DEPARTMENT OF ELECTRONIC AND TELECOMMUNICATION ENGINEERING
UNIVERSITY OF MORATUWA

MUDALIGE DINETH NAVODYA

170401V

EN 2570 - DIGITAL SIGNAL PROCESSING

FIR band-stop filter design

This is submitted as a partial fulfillment for the module
EN 2570 - Digital Signal Processing

—

December 26, 2019

Abstract

This is report discusses the complete procedure needed in designing a Finite-Duration Impulse Response (FIR) bandstop filter for a set of given specifications and its implementation using MATLAB.Windowing method or the Fourier series method is used in implementing the filter as a closed form method using the predefined equations[1]. The Kaiser Window function is used to achieve the transfer function. The report discusses about the magnitude responses of the designed filter. Later it assesses the performance based on sinusoidal input. Finally it is compared to an ideal stopband filter with the same specifications.

Contents

List of Figures

List of Tables

1 Introduction

This project describes the process of designing a FIR bandstop filter using the Kaiser window method in a stepwise manner. The software implementation is done using MATLAB. The Kaiser window method is a closed form method. The final result is analysed based on an input signal and an ideal stopband filter.

2 Method

Question 1

2.1 Specifications

2.1.1 Required specifications

Parameter	Value
Maximum passband ripple	0.07 dB
Minimum stopband attenuation	45 dB
Lower passband edge	500 $rads^{-1}$
Upper passband edge	1050 $rads^{-1}$
Lower stopband edge	600 $rads^{-1}$
Upper stopband edge	900 $rads^{-1}$
Sampling frequency	2800 $rads^{-1}$

Table 1: The required parameters

The required parameters are generated for the index number by the Listing[1] in the Appendix[A].

2.1.2 Derived specifications

Parameter	Value
Lower transition width	100 $rads^{-1}$
Upper transition width	150 $rads^{-1}$
Critical transition width	100 $rads^{-1}$
Lower cut-off frequency	550 $rads^{-1}$
Upper cut-off frequency	1000 $rads^{-1}$
Sampling period	2.24×10^{-3} s
Actual stopband attenuation	47.89 dB
Actual passband ripple	0.07 dB
Actual sideband ripple	47.89 dB
Filter order	79

Table 2: The derived parameters

The Listing[2] in Appendix[A] is used in obtaining these values.

2.2 Kaiser Window

The Kaiser Window function is given by

$$w_K(nT) = \begin{cases} \frac{I_0(\beta)}{I_0(\alpha)} & |x| \leq \frac{N-1}{2} \\ 0 & otherwise \end{cases}$$

where N is the order of the filter, α is an independent parameter and

$$\beta = \alpha\sqrt{1-\left(\frac{2n}{N-1}\right)^2} \qquad\qquad I_0(\alpha) = 1 + \sum_{k=1}^{\infty}\left[\frac{1}{k!}\left(\frac{x}{2}\right)^k\right]^2$$

$$\delta = min\left(\tilde{\delta_p}, \tilde{\delta_a}\right)$$

where

$$\tilde{\delta_p} = \frac{10^{0.05\tilde{A_p}} - 1}{10^{0.05\tilde{A_p}} + 1} \qquad and \qquad \tilde{\delta_a} = 10^{0.05\tilde{A_a}} - 1$$

Now, with the defined δ, we calculate the actual stop band loss

$$A_a = -20log|x|$$

and the actual pass band ripple

$$A_p = 20log\frac{|1+\delta|}{|1-\delta|}$$

We can chose α as

$$\alpha = \begin{cases} 0 & A_a \leq 21dB \\ 0.5842(A_a-21)^{0.4} + 0.07886(A_a-21) & 21 < A_a \leq 50dB \\ 0.1102(A_a-8.7) & A_a > 50dB \end{cases}$$

A parameter D is chosen in order to obtain N, as

$$D = \begin{cases} 0.9222 & for\ A_a \leq 21dB \\ \frac{A_a-7.95}{14.36} & for\ A_a > 21dB \end{cases}$$

N is chosen such that it is the smallest odd integer value satisfying the inequality

$$N \geq \frac{\Omega_s D}{B_t} + 1$$

These calculations can be seen in the Listing[3].

