DEPARTMENT OF ELECTRONIC & TELECOMMUNICATION ENGINEERING

UNIVERSITY OF MORATUWA

EN2570 – DIGITAL SIGNAL PROCESSING

PROJECT REPORT

DESIGN OF A FINITE DURATION IMPULSE RESPONSE(FIR) BAND PASS FILTER

T.M. AQEEL

180039C

THIS REPORT IS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE MODULE EN 2570 - DIGITAL SIGNAL PROCESSING

04TH OF MARCH 2021

Abstract – **In this report, we discuss the process of designing a Finite Impulse Response (FIR) band pass filter which satisfies a set of required specifications. The software implementation and evaluation of this filter was done by MATLAB (2018a). This digital filter is designed for the prescribed specifications using the Fourier Series method. Kaiser Window function is used to achieve the truncations of the impulse response. The filter is then evaluated for its performance by analyzing the outputs obtained for a test input consisting of a combination of sinusoidal signals.**

INTRODUCTION

In this report, we can see the procedure used to design a band pass FIR Digital Filter for prescribed specifications using the windowing technique in conjunction with the Kaiser window. The software implementation and evaluation of this filter was done by MATLAB (Version R2018a, MathWorks Inc.). To satisfy a given set of specifications, the filter is designed. Although there are many methods used to develop FIR digital filters, in this report, the digital filter is designed for the prescribed specifications using the Fourier Series method. Kaiser Window function is used to achieve the truncations of the impulse response. Throughout the different design stages, the time domain and frequency domain representations of the filter are obtained to evaluate the filter characteristics. Through the Fast Fourier Transform (FFT) algorithm implementation of the Discrete Fourier Transform (DFT), the frequency response of the filter was obtained primarily. To test the resulting filter implementation, a sinusoidal signal composed of 3 frequencies, out of which one lies in the pass band is input to the filter is used. The output is then compared with the original sinusoidal components of the input signal, in the pass band.

METHOD

There are two basic tasks of the project. One is to implement a band pass filter and the next is to evaluate its performance. The methodology of these two tasks will be discussed separately.

The design stage of the band pass filter is done in several stages. The first stage involves computation of design parameter according to the filter requirements that are provided. Then, obtaining the ideal frequency response of the pass band filter is the next stage. A Kaiser Window function is then obtained such that the ideal frequency response is truncated preserving the required filter parameters. The final stage is to obtain the frequency domain and time domain representation of the designed filter.

The task of evaluation involves generation of the input signals following which the output is obtained by convolution of the input with the designed filter. Comparison between the expected output and the filter response is done by frequency and time domain plots of the inputs and the outputs.

I. Filter Implementation

1. Prescribed Filter Specifications

The required parameters for implementing the filter were calculated from the provided filter specifications which are shown in Table 1.

Index Number: - 180039C A = 0, B= 3, C=9, ·= C

Specification	Symbol	Calculation	Value	Units
Maximum pass band ripple	$\tilde{A}_p$	0.03 + (0.01×0)	0.03	dB
Minimum stopband attenuation	$\tilde{A}_a$	45 + 3	48	dB
Lower pass band edge	Ω_{p1}	(9×100) + 300	1200	rad/s
Upper pass band edge	Ω_{p2}	(9×100) + 700	1600	rad/s
Lower stop band edge	Ω_{a1}	(9×100) + 150	1050	rad/s
Upper stop band edge	Ω_{a2}	(9×100) + 800	1700	rad/s
Sampling frequency	Ω_s	2[(9×100) + 1200]	4200	rad/s

Table 1 : Specifications

2. Derived Filter Specifications

Based on the provided specifications, the following parameters shown in Table 2 were derived to design the filter and the Kaiser Window.

Derived Parameter	Symbol	Derivation	Value	Units
Lower transition width	B_{t1}	Ω_{a1}-Ω_{p1}	150	rad/s
Upper transition width	B_{t2}	Ω_{a2}-Ω_{p2}	100	rad/s
Critical transition width	B_t	Min [B_{t1}, B_{t2}]	100	rad/s
Lower cutoff frequency	Ω_{c1}	$\Omega_{p1} - \frac{B_t}{2}$	1150	rad/s
Upper cutoff frequency	Ω_{c2}	$\Omega_{p2} + \frac{B_t}{2}$	1650	rad/s
Sampling period	T	$\frac{2\pi}{\Omega_s}$	0.0015	s

Table 2 : Derived Specifications

3. Derivation of the Kaiser Window Parameters

- Using the derived parameters, the Kaiser window is obtained in the following form,

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

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

- The parameters α and N are calculated as follows,

A parameter δ is defined in a way that the actual pass band ripple $A_p \le \tilde{A}_p$, and the actual minimum stopband attenuation, $A_a \ge \tilde{A}_a$.

This is achieved when,

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

$$Where; \qquad \delta_p = \frac{10^{0.05\tilde{A}_p} - 1}{10^{0.05\tilde{A}_p} + 1} \qquad and \qquad \delta_a = 10^{-0.05\tilde{A}_a}$$

With the δ obtained thus, $\qquad A_a = -20\log(\delta)$

Now α is chosen as

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

To obtain N, we define a parameter D as,

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

N is chosen so that it is the smallest, odd integer satisfying the following inequality.

