Department of Electronic &
Telecommunication Engineering
University of Moratuwa

EN2570 Digital Signal Processing

Project Report

FIR Filter Design Project

Author:
Ramith Udara Hettiarachchi

Index Number:
170221T

This report is submitted as a partial fulfillment for the EN2570 - Digital Signal Processing module.
January 2, 2020

Contents

List of Figures

List of Tables

1 Abstract

Digital filters play a major role in a vast variety of fields. Thus, it is important to understand the fundamental concepts of filter design. In this project, the objective is to get hands on experience building a Finite Impulse Response(FIR) Filter.
FIR filters are designed based on the approximation of an Ideal Filter.

In our scope, an FIR filter will be designed using the windowing method in conjunction with the Kaiser window. The specifications are given in Table 1.

2 Introduction

In this report, my step by step approach in designing the Bandstop FIR Filter is explained. The objective behind this project was to develop our own code without using the built-in function in MATLAB to design the filter. Thus, several helper functions were also written to calculate the Bessel function and other tasks.
I carried out the design process using the MATLAB R2019b software, and my code files can be viewed here. https://github.com/ramithuh/BandStop-FIR-Filter-Design

The filter specifications corresponding to my index number are listed below.
(170ABC == 170221)

Table 1: Filter Specifications Given

Parameter	**Symbol**	**Relationship**	**Value**
Maximum passband ripple	$\tilde{A}_p$	$0.03 + (0.01 \times A)$ dB	**0.05 dB**
Minimum stopband attenuation	$\tilde{A}_a$	45+B dB	**47 dB**
Lower passband edge	Ω_{p1}	$(C \times 100) + 400$ rad/s	**500 rad/s**
Upper passband edge	Ω_{p2}	$(C \times 100) + 950$ rad/s	**1050 rad/s**
Lower stopband edge	Ω_{a1}	$(C \times 100) + 500$ rad/s	**600 rad/s**
Upper stopband edge	Ω_{a2}	$(C \times 100) + 800$ rad/s	**900 rad/s**
Sampling frequency	Ω_s	$2[(C \times 100) + 1300]$ rad/s	**2800 rad/s**

3 Basic Theory

3.1 Window Method

Windowing method is one of the simplest ways in which a Finite Impulse Response(FIR) filter can be approximated. Equation (1) represents the **ideal desired frequency response.** It is clear that this is non-causal as well as infinite. Therefore, to get an FIR approximation, we need to truncate this ideal desired frequency response. One way to obtain such truncation is known as *windowing method*.

$$H_d\left(e^{j\Omega}\right) = \sum_{n=-\infty}^{\infty} h_d[n]\ e^{-j\Omega n} \tag{1}$$

We can obtain a causal FIR filter if we truncate $h_d[n]$ as described in the equation below.

$$h[n] \ = \ \begin{cases} h_d[n], & 0 \leqslant n \leqslant M \\ 0, & otherwise. \end{cases}$$

Equation (2), shows the *general* way how the ideal frequency response is truncated in the windowing method according to an **arbitrary window function w[n]**.

$$h[n] \ = \ h_d[n]\ w[n] \tag{2}$$

There are various window types/functions introduced over the years. The four window types listed below have only 1 adjustable parameter:*window size*.

- Hamming
- von Hann
- Blackman
- Barlett

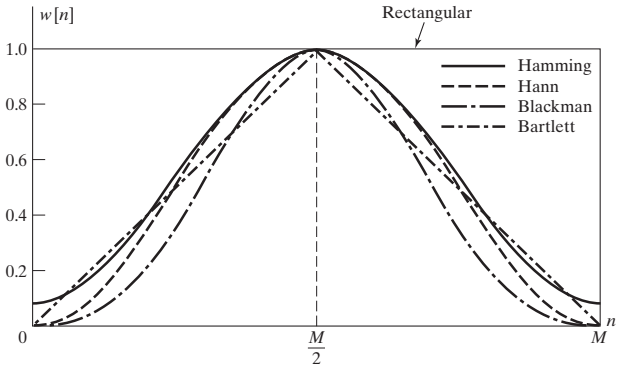


Figure 1: Commonly used windows $w[n]$ [1]

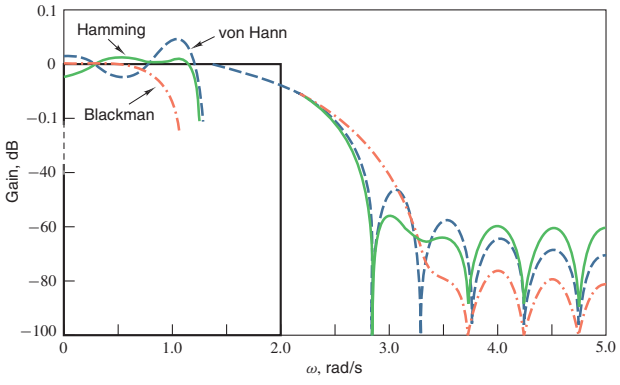


Figure 2: Amplitude response of common window types $w[n]$ [2]

The main problem with most of these windows is that, we cannot control the *peak approximation error* ($20 \log \delta$ *dB)* and the *transition width* (Δw, *Figure 3)* simultaneously. **Kaiser window method**, provides much convenience to overcome these issues. Compared to other window types, the kaiser window method has two parameters: *window length & β*