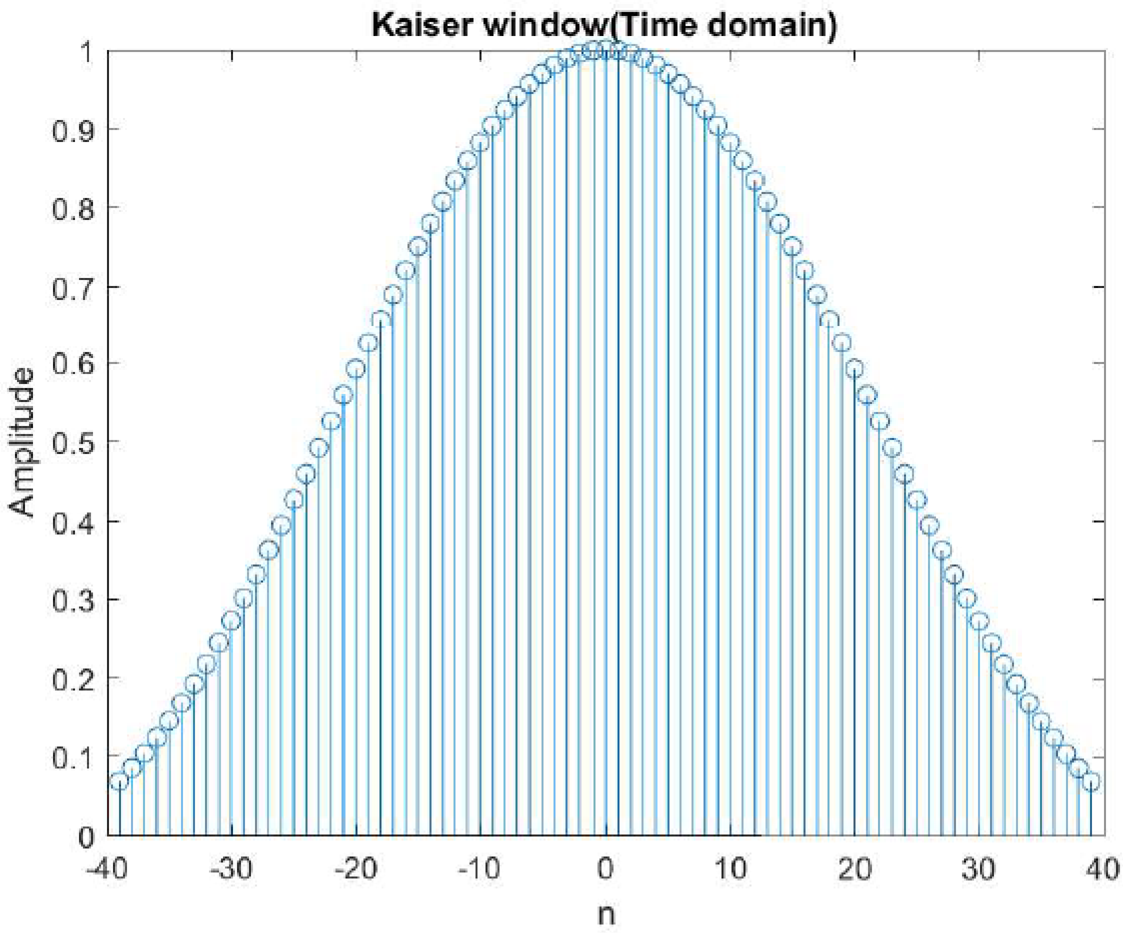

Figure 1: The Kaiser Window

2.3 Ideal impulse response filter

The frequency response of an ideal bandstop filter with cutoff frequencies Ω_{c1} and $\Omega c2$ is given by

$$H(e^{j\Omega T}) = \begin{cases} 1 & 0 \leq |\Omega| \leq \Omega_{c1} \\ 0 & \Omega_{c1} \leq |\Omega| \leq \Omega_{c2} \\ 1 & \Omega_{c2} \leq |\Omega| \leq \frac{\Omega_s}{2} \end{cases}$$

Applying the Fourier inverse transform for the above cases;

$$h(nT) = \frac{1}{\Omega_s} \int_{-\Omega_s/2}^{\Omega_s/2} H(e^{j\Omega T}) e^{j\Omega nT} d\Omega$$

$$h(nT) = \frac{1}{\Omega_s} \left[\int_{-\Omega_s/2}^{-\Omega_{c2}} e^{j\Omega nT} d\Omega + \int_{-\Omega_{c1}}^{0} e^{j\Omega nT} d\Omega + \int_{0}^{\Omega_{c1}} e^{j\Omega nT} d\Omega + \int_{\Omega_{c2}}^{\Omega_s/2} e^{j\Omega nT} d\Omega \right]$$

When $n \neq 0$:

$$h(nT) \ = \ \frac{2j}{jnT\Omega_s} \left[\sin \frac{\Omega_s}{2} nT + \sin \Omega_{c1} nT - \sin \Omega_{c2} nT \right]$$

As $\frac{\Omega_s}{2} T \ = \ \pi$

$$h(nT) = \frac{2j}{jnT\Omega_s}[\sin\Omega_{c1}nT - \sin\Omega_{c2}nT]$$

When n=0;

$$h(nT) = 1 + \frac{2}{\Omega_s}(\Omega_{c1} - \Omega_{c2})$$

The ideal impulse stopband filter is obtained based on Listing [4]

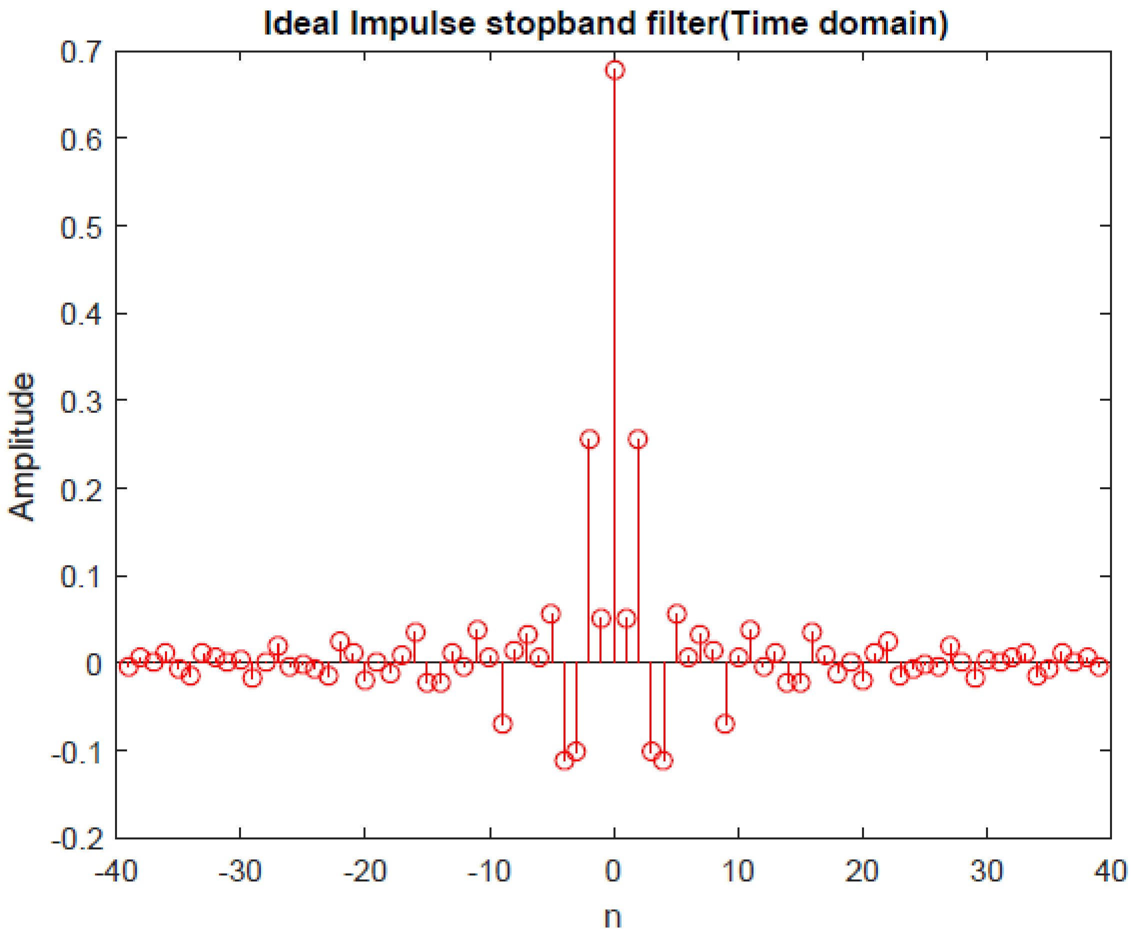

Figure 2: The ideal impulse stopband filter

2.4 The Non-causal filter

The finite order non-causal impulse response of the windowed filter $h_w(nT)$ by the multiplication of the Ideal impulse response $h(nT)$ by the Kaiser Window function $w_K(nT)$

