfreqz(x2);
    x2_abs_dB = 20*log10(calc_abs(x2_fft));

    figure
    subplot(2,2,1);
    stem(x1,'k.-');
    title(name,'time domain - N = 15')
    xlabel(' n ---> ');
    ylabel(' Amplitude ---> ')
    ylim([0,1.2]);
    xlim([0,16]);

    subplot(2,2,2);
    plot(f1,x1_abs_dB);
    title(name,'frequency domain - N = 15')
    xlabel(' w ---> ');
    ylabel(' Amplitude in dB ---> ')

    subplot(2,2,3);
    stem(x2,'k.-');
    title(name,'time domain - N = 55')
    xlabel(' n ---> ');
    ylabel(' Amplitude ---> ')
    ylim([0,1.2]);
    xlim([0,56])

    subplot(2,2,4);
    plot(f1,x2_abs_dB);
    title(name,'frequency domain - N = 55')
    xlabel(' w ---> ');
    ylabel(' Amplitude in dB ---> ')

end
```

The code for this question is given above here we have used a function answer() to plot all the window functions.

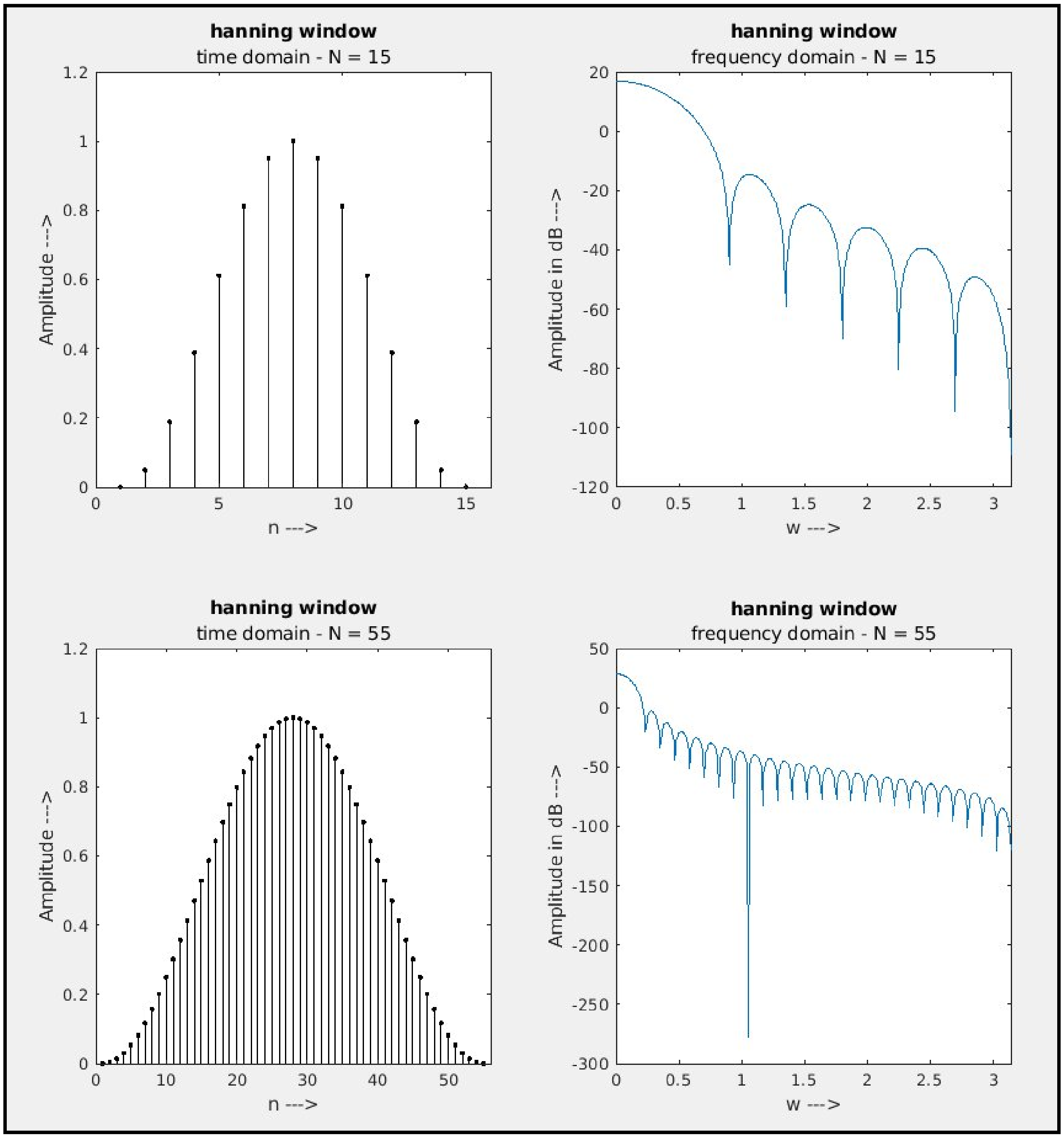
Output :

Hamming window:



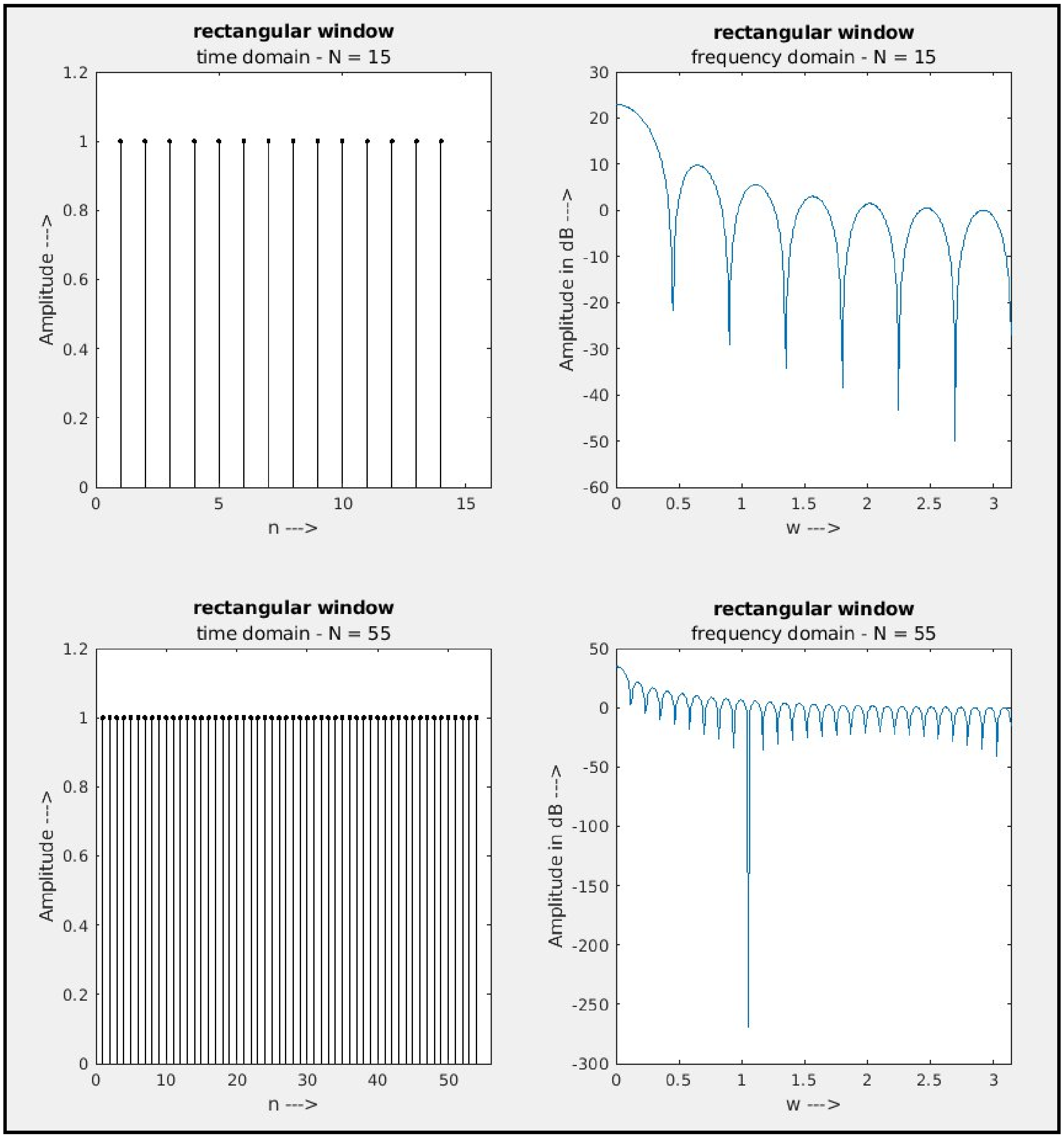
- Observed SLA for hamming window from the plot is 43dB, and the observed MLW for hamming window is 0.49, these observed values are exactly matching with that of theoretical values.
- We can also observe that the MLW decreases as the N increases this is because the equation for MLW is $8\pi/N$.

Hanning window :



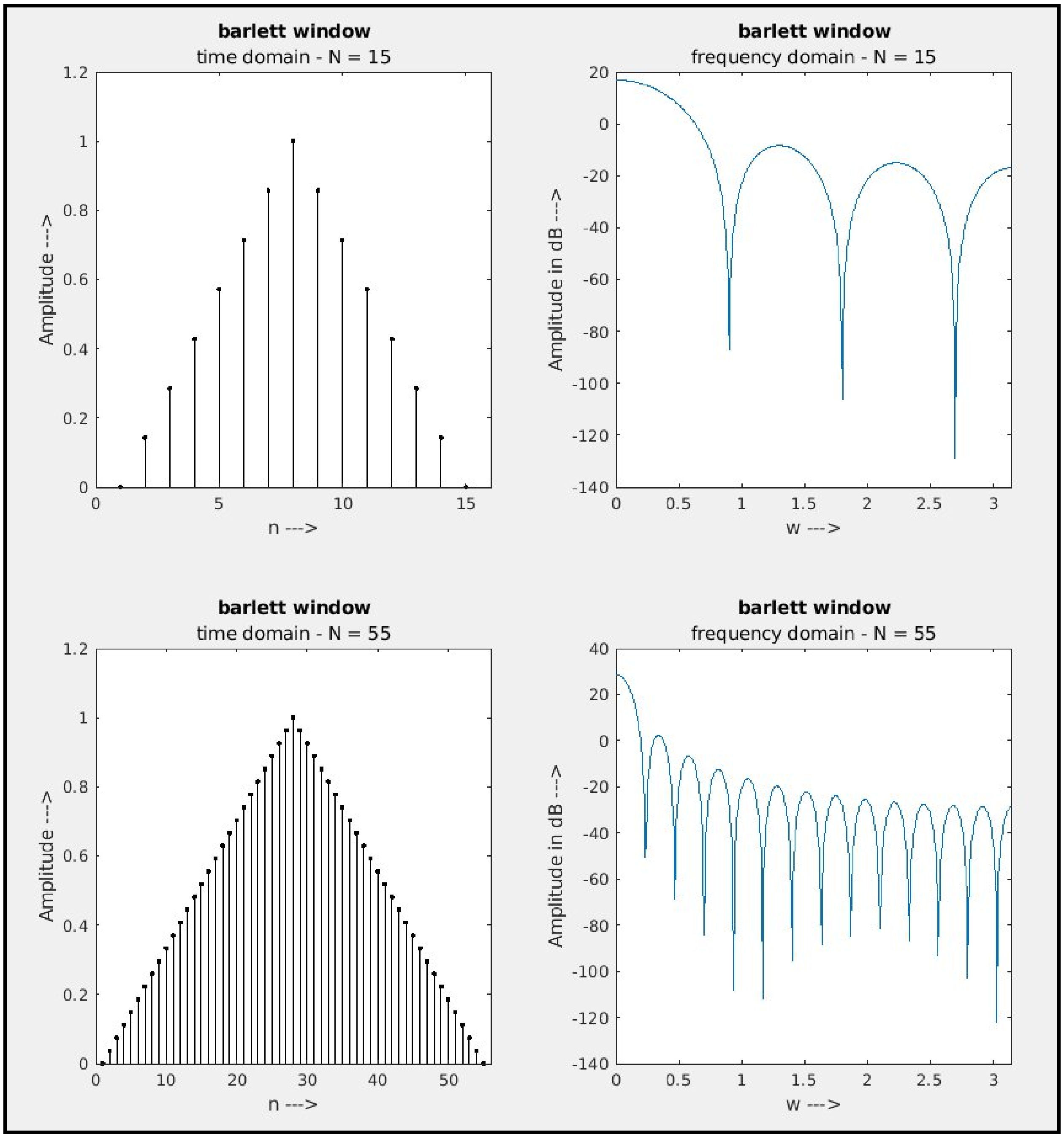
- Observed SLA for hanning window from the plot is 31.4 dB, and the observed MLW for hanning window is 0.466, these observed values are exactly matching with that of theoretical values.
- We can also observe that the MLW decreases as the N increases this is because the equation for MLW is $8\pi/N$.

Rectangular window :



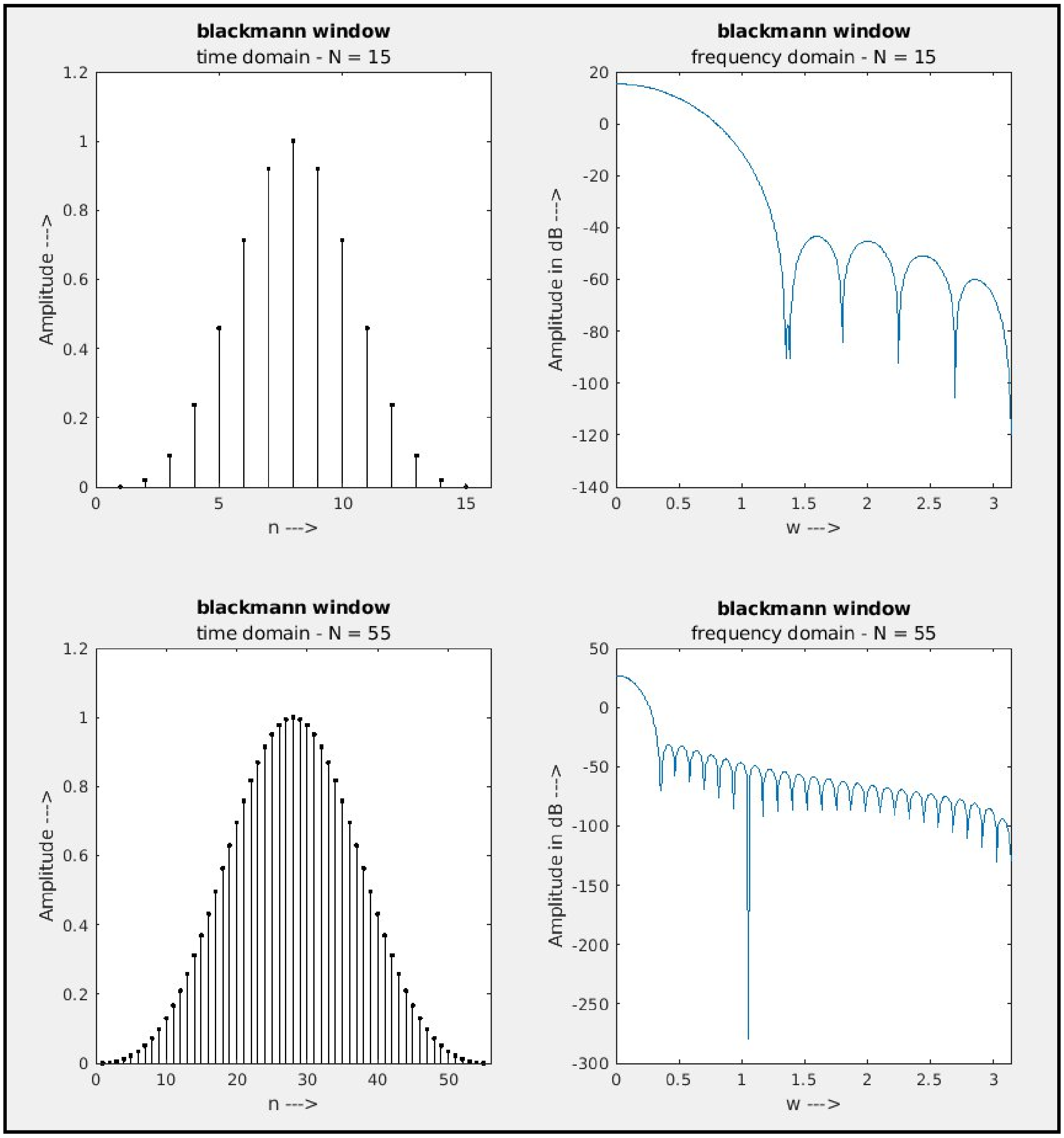
- Observed SLA for rectangular window from the plot is 13.26dB, and the observed MLW for hamming window is 0.2336, these observed values are exactly matching with that of theoretical values.
- We can also observe that the MLW decreases as the N increases this is because the equation for MLW is $4\pi/N+1$.

Barlett or triangular window :