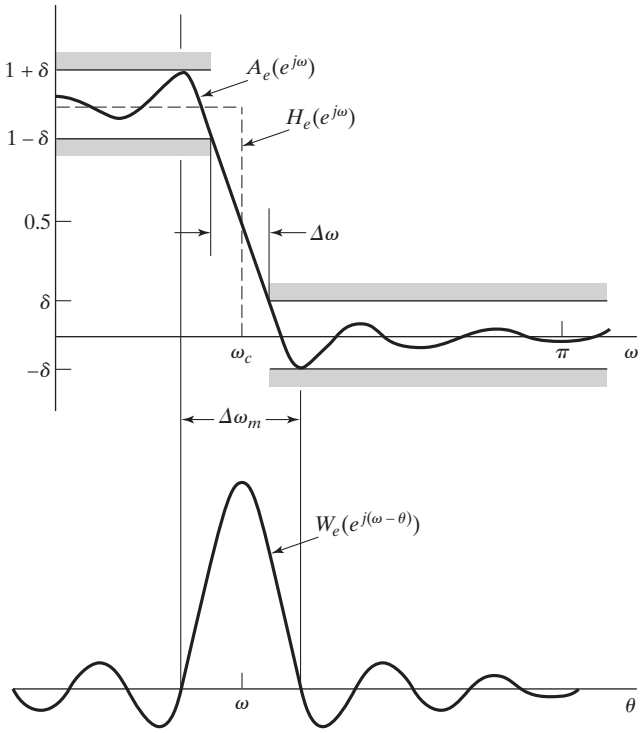


Figure 3: Approximation Errors [1]

3.2 Kaiser Window Method

Our intention when designing a FIR filter is to make the approximation very much closer to an ideal filter. However, the main two causes to approximation errors, *Main-lobe width & Side-lobe area* are very well linked to each other, which makes it hard to get an optimal filter design in a straightforward manner. Kaiser, was able to derive a near-optimal window design which can be formed using the zero$^{\text{th}}$ order Bessel Function of the first kind.

The Kaiser window is defined as,

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

where α is an independent parameter and

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

As mentioned previously, in this windowing method there are two parameters to be configured.

1. Window size
2. β Parameter

As the name suggests, the Window size means the length (N) of the window. The β parameter however is not straightforward. β controls the side-lobe levels and the main-lobe width.[3] **when β increases, the side-lobe levels decreases. However, at the same time, the main lobe becomes wider, which is an adverse effect.**

4 Kaiser Window - Design Steps

4.1 Deriving the Impulse Response of the Ideal Filter

4.1.1 Calculating the Realistic Specifications

As this is a bandstop filter, there will be two transition widths. Therefore, we need to select the narrowest transition width.

$$B_{t1} = \Omega_{a1} - \Omega_{p1} = \mathbf{100\ rad/s}$$

$$B_{t2} = \Omega_{p2} - \Omega_{a2} = \mathbf{150\ rad/s}$$

$$\star\ \mathbf{B_t} = min(B_{t1}, B_{t2})$$

Thus, Ω_{c1} & Ω_{c2} becomes, $\Omega_{c1} = \Omega_{p1} + \frac{\mathbf{B_t}}{2}$, $\Omega_{c2} = \Omega_{p2} - \frac{\mathbf{B_t}}{2}$.

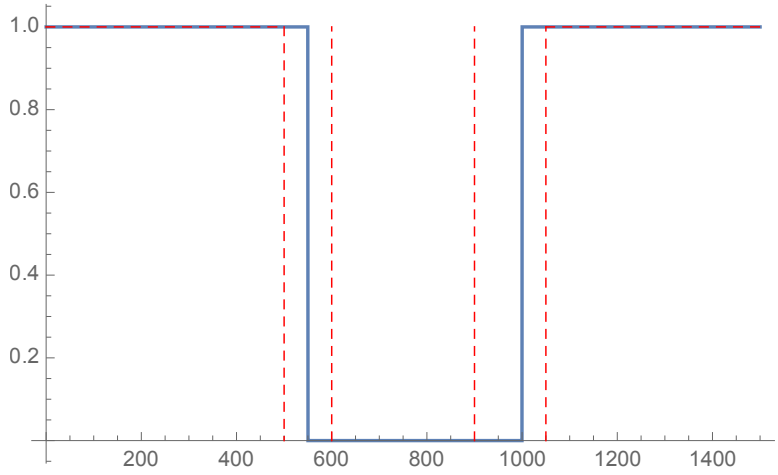


Figure 4: $H(e^{j\Omega T})$ - Red dotted lines represent $\Omega_{p1}, \Omega_{a1}, \Omega_{a2}, \Omega_{p2}$

4.1.2 Impulse Response of the Ideal Bandstop Filter

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

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

Using the inverse Fourier transform we get,

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

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

So the ideal impulse response $h(nT)$,

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

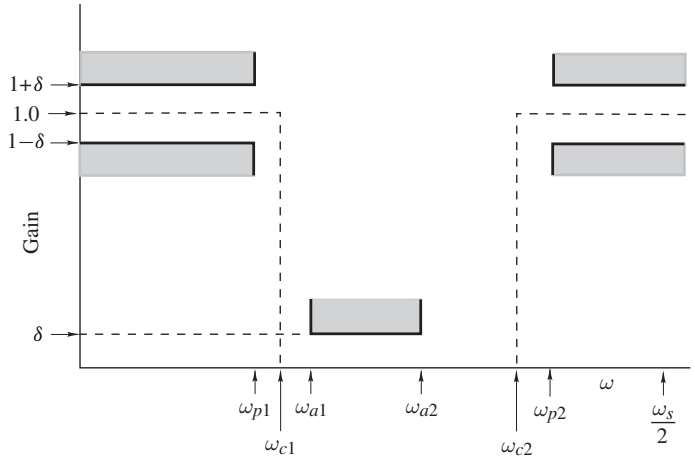


Figure 5: Intended Bandstop Filter Characteristics [2]

