Department of Electronic and Telecommunication Engineering
University of Moratuwa

EN2570 Digital Signal Processing

Project

Designing a Finite Impulse Response Bandstop Filter

Mannapperuma M.T.L 170373J

This report is submitted as a partial fulfillment for the module

EN2570 Digital Signal Processing

02nd of January 2020

Abstract - Bandstop Finite Impulse Response (FIR) filter design is discussed in this project and it satisfies some characteristics MATLAB is used to simulate this design and Kaiser Window and method of Fourier series are used. Performance of the filter is analyzed and evaluated by looking at the outputs if the test inputs. Inputs are a set of sinusoidal signals.

Table of Contents

1. Introduction

Bandstop FIR filter design and evaluation is described in this report in step by step. MATLAB (Version R2017a) is used for the implementation. Filter characteristics are calculated according to our index number and filter should be implemented in according to that characteristics. Windowing process is done by using Kaiser window method and Fourier series method is also used for filter implementation. In order to evaluate main characteristics, Time and frequency domain representation is obtained at different stages. Fast Fourier Transform (FFT) of Discrete Fourier Transform (DFT) is used to obtain frequency response of the filter. As input sinusoidal signals with three frequencies and one is in the stop band is used and then output is compared with the input signal and then to analyze output is compared to an ideal filter. Accuracy can be determined by that.

2. Method

Designing a band stop filter using given characteristics and filter output analyzation are the two main tasks at hand in this project. Determining parameters for implementation of Kaiser window and generating the window function is first.

Ideal filter impulse response is generated by Fourier Series method and it will be multiplied by the windows function to get required filter. Time and Frequency domain representations are calculated to analyze the filter behavior.

At last as inputs, sinusoidal signals with three frequencies and one is in the stop band was given and then output was compared with the input signal and then to analyze, output was compared to an ideal filter. Accuracy was determined by that.

2.1 Filter Specifications

Below parameters are used to design the filter.

Specification	*Symbol*	*Value*	*Units*
Maximum pass band ripple	Ãp	0.06	dB
Minimum stop band attenuation	Ãa	52	dB
Lower pass band edge	Ω_{p1}	700	rad/s
Upper pass band edge	Ω_{p2}	1250	rad/s
Lower stop band edge	Ω_{a1}	800	rad/s
Upper stop band edge	Ω_{a2}	1100	rad/s
Sampling frequency	Ω_s	3200	rad/s

2.2 Filter Parameters

Below parameters are calculated for the Kaiser window

Derived Parameter	*Symbol*	*Derivation*	*Value*	*Units*
Lower transition width	B_{t1}	$\Omega_{a1} - \Omega_{p1}$	100	rad/s
Upper transition width	B_{t2}	$\Omega_{p2} - \Omega_{a2}$	150	rad/s
Critical transition width	B_t	$\min(B_{t1}, B_{t2})$	100	rad/s
Lower cutoff frequency	Ω_{c1}	$\Omega_{p1} + \frac{B_t}{2}$	750	rad/s
Upper cutoff frequency	Ω_{c2}	$\Omega_{p2} - \frac{B_t}{2}$	1200	rad/s
Sampling period	T	$\frac{2\pi}{\Omega_s}$	0.001963	s

2.3 Kaiser Window parameters

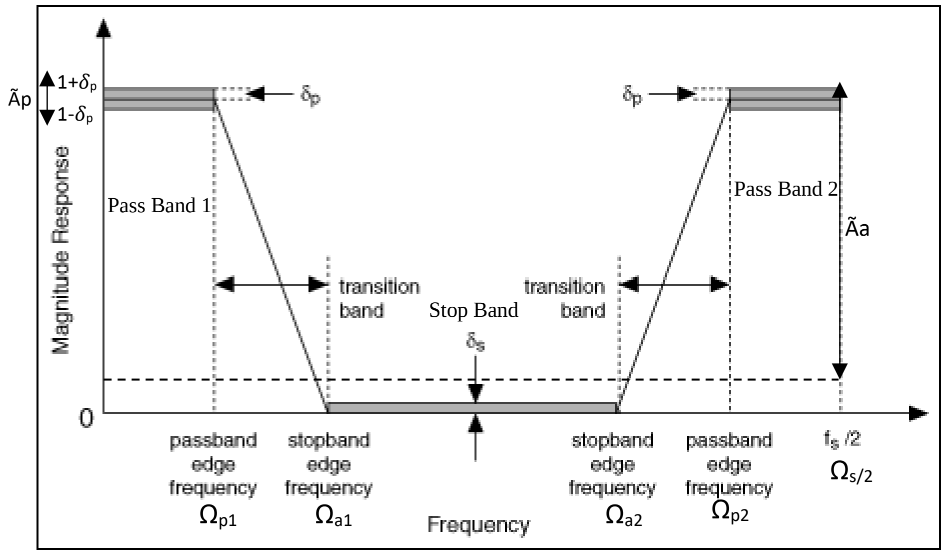

Figure 1

Maximum passband ripple is Ãp and Ãa is the stopband attenuation. It is shown in the above figure.

$$\delta_p = \frac{10^{0.05\,\tilde{A}p}-1}{10^{0.05\,\tilde{A}p}+1} \quad \text{and} \quad \delta_a = 10^{-0.05\tilde{A}_a}$$

Actual passband ripple and stopband attenuations should be $A_p \leq \tilde{A}_p$ and $A_a \geq \tilde{A}_a$ accordingly and δ should be take according to above conditions of actual passband ripple and stopband attenuation.

$$\delta = \min(\delta_p, \delta_a)$$
$$A_p = 20\log\left(\frac{1+\delta}{1-\delta}\right) \quad \text{and} \quad A_a = -20\log(\delta)$$

A_p and A_a are the actual passband ripple and stopband attenuation. [1] [2]

D can be defined as below

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

N is the length of the filter in below equation and it should be chosen as N is the smallest integer that satisfies below inequality.