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

Now let us calculate the Kaiser Window parameters.

$$\delta_p = \frac{10^{0.05(0.03)} - 1}{10^{0.05(0.03)} + 1} = 1.7269 \times 10^{-3} \quad and \quad \delta_a = 10^{-0.05(48)} = 3.9810 \times 10^{-3}$$

$$\delta = 1.7269 \times 10^{-3}$$

$$A_a = 55.2545\ dB$$

$$\alpha = 0.1102(55.2545 - 8.7) = 5.1303$$

$$D = \frac{55.2545 - 7.95}{14.36} = 3.2942$$

$$N \geq \frac{4200 \times 3.2942}{100} + 1 = 139.35 \qquad N = 141$$

A summary of Kaiser Window parameters obtained is shown in Table 3,

Parameter	Value	Units
δ	3.981×10^{-3}	-
A_a	55.2545	dB
α	5.1303	-
D	3.2942	-
N	141	-

Table 3 : Kaiser Window Parameters

4. Derivation of The Ideal Impulse Response:

The frequency response of an ideal band pass filter with cut off frequencies Ω_{c1} and Ω_{c2} are given by,

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

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

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

5. Obtaining the Impulse Response of the Windowed Filter:

The impulse response of the filter $w(nT)$ can be obtained by the multiplication of the Ideal Impulse Response $h[nT]$ and the Kaiser Window $w_k(nT)$.

$$filter(nT) = w_k(nT)\text{h(nT)}$$

The z – Transform of which takes the form,

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

Upon shifting for causality which becomes

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

$filter(nT)$ in time domain is the final filter. In order for comparison, we also obtain a filter that is windowed with a rectangular window.

II. Filter Evaluation

To evaluate the performance of the generated filter, an input signal $x(nT)$ is constructed as the sum of three sinusoid each having frequencies below the pass band, within the pass band and above the pass band as shown in Table 4. The output can be obtained by the convolution of the filters impulse response $filter[nT]$ and $x(nT)$. To avoid the use of convolution operation, we obtain the Discrete Fourier Transform (DFT) of these two signals and multiply them in the frequency domain to obtain the frequency domain representation of the output, which is again converted back to time domain by Inverse Discrete Fourier Transform (IDFT). DFT and IDFT are implemented respectively by Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform IFFT methods in MATLAB.

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

Frequency Component	Value	Unit
Ω1	575	rad/s
Ω2	1400	rad/s
Ω3	1875	rad/s

Table 4 : Input Frequency Components

RESULTS

The results of the filter design can be presented in two forms. The characteristics of the filter in each stage of the filter design process can be seen by the impulse response and frequency response plots. The performance of the filter can be evaluated by comparing the input and output obtained by the filter evaluation stage.

