I. INTRODUCTION

This report explains the way of designing and evaluating an FIR bandpass filter. The filter designing technique that is used in this project is the Kaiser windowing technique and project implementation is done with **MATLAB** R2018a Software. There are many methods for filter designing (Kaiser windowing technique, Han and Hamming window technique, triangular windowing technique, rectangular windowing technique, etc). The direct approach of this project is using Fourier transform method to implement the bandpass filter with the Kaiser windowing technique. The parameters of the Kaiser window were used to tune the filter to prescribed characteristics. The time domain and frequency domain representations of the signals were subjected to evaluate the filter characteristics. Frequency responses are obtained by the Fast Fourier Transform (FFT) algorithm implementation of the Discrete Fourier Transform (DFT). For obtaining the FFT of a function, the direct function method has been introduced in **MATLAB** software. Also, a direct function for obtaining the IFFT of a function is included. For evaluating purposes summation of three sinusoidal signals are defined. These three signals contain the middle frequency of the lower stopband, the middle frequency of the passband, and the middle frequency of the upper stopband. By observing the output of the above signal in the frequency domain and time domain, we can come to a clear conclusion. After analyzing all the results, identification of the problems in the filter, and the improvement for the filter can be suggested.

II. METHOD

This project serves two objectives regarding the basics of filter designing. They are FIR bandpass filter designing and implementing and evaluating the performance of the designed filter. The methodology of both these objectives will be discussed in detail separately. As the first step of the filter implementation, calculations for prescribed filter specifications and derived filter specifications would be done. Next calculated specifications, principles of windowing filter designing, and principles of digital signal processing are used to create the background for designing the relevant bandpass filter. As the final step of the filter implementation, a Kaiser Window function is obtained such that the ideal frequency response is truncated preserving the required filter parameters, and time and frequency domain representations of the designed passband filter is obtained. For the filter evaluation previously mentioned summation of the sinusoidal signal is used and their input and output representations can be used to analyze and compare with the expected filter response output.

Filter Implementation

1.1 Prescribed Filter Specifications

The following table describes the required filter specifications for designing the filter. (Calculations were based on given specifications for this project)

Table 1: Required filter specifications

Parameter	Symbol	Value
Maximum passband ripple	$\tilde{A}_p$	0.05 dB
Minimum stopband attenuation	$\tilde{A}_a$	48 dB
Lower passband edge	Ω_{p1}	1000 $rads^{-1}$
Upper passband edge	Ω_{p2}	1400 $rads^{-1}$
Lower stopband edge	Ω_{a1}	850 $rads^{-1}$
Upper stopband edge	Ω_{a2}	1500 $rads^{-1}$
Sampling frequency	Ω_s	3800 $rads^{-1}$

1.2 Derived Filter Specifications

The following table describes the derived filter specifications for designing the bandpass digital filter and the Kaiser Window.

Table 2: Derived filter specifications

Parameter	Symbol	Derivation	Value
Lower transition bandwidth	$\mathbf{B_{t1}}$	$\mathbf{\Omega_{p1} - \Omega_{a1}}$	150 rads^{-1}
Upper transition width	$\mathbf{B_{t2}}$	$\mathbf{\Omega_{a2} - \Omega_{p2}}$	100 rads^{-1}
Critical transition width	$\mathbf{B_t}$	min($\mathbf{B_{t1}}$, $\mathbf{B_{t2}}$)	100 rads^{-1}
Lower cutoff frequency	$\mathbf{\Omega_{c1}}$	$\mathbf{\Omega_{p1} - (B_t/2)}$	950 rads^{-1}
Upper cutoff frequency	$\mathbf{\Omega_{c2}}$	$\mathbf{\Omega_{p2} + (B_t/2)}$	1450 rads^{-1}
Sampling period	**T**	$\mathbf{2\pi/\Omega_s}$	1.65×10^{-3} s

2. Derivation of the Kaiser Window

The Kaiser Window function is defined as,

$$w_k(nT) = \begin{cases} \dfrac{I_0(\beta)}{I_0(\alpha)} & for\ |n| \le \dfrac{(N-1)}{2} \\ 0 & Otherwise \end{cases}$$

Where $\boldsymbol{\alpha}$ is an independent parameter and,

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

Considering **Table 1** and **Table 2** following calculations can be done to find $\boldsymbol{a}$ and $\boldsymbol{N}$ parameters.$\boldsymbol{\delta}$ Can be defined such that actual passband ripple($\mathbf{A_p}$) less than $\tilde{\mathbf{A}}_\mathbf{p}$ and the actual minimum stopband attenuation greater than $\tilde{\mathbf{A}}_\mathbf{a}$.

$$\delta = \min(\delta_p, \delta_a)$$

$$\delta_p = \frac{10^{o.o5\,\tilde{A}_p} - 1}{10^{o.o5\,\tilde{A}_p} + 1} \quad and \quad \delta_a = 10^{-o.o5\tilde{A}_a}$$

Using $\boldsymbol{\delta}$ actual stopband loss ($\mathbf{A_a}$) can be calculated,

$$A_a = -20\log(\delta)$$

Define $\boldsymbol{\alpha}$ parameter as

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

Define **D** parameter as

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

N is selected that is the smallest odd integer satisfying the following inequality

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

The following **Table 3** defines the calculated Kaiser window parameters.