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

α and β can be defined as below.

$$\alpha = \begin{cases} 0 & for\ A_a \leq 21 \\ 0.5842(A_a - 21)^{0.4} + 0.07886(A_a - 21) & for\ 21 < A_a \leq 50 \\ 0.1102(A_a - 8.7) & for\ A_a > 50 \end{cases} \quad , \quad \beta = \alpha\sqrt{1 - \left(\frac{2n}{N-1}\right)^2}$$

$w_k(nT)$ is tha Kaiser Window function and it can be derived by above parameters as below.

$$w_k(nT) = \begin{cases} \frac{I_0(\beta)}{I_0(\alpha)} & for\ |n| \le (N-1)/2 \\ 0 & Otherwise \end{cases} \quad , where\ \left(I_0(x) = 1 + \sum_{k=1}^{\infty} \left[\frac{1}{k!} \left(\frac{x}{2}\right)^k \right]^2 \right)$$

Calculated Kaiser Window parameters are in the below table.

Parameter	*Value*	*Units*
δ	0.002512	-
A_a	52	dB
α	4.7717	-
D	3.067548747	-
N	101	-

2.4 Ideal Impulse Response derivation

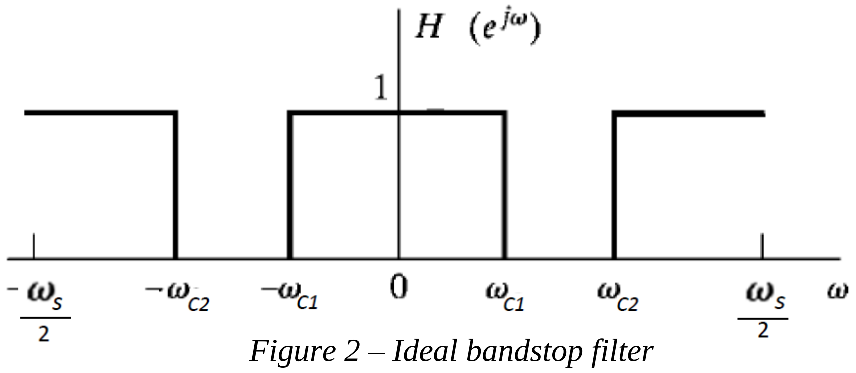

Figure 2 – Ideal bandstop filter

Cutoff frequencies Ω_{c1} and Ω_{c2} can be given as below,

$$\Omega_{c1} = \Omega_{p1} + \frac{B_t}{2} \text{ and } \Omega_{c2} = \Omega_{p2} + \frac{B_t}{2}$$

Ideal band stop filter frequency response can be derived as below with above cutoff frequencies.

$$\mathrm{H}\left(e^{j\omega t}\right) = \begin{cases} 1 & for\ 0 \le |\omega| < \Omega_{c1} \\ 0 & for\ \Omega_{c1} \le |\omega| < \Omega_{c2} \\ 1 & for\ \Omega_{c2} \le |\omega| < \frac{\Omega_s}{2} \end{cases}$$

Impulse response of $\mathrm{H}\left(e^{j\omega t}\right)$ can be derived by inverse Fourier transform as below

$$h[nT] = \begin{cases} 1 + \frac{2}{\Omega_s}(\Omega_{c1} - \Omega_{c2}) & for\ n = 0 \\ \frac{1}{n\pi}(\sin \Omega_{c2} nT - \sin \Omega_{c1} nT) & Otherwise \end{cases}$$

2.5 Causal filter derivation

By multiplying Ideal Impulse Response $h[nT]$ and Kaiser Window $wk(nT)$ we can get the non-causal impulse response $h_w(nT)$

$$h_w(nT) = w_k(nT)h(nT)$$

Z transform of the filer can be obtained as below,

$$H_w(z) = \mathcal{Z}[w_k(nT)\mathrm{h(nT)}]$$

Causal impulse response $H'_w(z)$ can be obtained by shifting $H_w(z)$ by $\left(\frac{N-1}{2}\right)$ samples forward.

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

2.6 Filter evaluation

For the evaluation purpose of the filter, a sample signal with 3 sinusoids consisting components in three bands which are upper, lower and stop is given as the input for the filter.

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

Input signal parameters are given in the below table

Frequency Component	Derivation	Value	Unit
Ω_1	$\frac{\Omega_{c1}}{2}$	375	rad/s
Ω_2	$\frac{\Omega_{c1} + \Omega_{c2}}{2}$	975	rad/s
Ω_3	$\frac{\Omega_{c2} + \frac{\Omega_s}{2}}{2}$	1400	rad/s

By convoluting input signal $x(nT)$ with the impulse response of the filter $h_w(nT)$, output can be obtained. First, Discrete Time Fourier Transform (DFT) of $x(nT)$ and $h_w(nT)$ is obtained since convoluting in time domain is multiplication in frequency domain. Then those Fourier Transformed signals are multiplied with each other. Finally using Inverse Discrete Time Fourier Transform (IDFT) multiplied signal in converted back to time domain.

3. Simulation Results

3.1 Filter Design

Both time and frequency domain characteristics are analyzed, and simulation results can be shown as below figures.