- Observed SLA for barlett window from the plot is 26.45 dB, and the observed MLW for barlett window is 0.446, these observed values are exactly matching with that of theoretical values.
- We can also observe that the MLW decreases as the N increases this is because the equation for MLW is $8\pi/N$.

Blackmann window :



- Observed SLA for blackmann window from the plot is 57 dB, and the observed MLW for blackmann window is 0.712, these observed values are exactly matching with that of theoretical values.
- We can also observe that the MLW decreases as the N increases this is because the equation for MLW is $12\pi/N$.

3. Design a linear Phase Type 1 FIR filter with the following specifications using window method. Choose appropriate window function based on the given specifications and compute the order of the filter. Plot the window function, ideal impulse response and impulse response of the designed filter. Plot the magnitude and phase spectrum of the designed filter and verify that the given specifications and linear phase property is met.

   a. 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

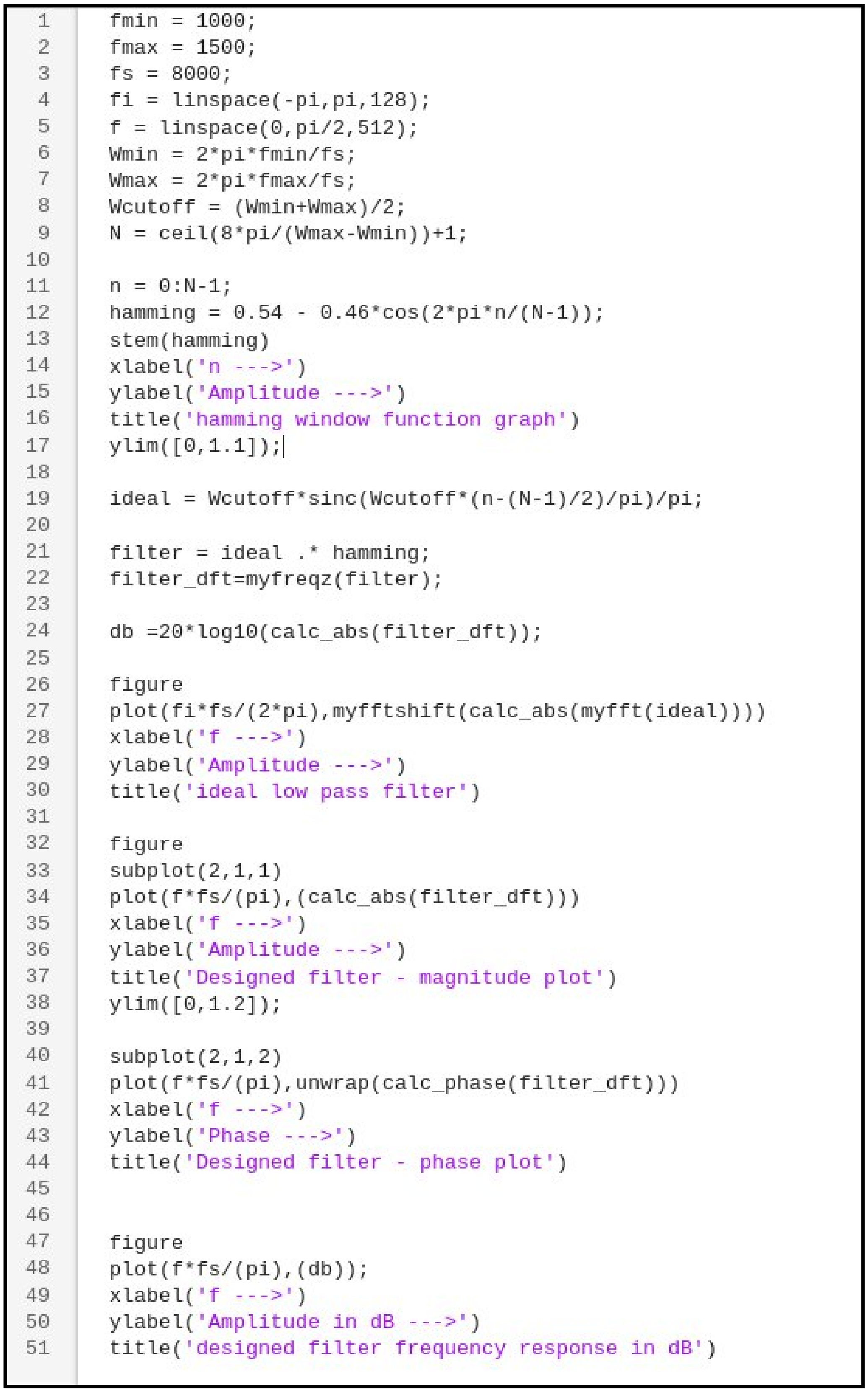
   b. Pass band edge frequency = 1500Hz
      Stop band edge frequency = 1000Hz
      Minimum stop band attenuation = 50 dB
      Maximum pass band attenuation = 0.9 dB
      Sampling Frequency = 8000Hz

Here we are supposed to find the window function based on the given characteristics, here it is given that minimum stop band attenuation of 50 dB should be there so we can choose either hamming window or blackmann window, here for doing this question we have chosen hamming window.

The first step for us is to calculate the length of the filter , we can do that easily as the length of the transition lobe is given, in case of hamming window the length of transition lobe is $8\pi/N$ by equating this we can find the filter length.

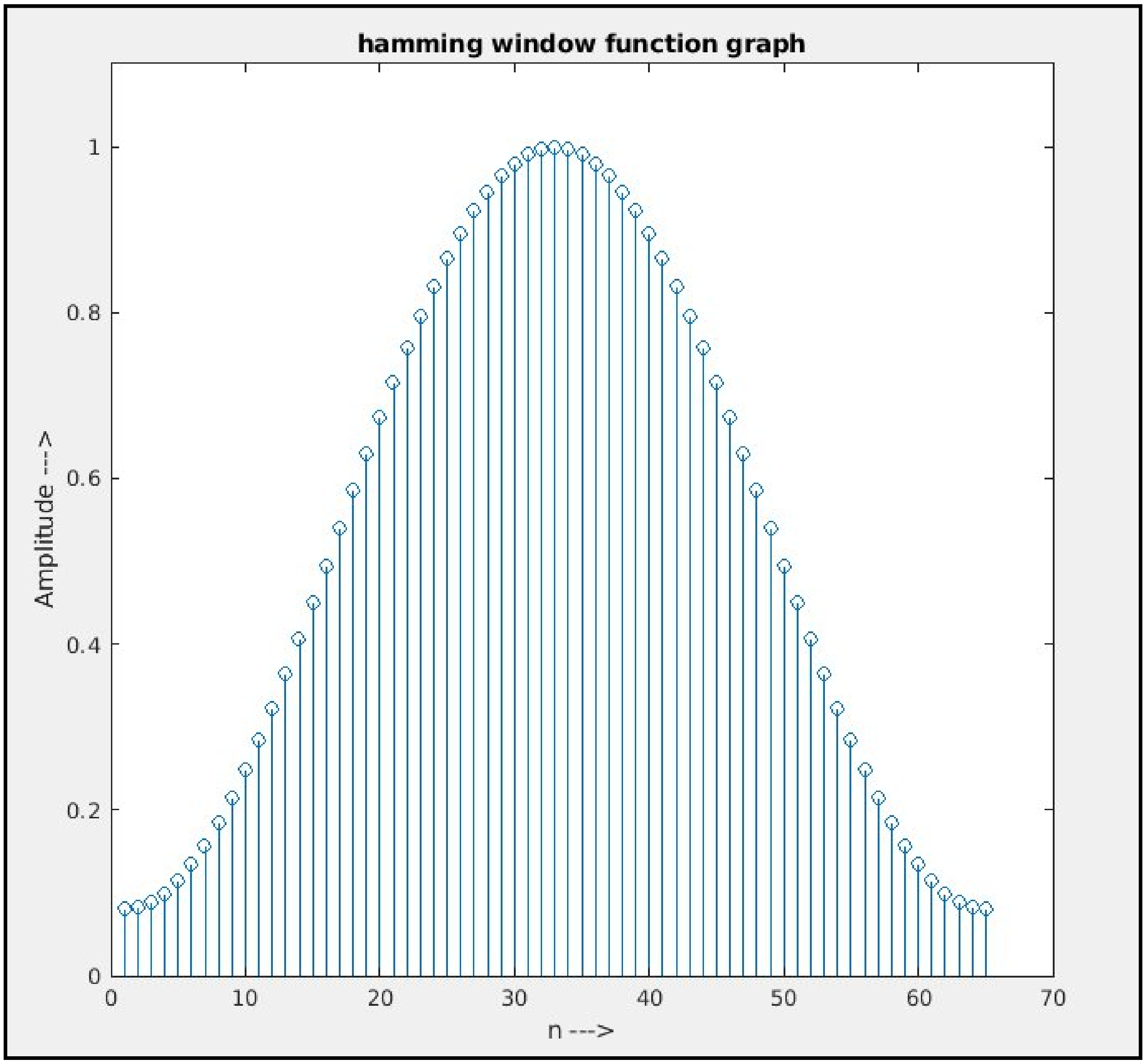
The code for the part a of the question is given in the next page

Part a code :

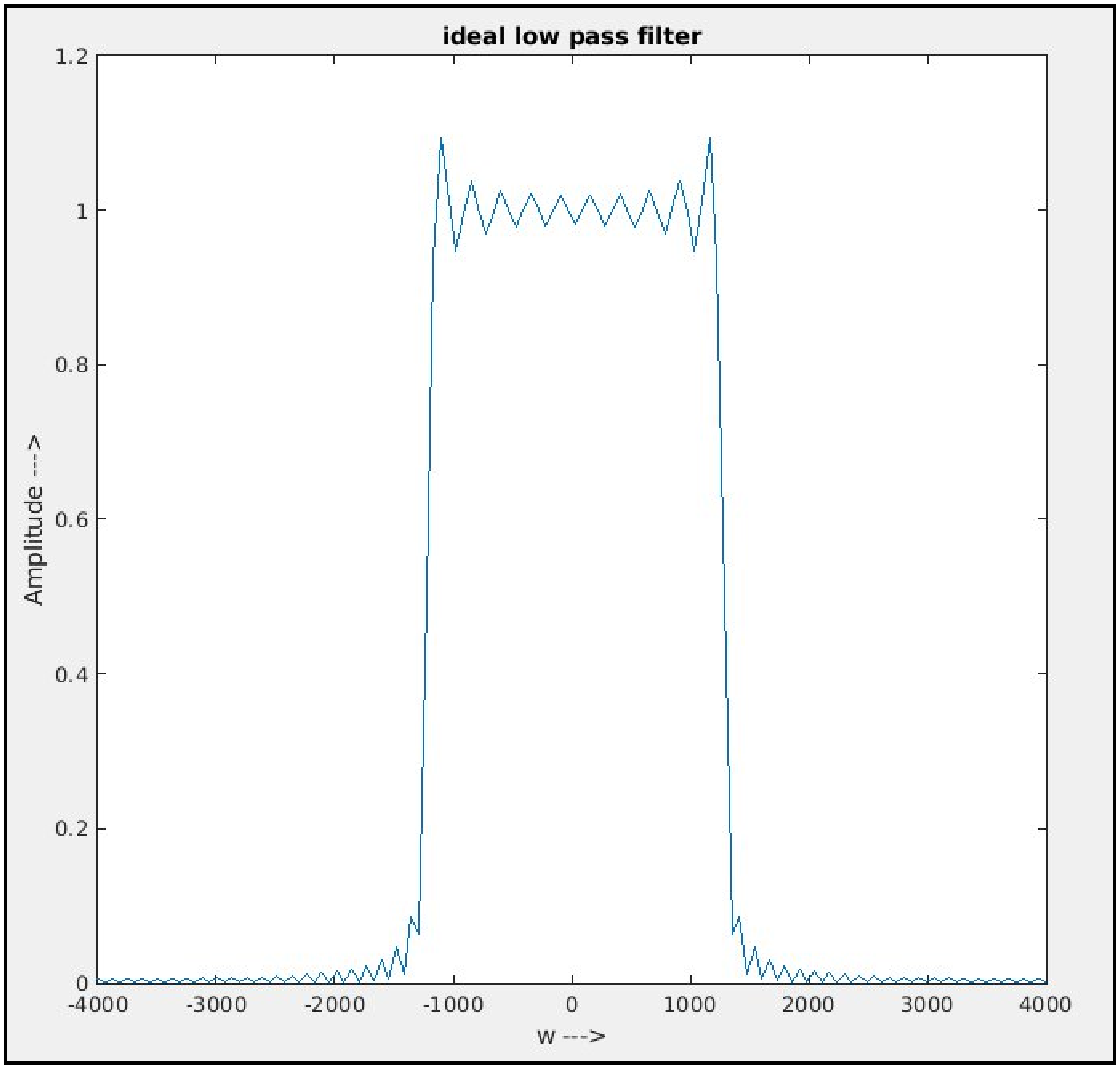