$$h_w(nT) = w_K(nT)h(nT)$$

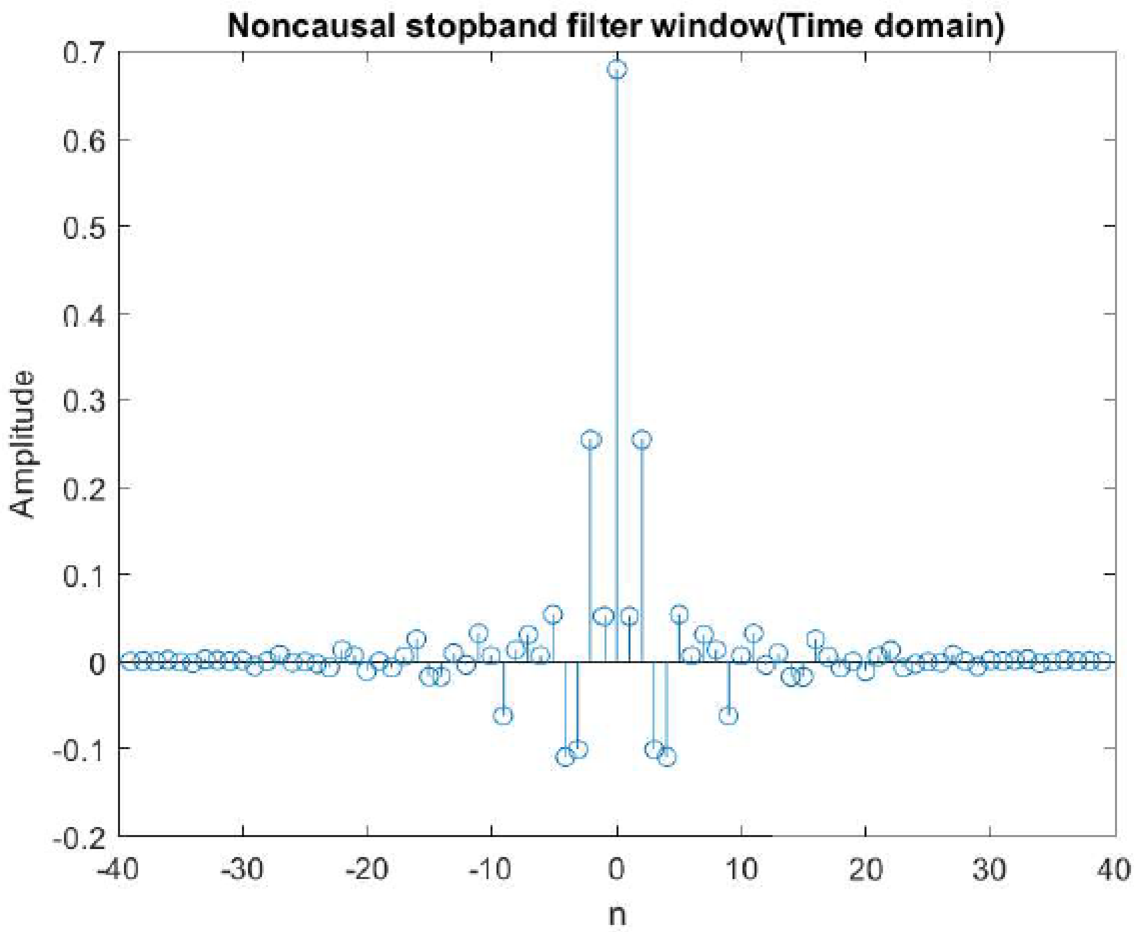

Figure 3: The non-causal stopband filter

Question 2

2.5 Causal stopband filter

The $\mathscr{Z}$-transform of $h_w(nT)$ should be obtained.

$H_w(z) = \mathscr{Z}[h_w(nT)] = \mathscr{Z}[w_K(nT)h(nT)]$
After shifting for causality it becomes

$$H'_w(z) = z^{-(N-1)/2} H_w(z)$$

3 Performance

Question 3

3.1 Magnitude response of the stopband filter

The magnitude response can be easily obtained from the fvtool function as seen on Figure[5] but can be also plotted using freqz as in Figure[4].