1. Frequency and Time Domain plots of the Filter

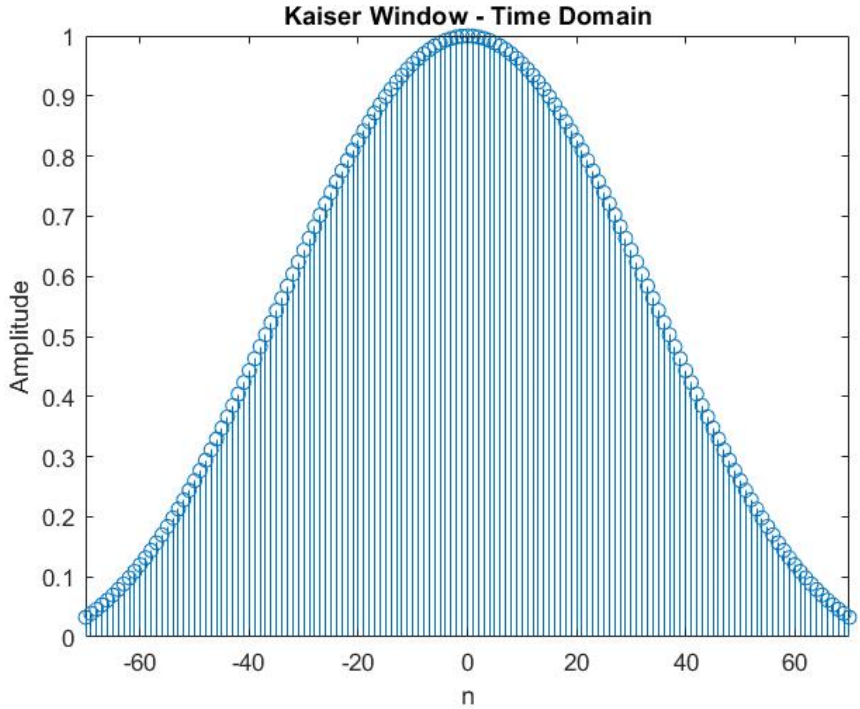

Figure 1 : Impulse response of Kaiser Window

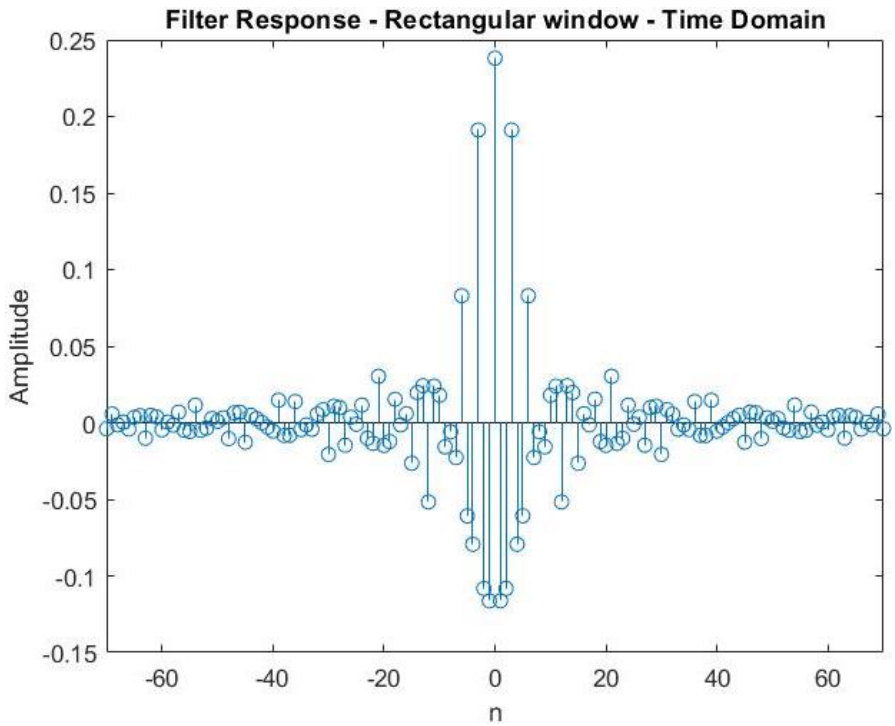

Figure 2: Time Domain Representation of the Rectangular Window Function filter design

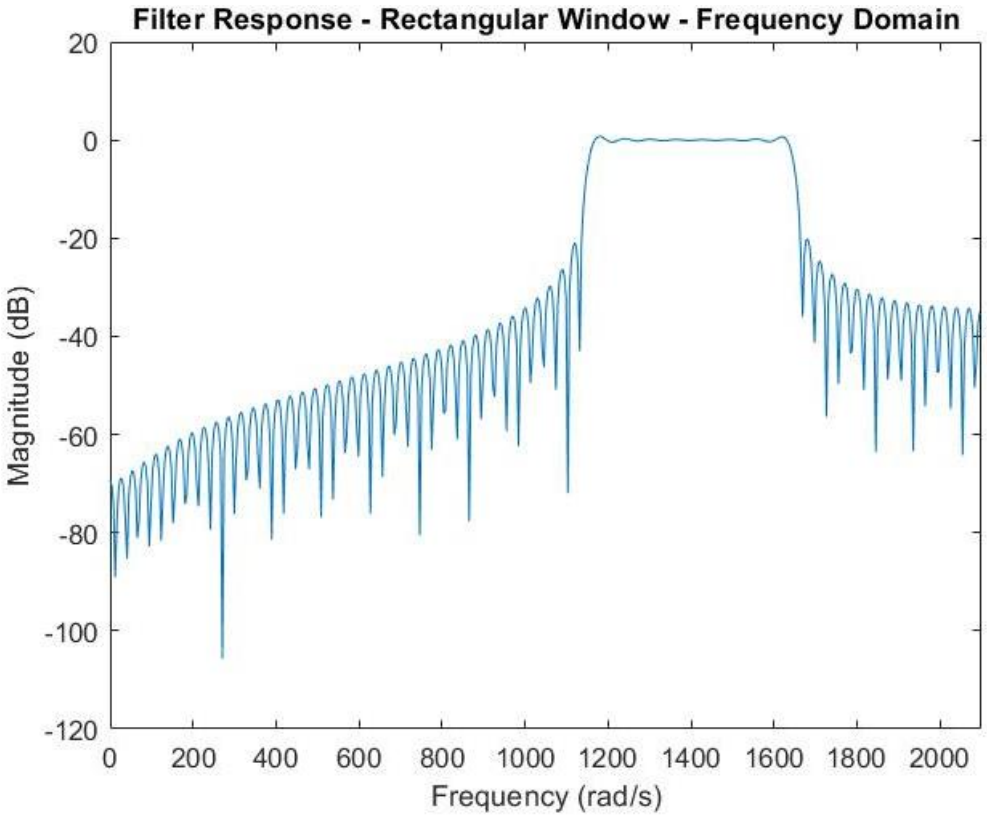

Figure 3 : Frequency Domain Representation of the Rectangular Window Function filter design