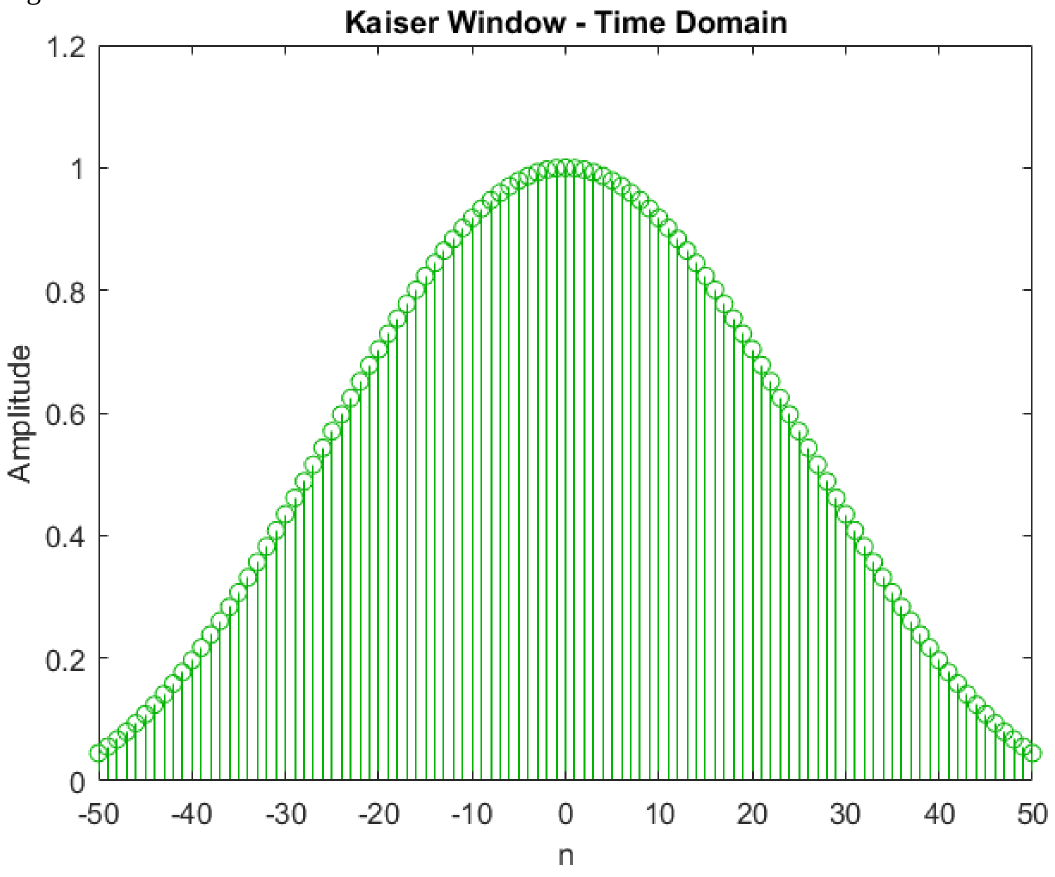

Figure 3 – Kaiser Window (time domain)

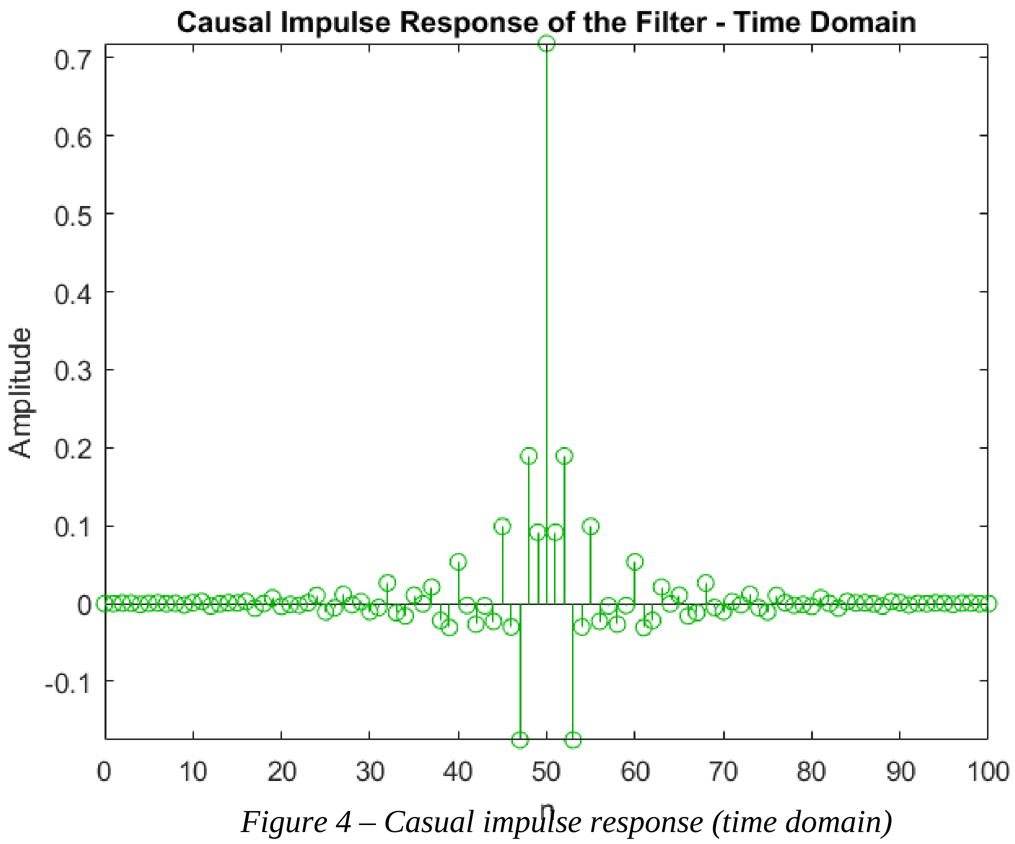

Figure 4 – Casual impulse response (time domain)