Table 3: Kaiser Window parameters

Parameters	Values
δ	2.878×10^{-3}
A_a	50.82 dB
α	4.64
D	2.985
N	115

3. Derivation of The Ideal Impulse Response

Ideal bandpass filter frequency response according to cutoff frequencies(Ω_{c1} and Ω_{c2})

$$H(e^{j\omega t}) = \begin{cases} 1 & for -\Omega_{c2} \leq \Omega \leq -\Omega_{c1} \\ 1 & for\ \Omega_{c1} \leq \Omega \leq \Omega_{c2} \\ 0 & Otherwise \end{cases}$$

Using inverse Fourier transform impulse response of $H(e^{j\Omega t})$ is obtained to be

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

4. Obtaining the Impulse Response of the Windowed Filter

By multiplying ideal impulse response and $h[nT]$ and Kaiser window function $w_k(nT)$, finite order non-causal impulse response of the windowed filter $H_w(nT)$ can be obtained.

$$H_w(nT) = w_k(nT)h(nT)$$

The impulse response of the windowed filter in Z form

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

which upon shifting for causality becomes

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

We also present the filter that is obtained from a rectangular windowing, to compare.

$$w_R(nT) = \begin{cases} 1; & for\ |n| \leq \dfrac{(N-1)}{2} \\ 0; & otherwise \end{cases}$$

Filter Evaluation

For evaluating the performance of the created filter, the summation of three sinusoidal signals is added to the input signal. Mentioned three signals contain the middle frequency of the lower stopband, the middle frequency of the passband, and the middle frequency of the upper stopband. **Table: 4** contains calculated values for mentioned frequencies. To obtain the output of the regarded sinusoidal equation, the convolution of the filter function and input function has to use but using Discrete-Time Fourier Transform techniques and principles that method can be simplified. For obtaining the Fourier transform of filter function and input function, **FFT**(Fast Fourier Transform) technique is used in **MATLAB**. After the multiplication of the obtained DTFT functions, the frequency representation of the output signal can be obtained, and to represent its time domain, **IFFT**(Inverse Fast Fourier Transform) technique is used.

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

Table 4: Input Frequency Components

Parameters	Derivation	Values
Ω_1	$\frac{(\Omega_{c1})}{2}$	475 rads^{-1}
Ω_2	$\frac{(\Omega_{c1} + \Omega_{c2})}{2}$	1200 rads^{-1}
Ω_3	$\frac{(\Omega_{c2} + \Omega_s/2)}{2}$	1675 rads^{-1}

III RESULTS

Filter design results can be present in two forms. In the first form, frequency and impulse response plots are used to observe the filter characteristics. By comparing the input and the output of the signal, the remaining form of demonstrating the performance of the filter can be done.

Frequency and Time Domain plots of the Filter

The following figures(**Figure 1-Figure 6**) are obtained during the process of designing the filter. **Figure 1**, shows the impulse response representation of the Kaiser Window. **Figure 2** and **Figure 3** show the frequency domain and time-domain representation of another window function (rectangular window). **Figure 4** and **Figure 5** show the frequency domain and time-domain representation of the Kaiser Window function. **Figure 6** is a plot that shows the passband of the filter designed using the Kaiser Window and it is a zoomed plot of **Figure 4. Figure 7** is a plot that shows the passband of the filter designed using the rectangular Window and it is a zoomed plot of **Figure 2.**