```
fmin = 1000;
fmax = 1500;
fs = 8000;
fi = linspace(-pi,pi,128);
f = linspace(0,pi/2,512);
Wmin = 2*pi*fmin/fs;
Wmax = 2*pi*fmax/fs;
Wcutoff = (Wmin+Wmax)/2;
N = ceil(8*pi/(Wmax-Wmin))+1;

n = 0:N-1;
hamming = 0.54 - 0.46*cos(2*pi*n/(N-1));
stem(hamming)
xlabel('n --->')
ylabel('Amplitude --->')
title('hamming window function graph')
ylim([0,1.1]);

ideal = Wcutoff*sinc(Wcutoff*(n-(N-1)/2)/pi)/pi;

filter = ideal .* hamming;
filter_dft=myfreqz(filter);

db =20*log10(calc_abs(filter_dft));

figure
plot(fi*fs/(2*pi),myfftshift(calc_abs(myfft(ideal))))
xlabel('f --->')
ylabel('Amplitude --->')
title('ideal low pass filter')

figure
subplot(2,1,1)
plot(f*fs/(pi),(calc_abs(filter_dft)))
xlabel('f --->')
ylabel('Amplitude --->')
title('Designed filter - magnitude plot')
ylim([0,1.2]);

subplot(2,1,2)
plot(f*fs/(pi),unwrap(calc_phase(filter_dft)))
xlabel('f --->')
ylabel('Phase --->')
title('Designed filter - phase plot')


figure
plot(f*fs/(pi),(db));
xlabel('f --->')
ylabel('Amplitude in dB --->')
title('designed filter frequency response in dB')
```

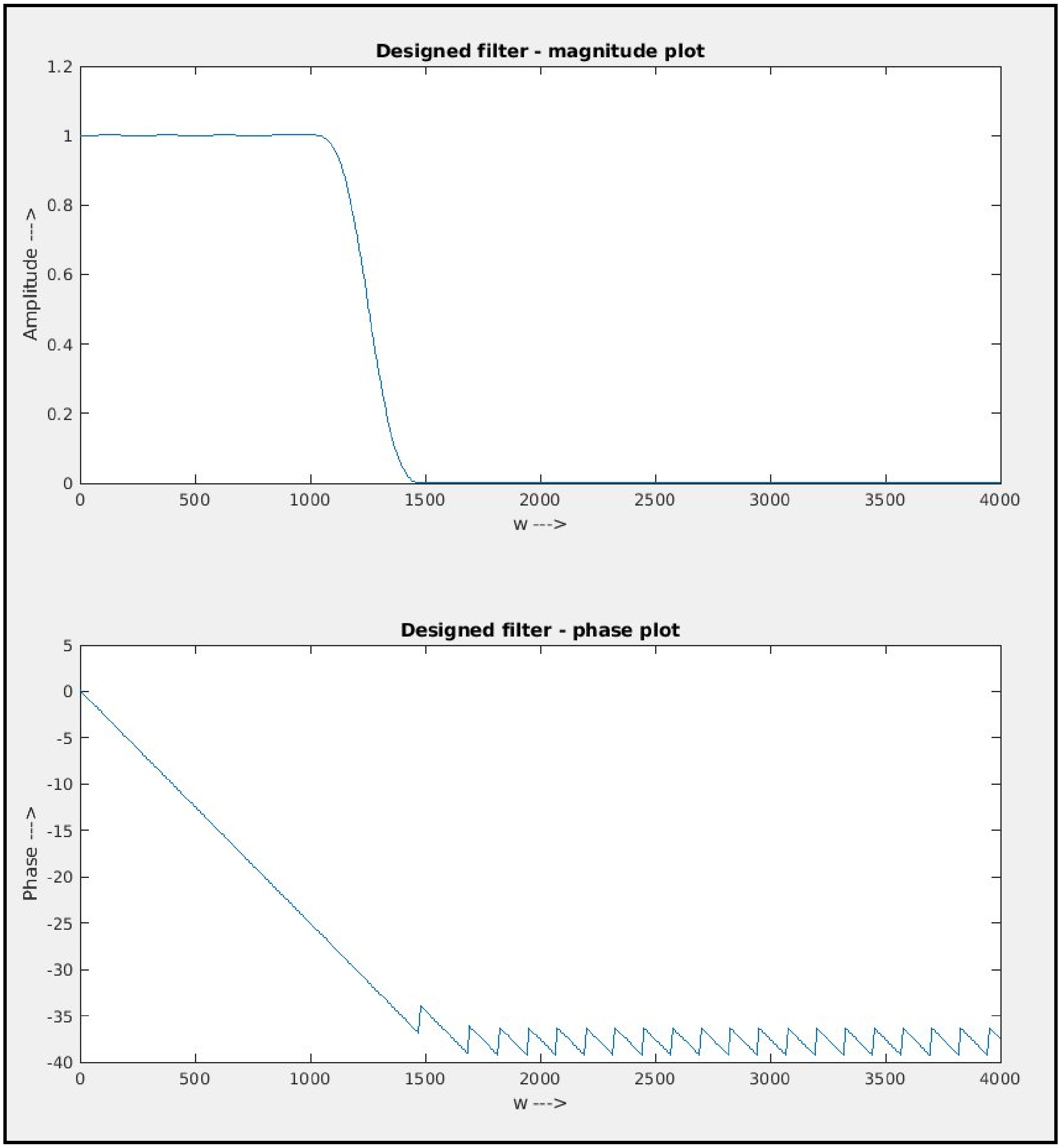
Output :



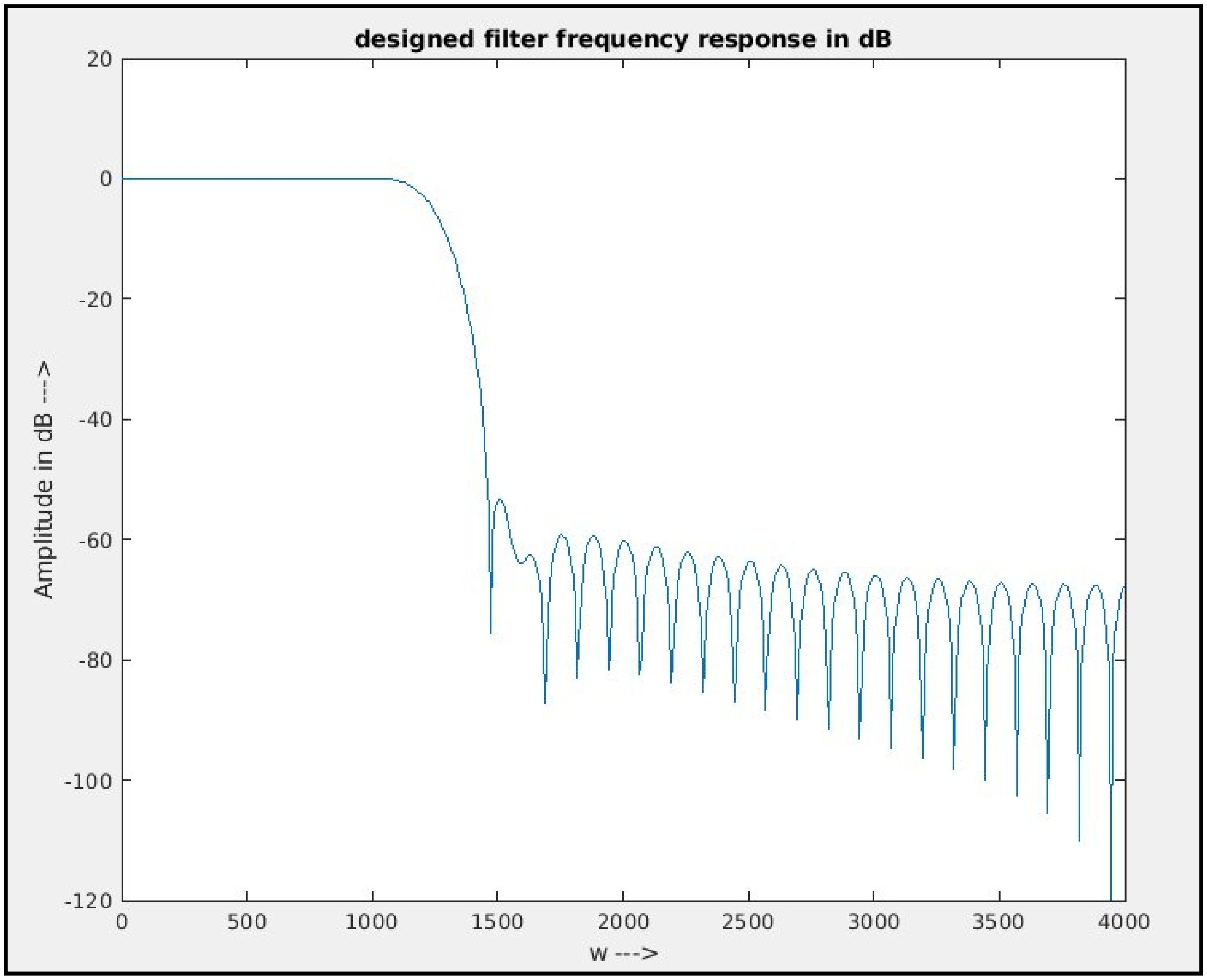
- This is the plot of the hamming window function which we are going to use in this question.



- This is the ideal low pass filter, we can clearly see that it has lots of ripples in the pass band and the stop band, in order to smoothen this ideal filter, we are multiplying it with the hamming window function.

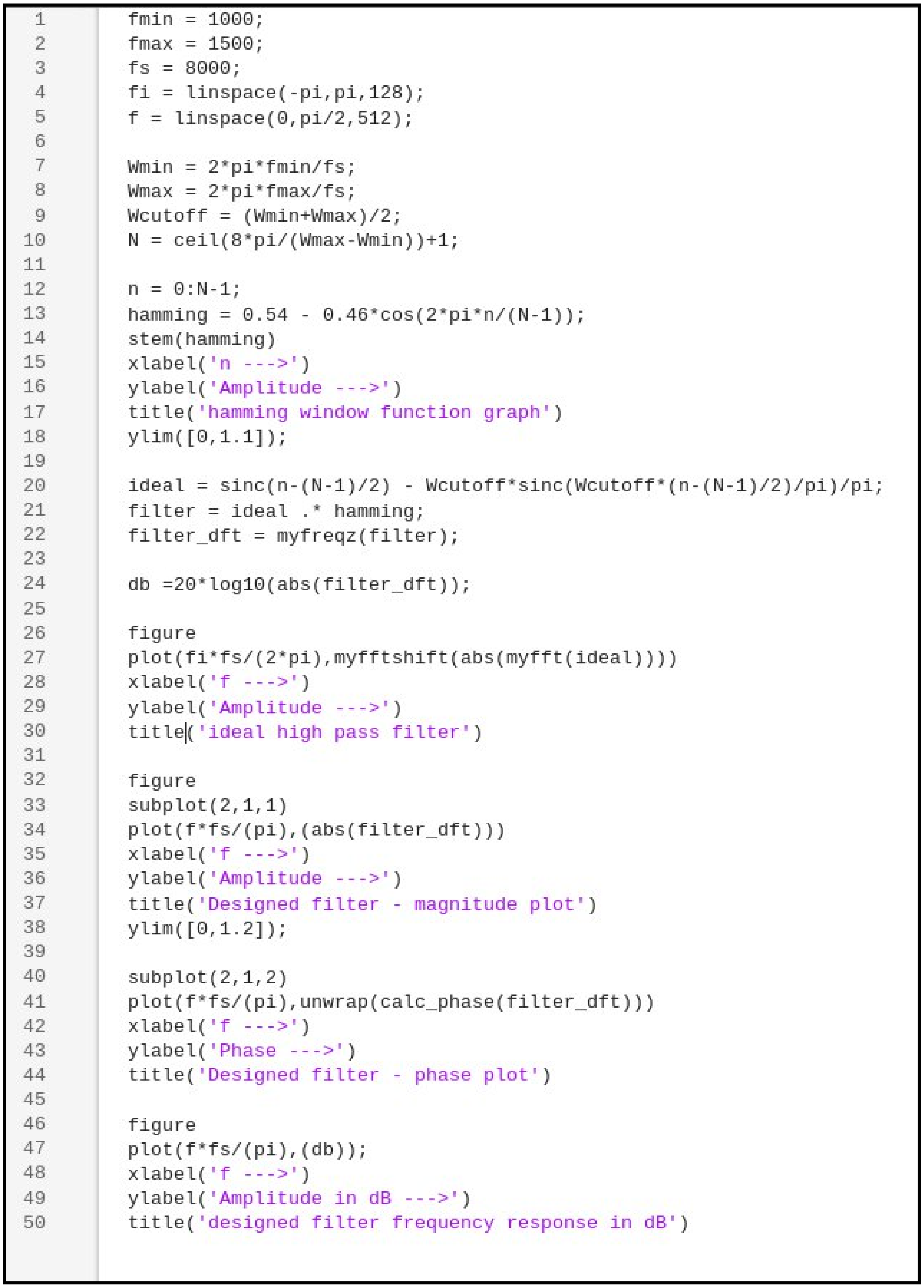


- Given here is the magnitude and phase plot of the designed filter, this is obtained by multiplying the ideal filter response and the hamming window function.
- We can see that the designed filter is linear phase in the passband and it has got only some minute ripples in the phase in the stop band.
- The cutoff frequencies of 1000Hz and 1500Hz are correctly met, as given in the question.



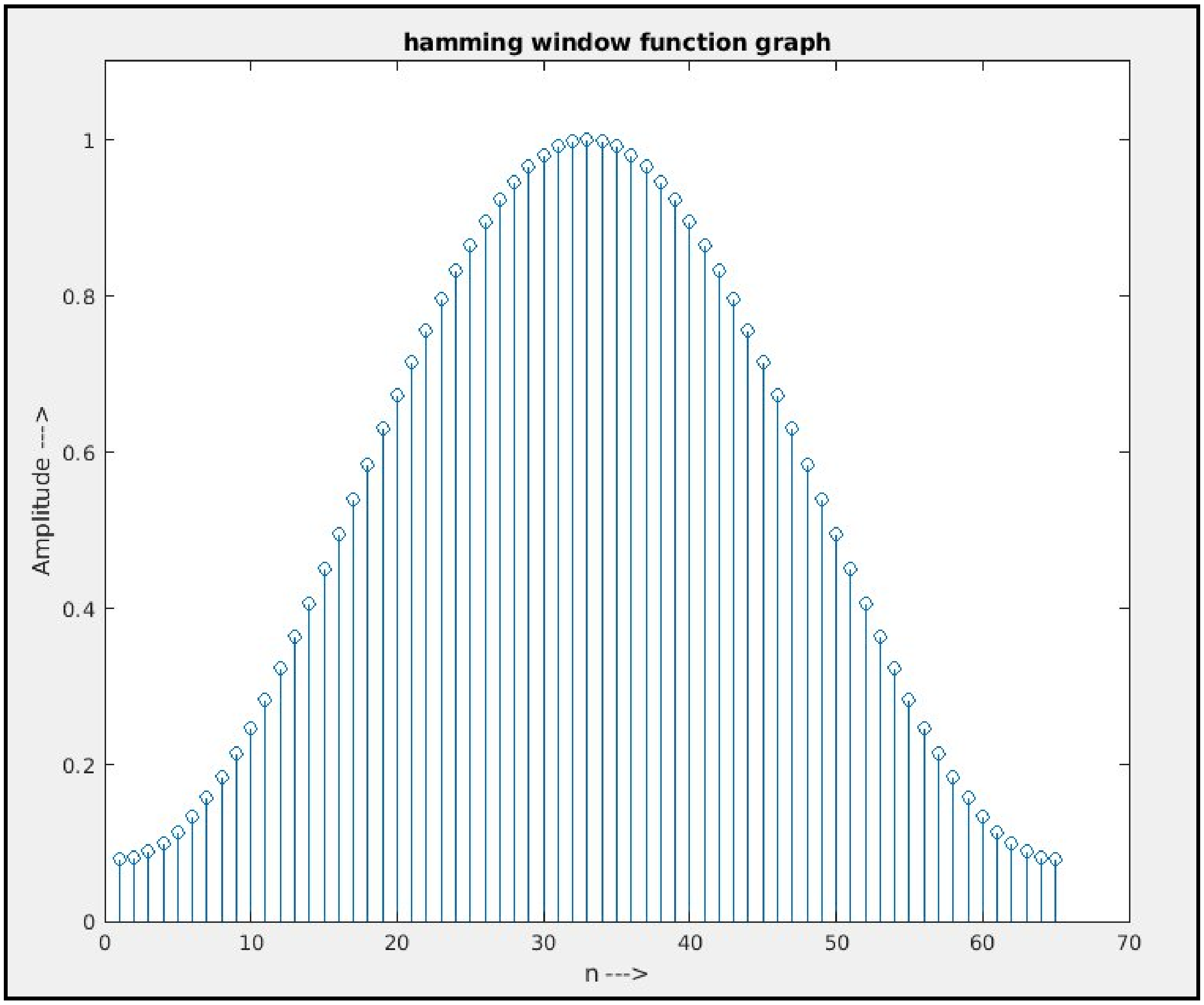
- This is the magnitude response of the given filter, here the y axis is plotted in decibel scale.
- From the plot it is clearly visible that the filter has a stop band attenuation of about -60 dB which meets the design constrains given in the question.
- Now we can conclude that we have created a filter which exactly matches the design constrains given in the question.

Part b code :

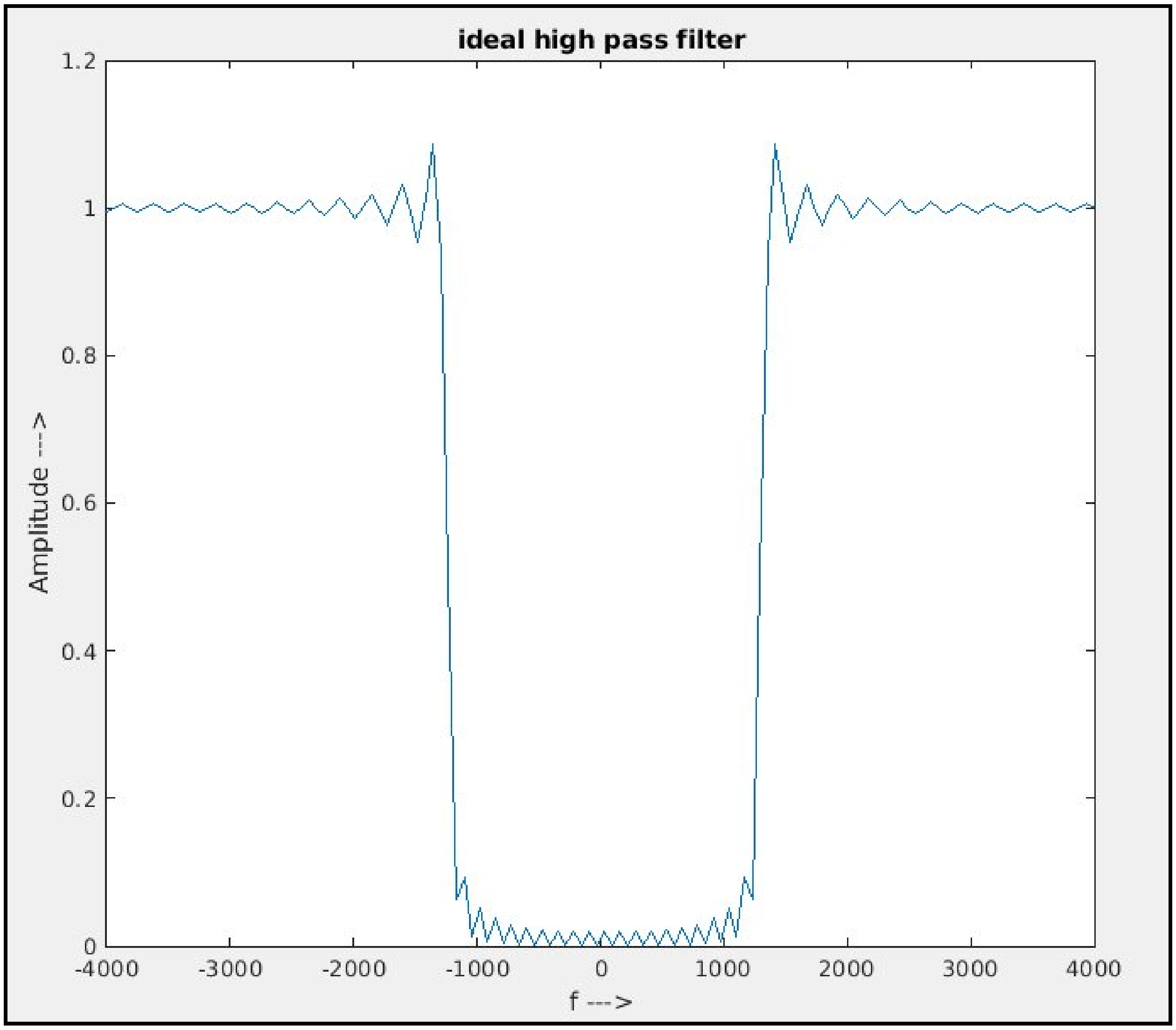