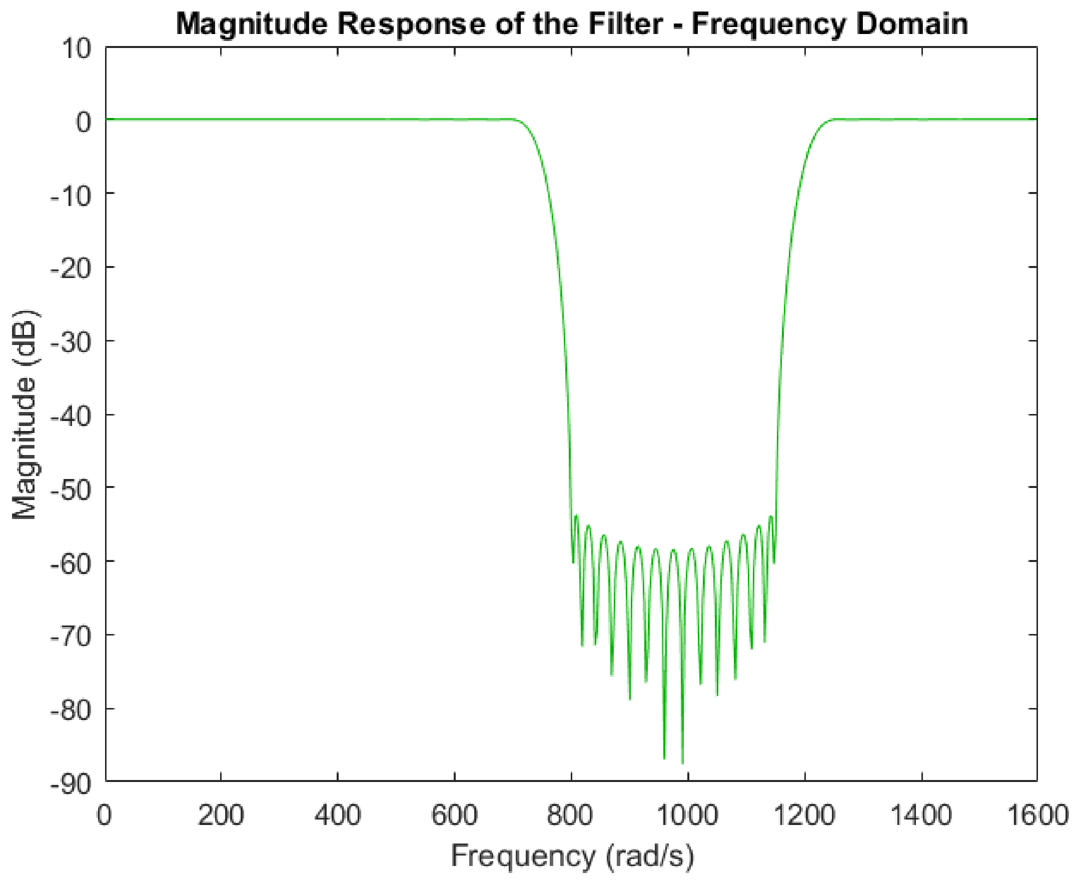

Figure 5 – Magnitude response (frequency domain)

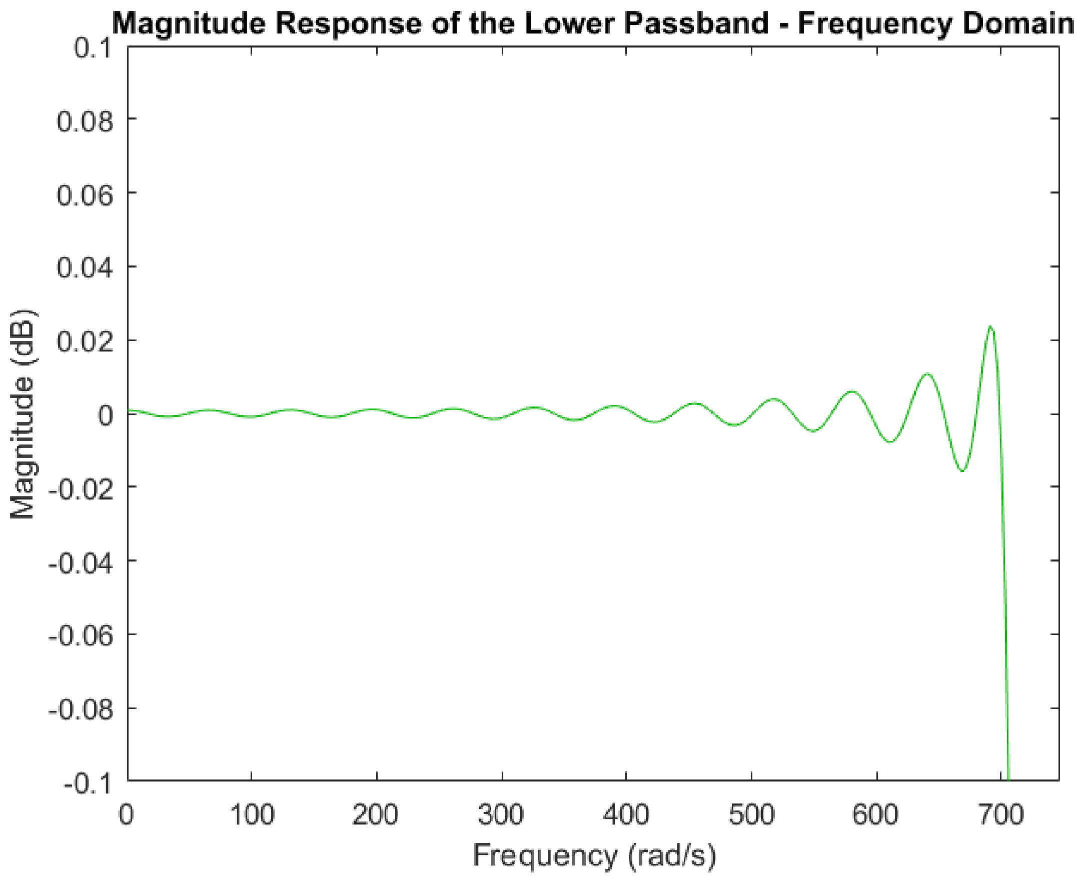

Figure 6 – Magnitude response of the lower passband (frequency domain)