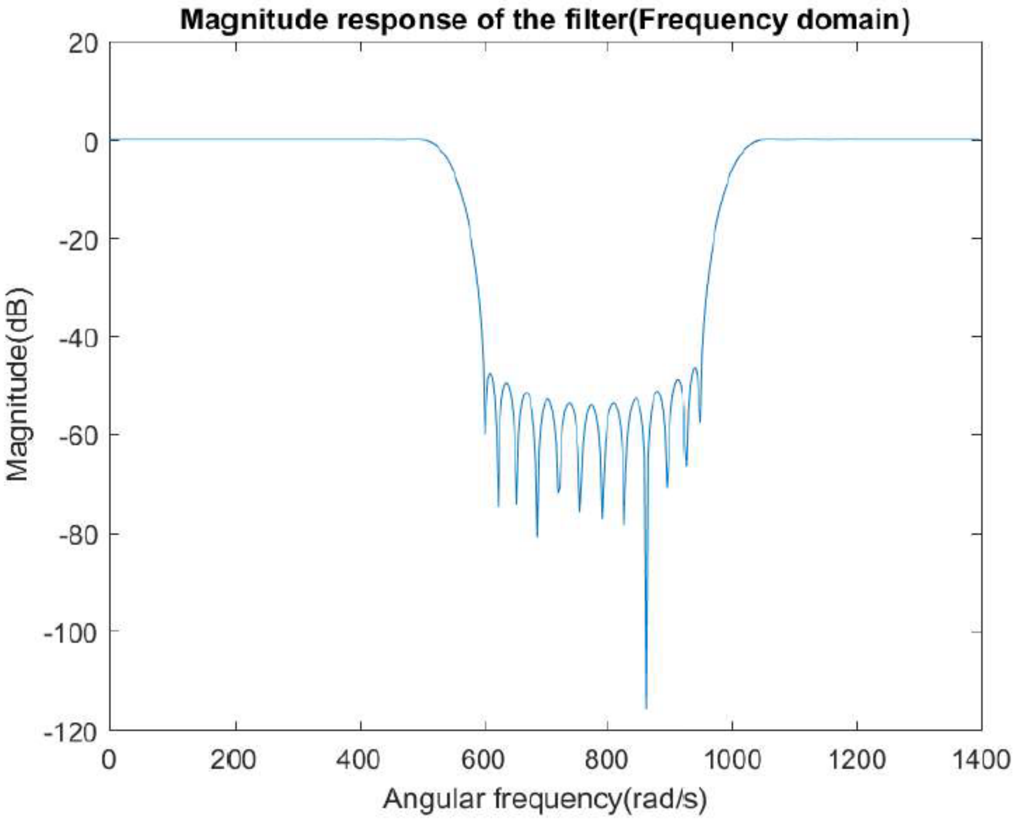

Figure 4: The magnitude response of the filter in frequency domain

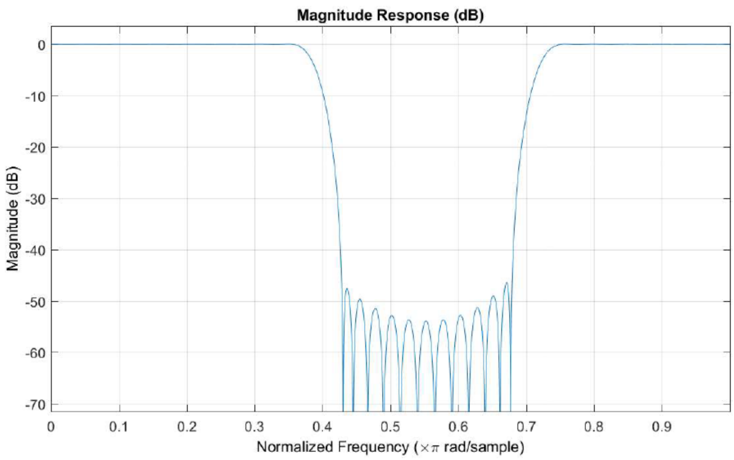

Figure 5: The magnitude response of the filter using fvtool function

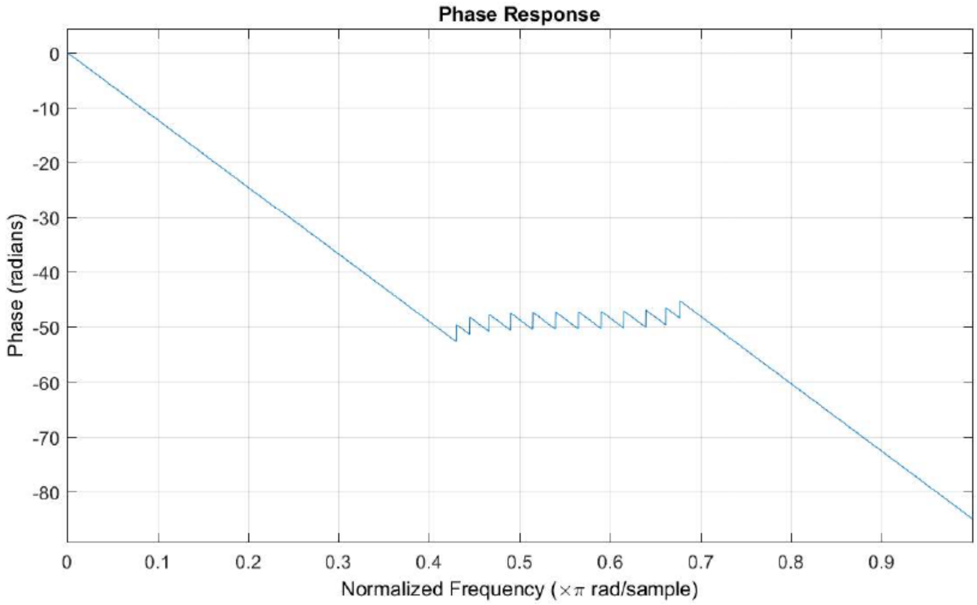

Figure 6: The phase response of the filter using fvtool function

Question 4

3.2 Magnitude responses of passbands

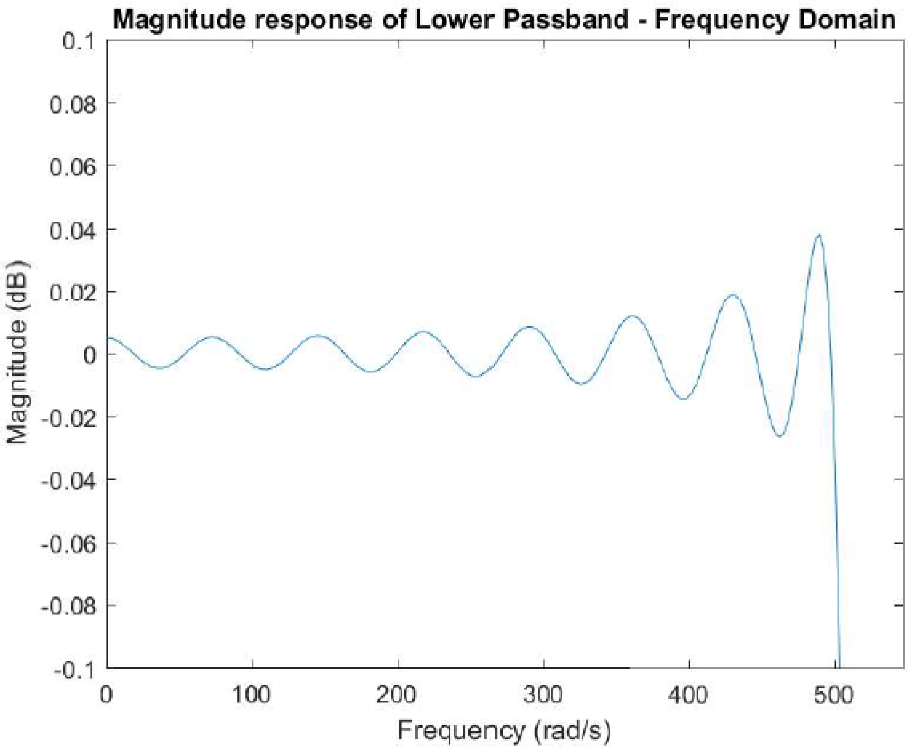

Figure 7: The magnitude response of lower passband in frequency domain

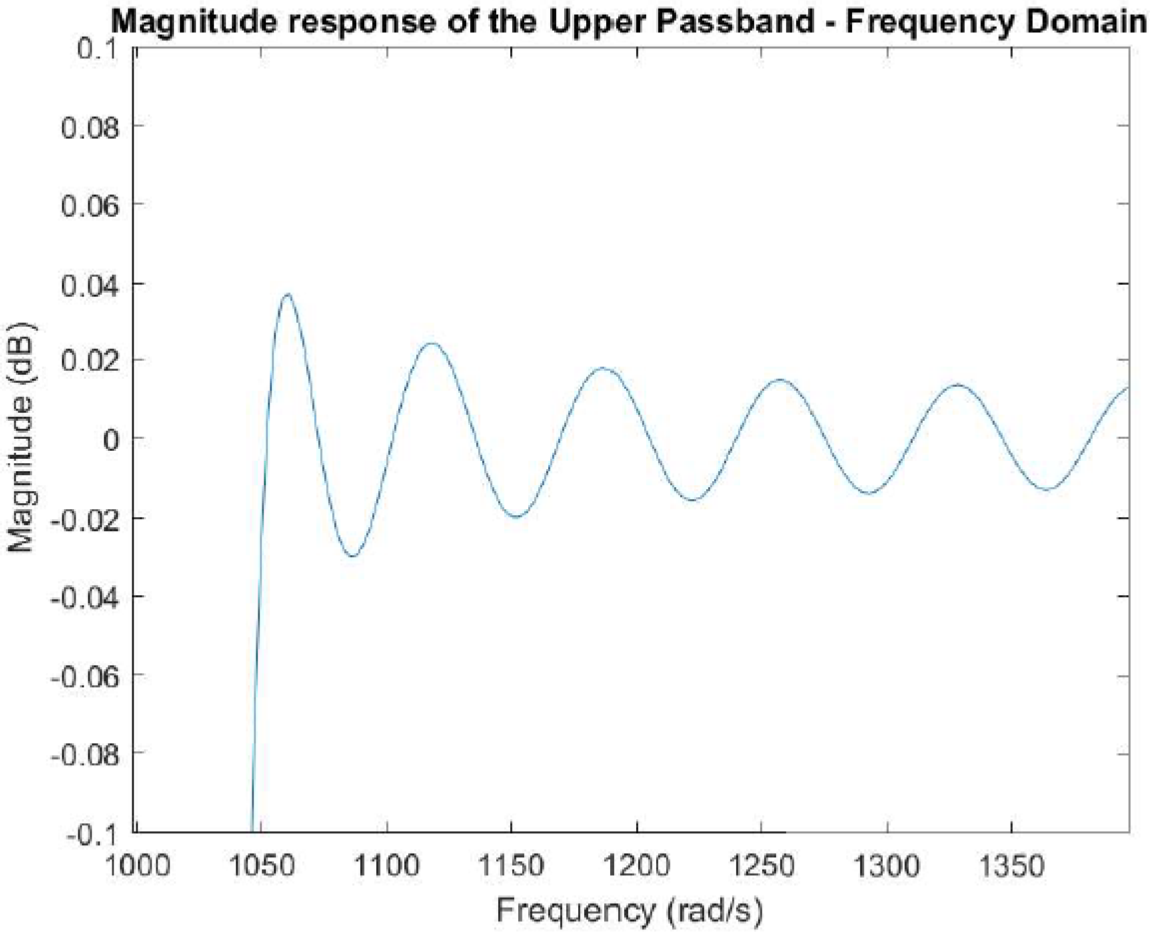

Figure 8: The magnitude response of the upper passband in frequency domain

4 Testing and evaluation

Question 5

To evaluate the performance of the designed filter, we use an input signal $x(nT)$ as in Figure[9] which is the sum of three