```matlab
fmin = 1000;
fmax = 1500;
fs = 8000;
fi = linspace(-pi,pi,128);
f = linspace(0,pi/2,512);

Wmin = 2*pi*fmin/fs;
Wmax = 2*pi*fmax/fs;
Wcutoff = (Wmin+Wmax)/2;
N = ceil(8*pi/(Wmax-Wmin))+1;

n = 0:N-1;
hamming = 0.54 - 0.46*cos(2*pi*n/(N-1));
stem(hamming)
xlabel('n --->')
ylabel('Amplitude --->')
title('hamming window function graph')
ylim([0,1.1]);

ideal = sinc(n-(N-1)/2) - Wcutoff*sinc(Wcutoff*(n-(N-1)/2)/pi)/pi;
filter = ideal .* hamming;
filter_dft = myfreqz(filter);

db =20*log10(abs(filter_dft));

figure
plot(fi*fs/(2*pi),myfftshift(abs(myfft(ideal))))
xlabel('f --->')
ylabel('Amplitude --->')
title('ideal high pass filter')

figure
subplot(2,1,1)
plot(f*fs/(pi),(abs(filter_dft)))
xlabel('f --->')
ylabel('Amplitude --->')
title('Designed filter - magnitude plot')
ylim([0,1.2]);

subplot(2,1,2)
plot(f*fs/(pi),unwrap(calc_phase(filter_dft)))
xlabel('f --->')
ylabel('Phase --->')
title('Designed filter - phase plot')