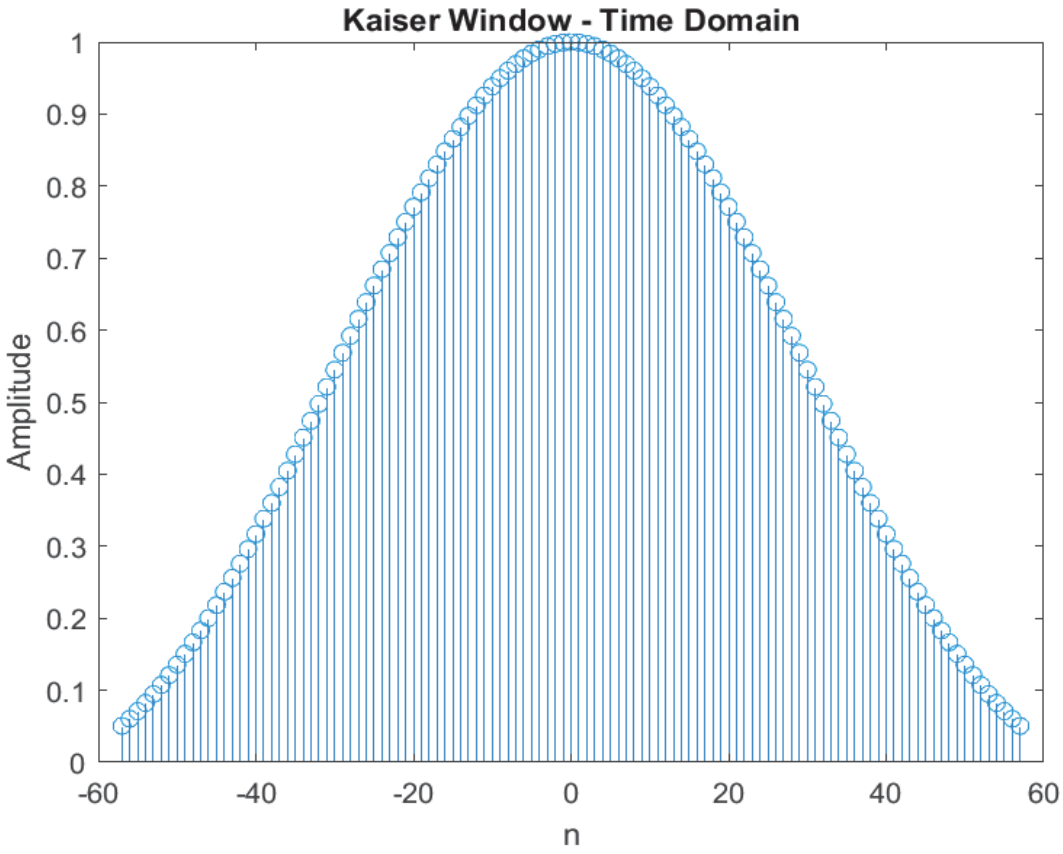

Figure 1: Impulse response representation of Kaiser Window

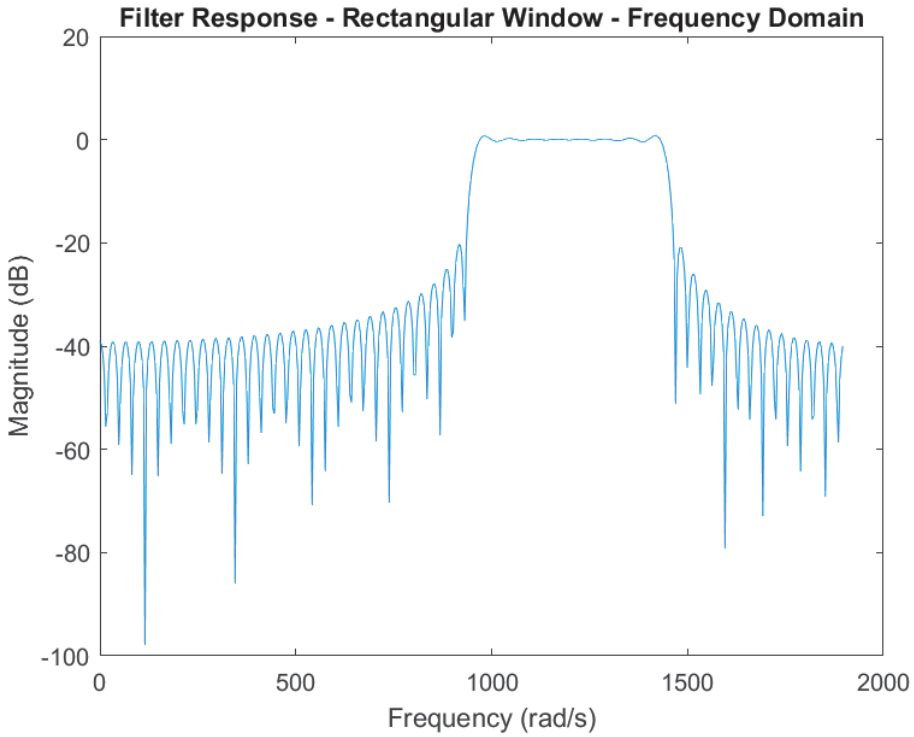

Figure 2: Frequency Domain Representation of the filter designed using a Rectangular Window Function

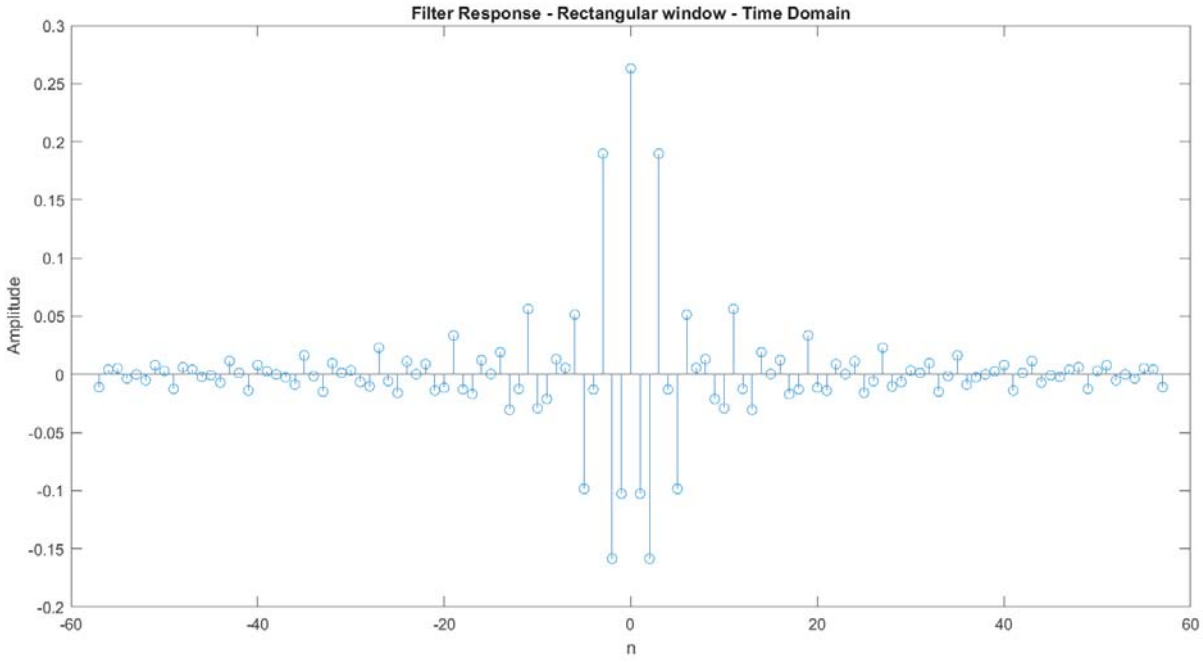

Figure 3: Time Domain Representation of the filter designed using a Rectangular Window Function