sinusoidal signals, each of which has a frequency in the lower pass band, the stop band and the upper pass band as shown below.

$$x(nT) = \sum_{i=1}^{3} \cos(\omega_i nT)$$

- $\Omega_1 = \frac{\Omega_{c1}}{2} = 275 \text{ rad}s^{-1}$
- $\Omega_2 = \frac{\Omega_{c1}+\Omega_{c2}}{2} = 825 \text{ rad}s^{-1}$
- $\Omega_3 = \frac{\Omega_{c1}+\Omega_s/2}{2} = 975 \text{ rad}s^{-1}$

The way of obtaining the input signal is shown in Listing[5]. For the evaluation, I obtained 600 samples from the input signal.

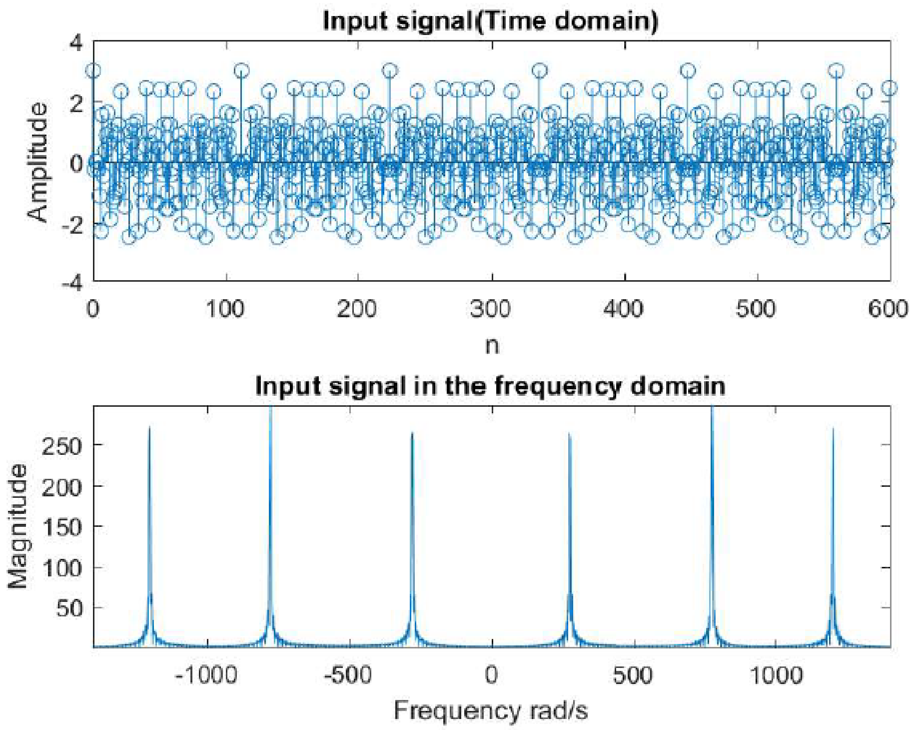

Figure 9: The input signal used for the evaluation of the filter

Question 6

The output of the designed filter for the input signal shown in Figure[9] is obtained using the FFT as it eliminates the need of convolution.