figure
plot(f*fs/(pi),(db));
xlabel('f --->')
ylabel('Amplitude in dB --->')
title('designed filter frequency response in dB')
```

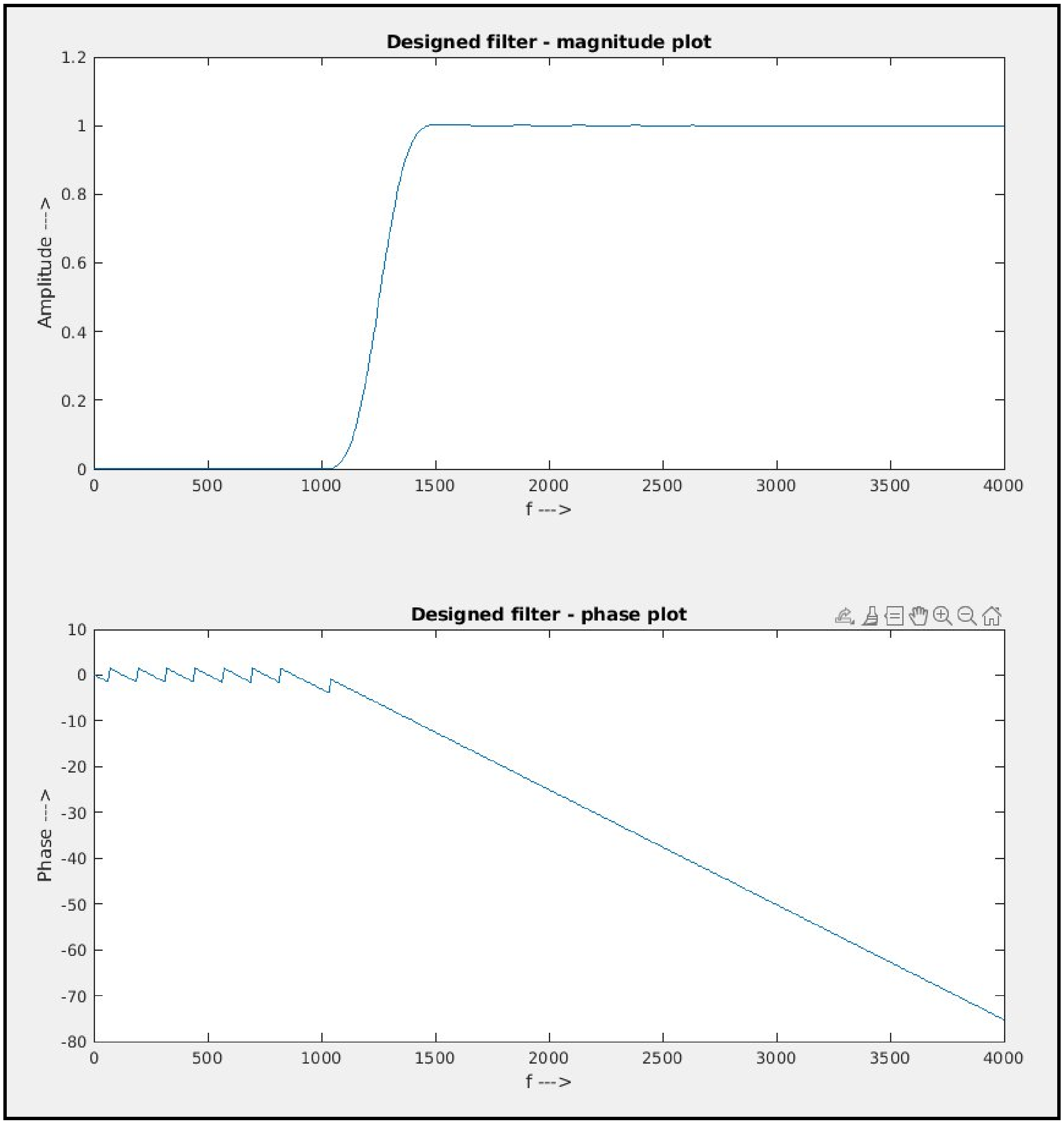
Output :



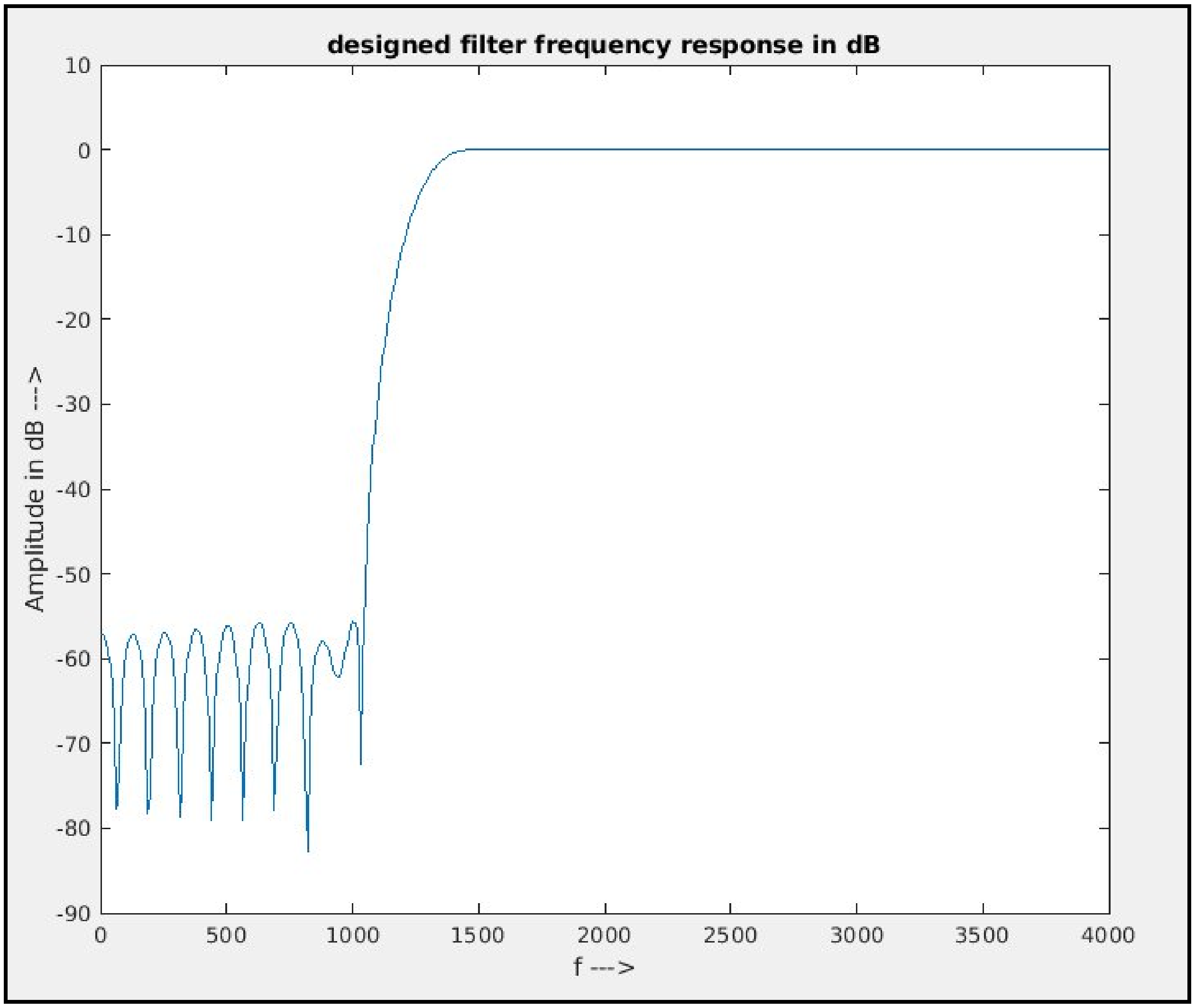
- This is the plot of the hamming window function.



- This is the plot of the ideal high-pass filter with Pass band edge frequency = 1500Hz and Stop band edge frequency = 1000Hz.
- To remove these ripples in the passband and stopband we have to multiply this function with the window function given above.



- This is the magnitude and phase response of the high pass filter that we designed using the hamming window function.
- Here we can clearly see that this filter meets all the filter specifications given in the question like passband edge frequency = 1500Hz and stopband edge frequency = 1000Hz.
- The phase response of the filter is also linear in the pass band, and hence it is a linear phase filter.



- Here we have plotted the magnitude response of the filter with y-axis in dB
- We can see that here we have a stop band attenuation of -60dB which is matching with the problem statement given in the question.
- From all these plots we can say that we have successfully designed a filter with
  Pass band edge frequency = 1500Hz
  Stop band edge frequency = 1000Hz
  Minimum stop band attenuation = 50 dB
  Maximum pass band attenuation = 0.9 dB
  Sampling Frequency = 8000Hz
- So we can conclude that the designed filter exactly matches the design constrains given in the question.

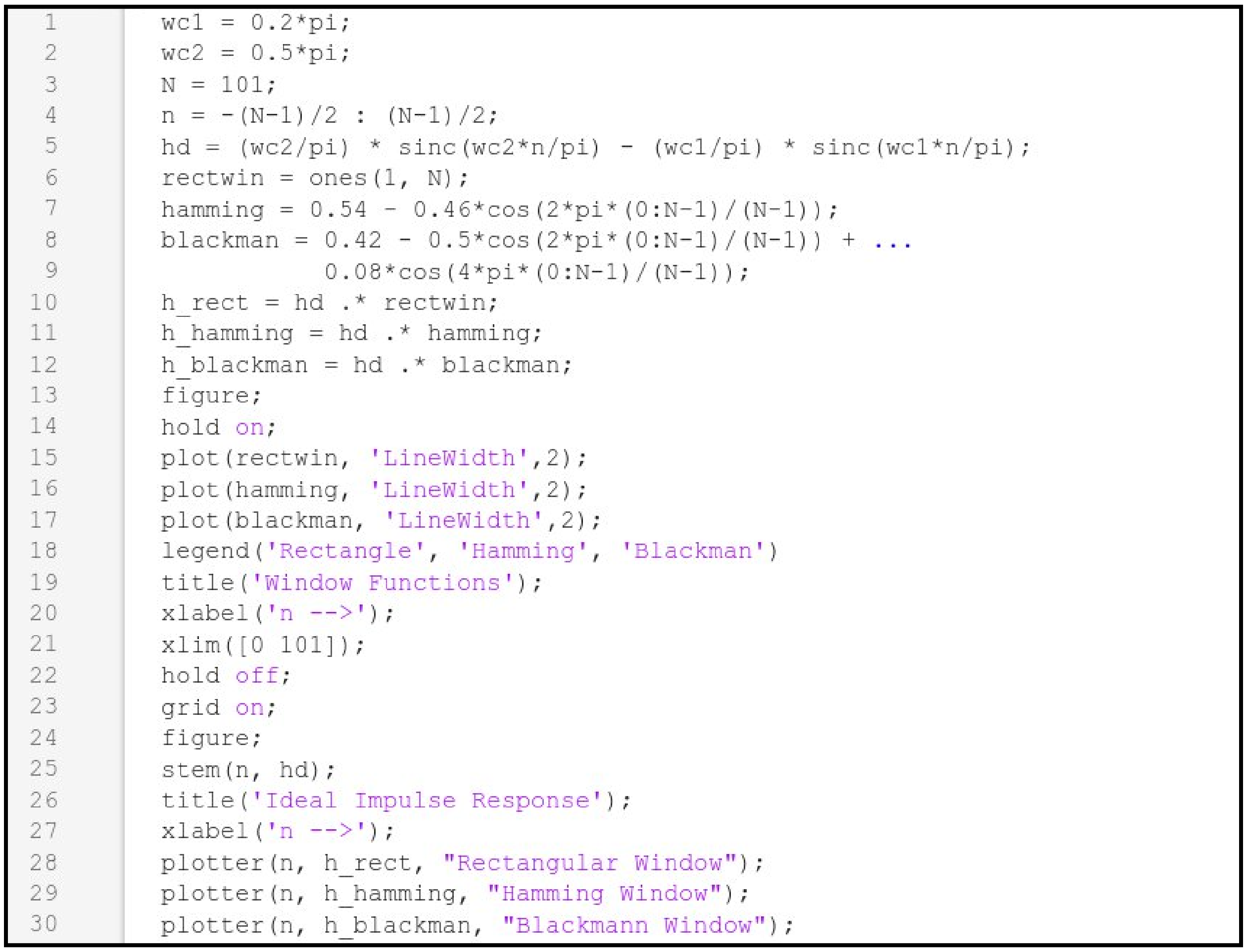
4. An FIR filter having the following specifications is to be designed using window method. Design the given filter using Rectangular, Hamming and Blackman windows. Plot the window function, ideal impulse response and implulse response of the designed filter. Also plot the magnitude and phase spectrum of the designed filter. Compare the performance of the filters designed using different windows and give appropriate comments based on your results.