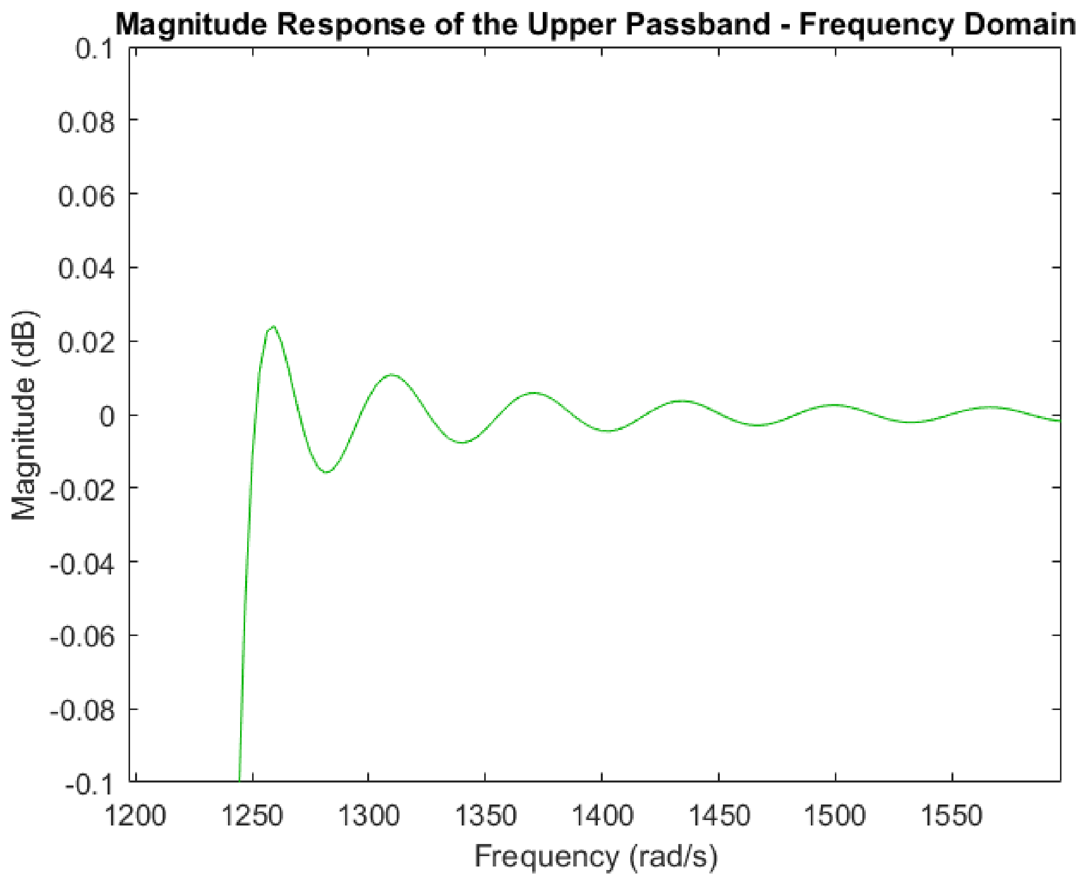

Figure 7 – Magnitude response of the upper passband (frequency domain)