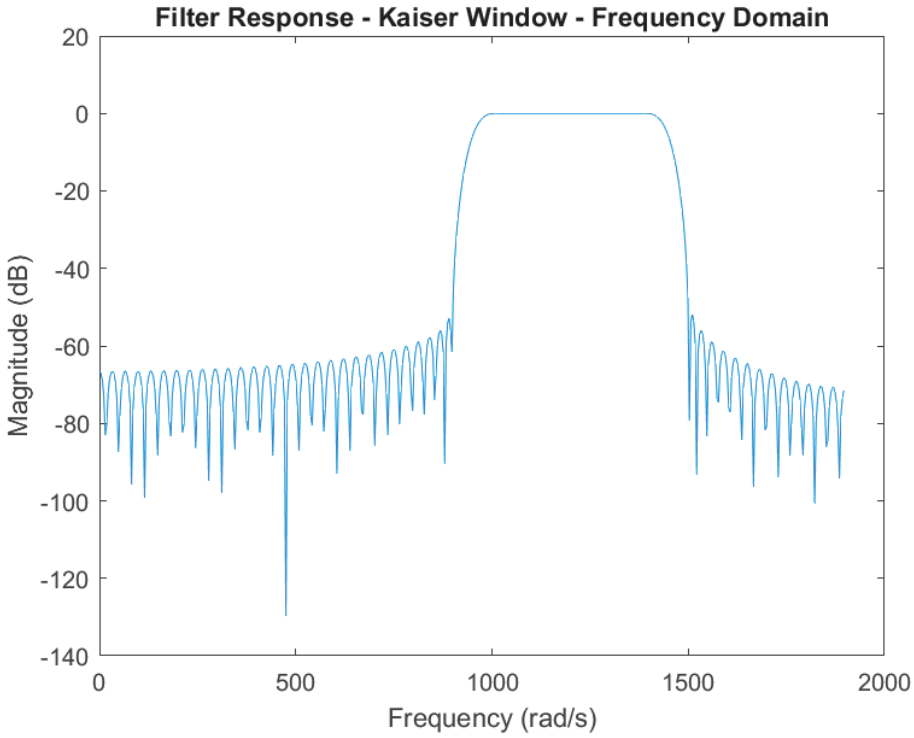

Figure 4: Frequency Domain Representation of the filter designed using a Kaiser Window Function

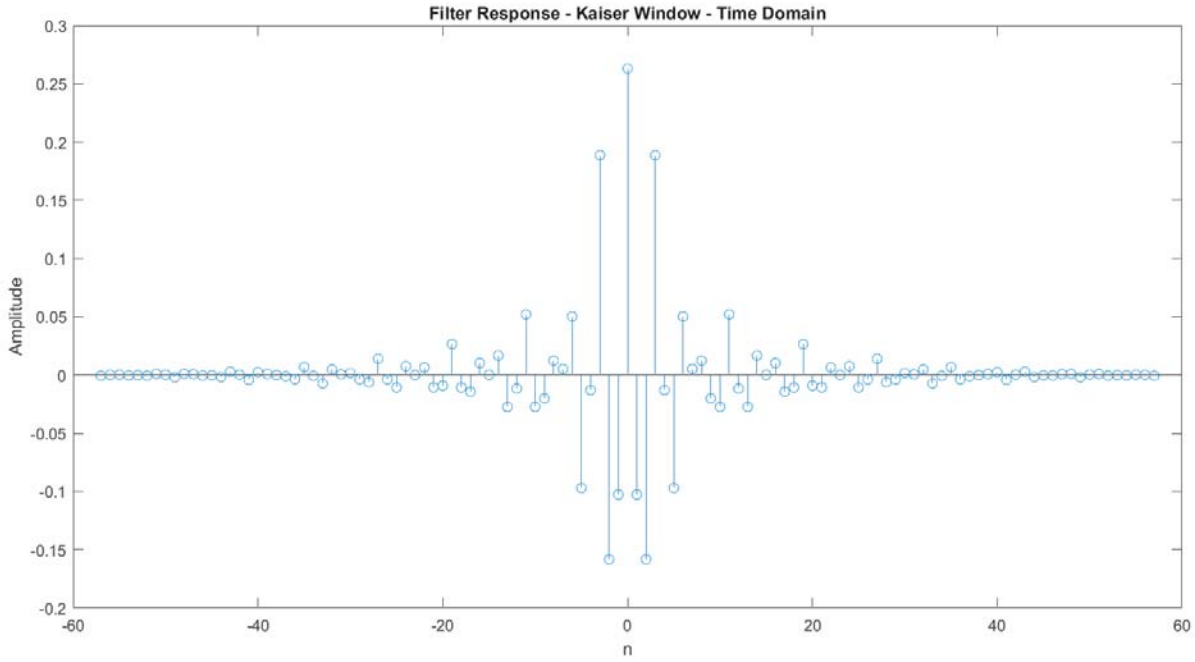

Figure 5: Time Domain Representation of the filter designed using a Kaiser Window Function