For the evaluation of the filter that we have designed, the output of the ideal filter is obtained based on the notion

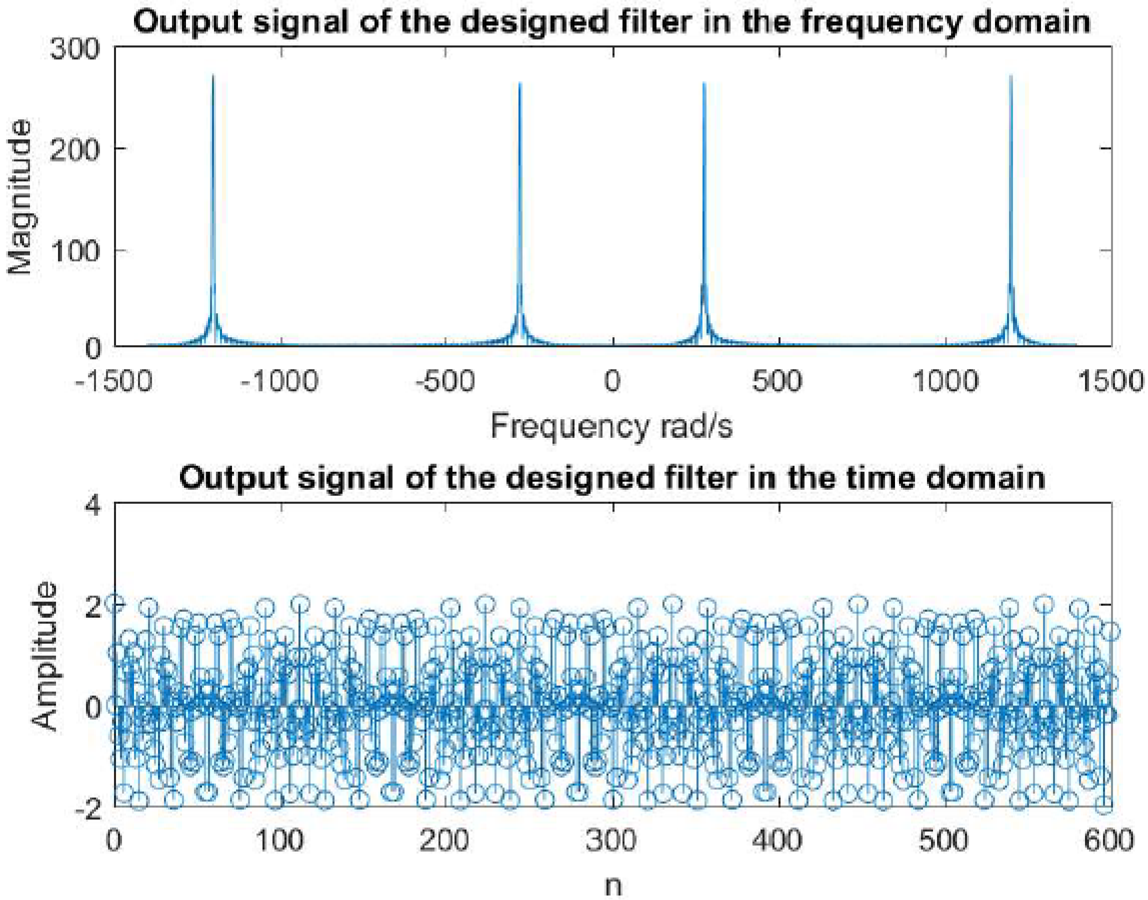

Figure 10: The output of the designed filter in time and frequency domains

that the ideal filter stops all frequencies in the stop band.

I obtained the absolute deviation between the output of the ideal filter and the designed filter as shown in the Figure[12]. The root mean square error(RMSE) between the two outputs was obtained as 0.01541.

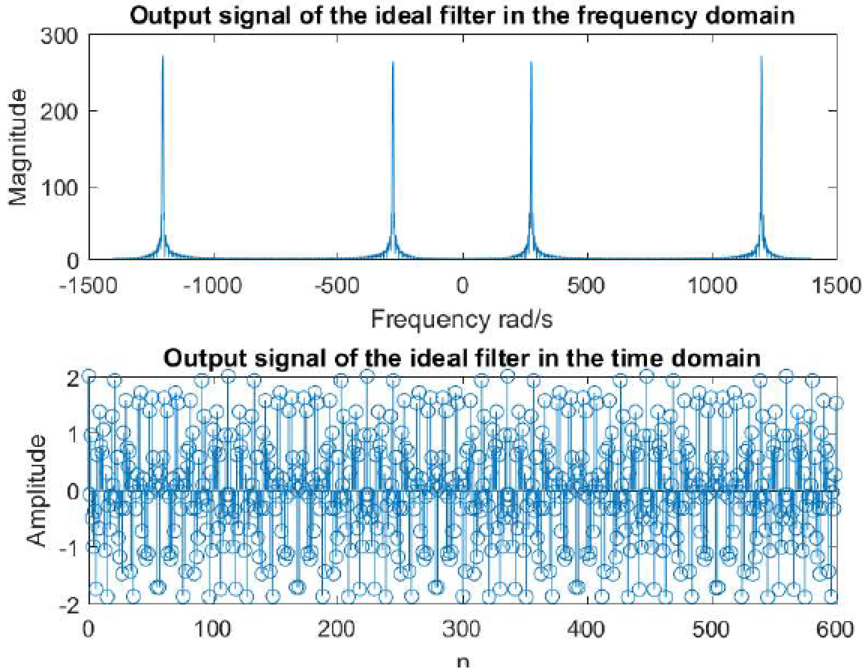

Figure 11: The output of the filter in frequency and time domains

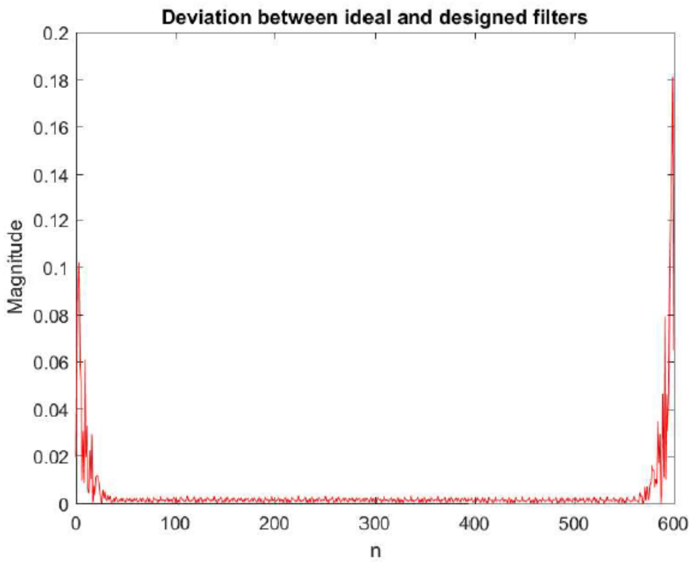

Figure 12: The deviation between the outputs of designed and ideal filters

5 Discussion

From the evaluation section it is clear that the filter has achieved its purpose. From Figure[4], we can see that the maximum stop band attenuation is around 48dB. The passband ripple is small as seen in Figures 7 and 8.