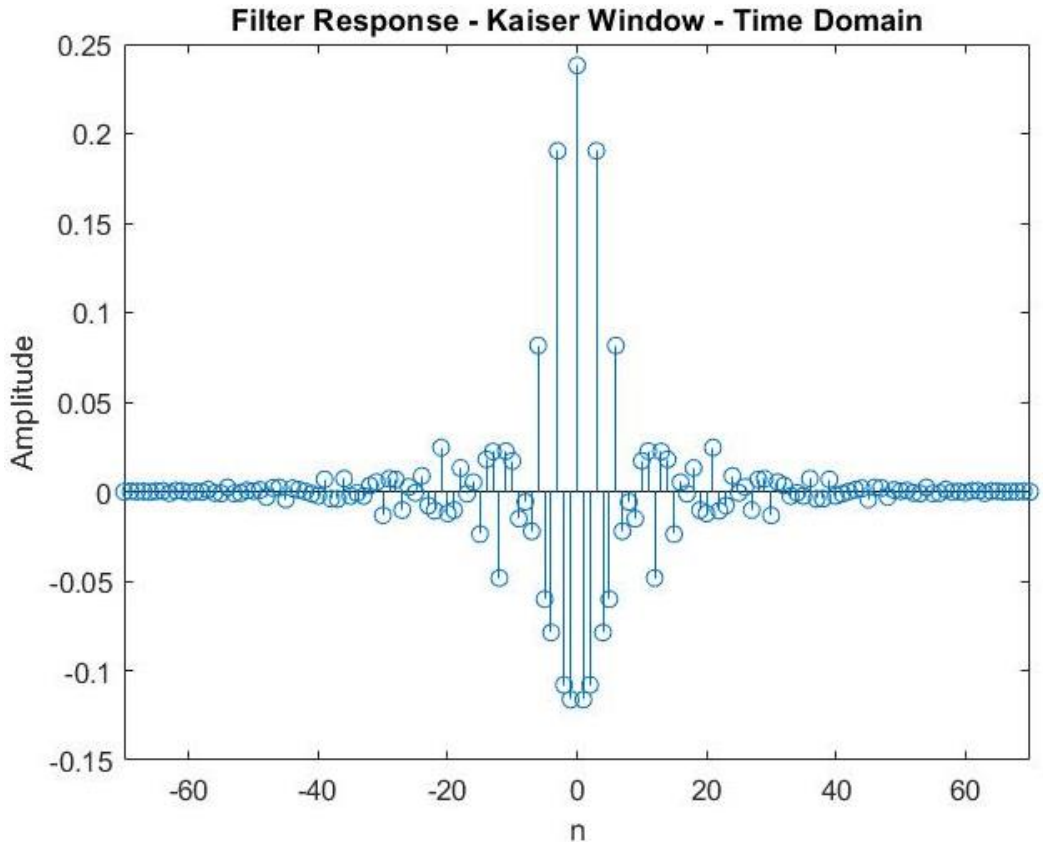

Figure 4 : Time Domain Representation of the Kaiser Window Function filter design

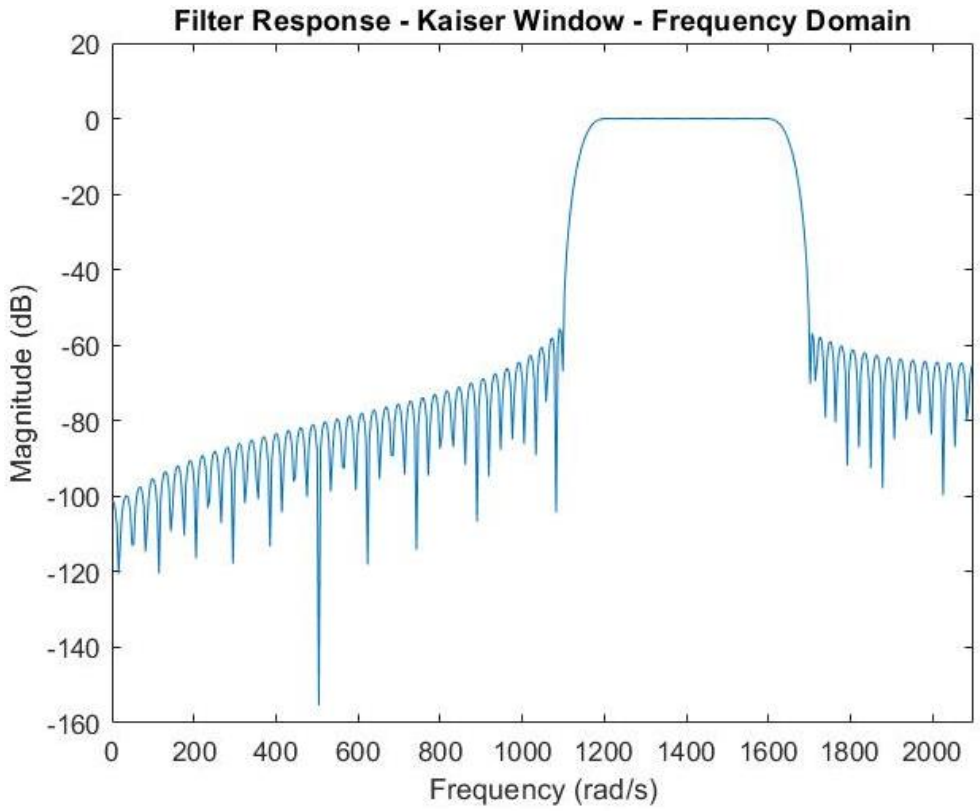

Figure 5 : Frequency Domain Representation of the Kaiser Window Function filter design