In summary we have the derived parameters,

Table 2: Derived Filter Specifications

Specification	Symbol	Derivation	Value	Units
Lower transition width	B_{t1}	$\Omega_{a1} - \Omega_{p1}$	100	$rads^{-1}$
Upper transition width	B_{t2}	$\Omega_{p2} - \Omega_{a2}$	150	$rads^{-1}$
Critical transition width	B_t	$min(B_{t1}, B_{t2})$	100	$rads^{-1}$
Lower cutoff frequency	Ω_{c1}	$\Omega_{p1} + \frac{B_t}{2}$	550	$rads^{-1}$
Upper cutoff frequency	Ω_{c2}	$\Omega_{p2} - \frac{B_t}{2}$	1000	$rads^{-1}$
Sampling period	T	$\frac{2\pi}{\Omega_s}$	0.002244	s

4.2 Kaiser Parameters

4.2.1 Deriving δ

The passband amplitude response oscillates between (1 - σ) and (1 + σ). The stopband amplitude response oscillates between 0 and σ.
Therefore we are constrained under, **Maximum passband ripple $\tilde{\mathbf{A}_p}$** and **Minimum stopband attenuation $\tilde{\mathbf{A}_a}$**
Since these constraints are given in decibels, we need to establish a relationship between σ, A_p & A_a

$$A_a = -20log|\delta| \;\; (stopband\ attenuation) \;\Rightarrow\; \tilde{\delta_a} = 10^{-0.05\tilde{A_a}}$$

$$A_p = 20log\frac{1+\delta}{1-\delta} \;\; (passband\ ripple) \;\Rightarrow\; \tilde{\delta_p} = \frac{10^{0.05\tilde{A_p}} - 1}{10^{0.05\tilde{A_p}} + 1}$$

Because it might not be possible to achieve the Maximum passband ripple $\tilde{\mathbf{A}_p}$ and Minimum stopband attenuation $\tilde{\mathbf{A}_a}$ in exact value with the same δ, we build a filter that might over-satisfy one

or both specifications. So we pick δ such that $A_p \leqslant \tilde{A}_p \quad and \quad A_a \geqslant \tilde{A}_a$ conditions both satisfy at the same time. Therefore,

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

Minimum stopband attenuation $\tilde{\mathbf{A}_a} = \mathbf{47dB}$ $\Rightarrow$ $\tilde{\delta_a} = 0.00446684$
Maximum passband ripple $\tilde{\mathbf{A}_p} = \mathbf{0.05dB}$ $\Rightarrow$ $\tilde{\delta_p} = 0.00287822$

$$\therefore \delta = \mathbf{0.00287822}$$

4.2.2 Calculating Actual Stopband Loss A_a & Passband Ripple A_p

Now that we've found δ we can find the actual stopband loss/attenuation,

$$A_a = -20log|\delta| = -20log|0.00287822| = 50.8175$$

$$A_p = 20log|\frac{1 + 0.00287822}{1 - 0.00287822}|0 = 0.05$$

4.2.3 Choosing Parameter α

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

Since $A_a > 50dB$, $\alpha = 0.1102(A_a - 8.7) = 0.1102(50.8175 - 8.7) = 4.64135$

4.2.4 Choosing Parameter D & Window size N

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

Since $A_a > 21dB$, $D = \frac{A_a - 7.95}{14.36} = \frac{50.8175 - 7.95}{14.36} = 2.9852$

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

$$N \geq \frac{\Omega_s D}{B_t} + 1 = \frac{2800rad/s * 2.9852}{100rad/s} + 1 = 84.5856$$

Thus, N = 85.

All these parameters were calculated manually, and the results were cross checked with the relevant values got from the MATLAB code.

Table 3: Kaiser Parameters Summary

Parameter	Value	Units
δ	0.00287822	-
A_a	50.8175	dB
A_p	0.05	dB
α	4.64135	-
$I_0(\alpha)$	19.8032	-
D	2.9852	-
N	85	-

5 Results

5.1 Forming the window sequence $w_k(nT)$

Since the window length (N) was computed to be 85, We have the kaiser window from $n \leq \left|\frac{N-1}{2}\right|$. n = -42 to n = +42.

$$w_K(nT) = \begin{cases} I_0(\beta)/I_0(\alpha) & for\ |n| \leq \frac{N}{2} \\ 0 & otherwise \end{cases}$$