Apart from that, the comparison with the output of an ideal filter for the same input given in the Figure[9] justifies the design. The deviation is less as in the Figure[12] but there is a larger deviation at the edges. It is due to the truncation of the filter. There will be no such deviation if it extends to the infinity.

6 Conclusion

The Kaiser Window that is used for this stopband filter implementation has proved to be fruitful. It is a closed form method. Thus, it is easy to implement.

When looking at the filter specifications in the Table[2], the order of the implemented is high. It means that a large number of calculations needed to be performed. It becomes an issue in the hardware design as the complexity increases, making the design more complex and expensive.

References

[1] A. Antoniou, *Digital Signal Processing: Signals, Systems, and Filters*. McGraw-Hill Education, 2006, ISBN: 9780071454247. [Online]. Available: `https://books.google.lk/books?id=JQ4fAQAAIAAJ`.

A MATLAB codes

Listing 1: Generation of the required specification by the index number

```
function filterparams(index_num)
%Generates both given and derived parameters to implement the filter
global A B C;%extracts from the index number
global A_p;%max passband ripple
global A_a;%min stopband ripple
global O_p1;%lower passband edge
global O_p2;%upper passband edge
global O_a1;%lower stopband edge
global O_a2;%upper stopband edge
global O_s;%sampling frequency
%Deriving the information from the index number
A = mod(floor(index_num/100),10);
B = mod(floor(index_num/10),10);
C = mod(index_num,10);
A_p = 0.03+(0.01*A);
A_a = 45+B;
O_p1 = (C*100)+400;
O_p2 = (C*100)+950;
O_a1 = (C*100)+500;
O_a2 = (C*100)+800;
O_s = 2*((C*100)+1300);
fprintf('For the index number %d:\n.....The required parameters.....\n',index_num);
fprintf('Maximum passband ripple = %.2f\nMinimum stopband attenuation = %d\nLower passband edge = %d\n',A_p,A_a,O_p1);
fprintf('Upper passband edge = %d\nLower stopband edge = %d\nUpper stopband edge = %d\nSampling frequency = %d\n',O_p2,O_a1,O_a2,O_s);
```

Listing 2: Obtaining the specifications of the filter

```
function deriveparams
%Using the required specifications
global A_p;%max passband ripple
global A_a;%min stopband ripple
global O_p1;%lower passband edge
global O_p2;%upper passband edge
global O_a1;%lower stopband edge
global O_a2;%upper stopband edge
global O_s;%sampling frequency
%specifications for filter
global B_t1;%Lower transition width
global B_t2;%Upper transistion width
global B_t;%critical transition width
global O_c1;%Lower cutoff frequency
global O_c2;%Upper cutoff frequency
global A;%Stopband attenuation
global T;%Sampling period
global Aa;%Stopband ripple
global Ap;%Passband ripple
B_t1 = O_a1-O_p1;
```

```
B_t2 = O_p2-O_a2;
B_t = min(B_t1,B_t2);
O_c1 = O_p1+(B_t/2);
O_c2 = O_p2-(B_t/2);
T = 2*pi/O_s;
dp = ((10^(0.05*A_p))-1)/(1+(10^(0.05*A_p)));
da = 10^(-0.05*A_a);
d = min(dp,da);
A = -20*log10(d);
Ap = 20*log10((1+d)/(1-d));
Aa = -20*log10(d);
fprintf('....The derived parameters......\nLower transition width=%d\nUpper transition
    width=%d\n',B_t1,B_t2);
fprintf('Sampling period = %.5f\n,Actual passband ripple = %.2f\nActual sideband ripple =
     %.2f\n',T,Ap,Aa);
fprintf('Critical transition width = %d\nLower cutoff frequency=%.2f\n',B_t,O_c1);
fprintf('Upper cutoff frequency = %.2f\nActual Stopband attenuation=%.2f\n',O_c2,A);
```

Listing 3: Obtaining the Kaiser Window

```
function wk_nT = kaiser
global A;
global order n O_s B_t;
if A<=21
    alpha = 0;
elseif A>21 && A<=50
    alpha = 0.5842.*(A-21).^0.4+0.07886.*(A-21);
else
    alpha = 0.1102.*(A-8.7);
end
%Calculating D
if A<=21
    D = 0.9222;
else
    D = (A-7.95)/14.36;
end
%Finding the order of the filter
N = ceil((O_s*D/B_t)+1);
%Order of the filter should be odd
if mod(N,2) == 0
    order = N+1;
else
    order =N;
end
n = -(N-1)/2:1:(N-1)/2;
beta = alpha*sqrt(1-(2*n/(N-1)).^2);
%Generating Io_alpha
bessellimit = 125;
Io_alpha = 1;
for k = 1:bessellimit
    val_k = ((1/factorial(k))*(alpha/2).^k).^2;
    Io_alpha = Io_alpha + val_k;
```

```
end
%Generating Io_beta
Io_beta = 1;
for m = 1:bessellimit
    val_m = ((1/factorial(m))*(beta/2).^m).^2;
    Io_beta = Io_beta +val_m;
end
wk_nT = Io_beta/Io_alpha;
%Printing the results
fprintf('Filter order = %d',order);
%Plotting the kaiser function
figure;
stem(n,wk_nT);
xlabel('n');
ylabel('Amplitude');
title('Kaiser window(Time domain)');
```

Listing 4: Obtaining the ideal impulse stopband filter

```
function h_nT = idealfilter
global O_c1 O_c2 O_s T order;
%Generates the ideal impulse response stopband filter
n_L = -(order-1)/2:1:-1;
hn_L = (1./(n_L*pi)).*(sin(O_c1*n_L*T)-sin(O_c2*n_L*T));
n_R = 1:1:(order-1)/2;
hn_R = (1./(n_R*pi)).*(sin(O_c1*n_R*T)-sin(O_c2*n_R*T));
hn_0 = 1+(2/O_s).*(O_c1-O_c2);
n = [n_L,0,n_R];
h_nT = [hn_L,hn_0,hn_R];
%Plotting the ideal filter
figure;
stem(n,h_nT,'-r');
xlabel('n');
ylabel('Amplitude');
title('Ideal Impulse stopband filter(Time domain)');
```