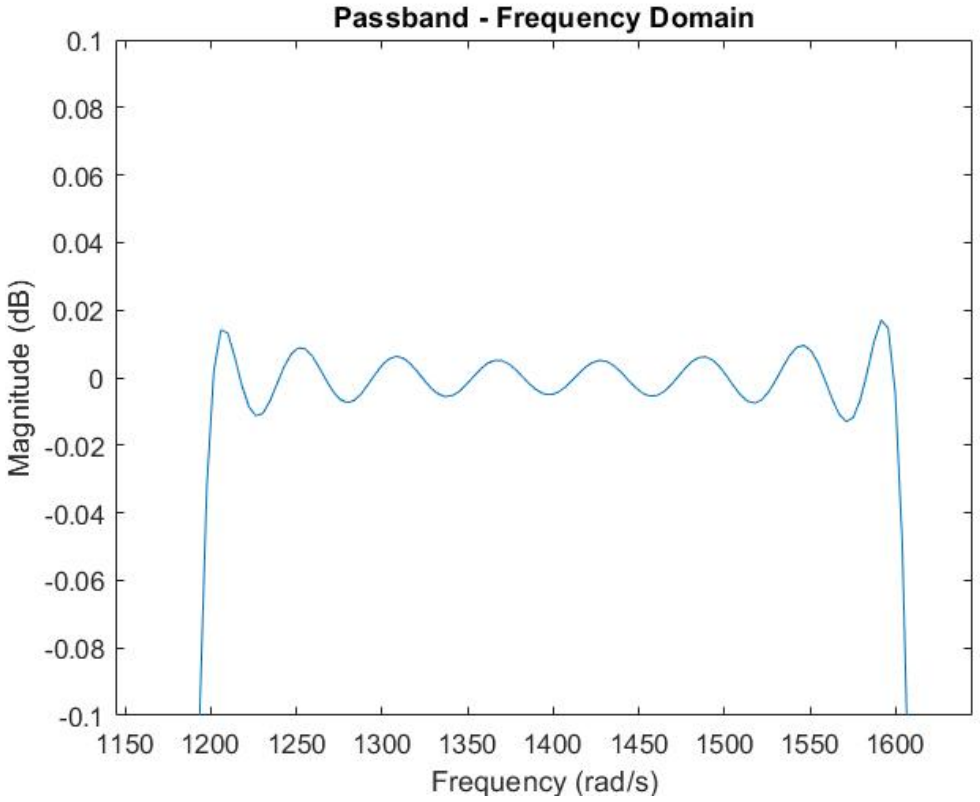

Figure 6 : Pass band of the Filter designed using the Kaiser Window

2. Input and Output of the Filter

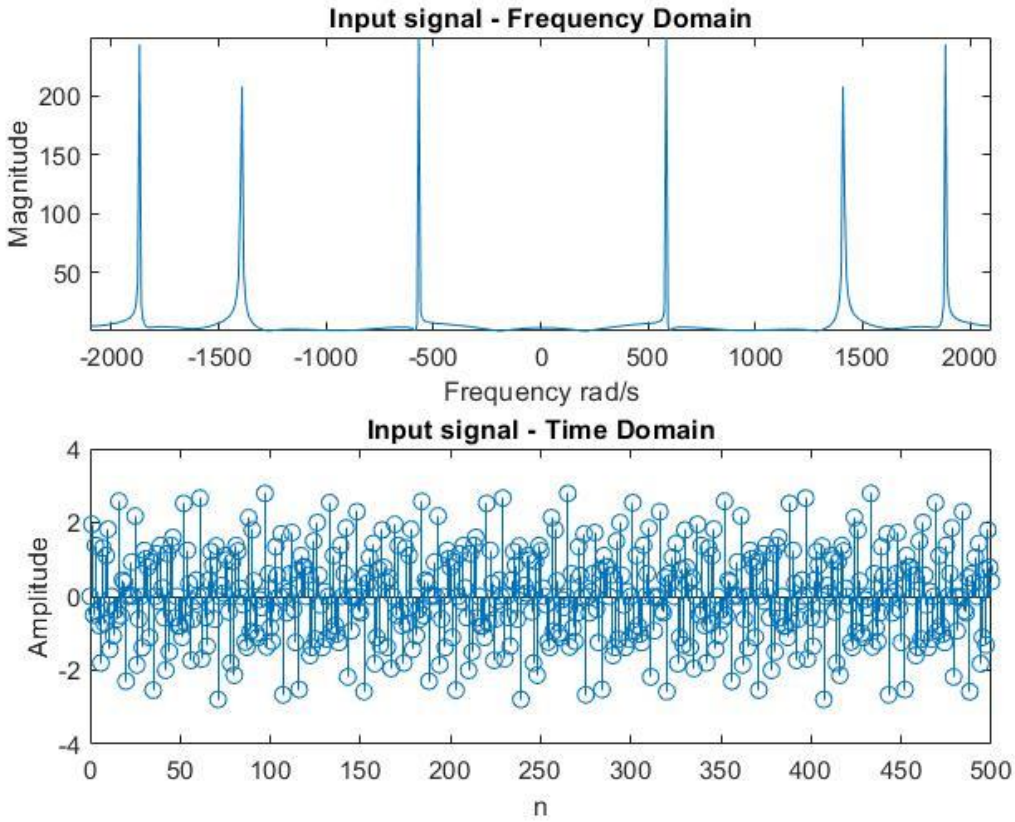

Figure 7 : Input signal - (Frequency domain & Time domain)