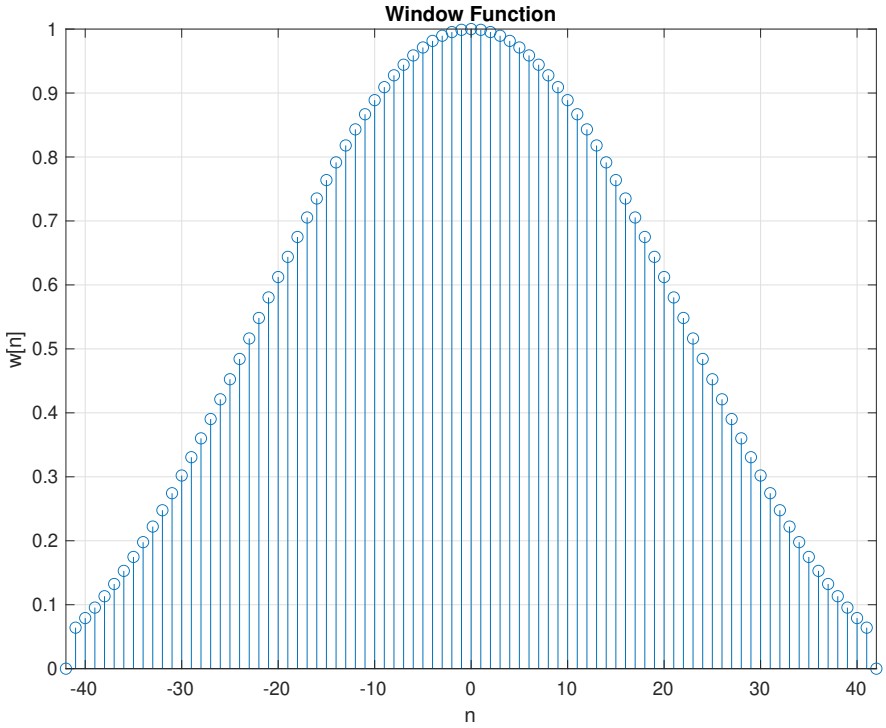


Figure 6: Constructed Kaiser Window

5.2 Causal Impulse Response of the Bandstop Filter

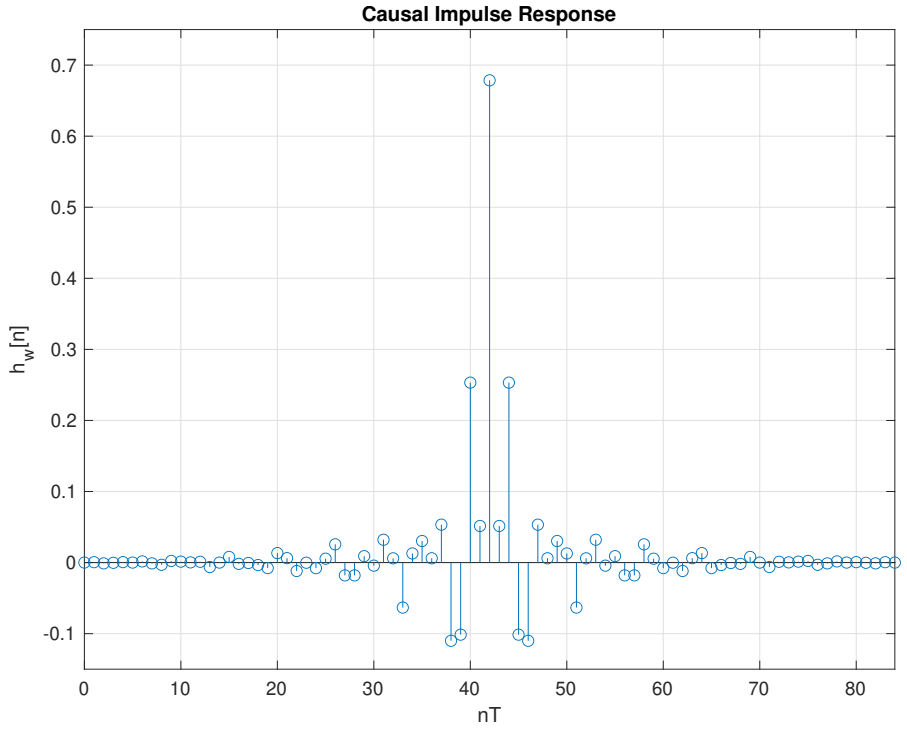


Figure 7: Causal Impulse Response

5.3 Magnitude Response of the Digital Filter

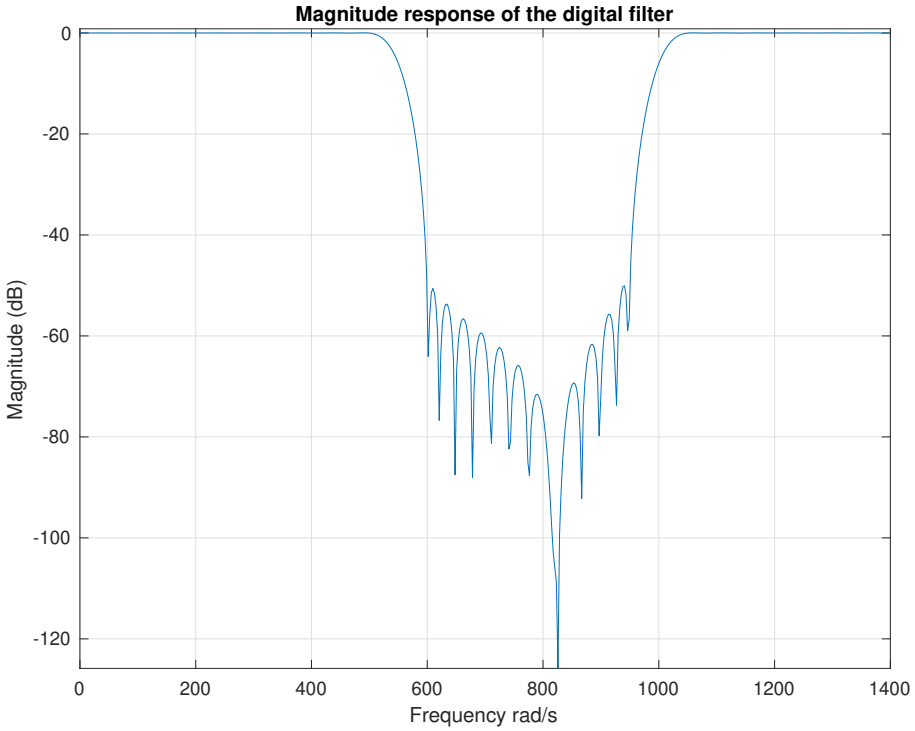


Figure 8: Magnitude Response of the BandStop Filter

5.4 Magnitude Responses of the Passbands