Listing 5: Generating the input signal

```
function X = inputsignal(samples)
global O_c1 O_c2 O_s T;
global O_1 O_2 O_3 n1;
%Component frequencies of the input
O_1 = O_c1/2;
O_2 = O_c1 + (O_c2-O_c1)/2;
O_3 = O_c2 + (O_s/2-O_c2)/2;
%Generating the discrete signal
n1 = 0:1:samples;
X = cos(O_1.*n1.*T)+cos(O_2.*n1.*T)+cos(O_3.*n1.*T);
figure;
subplot(2,1,1);
stem(n1,X);
xlabel('n');
ylabel('Amplitude');
```

```
title('Input signal(Time domain)')
subplot(2,1,2);
len_fft = 2^nextpow2(numel(n1))−1;
x_fft = fft(X,len_fft);
x_fft_plot = [abs([x_fft(len_fft/2+1:len_fft)]),abs(x_fft(1)),abs(x_fft(2:len_fft/2+1))];
f = O_s∗linspace(0,1,len_fft)−O_s/2;
plot(f,x_fft_plot);
xlabel('Frequency rad/s');
ylabel('Magnitude');
title('Input signal in the frequency domain');
axis tight;
```

Listing 6: The main program

```
close all;
clear all;
clc;
global n order O_s O_c1 O_c2 T;
global O_1 O_3 n1;
%Generating the given filter parameters
filterparams(170401);
%Generating the derived filter parameters
deriveparams;
%Generating the kaiser window
wk_nT = kaiser;
%Obtaining the ideal impulse stopband filter
h_nT = idealfilter;
%Obtaining the noncausal stopband filter
hw_nT = h_nT.∗wk_nT;
%Plotting the noncausal stopband filter
figure;
stem(n,hw_nT);
xlabel('n');
ylabel('Amplitude');
title('Noncausal stopband filter window(Time domain)');
%Question 2
%Plotting the causal stopband filter
n_shifted = [0:1:order−1];
figure;
stem(n_shifted,hw_nT);
xlabel('n');
ylabel('Amplitude');
title('Causal Impulse Response filter(Time Domain)');
%obtaining the frequency domain impulse response response
fvtool(hw_nT);
%Question 3
[Hw,f] = freqz(hw_nT);%obtaining the frequency response and corresponding frequencies
w = f∗O_s/(2∗pi);%Angular frequency
log_Hw = 20.∗log10(abs(Hw));
figure;
plot(w,log_Hw);
xlabel('Angular frequency(rad/s)');
```

```
ylabel('Magnitude(dB)');
title('Magnitude response of the filter(Frequency domain)');
%Question 4
%Plotting the magnitude response of the passbands
%considering the lower passband
figure;
finish = round((length(w)/(O_s/2)*O_c1));
wpass_l = w(1:finish);
hpass_l = log_Hw(1:finish);
plot(wpass_l,hpass_l);
axis([-inf, inf, -0.1, 0.1]);
xlabel('Frequency (rad/s)');
ylabel('Magnitude (dB)');
title('Magnitude response of Lower Passband - Frequency Domain');
%Considering the upperpassband
figure;
start = round(length(w)/(O_s/2)*O_c2);
wpass_h = w(start:length(w));
hpass_h = log_Hw(start:length(w));
plot(wpass_h,hpass_h);
axis([-inf, inf, -0.1, 0.1]);
xlabel('Frequency (rad/s)');
ylabel('Magnitude (dB)');
title('Magnitude response of the Upper Passband - Frequency Domain');
%Question 5
%Generating the input of desired number samples
X = inputsignal(600);
%Question 6
% Filtering using frequency domain multiplication
len_fft = length(X)+length(hw_nT)-1; % length for fft in x dimension
x_fft = fft(X,len_fft);
hw_nT_fft = fft(hw_nT,len_fft);
out_fft = hw_nT_fft.*x_fft;
out = ifft(out_fft,len_fft);
rec_out = out(floor(order/2)+1:length(out)-floor(order/2));
% Ideal Output Signal
ideal_out = cos(O_1.*n1.*T)+cos(O_3.*n1.*T);
%O_2 is left out because it is in the  stopband
%Obtaining the output waveforms
% Frequency domain representation of output signal after filtering using
% the designed filter
figure;
subplot(2,1,1);
len_fft = 2^nextpow2(numel(n1))-1;
xfft_out = fft(rec_out,len_fft);
x_fft_out_plot = [abs([xfft_out(len_fft/2+1:len_fft)]),abs(xfft_out(1)),abs(xfft_out(2:
    len_fft/2+1))];
f = O_s*linspace(0,1,len_fft)-O_s/2;
plot(f,x_fft_out_plot);
xlabel('Frequency rad/s');
ylabel('Magnitude');
```

```
title('Output signal of the designed filter in the frequency domain');
% Time domain representation of output signal after filtering using the
% designed filter
subplot(2,1,2);
stem(n1,rec_out);
xlabel('n');
ylabel('Amplitude');
title('Output signal of the designed filter in the time domain');
%Obtaining the outputs of the ideal filter
figure;
subplot(2,1,1);
xfft_outideal = fft(ideal_out,len_fft);
x_fft_outideal_plot = [abs([xfft_outideal(len_fft/2+1:len_fft)]),abs(xfft_outideal(1)),
    abs(xfft_outideal(2:len_fft/2+1))];
plot(f,x_fft_outideal_plot);
xlabel('Frequency rad/s');
ylabel('Magnitude');
title('Output signal of the ideal filter in the frequency domain');
% Time domain representation of output signal after filtering using ideal filter
subplot(2,1,2);
stem(n1,ideal_out);
xlabel('n');
ylabel('Amplitude');
title('Output signal of the ideal filter in the time domain');
%Obtaining the RMSE between the output of the outputs of the designed and
%ideal filters
RMSE = sqrt(mean((rec_out - ideal_out).^2));
deviation = abs(rec_out-ideal_out);
figure;
plot(n1,deviation,'-r');
xlabel('n');
ylabel('Magnitude');
title('Deviation between ideal and designed filters');
fprintf('The root mean square error between the ideal and designed filters = %.5f\n',RMSE
    );
```