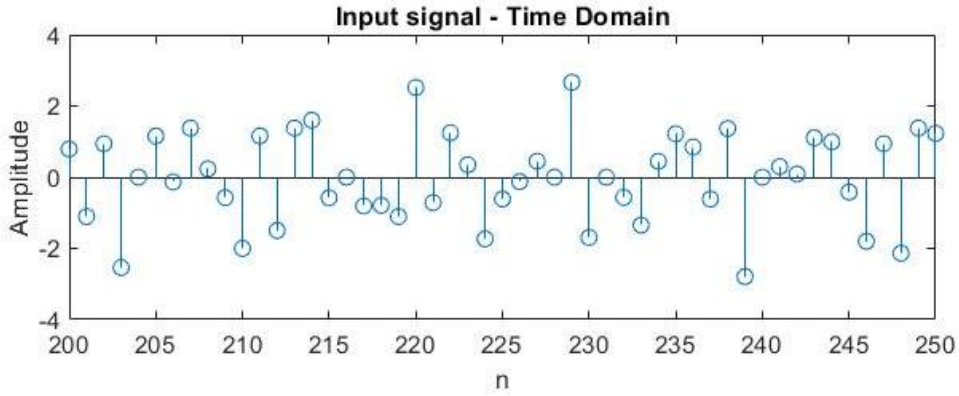

Figure 8 : Input Signal Time domain (zoomed)

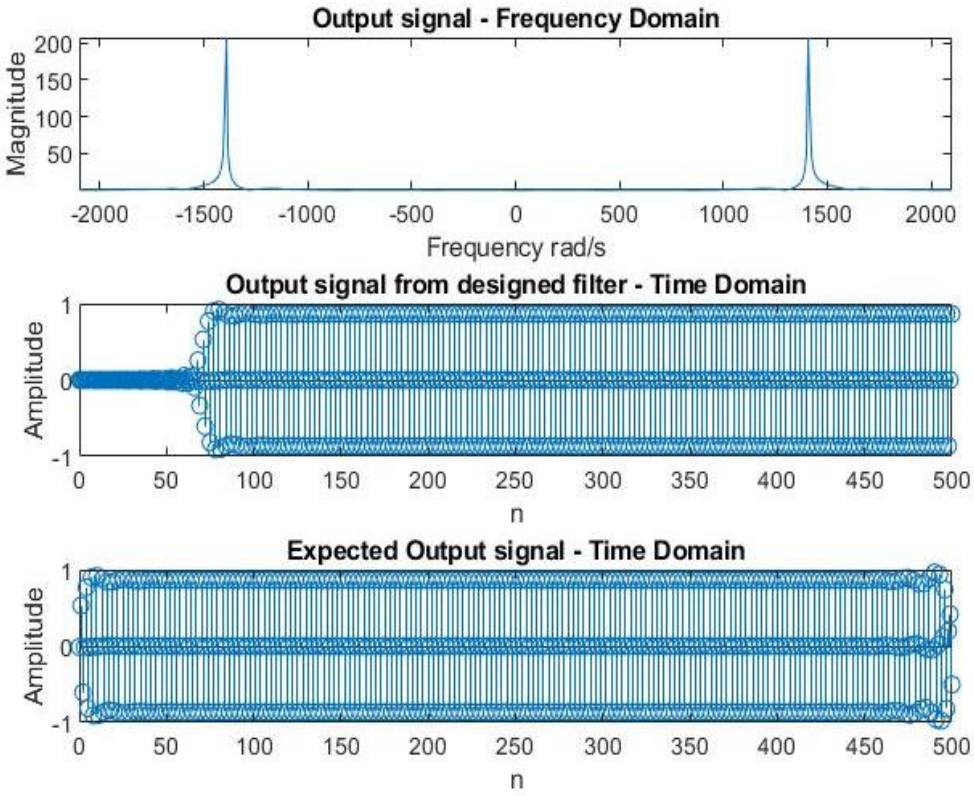

Figure 9 : Output Signal - (Frequency domain & Time domain)