<u>3.2 Filter Analyzation – Simple signal</u>

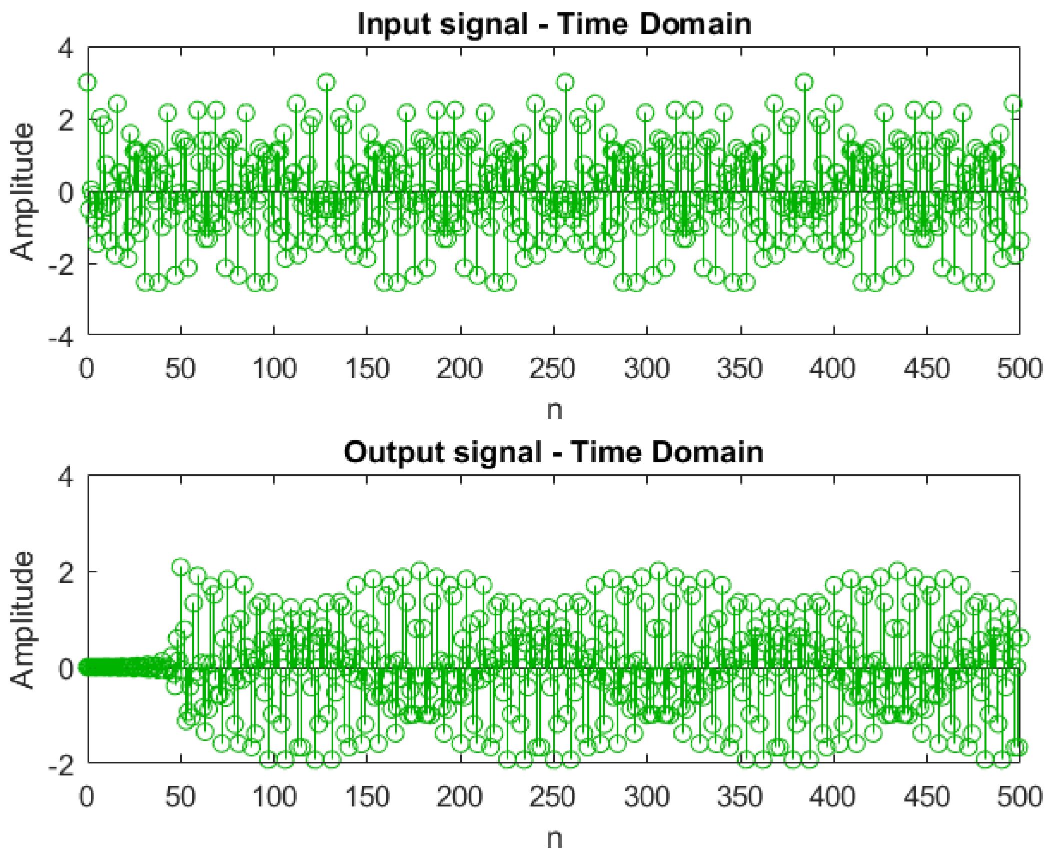

Figure 8 – Input and Output signals (Time domain)

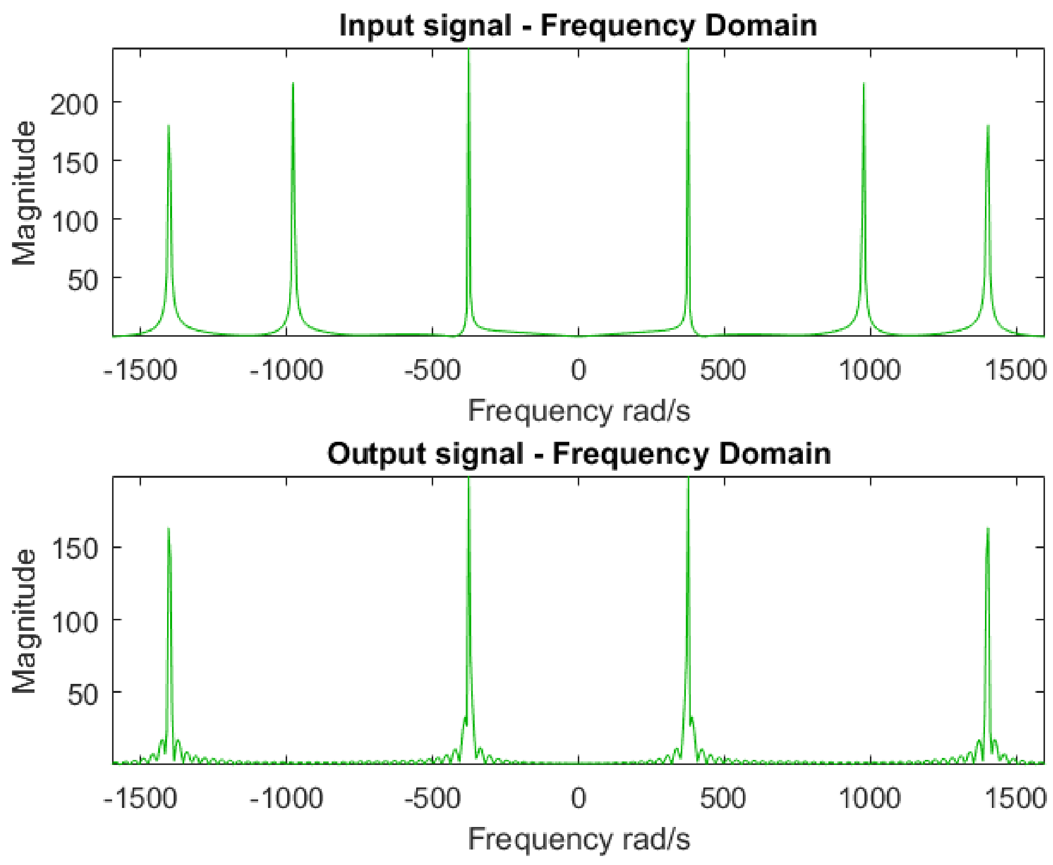

Figure 9 – Input and Output signals (frequency domain)