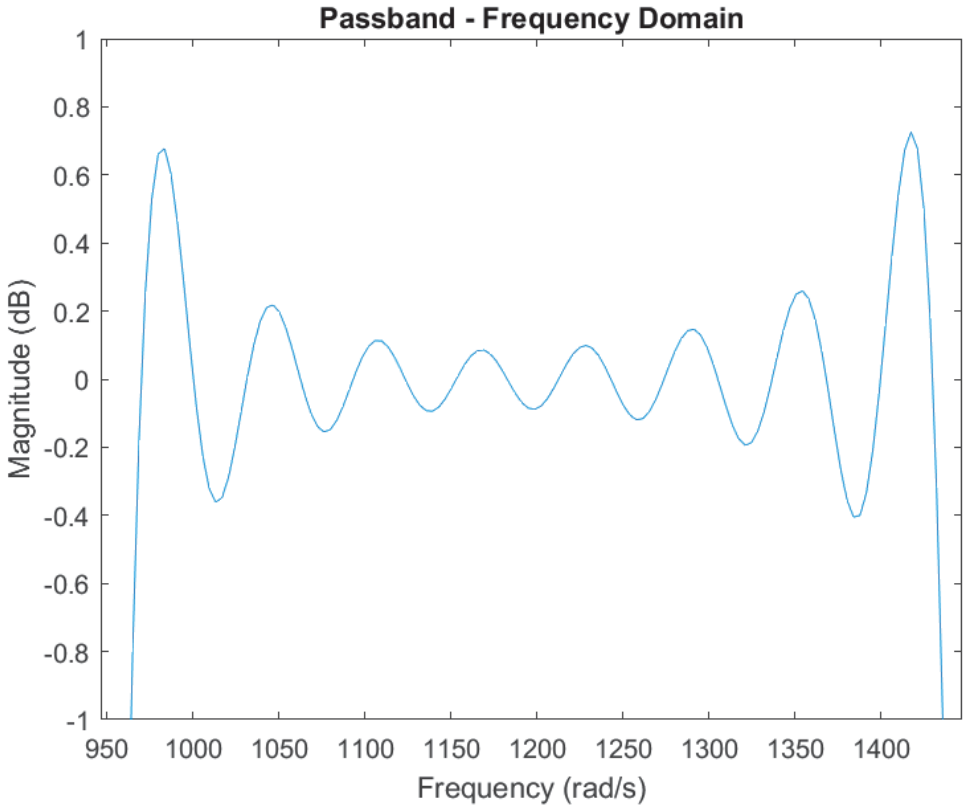

Figure 4: Passband of the Filter designed using the rectangular Window

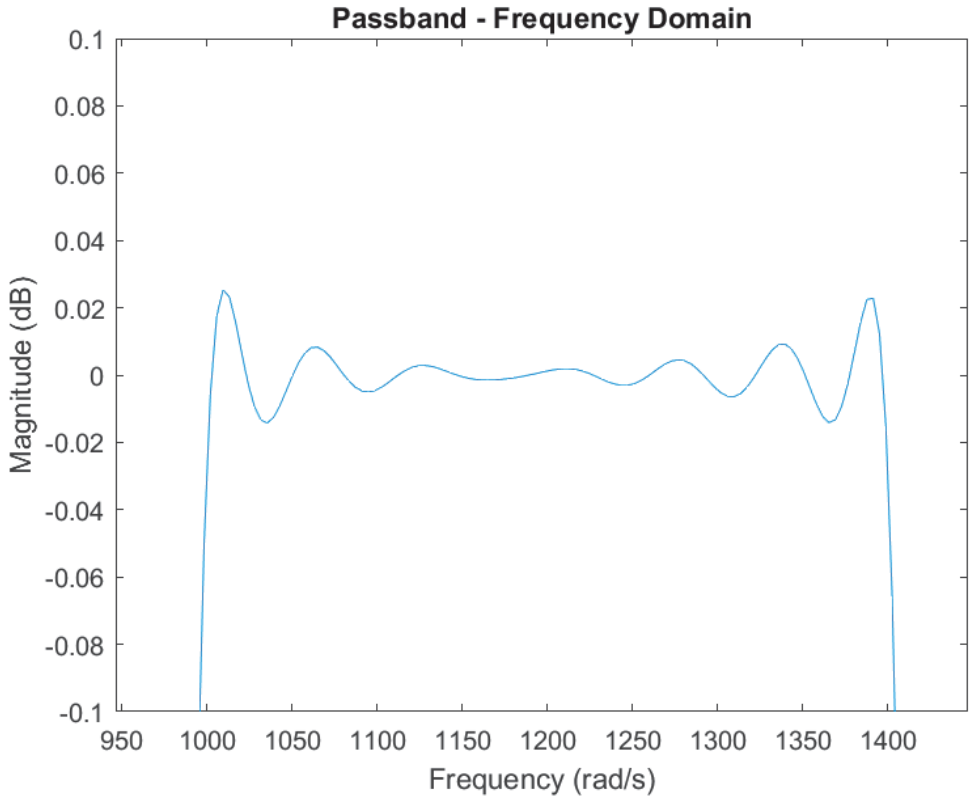

Figure 7: Passband of the Filter designed using the Kaiser Window

Input and Output of the Filter

The filter is given a combination of three sinusoidal signals which includes three mid-frequency bands. The following figures show the time domain and the frequency domain input signals(**Figure 8**) and output signals (**Figure 9**). In time-domain representation input and output, both impulses show in blue color and envelop of the signal is shown in orange color. Output signals contain only the frequency component in the passband frequency range.