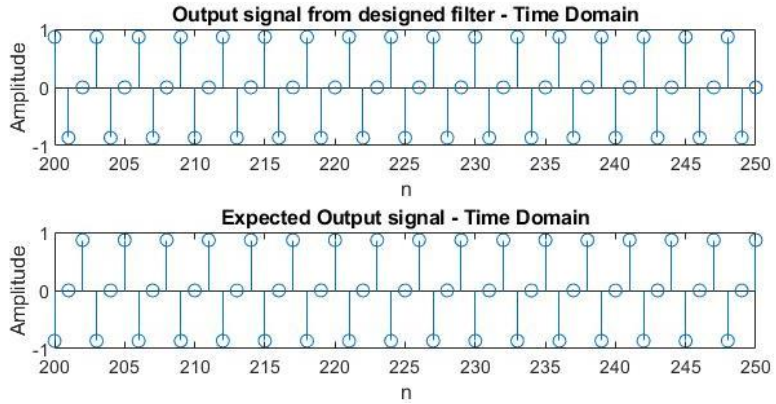

Figure 10 : Output Signal Time domain (Zoomed)

DISCUSSION

We could see several significant differences between the filter obtained by Rectangular Window and the Kaiser Window. Comparing Figures 3 and 5 it was clearly seen that the stop band ripple in Rectangular Window filter was significantly high and non-uniform compared to the Kaiser Window. Further only in the Kaiser Window method the required stop band attenuation is achieved. Compared to the Rectangular Window method, the pass band ripple in the Kaiser Window method is clearly seen to be smaller. The time domain plots of the filter clearly shows that the Kaiser Window method gradually attenuates samples at the edge of the window, while the rectangular window truncates the impulse response abruptly. This is the reason for smoother pass band in the Kaiser Window method. The frequency domain representation of the output signal to the filter shown in Figure 9 clearly shows that the stop band components with frequencies 1875 rad/s and 575rad/s have been attenuated while the 1400 rad/s pass band component has been preserved. From the time domain representation of the input and output signals, this fact is evident, where the output is clearly of single frequency valued.

CONCLUSION

The flexibility of the Kaiser Window is evident through the above results. We cannot practically implement ideal filters. So, it is useful to make a flexible filter of which the limitations can be controlled. This is a practical approach since small imperfections such as pass band ripple will not cause an observable difference in the filtered output. This means that the parameters of the filter can be adjusted until the differences between the output and an ideal output become indistinguishable for all practical purposes. We can see that the results of the designed filter are identical to that of an ideal filter from the comparisons that we have done above. So, we can get an acceptable result from the Kaiser Window method. Even though it is not an ideal filter, this method is flexible and can be adjusted using the input parameters. Also, we need a small computational effort to design in this method. Requirement of higher order of filter is the major drawback of this filter design. There can be optimal filters of lower order that can achieve the same specifications. When implemented in hardware higher order filters can be inefficient due to the usage of more unit delays, adders, and multipliers. In the software implementation, higher order filters require more computations per sample. More advanced design techniques must be used to obtain optimality.

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] Digital Signal Processing: Signals, Systems, and Filters by Andreas Antoniou

APPENDIX

The following code was used to design, simulate, and validate the band stop filter.

Contents

EN2570 Digital Signal Processing Project

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

Calculate the paramteres

```
indexNo = str2double('180039');
%indexNo = str2double(inputdlg('Enter Index Number','Index Number',1));

%Get A,B,C in 180ABC ( index )
A = mod(floor(indexNo/100),10);
B = mod(floor(indexNo/10),10);
C = mod(indexNo,10);
%A = 0; B = 3; C = 9;
A_tilda_p = 0.03 + (0.01 * A) ; %dB Maximum passband ripple, A~p
A_tilda_a = 45 + B;             %dB Minimum stopband attenuation, A~a
Omega_p1 = C*100 + 300;         %rad/s Lower passband edge
Omega_p2 = C*100 + 700;         %rad/s Upper passband edge
Omega_a1 = C*100 + 150;         %rad/s Lower stopband edge
Omega_a2 = C*100 + 800;         %rad/s Upper stopband edge
Omega_s = 2*(C*100 + 1200);     %rad/s Sampling frequency
```

Derived Filter Specifications

```
B_t1 = Omega_p1-Omega_a1;       %rad/s lower transition width
B_t2 = Omega_a2-Omega_p2;       %rad/s upper transisiton width
B_t = min(B_t1,B_t2);           %rad/s critical transition width
Omega_c1 = Omega_p1-B_t/2;      %rad/s lower cutoff frequency
Omega_c2 = Omega_p2+B_t/2;      %rad/s upper cutoff frequency
T = 2*pi/Omega_s;               %s sampling period
```

Kaiser Window Parameters

```
%Choose Delta
delta_p = ((10^(0.05*A_tilda_p))-1)/((10^(0.05*A_tilda_p))+1);
delta_a = 10^(-1*(0.05*A_tilda_a));
delta = min(delta_p,delta_a);
%Actual Stopband attenuation
A_a = -20*log10(delta);
%Choose Alpha
if A_a<=21
    alpha = 0;
elseif A_a>21 && A_a<= 50
    alpha = 0.5842*(A_a-21)^0.4 + 0.07886*(A_a-21);
else
    alpha = 0.1102*(A_a-8.7);
end
```

```
%Choose D
if A_a <= 21
    D = 0.9222;
else
    D = (A_a-7.95)/14.36;
End

% Calculating order of the filter N
N = ceil((Omega_s*D/B_t) +1);
if mod(N,2) == 0
    N = N+1;
End
%Range of N
n_lim = floor(N-1)/2;

% Length of the filter
n = -n_lim:1:n_lim;

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

Generating I_0_alpha