$$-1 \le \left|H(e^{j\omega})\right|_{dB} \le 0 \qquad \text{for} \qquad 0.2\pi \le \omega \le 0.5\pi$$

$$\left|H(e^{j\omega})\right|_{dB} \le -60 \qquad \text{for} \qquad 0 \le \omega \le 0.15\pi \text{ and } \; 0.7\pi \le \omega \le \pi$$

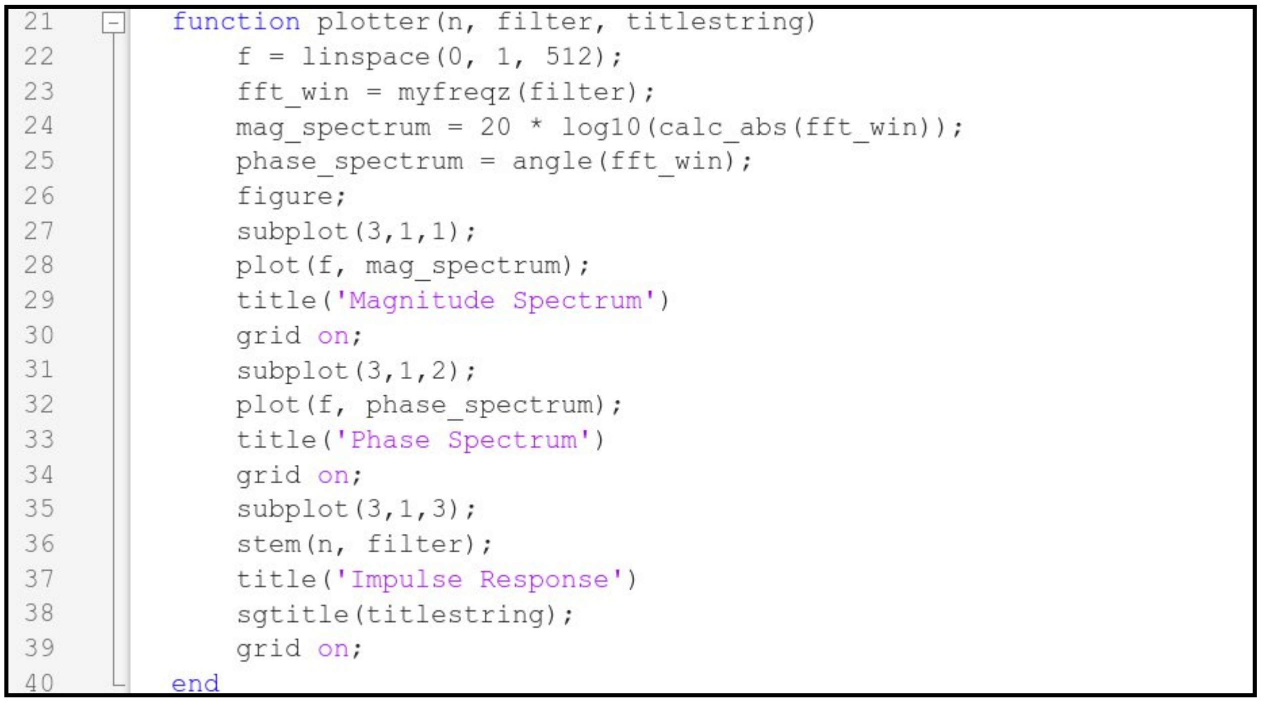
Code:

- We’re asked to design a bandpass FIR filter with cutoff frequencies 0.2 π and 0.5 π.



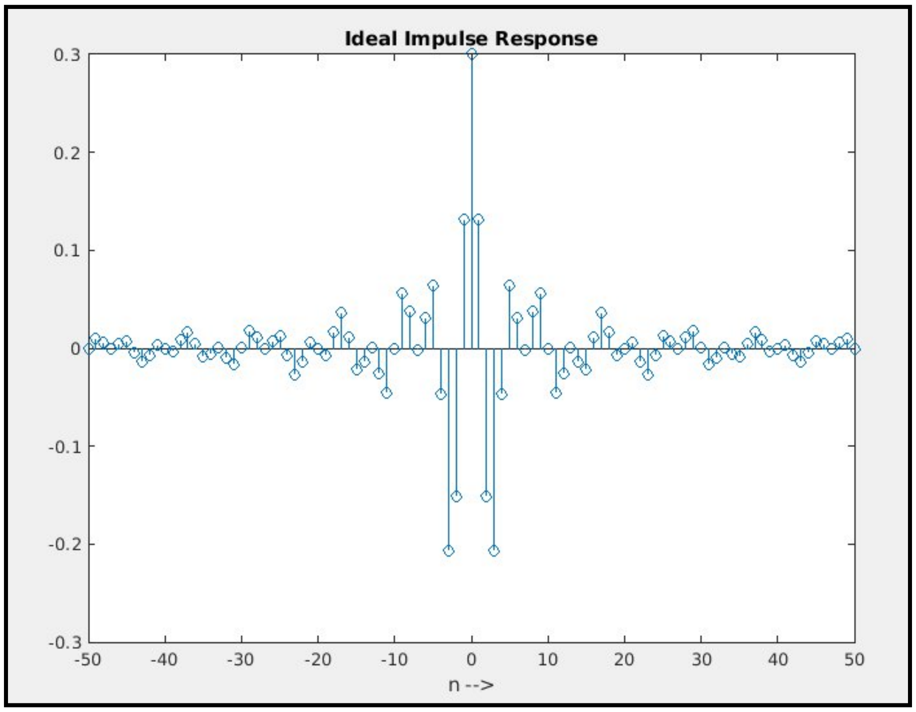
```
wc1 = 0.2*pi;
wc2 = 0.5*pi;
N = 101;
n = -(N-1)/2 : (N-1)/2;
hd = (wc2/pi) * sinc(wc2*n/pi) - (wc1/pi) * sinc(wc1*n/pi);
rectwin = ones(1, N);
hamming = 0.54 - 0.46*cos(2*pi*(0:N-1)/(N-1));
blackman = 0.42 - 0.5*cos(2*pi*(0:N-1)/(N-1)) + ...
           0.08*cos(4*pi*(0:N-1)/(N-1));
h_rect = hd .* rectwin;
h_hamming = hd .* hamming;
h_blackman = hd .* blackman;
figure;
hold on;
plot(rectwin, 'LineWidth',2);
plot(hamming, 'LineWidth',2);
plot(blackman, 'LineWidth',2);
legend('Rectangle', 'Hamming', 'Blackman')
title('Window Functions');
xlabel('n -->');
xlim([0 101]);
hold off;
grid on;
figure;
stem(n, hd);
title('Ideal Impulse Response');
xlabel('n -->');
plotter(n, h_rect, "Rectangular Window");
plotter(n, h_hamming, "Hamming Window");
plotter(n, h_blackman, "Blackmann Window");
```

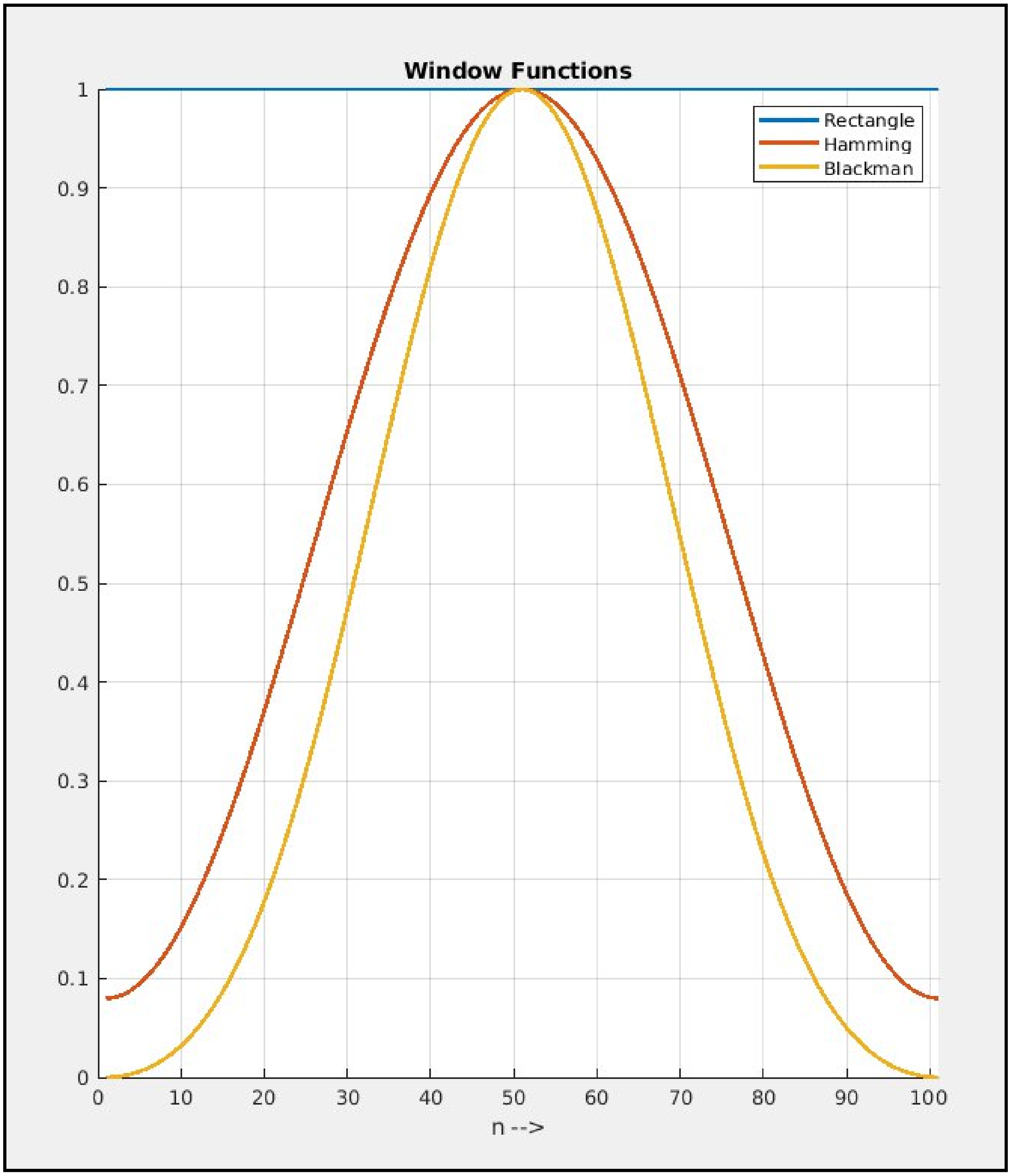
- A custom function plotter is defined to find the magnitude and phase spectrums of the filters and plot them.



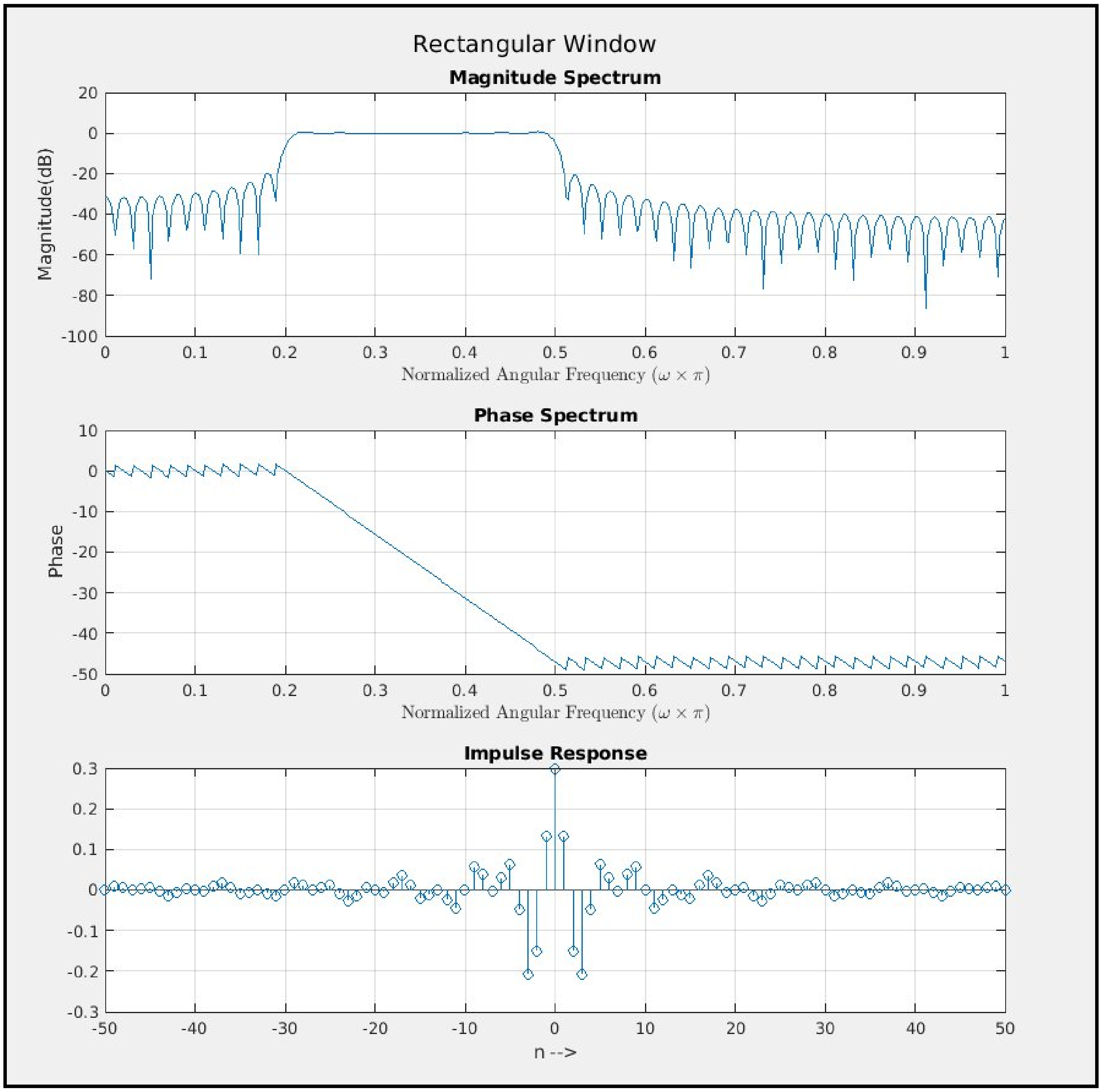
```
function plotter(n, filter, titlestring)
    f = linspace(0, 1, 512);
    fft_win = myfreqz(filter);
    mag_spectrum = 20 * log10(calc_abs(fft_win));