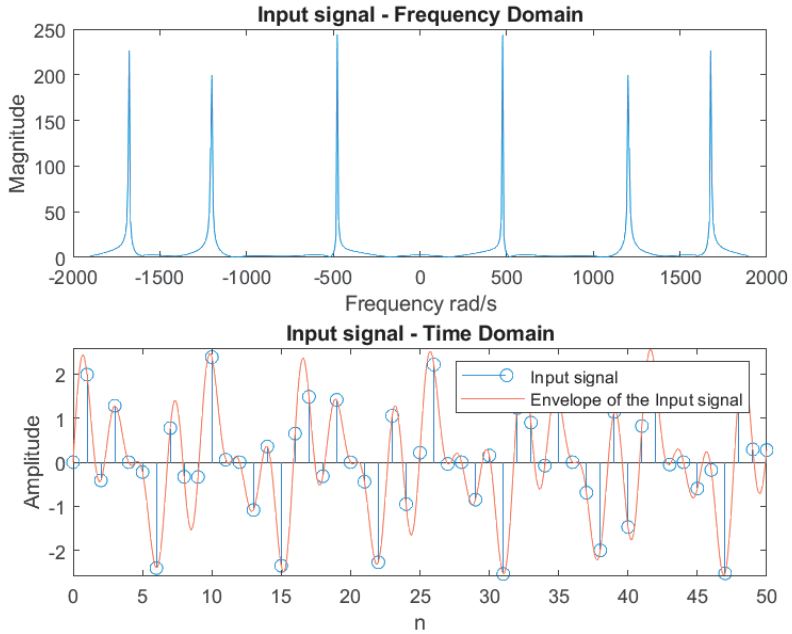

Figure 8: Frequency Domain and Time Domain representation of the input signal

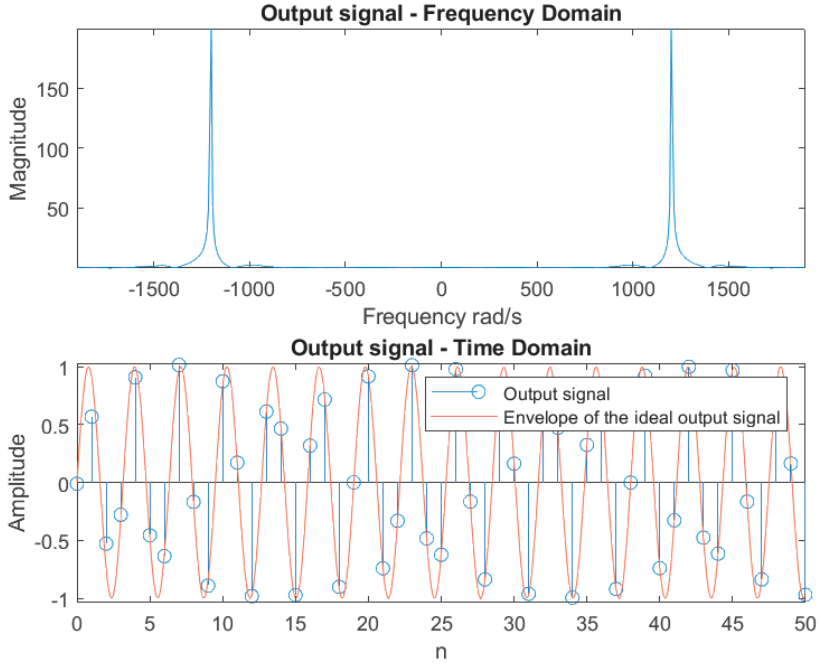

Figure 9: Frequency Domain and Time Domain representation of the output signal

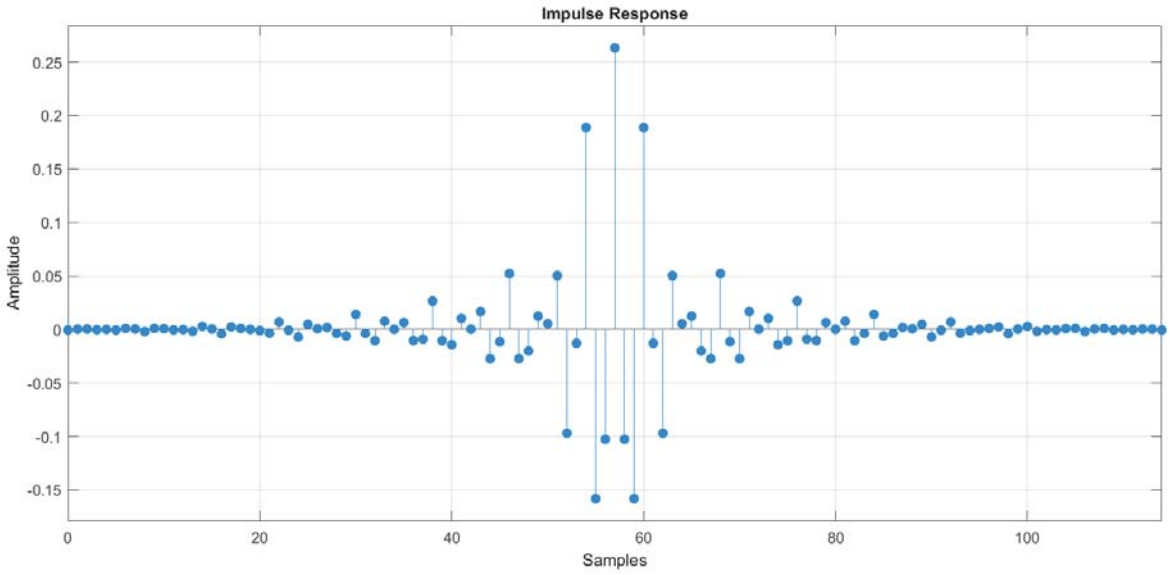

Figure 10: Time Domain representation of bandpass filter (Kaiser Window-fvtool)