From figures 9 & 10, it is clear that the passband ripples A_p are well within the range $\tilde{A}_p$=0.05.

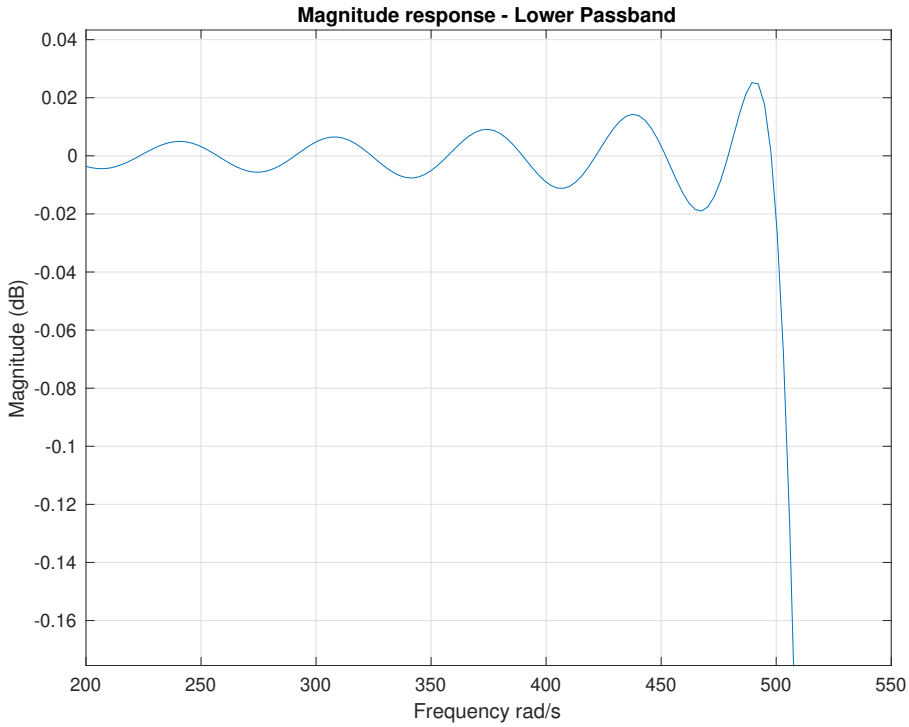


Figure 9: Upper passband (Zoomed)

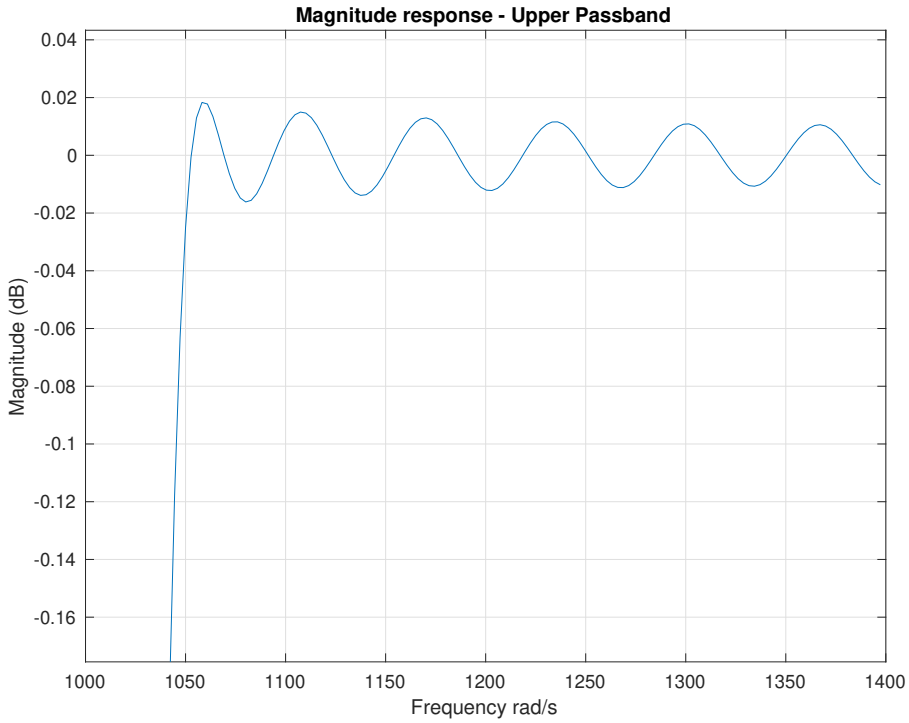


Figure 10: Lower passband (Zoomed)

5.5 Input Signal

5.5.1 Time Domain

Note that, only 100 samples from $\mathbf{x(nT) = \sum_{i=1}^{3} cos[\Omega_i nT]}$ are displayed in this plot.