    phase_spectrum = angle(fft_win);
    figure;
    subplot(3,1,1);
    plot(f, mag_spectrum);
    title('Magnitude Spectrum')
    grid on;
    subplot(3,1,2);
    plot(f, phase_spectrum);
    title('Phase Spectrum')
    grid on;
    subplot(3,1,3);
    stem(n, filter);
    title('Impulse Response')
    sgtitle(titlestring);
    grid on;
end
```

Outputs:

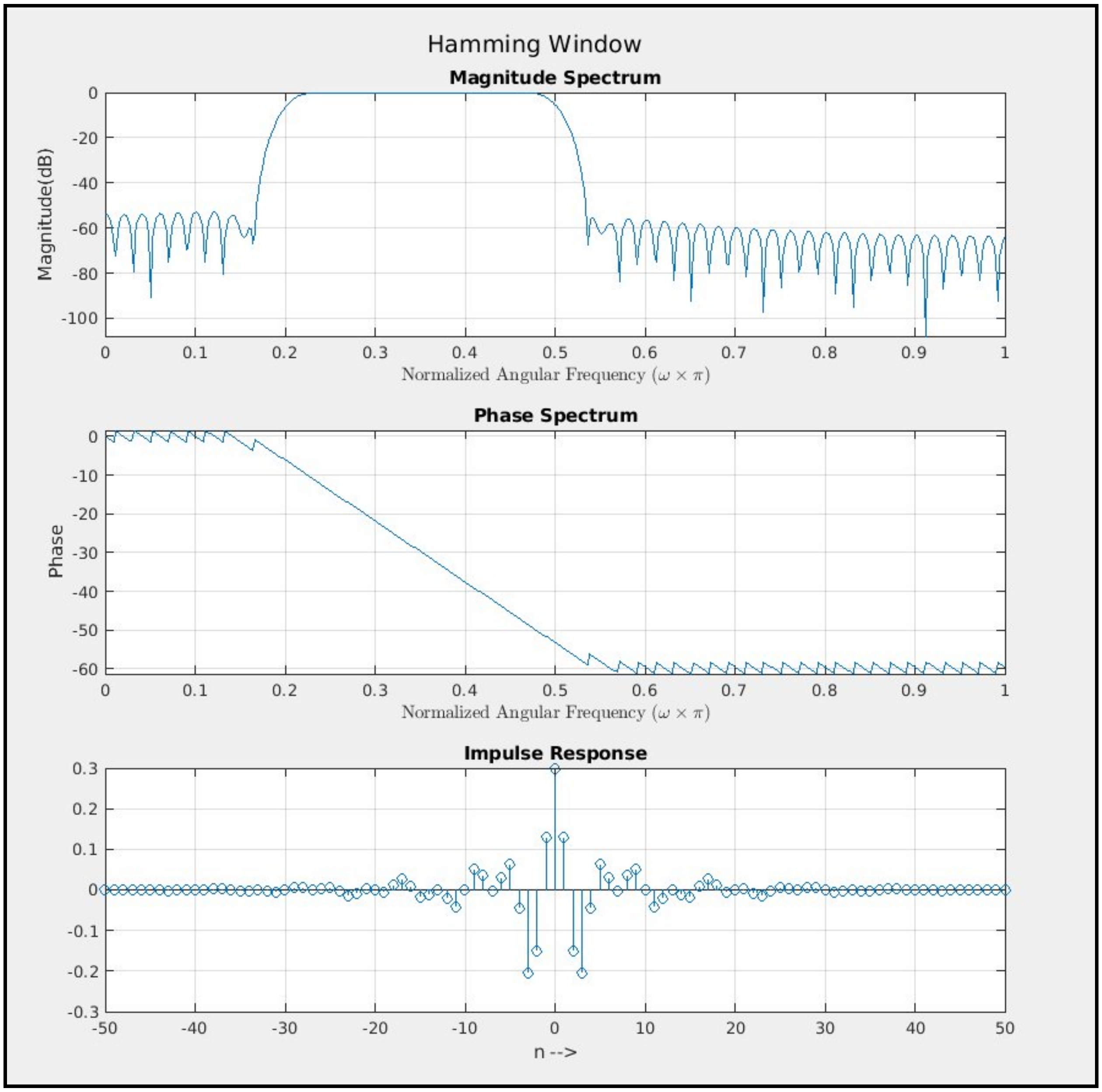




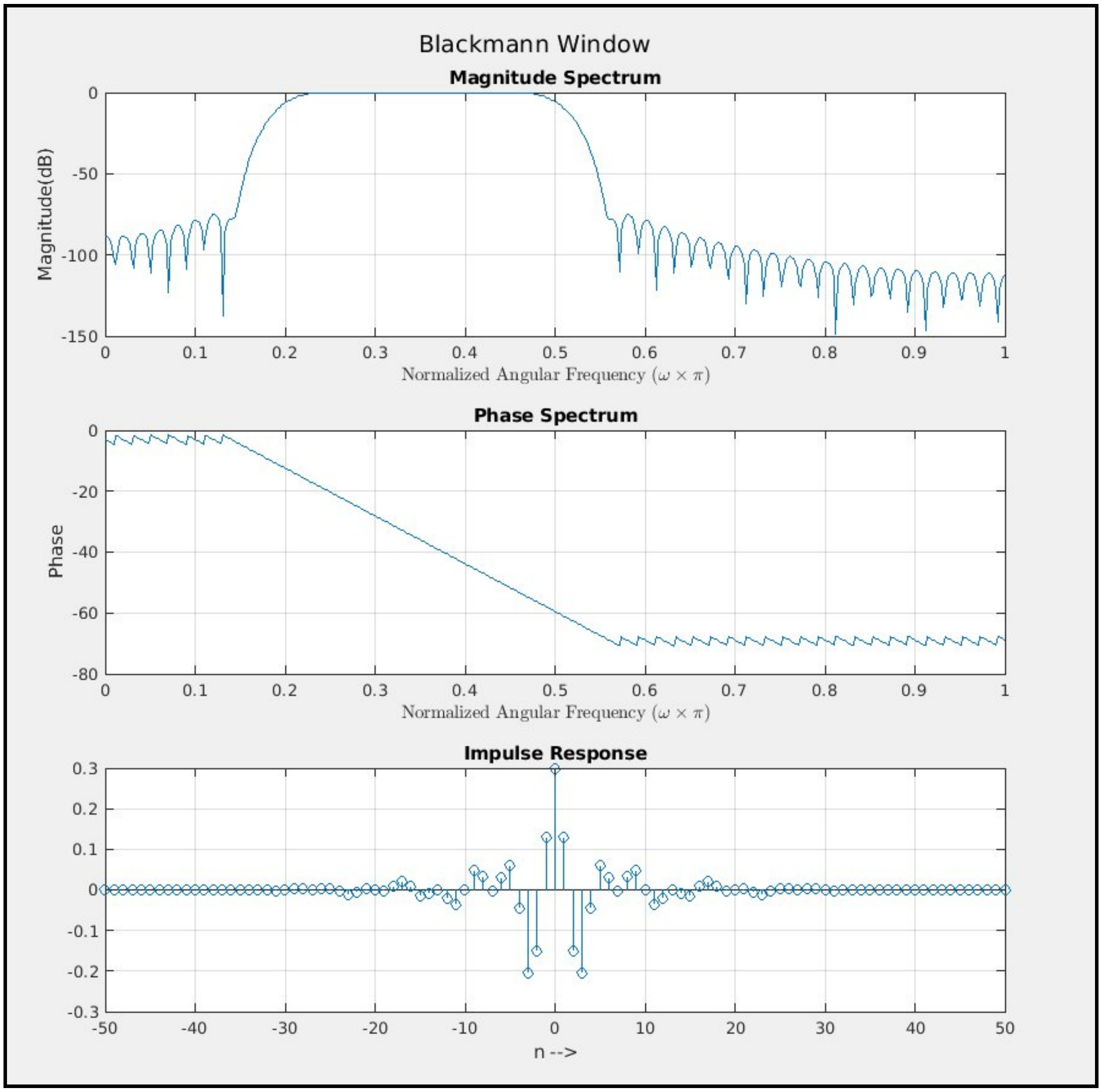
- In order to smoothen the ideal filter, we are multiplying it with the window functions



- The cutoff frequencies are clearly at $0.2\ \pi$ and $0.5\ \pi$ as required.



- The cutoff frequencies are clearly at $0.2\,\pi$ and $0.5\,\pi$ as required.



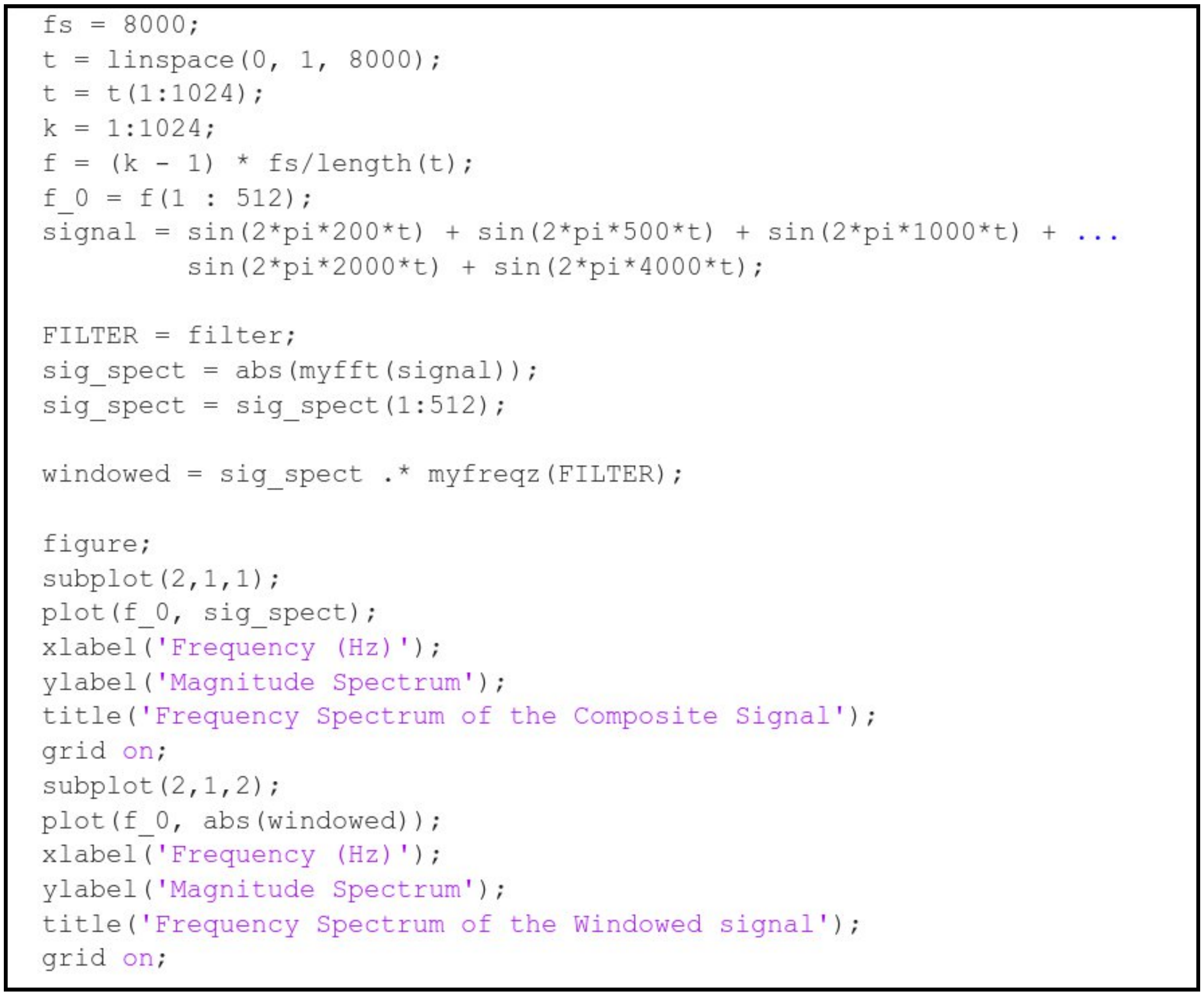
- The cutoff frequencies are clearly at $0.2\,\pi$ and $0.5\,\pi$ as required.

Performance Analysis of filters			
Window Used	**Main Lobe Width**	**Side Lobe attenuation**	**Transition Bandwidth**
Rectangular Window	**0.287π**	**-20dB**	**0.0215π**
Hamming Window	**0.284π**	**-54.41dB**	**0.0644π**
Blackmann Window	**0.28π**	**-74.92dB**	**0.0947π**

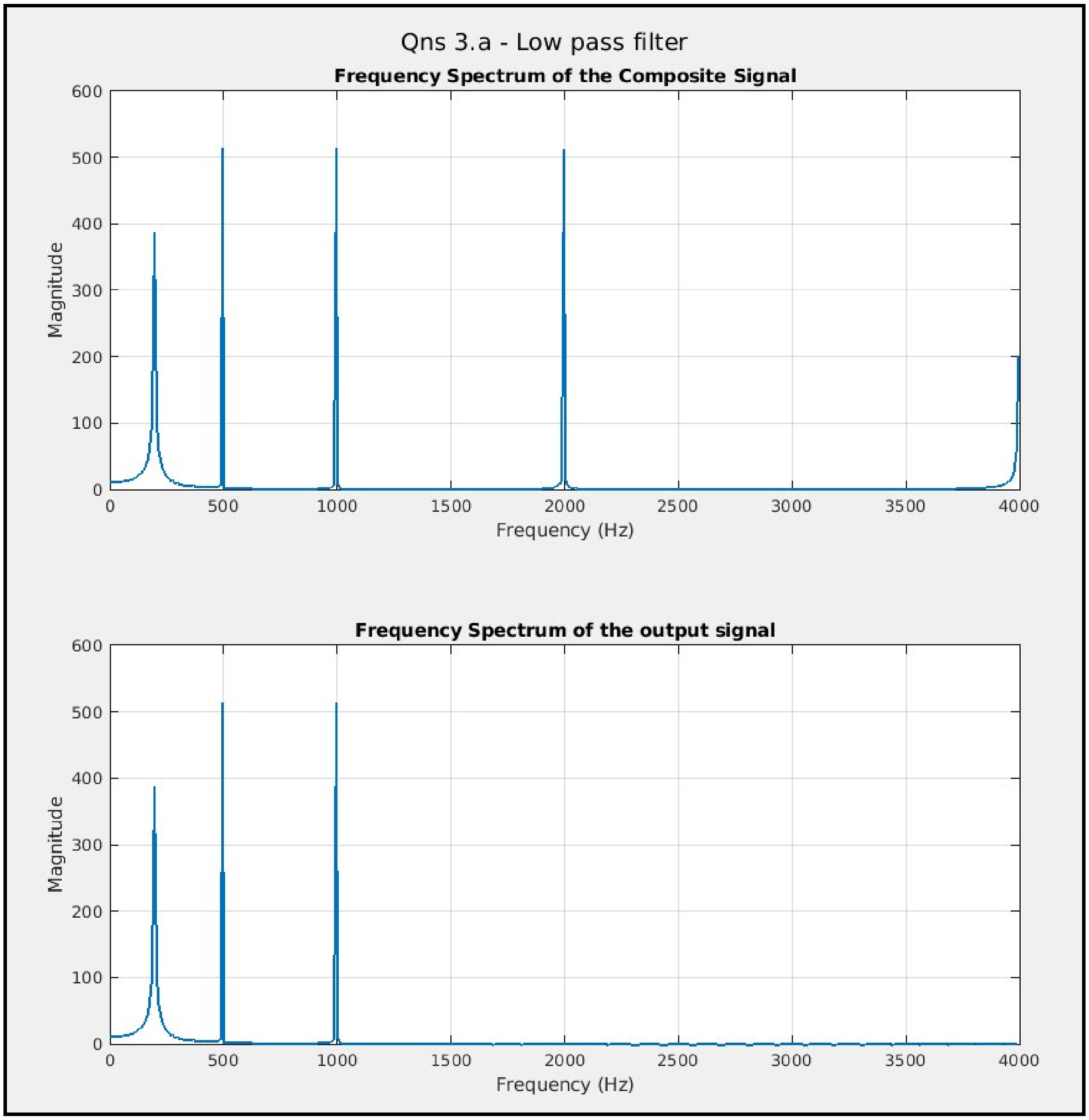
5. Generate a composite signal by adding sinusoids of different frequencies (200Hz, 500Hz,1000Hz, 2000Hz, 4000Hz). Compute the output of the filters designed in Qns. 3 and 4 by giving the composite signal as input and verify the design.

Code:

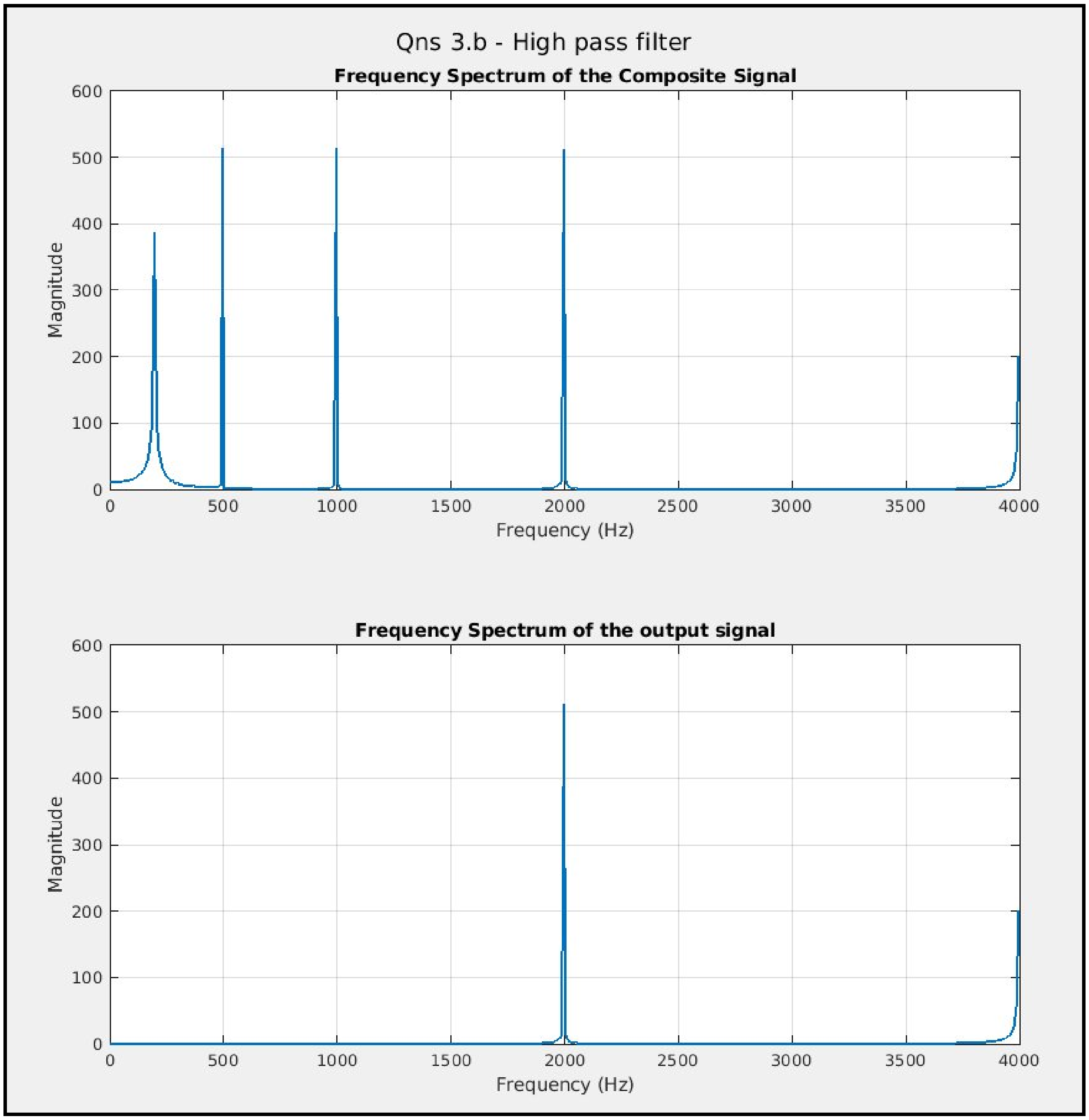
- The sampling frequency is taken as 8000Hz.
- FILTER is the variable corresponding to the filters from each question.



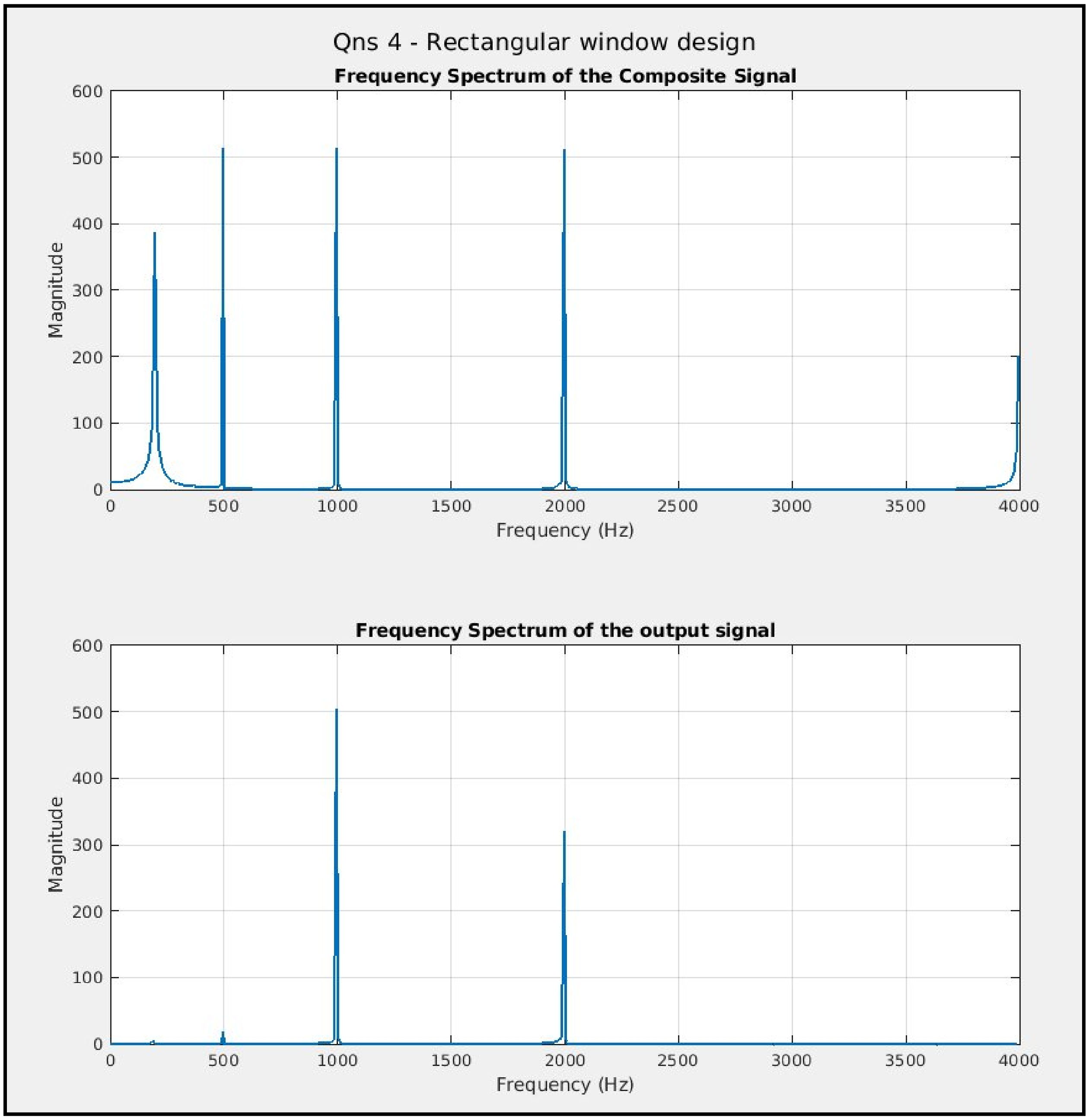
```
fs = 8000;
t = linspace(0, 1, 8000);
t = t(1:1024);
k = 1:1024;
f = (k - 1) * fs/length(t);
f_0 = f(1 : 512);
signal = sin(2*pi*200*t) + sin(2*pi*500*t) + sin(2*pi*1000*t) + ...
         sin(2*pi*2000*t) + sin(2*pi*4000*t);