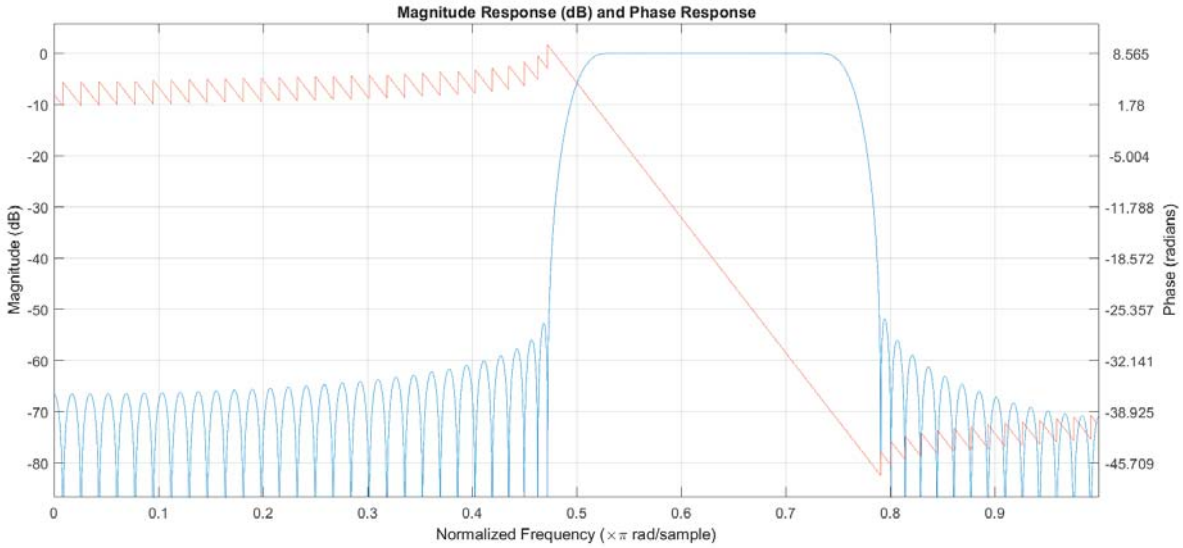

Figure 11: Frequency Domain representation of bandpass filter (Kaiser Window-fvtool)

IV DISCUSSION

Through the observations of figures in the results section, it appears to has been achieved the expected outcomes. Filter design has been completed successfully and the accuracy of the designed Kaiser window filter is at a good level compared to the rectangular window filter. Stopband attenuation of the Kaiser window method is less than -50dB (maximum approximate value is about -53dB). For the rectangular method that value is nearly -25dB. In **Figure 2**, **Figure 4**, and **Figure 11,** those critical observations can be identified clearly. When stopband attenuation is decreasing (negative value is increasing) accuracy of the method is increasing as the unnecessary frequency components are removed. Therefore, the Kaiser window technique accuracy is good instead of the accuracy of using rectangular window technique considering the above fact.

Another considerable factor is the passband ripple that describes the oscillation of the signal. Passband ripple of the two windows are shown in **Figure 5** (Passband of the Filter designed using the rectangular Window) and **Figure 7** (Passband of the Filter designed using the Kaiser Window). According to the figures in the rectangular window method, higher oscillations (nearly between 0.7 dB to - 0.4dB) can be seen compared to the Kaiser Window method (nearly between 0.02dB to -0.02dB). According to time-domain plot observation in the Kaiser window method gradually attenuates the samples at the edge of the window but in the Rectangular window, the method truncates impulse response abruptly. In **Figure 8** and **Figure 9** clear observations of the input and output signal in the time domain and frequency domain and frequency domain, can be identified. Input frequencies are 475 $rads^{-1}$, 1200 $rads^{-1,}$ and 1675 $rads^{-1}$; the passband frequency range is between 950 $rads^{-1}$ and 1450 $rads^{-1}$. Therefore, 1200 $rads^{-1}$ frequency range components are included in the output signal. This fact is evident from the tie domain representation of the input and output signals, where the output is clearly of single frequency valued.

V CONCLUSION

Allover this project the Kaiser Window method was able to show good filtered output. As ideal filters are not practicable filters, the Kaiser window filter is a good solution instead of using a Rectangular window filter which means the Kaiser window technique can give acceptable practical results. Although the Kaiser window filter passband signal contains small imperfection results such as small ripples (0.02dB to -0.02dB), there is no observable difference from the ideal filter output (**Figure 4**). Therefore, the parameters of the filter can be changed until it is achieving optimistic results for filter output.

For designing this filter low computational effort is required. However, this filter design is a high-order filter design. There may exist low order filters better than this filter with the same specifications. Hardware high order filters maybe not efficient because those filters use many delays, adders, and multiplexers. Software-implemented filters required more computational samples for the higher-order filters. Therefore for a more optimistic filter design, the filter method should be more advanced.

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Chameera U.S. Edussooriya, the project supervisor, for the constant guidance he provided throughout this project. I would also like to thank my colleagues for sharing their knowledge and experience on this project.

REFERENCES

[1] Antoniou.A "Digital Signal Processing - Signals Systems and Filters"(1st ed.).(2005). McGraw-Hill.

[2] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed., Pearson HigherEducation, 2010