```
bessellimit =50;
I_0_alpha = 1;
for k = 1:bessellimit
    termk = (1/factorial(k)*(alpha/2).^k).^2;
    I_0_alpha = I_0_alpha + termk;
end
```

Generating I_0_beta

```
I_0_beta = 1;
for k = 1:bessellimit
    termk = (1/factorial(k)*(beta/2).^k).^2;
    I_0_beta = I_0_beta + termk;
end
```

Obtaining Kaiser Window w_k(nT)

```
wk_nt = I_0_beta/I_0_alpha;
figure
stem(n,wk_nt)
xlim([-70 70])
xlabel('n')
ylabel('Amplitude')
title('Kaiser Window - Time Domain');
```

Generating Impulse Response h(nT)

```
n_left = -(N-1)/2:-1;
hnt_left = 1./(n_left*pi).*(sin(Omega_c2*n_left*T)-sin(Omega_c1*n_left*T));

n_right = 1:(N-1)/2;
hnt_right = 1./(n_right*pi).*(sin(Omega_c2*n_right*T)-sin(Omega_c1*n_right*T));

hnt_0 = 2/Omega_s*(Omega_c2-Omega_c1);
hnt = [hnt_left,hnt_0,hnt_right];

figure
stem(n,hnt)
xlim([-70 70])
xlabel('n')
ylabel('Amplitude')
title(strcat(['Filter Response - Rectangular window - Time Domain']));

figure
[h,w] = freqz(hnt);
w = w/T;
h = 20*log10(abs(h));
plot(w,(h))
xlim([0 2100])
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title(strcat(['Filter Response - Rectangular Window - Frequency Domain']));
```

Applying the Kaiser Window to the filter

```
Hw_nT = hnt.*wk_nt;

Figure
stem(n,Hw_nT);
xlim([-70 70])
xlabel('n')
ylabel('Amplitude')
title(strcat(['Filter Response - Kaiser Window - Time Domain']));
%fvtool(Hw_nT);
Figure
[h,w] = freqz(Hw_nT);
w = w/T;
h = 20*log10(abs(h));
plot(w,h)
xlim([0 2100])
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title(strcat(['Filter Response - Kaiser Window - Frequency Domain']));
```

Plotting the Pass band

```
Figure
start = round(length(w)/(Omega_s/2)*Omega_c1);
finish = round(length(w)/(Omega_s/2)*Omega_c2);
w_pass = w(start:finish);
h_pass = h(start:finish);
plot(w_pass,h_pass)
axis([-inf, inf, -0.1, 0.1]);
xlabel('Frequency (rad/s)')
ylabel('Magnitude (dB)')
title('Passband - Frequency Domain');
```

Input signal generation

```
% component frequencies of the input
Omega_1 = Omega_c1/2;
Omega_2 = Omega_c1 + (Omega_c2-Omega_c1)/2;
Omega_3 = Omega_c2 + (Omega_s/2-Omega_c2)/2;

% generate discrete signal and evelope
samples = 500;
n1 = 0:1:samples;
n2 = 0:0.1:samples;
X_nT = sin(Omega_1.*n1.*T)+sin(Omega_2.*n1.*T)+sin(Omega_3.*n1.*T);
X_env = sin(Omega_1.*n2.*T)+sin(Omega_2.*n2.*T)+sin(Omega_3.*n2.*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; % 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(Omega_1.*n2.*T)+sin(Omega_3.*n2.*T);

% 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(floor(len_fft/2)+1:len_fft)]),abs(x_fft(1)),abs(x_fft(2:floor(len_fft/2
))))];
```

```
f = Omega_s*linspace(0,1,len_fft)-Omega_s/2;
plot(f,x_fft_plot); axis tight;
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)
%xlim([200 250])  %Uncomment for zoomed plot
xlabel('n')
ylabel('Amplitude')
title(strcat(['Input signal',' ','- Time Domain']));

% Frequency domain representation of output signal after filtering
figure
subplot(3,1,1)
len_fft = 2^nextpow2(numel(n1))-1;
xfft_out = fft(rec_out,len_fft);
x_fft_out_plot
=[abs([xfft_out(floor(len_fft/2)+1:len_fft)]),abs(xfft_out(1)),abs(xfft_out(2:floor
(len_fft/2)))];
f = Omega_s*linspace(0,1,len_fft)-Omega_s/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(3,1,2)
stem(n1,out(1:samples+1))
%xlim([200 250])    %Uncomment for zoomed plot
xlabel('n')
ylabel('Amplitude')
title(strcat(['Output signal from designed filter',' ','- Time Domain']));
subplot(3,1,3)
stem(n1,rec_out)
%xlim([200 250])    %Uncomment for zoomed plot
xlabel('n')
ylabel('Amplitude')
title(strcat(['Expected Output signal',' ','- Time Domain']));
```

Published with MATLAB® R2018a