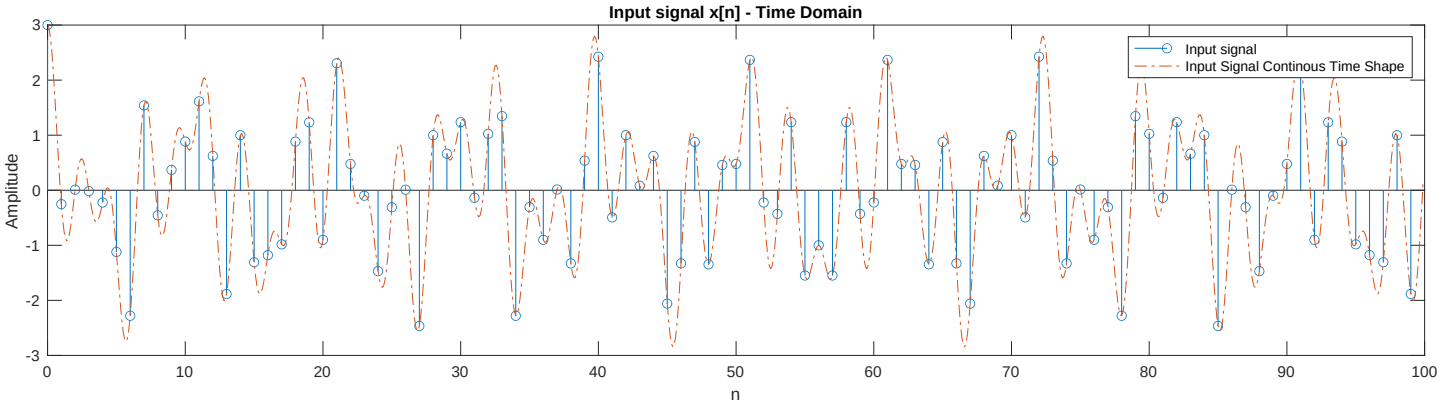


Figure 11: $x(nT) = \sum_{i=1}^{3} cos[\Omega_i nT]$

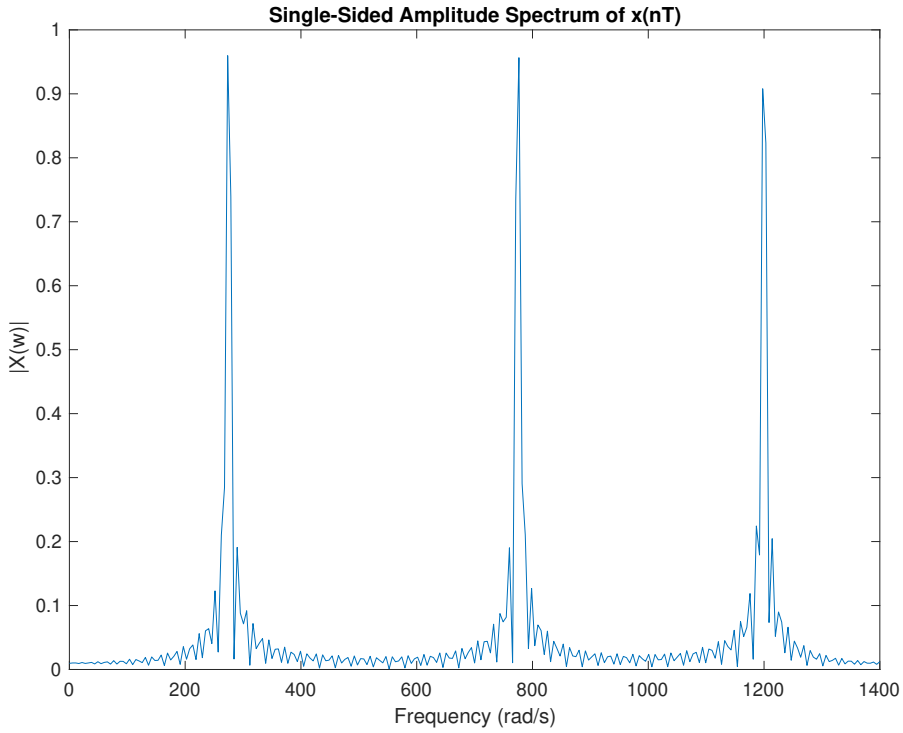
5.5.2 Frequency Domain

The raw input signal is a sum of three cosines, therefore, they clearly have peaks at,