FILTER = filter;
sig_spect = abs(myfft(signal));
sig_spect = sig_spect(1:512);

windowed = sig_spect .* myfreqz(FILTER);

figure;
subplot(2,1,1);
plot(f_0, sig_spect);
xlabel('Frequency (Hz)');
ylabel('Magnitude Spectrum');
title('Frequency Spectrum of the Composite Signal');
grid on;
subplot(2,1,2);
plot(f_0, abs(windowed));
xlabel('Frequency (Hz)');
ylabel('Magnitude Spectrum');
title('Frequency Spectrum of the Windowed signal');
grid on;
```



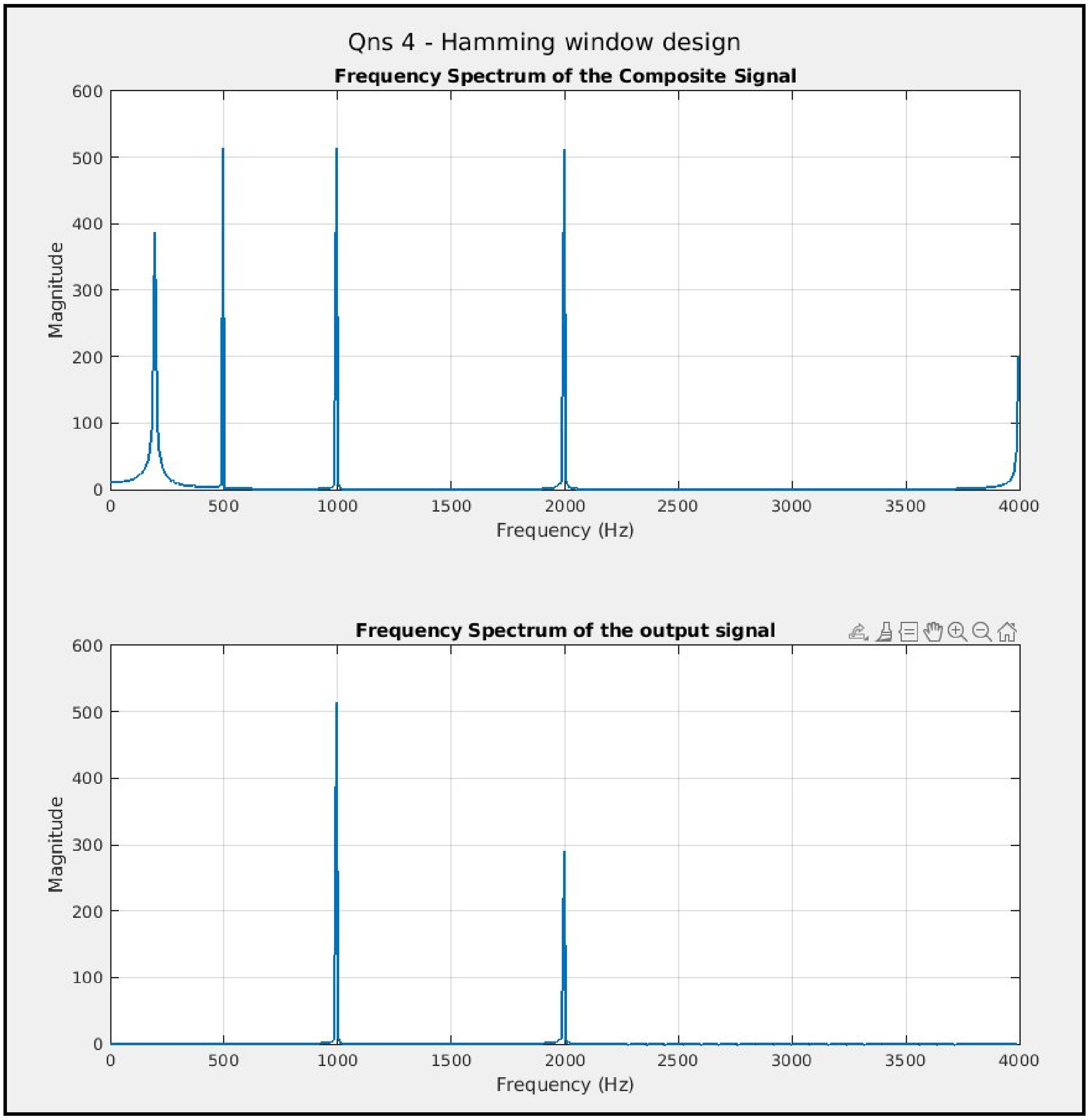
- Filter specification – Low pass filter with Pass band edge frequency at 1000Hz
- Expected Frequency components - 200Hz, 500Hz, 1000Hz
- Verified from output



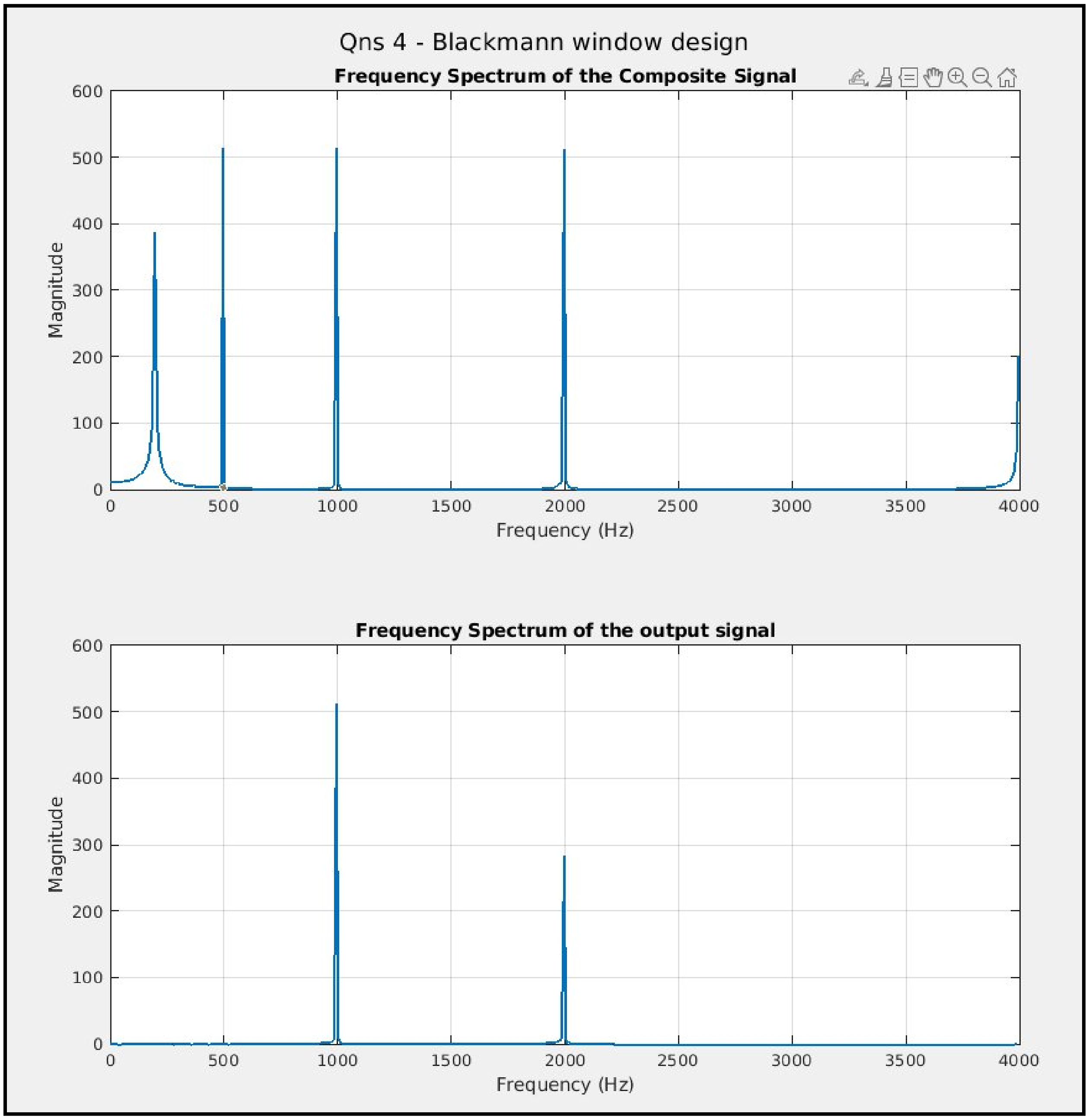
- Filter specification – Low pass filter with Pass band edge frequency at 1500Hz
- Expected Frequency components - 2000Hz, 4000Hz
- Verified from output



- Filter specification – Band pass filter with Cutoff frequency at 0.2π (800Hz) and 0.5π (2000Hz).
- Expected Frequency components - 1000Hz, 2000Hz
- Verified from output



- Filter specification – Band pass filter with Cutoff frequency at 0.2π (800Hz) and 0.5π(2000Hz)
- Expected Frequency components - 1000Hz, 2000Hz
- Verified from output



- Filter specification – Band pass filter with Cutoff frequency at 0.2π (800Hz) and 0.5π(2000Hz)
- Expected Frequency components - 1000Hz, 2000Hz
- Verified from output

Therefore, the filters are all working as per the specified design parameters.