3.3 Output and Expected Output Comparison

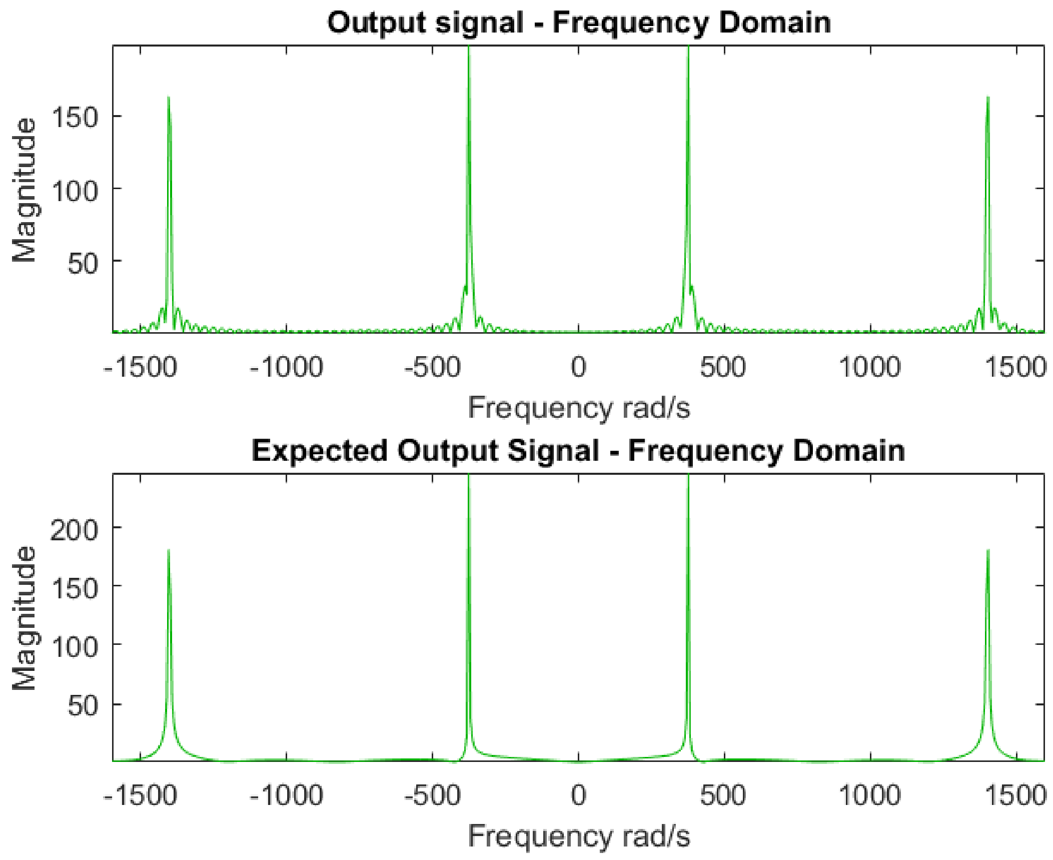

Figure 10 – Output vs Expected Output

4. Discussion

Filters are used in most of the day to day use cases. The purpose of a filter can be depicted as to isolate, or selection of a certain frequency range from a given signal. A band stop filter is discussed in this project with pre given specification. By looking at above results we can conclude that our goal is a success with this project.

Figure 5 shows that the gain is unity at the passband of the signal. In the stopband maximum gain is nearly -90dB. Amplitude of the signal is at a minimum in the stopband frequency. Maximum allowed passband ripple is about 0.06dB and in figure 6 and 7 we can see that the simulated passband ripple is less than that.

A sinusoid signal with 3 frequency components in lower, upper passbands and stopband is used as an input for the filter and filter functionality is tested through that. Figure 9 shows that stopband frequencies are not in the final output signal. By using rectangular window function, expected output signal can be obtained and can be compared to the output signal of the filter as in figure 10. By these figures, expected signal output and real output signal is almost identical with negligible ripple in the real output signal.

5. Conclusion

These results show that Kaiser Window method produces a filter with characteristics almost identical to an ideal filter. Kaiser window parameters are very flexible, and they can be easily implemented as a real-world ideal filter. This filter design is not very difficult but there is a one drawback which is the requirement of higher order. For practical visualization and implementation, it is required a complex electronic circuitry. So, by looking at above details we can conclude that using Kaiser Window method, FIR band stop filter can be successfully implemented.

6. Acknowledgement

I would like to express my gratitude towards Dr.Chamira Edusooriya for providing the guidance and knowledge which needed for this project.

7. Reference

[1] S. Hemachandra, "Medium.com," Medium, [Online]. Available: https://medium.com/@savinihemachandra/fir-filter-design-bandstop-filter-17bdace6a54e. [Accessed 29 12 2019].

[2] Antoniou.A, Digital Signal Processing - Signals Systems and Filters, 1 ed., McGraw-Hill, 2005.