APPENDIX

```
%%%%%%%%%%%%%%General specifications%%%%%%%%%%%%%%

A = 2;
B = 3;
C = 7;

A_p = 0.03+(0.01*A);  % dB max passband ripple
A_a = 45+B; %dB min stopband attenuation
wp1 = C*100+300; %rad/s lower passband edge
wp2 = C*100+700; %rad/s upper passband edge
wa1 = C*100+150; %rad/s lower stopband edge
wa2 = C*100+800; %rad/s upper stopband edge
ws  = 2*((C*100)+1200); %sampling freqency
```

```
%%%%%%%%%%%%%%Derived specifications%%%%%%%%%%%%%%

bt1 = wp1-wa1;  %lower transition width
bt2 = wa2-wp2;  % upper transisiton width
bt  = min(bt1,bt2); %critical transition width
wc1 = wp1-bt/2; % lower cutoff frequency
wc2 = wp2+bt/2; % upper cutoff frequency
T   = 2*pi/ws; % sampling period
```

```
%%%%%%%%%%%%%%Kaiser Window Parameters%%%%%%%%%%%%%%

delta_p = (10^(0.05*A_p) - 1)/ (10^(0.05*A_p) + 1); % calculating delta
delta_a = 10^(-0.05*A_a);
delta = min(delta_p,delta_a);
Aa = -20*log10(delta);  % Actual stopband attenuation
if Aa<=21               % Calculating alpha
    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

if Aa <= 21             % Calculating D
    D = 0.9222;
else
    D = (Aa-7.95)/14.36;
end

N = ceil(ws*D/bt +1); % order of the filter
if mod(N,2) == 0
    N = N+1;
End
n = -(N-1)/2:1:(N-1)/2;   % length of the filter
beta = alpha*sqrt(1-(2*n/(N-1)).^2);
```

```
%%%%%%%%%%%%%%Generating IoAlpha%%%%%%%%%%%%%%

bessellimit = 200;

IoAlpha = 1;
for k = 1:bessellimit
    var1 = (1/factorial(k)*(alpha/2).^k).^2;
    IoAlpha = IoAlpha + var1;
end
```

```
%%%%%%%%%%%%%%Generating IoBeta%%%%%%%%%%%%%%

IoBeta = 1;
for k = 1:bessellimit
    var2 = (1/factorial(k)*(beta/2).^k).^2;
    IoBeta = IoBeta + var2;
end
```

```
%%%%%%%%%%%%%%Obtaining Kaiser Window%%%%%%%%%%%%%%

wknt = IoBeta/IoAlpha;
figure
stem(n,wknt)
xlabel('n')
ylabel('Amplitude')
title('Kaiser Window - Time Domain');
```

```
%%%%%%%%%%%%%%Obtaining Impulse Response%%%%%%%%%%%%%%

n_left = -(N-1)/2:-1;     %negative part
hnt_left = 1./(n_left*pi).*(sin(wc2*n_left*T)-sin(wc1*n_left*T));

n_right = 1:(N-1)/2;      %Positive part
hnt_right = 1./(n_right*pi).*(sin(wc2*n_right*T)-sin(wc1*n_right*T));

hnt0 = 2/ws*(wc2-wc1);  %for n=0

hnt = [hnt_left,hnt0,hnt_right];
figure
stem(n,hnt)
xlabel('n')
ylabel('Amplitude')
title(strcat('Filter Response - Rectangular window - Time Domain'));

figure
[h,w] = freqz(hnt);
w = w/T;
h = 20*log10(h);
plot(w,h)
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title(strcat('Filter Response - Rectangular Window - Frequency Domain'));
```

```
%%%%%%%%%%%%%%%%%%Plotting the Passband(Rectangular)%%%%%%%%%%%%%%%%%%

[h,w] = freqz(hnt);
w = w/T;
h = 20*log10(h);
start = round(length(w)/(ws/2)*wc1);
finish = round((length(w)/(ws/2)*wc2));
wpass1 = w(start:finish);
hpass1 = (h(start:finish));
plot(wpass1,hpass1)
axis([-inf, inf, -1, 1]);
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title('Passband - Frequency Domain');
```

```
%%%%%%%%%%%%%%Applying the Kaiser window to the filter%%%%%%%%%%%%%%

filter = hnt.*wknt;
figure
stem(n,filter)
xlabel('n')
ylabel('Amplitude')
title(strcat('Filter Response - Kaiser Window - Time Domain'));

figure
[h,w] = freqz(filter);
w = w/T;
h = 20*log10(abs(h));
plot(w,h)
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title(strcat('Filter Response - Kaiser Window - Frequency Domain'));
```

```
%%%%%%%%%%%%%%Plotting the Passband%%%%%%%%%%%%%%

figure
start = round(length(w)/(ws/2)*wc1);
finish = round((length(w)/(ws/2)*wc2));
wpass = w(start:finish);
hpass = (h(start:finish));
plot(wpass,hpass)
axis([-inf, inf, -0.1, 0.1]);
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title('Passband - Frequency Domain');
```

```
%%%%%%%%%%%%%Input signal generation%%%%%%%%%%%%%

w1 = wc1/2;
w2 = (wc1 + wc2)/2;
w3 = (wc2 + (ws/2))/2;

%Generate discrete signal and evelope
samples = 500;
n1 = 0:1:samples;
n2 = 0:0.1:samples;
x_nT = sin(w1.*n1.*T)+sin(w2.*n1.*T)+sin(w3.*n1.*T);
x_env = sin(w1.*n2.*T)+sin(w2.*n2.*T)+sin(w3.*n2.*T);
```

```
%%%%%%%%%%%%%Filtering using frequency domain multiplication%%%%%%%%%%%%%

len_fft = length(x_nT)+length(filter)-1; %length for fft in x dimension
x_fft = fft(x_nT,len_fft);
filter_fft = fft(filter,len_fft);
out_fft = filter_fft.*x_fft; %a shift in time is added here
out = ifft(out_fft,len_fft);
rec_out = out(floor(N/2)+1:length(out)-floor(N/2)); %account for shifting delay

%Ideal Output Signal
ideal_out = sin(w2.*n2.*T);

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 = ws*linspace(0,1,len_fft)-ws/2;
plot(f,x_fft_plot);
xlabel('Frequency rad/s')
ylabel('Magnitude')
title(strcat(['Input signal',' ','- Frequency Domain']));
```

```
%%%%%%%%%%%%%Time domain representation of input signal before filtering%%%%%%

subplot(2,1,2)
stem(n1,x_nT)
axis([0, 50, -inf, inf]);
xlabel('n')
ylabel('Amplitude')
title(strcat(['Input signal',' ','- Time Domain']));
hold on
plot(n2,x_env)
legend('Input signal','Envelope of the Input signal');
```

```
%%%%%%%%%%%%%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 = ws*linspace(0,1,len_fft)-ws/2;
plot(f,x_fft_out_plot); axis tight;
xlabel('Frequency rad/s')
ylabel('Magnitude')
title(strcat(['Output signal',' ','- Frequency Domain']));
```

```
%%%%%%%%%%%%%Time domain representation of output signal after filtering%%%%%

subplot(2,1,2)
stem(n1,rec_out)
axis([0, 50, -inf, inf]);
xlabel('n')
ylabel('Amplitude')
title(strcat(['Output signal',' ','- Time Domain']));
hold on
plot(n2,ideal_out)
legend('Output signal','Envelope of the ideal output signal');

fvtool(filter);
```