⋆ $\Omega_1 = \frac{\Omega_{c1}}{2} = 275\ rad/s$

⋆ $\Omega_2 = \Omega_{c1} + \frac{\Omega_{c2} - \Omega_{c1}}{2} = 775\ rad/s$

⋆ $\Omega_3 = \Omega_{c2} + \frac{\Omega_s/2 - \Omega_{c2}}{2} = 1200\ rad/s$



5.6 Filtered Signal

$$Input\ Signal Was: \quad x(nT) = \sum_{i=1}^{3} cos[\Omega_i nT]$$

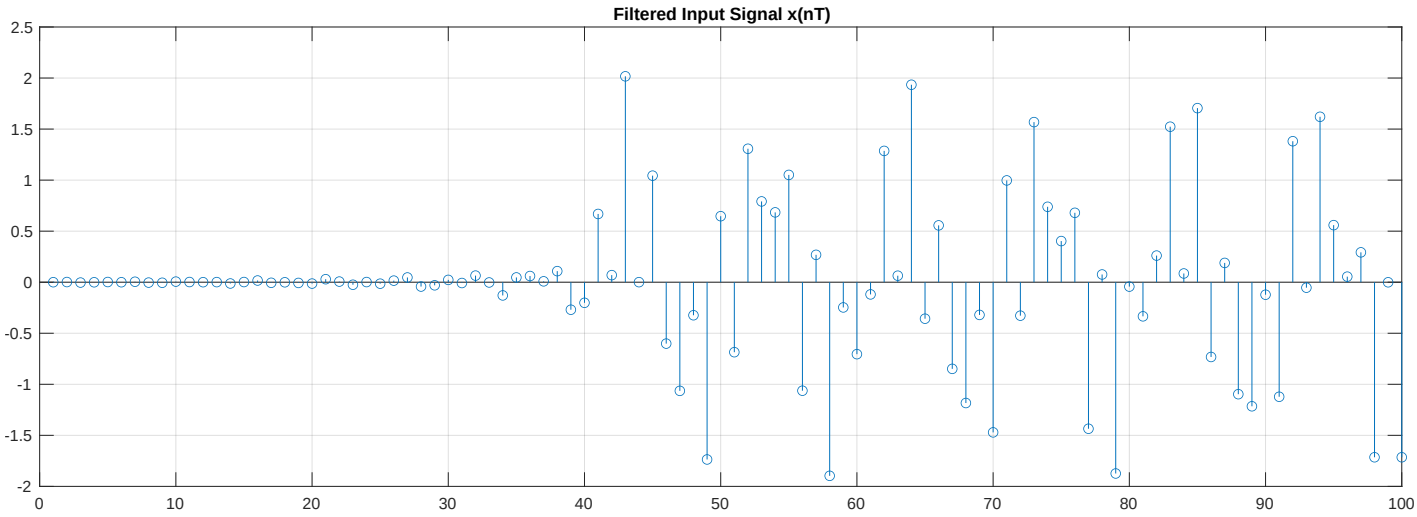


Figure 12: $x(nT) * h_w(nT)$

By the bandstop filter, our intention was to attenuate frequencies from 550 rad/s to 1000 rad/s, which has been done successfully.

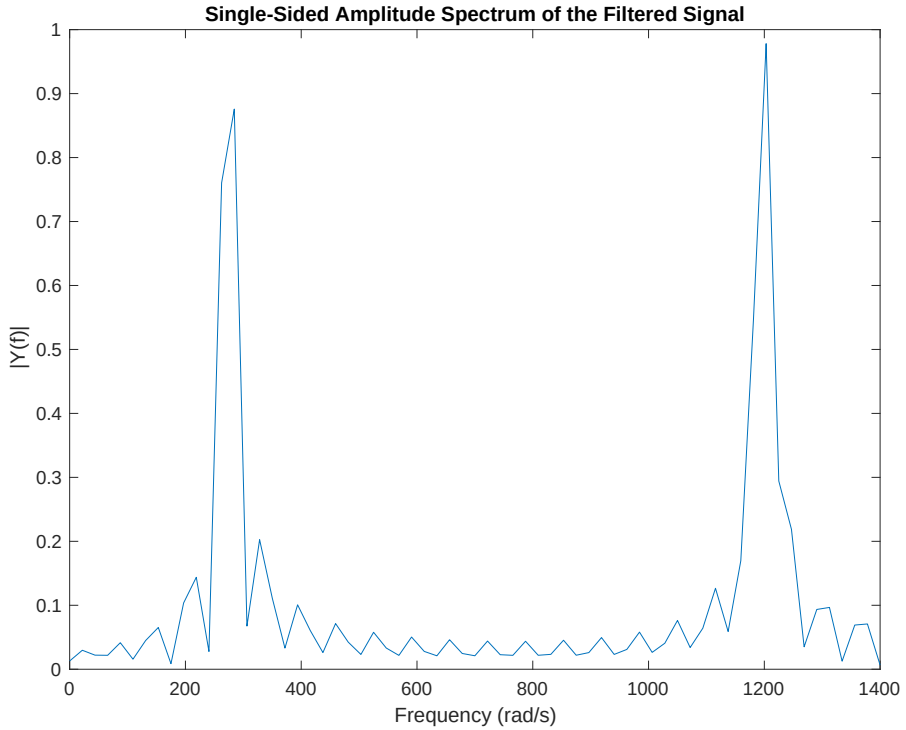


Figure 13: $\mathscr{F}\ (x(nT) * h_w(nT))$

5.7 Expected Outputs

Figure 14, shows the plot of the expected filtered signal which is, $cos[\Omega_1 nT] + cos[\Omega_3 nT]$

Figure 14: $\mathscr{F}\ (cos[\Omega_1 nT] + cos[\Omega_3 nT])$

6 Discussion

With the design steps that were mentioned, I was able to make a Bandstop FIR Filter using the Kaiser Method. It was evident that the designed filter conformed with the specifications in Table 1 - $\tilde{A}_p$ =0.05 $\tilde{A}_a$ = 47dB as per the magnitude spectrum in Figure 8

The filtered signal was obtained by multiplying the signal and the window in the frequency domain and then taking the inverse fourier transform of the result. And if we observe the filtered signal in Figure 12, it takes more than 40 samples for the signal to reach the steady state. The reason for this is because the filter we designed had the order 84. Therefore, higher order filters may take more time for the filtered signal to reach the steady state.
The designed filter was also compared with the builtin MATLAB functions, $kaiserord$ & $fir1$

7 Acknowledgment

I would like to extend my sincere gratitude towards Dr.Chamira Edussooriya for the guidance provided during this project.