8. Appendix – MATLAB Code

```matlab
clc;
clear all;
close all;

%% Filter Specifications

%Index Number 170373

A = 3; B = 7; C = 3;

P_r = 0.03+(0.01*A);     % dB    %%max passband ripple
S_a = 45+B;              %dB     %%min stopband attenuation
LP_e = (C*100)+400;      %rad/s  %%lower passband edge
UP_e = (C*100)+950;      %rad/s  %%upper passband edge
LS_e = (C*100)+500;      %rad/s  %%lower stopband edge
US_e = (C*100)+800;      %rad/s  %%upper stopband edge
S_f = 2*(C*100+1300);    %rad/s  %%sampling freqency



%% Filter Parameters

LT_w = LS_e-LP_e;        %rad/s   %%lower transition width
UT_w = UP_e-US_e;        %rad/s   %%upper transisiton width
CT_w = min(LT_w,UT_w);   %rad/s   %%critical transition width
LC_f = LP_e+CT_w/2;      %rad/s   %%lower cutoff frequency
UC_f = UP_e-CT_w/2;      %rad/s   %%upper cutoff frequency
T = 2*pi/S_f;            %s       %%sampling period

%% Kaiser Window Parameters

% calculating delta
d_p = (10^(0.05*P_r) - 1)/ (10^(0.05*P_r) + 1);
d_a = 10^(-0.05*S_a);
d = min(d_p,d_a);

Aa = -20*log10(d);           % Actual stopband attenuation
Ap = 20*log10(1+d/1-d);      % Actual passband ripple


% Calculating alpha
if Aa<=21
    alpha = 0;
elseif Aa>21 && Aa<= 50
    alpha = 0.5842*(Aa-21)^0.4 + 0.07886*(Aa-21);
else
    alpha = 0.1102*(Aa-8.7);
end

% Calculating D
if Aa <= 21
    D = 0.9222;
else
```

```
    D = (Aa-7.95)/14.36;
end

% Calculating filter order (N)
N = ceil(S_f*D/CT_w +1);
if mod(N,2) == 0
    N = N+1;
end

% Length of the filter
n = -(N-1)/2:1:(N-1)/2;

% Calculating beta
beta = alpha*sqrt(1-(2*n/(N-1)).^2);

%% Generating Io(alpha)

Io_alpha = 1;
for k = 1:100
    val_k = (1/factorial(k)*(alpha/2).^k).^2;
    Io_alpha = Io_alpha + val_k;
end

%% Generating Io(beta)

Io_beta = 1;
for m = 1:100
    val_m = (1/factorial(m)*(beta/2).^m).^2;
    Io_beta = Io_beta + val_m;
end

%% Generating Kaiser Window w_k(nT)

wk_nT = Io_beta/Io_alpha;

figure
stem(n,wk_nT,'Color',[0 0.7 0])
xlabel('n')
ylabel('Amplitude')
ylim([0 1.2])
title('Kaiser Window - Time Domain');

%% Generating Impulse Response h(nT)

n_L= -(N-1)/2:1:-1;
hnt_L = 1./(n_L*pi).*(sin(LC_f*n_L*T)-sin(UC_f*n_L*T));

n_R = 1:1:(N-1)/2;
hnt_R = 1./(n_R*pi).*(sin(LC_f*n_R*T)-sin(UC_f*n_R*T));

hnt_0 = 1+ 2/S_f*(LC_f-UC_f);

n = [n_L,0,n_R];
```

```
h_nT = [hnt_L,hnt_0,hnt_R];

%% Applying the Kaiser Window to the filter

Hw_nT = h_nT.*wk_nT;

%% Plotting the causal impulse response

n_shifted = 0:1:N-1;
figure
stem(n_shifted,Hw_nT,'Color',[0 0.7 0]); axis tight;
xlabel('n')
ylabel('Amplitude')
title(strcat(['Causal Impulse Response of the Filter - Time Domain']));

%% Plotting the magnitude response of filter in the range (0,S_f/2)
figure
[Hw,f] = freqz(Hw_nT);
w_1 = f*S_f/(2*pi);
log_Hw = 20*log10(abs(Hw));
plot(w_1,log_Hw,'Color',[0 0.7 0])
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title(strcat(['Magnitude Response of the Filter - Frequency Domain']));

%% Plotting the magnitude response of the Lower Passband

figure
lower_cutoff = round((length(w_1)/(S_f/2)*LC_f));
wpass_l = w_1(1:lower_cutoff);
hpass_l = log_Hw(1:lower_cutoff);
plot(wpass_l,hpass_l,'Color',[0 0.7 0])
axis([-inf, inf, -0.1, 0.1]);
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title('Magnitude Response of the Lower Passband - Frequency Domain');

%% Plotting the magnitude response of the Upper Passband

figure
upper_cutoff = round(length(w_1)/(S_f/2)*UC_f);
wpass_h = w_1(upper_cutoff:length(w_1));
hpass_h = log_Hw(upper_cutoff:length(w_1));
plot(wpass_h,hpass_h,'Color',[0 0.7 0])
axis([-inf, inf, -0.1, 0.1]);
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title('Magnitude Response of the Upper Passband - Frequency Domain');


%% Input signal generation

% frequency components
```

```
w1 = LC_f/2;
w2 = LC_f + (UC_f-LC_f)/2;
w3 = UC_f + (S_f/2-UC_f)/2;

% generate discrete signal and evelope
samples = 500;
n1 = 0:1:samples;
n2 = 0:0.1:samples;

X_nT = cos(w1.*n1.*T)+cos(w2.*n1.*T)+cos(w3.*n1.*T);


%% Using DFT to check the filtering

% Filtering using frequency domain multiplication

len_fft = length(X_nT)+length(Hw_nT)-1; % length for fft in x dimension
x_fft = fft(X_nT,len_fft);
Hw_nT_fft = fft(Hw_nT,len_fft);
out_fft = Hw_nT_fft.*x_fft;
out = ifft(out_fft,len_fft);
out = out(1:length(out)-floor(N)+1);
rec_out = out(floor(N/2)+1:length(out)-floor(N/2));% account for shifting delay



%% Plots for the evaluation of the filter

% Time domain representation of input signal before filtering
figure
subplot(2,1,1)
stem(n1,X_nT,'Color',[0 0.7 0])
xlabel('n')
ylabel('Amplitude')
title(strcat(['Input signal',' ','- Time Domain']));


% Time domain representation of output signal after filtering

subplot(2,1,2)
stem(n1,out,'Color',[0 0.7 0])
xlabel('n')
ylabel('Amplitude')
title(strcat(['Output signal',' ','- Time Domain']));


% Frequency domain representation of input signal before filtering

figure
subplot(2,1,1)
len_fft = 2^nextpow2(numel(n1))-1;
x_fft = fft(X_nT,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 = S_f*linspace(0,1,len_fft)-S_f/2;
plot(f,x_fft_plot,'Color',[0 0.7 0]);
axis tight;
xlabel('Frequency rad/s')
ylabel('Magnitude')
title(strcat(['Input signal',' ','- Frequency Domain']));


% Frequency domain representation of output signal after filtering
subplot(2,1,2)
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 = S_f*linspace(0,1,len_fft)-S_f/2;
plot(f,x_fft_out_plot,'Color',[0 0.7 0]);
axis tight;
xlabel('Frequency rad/s')
ylabel('Magnitude')
title(strcat(['Output signal',' ','- Frequency Domain']));
%% Expected output
ideal_out = cos(w1.*n1.*T)+cos(w3.*n1.*T);

% Frequency domain representation of output signal after filtering
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 = S_f*linspace(0,1,len_fft)-S_f/2;
plot(f,x_fft_out_plot,'Color',[0 0.7 0]); axis tight;
xlabel('Frequency rad/s')
ylabel('Magnitude')
title(strcat(['Output signal',' ','- Frequency Domain']));


% Frequency domain representation of expected output signal
subplot(2,1,2)
len_fft = 2^nextpow2(numel(n1))-1;
x_fft = fft(ideal_out,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 = S_f*linspace(0,1,len_fft)-S_f/2;
plot(f,x_fft_plot,'Color',[0 0.7 0]); axis tight;
xlabel('Frequency rad/s')
ylabel('Magnitude')
title(strcat(['Expected Output Signal',' ','- Frequency Domain']));
```