References

[1] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-time signal processing.* Pearson, 2010.

[2] Andreas Antoniou. *Digital signal processing.* McGraw-Hill, 2006.

[3] Center for Computer Research in Music and Stanford University Acoustics (CCRMA). - kaiser window beta parameter.

8 Appendix

```
%%Function to Calculate the kaiser window
function f = my_kaiser(N,alpha)
    % N should be odd
    f = zeros(1,N);
    d = bessel_i(alpha);

    for k=0:(N-1)/2-1
        f((N-1)/2+1+k) = bessel_i(beta_(alpha,k,N))/d;
        f((N-1)/2+1-k) = bessel_i(beta_(alpha,k,N))/d;
    end
    f=f.';
end

%%Function to calculate the modified Bessel function of the first kind
function f = bessel_i(n)
    ans = 1;
    n = n/2;

    for k=1:100
        ans = ans + ((n^k)/factorial(k))^2;
    end
    f = ans;
end

%%Function to calculate beta value for a given alpha
function f = beta_(alpha,n,N)
    f = alpha*(1-((2*n)/(N-1))^2)^0.5;
end
```

Published with MATLAB® R2019b

1

Appendix

Table of Contents


```
%%Author : R.U. Hettiarachchi %%
%%Index  : 170221T            %%
```

Given Design Parameters :

```
A_p_tilde = 0.05; %Max Passband Ripple
A_a_tilde = 47;   %Min stopband attenuation
Omega_p1 = 500;
Omega_p2 = 1050;
Omega_a1 = 600;
Omega_a2 = 900;
Omega_s  = 2800;
T        = 2*pi/Omega_s;
```

Calculating Realistic Specs :

```
B_t1 = Omega_a1 - Omega_p1;
B_t2 = Omega_p2 - Omega_a2;

B_t = min(B_t1,B_t2)
Omega_c1 = Omega_p1 + B_t/2
Omega_c2 = Omega_p2 - B_t/2
```

Calculating Kaiser Parameters %%

```
delta_p_tilde = (10^(0.05*A_p_tilde)-1)/(10^(0.05*A_p_tilde)+1)
delta_a_tilde =  10^(-0.05*A_a_tilde)

delta = min(delta_p_tilde,delta_a_tilde)

A_a = -20*log10(delta)                  %Actual Stopband Loss
A_p = 20*log10((1+delta)/(1-delta))     %Actual Passband Ripple


% -> Find alpha
alpha = 0;
```

Appendix

```
if (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

% -> Find D
D = 0.9222;
if(A_a > 21) D = (A_a - 7.95)/14.36; end

% -> Find N

N = (Omega_s*D)/B_t+1;
N = ceil(N) + mod(ceil(N)-1,2) %smallest odd integer satisfying the
 inequality
```

Construct Kaiser Window

```
n = -(N-1)/2 : 1 : (N-1)/2;
w = my_kaiser(N,alpha);

figure;
stem(n,w);
xlabel('n');
ylabel('w[n]');
axis([-(N-1)/2 (N-1)/2 0 1 ])
title('Window Function');
grid on;
saveas(gcf,'window','epsc')
```

Construct Impulse Response of the Ideal Filter

```
h = (1./(n*pi)).*(sin(Omega_c1*n*T) - sin(Omega_c2*n*T));
h((N-1)/2+1) = 1 + (2/Omega_s)*(Omega_c1 - Omega_c2) ;
n = 0 : (N-1);              %shifting by (N-1)/2
```

Causal Impulse Response of the Filter

```
close all;
h_w = h.*w.';

figure;
stem(n,h_w);
xlabel('nT');
ylabel('h_w[n]');
axis([0 (N-1) -0.15 0.75 ])
title('Causal Impulse Response');
grid on;
saveas(gcf,'cir','epsc')
```

Magnitude response of the digital filter & Pass-bands zoomed in

```
freqz(h_w);
```

Appendix

```
[z,w] = freqz(h_w);
w = w./T;

figure;
plot(w,20*log10(abs(z)))
grid on;
axis([0 1400 20*log10(0) 20*log10(1.1) ])
xlabel('Frequency rad/s');
ylabel('Magnitude (dB)');
title('Magnitude response of the digital filter');
saveas(gcf,'magnitude response of filter','epsc')


figure;
plot(w,20*log10(abs(z)))
grid on;
axis([200 550 20*log10(0.98) 20*log10(1.005) ])
xlabel('Frequency rad/s');
ylabel('Magnitude (dB)');
title('Magnitude response - Lower Passband');
saveas(gcf,'upper passband','epsc')

figure;
plot(w,20*log10(abs(z)))
grid on;
axis([1000 1400 20*log10(0.98) 20*log10(1.005) ])
xlabel('Frequency rad/s');
ylabel('Magnitude (dB)');
title('Magnitude response - Upper Passband');
saveas(gcf,'lower passband','epsc')
```

Input Signal

Freq components,

```
Omega1 = Omega_c1/2;
Omega2 = Omega_c1 + (Omega_c2-Omega_c1)/2;
Omega3 = Omega_c2 + (Omega_s/2-Omega_c2)/2;

% Generate the Input signal

samples = 300; % Enough to achieve a steady-state response.
n1 = 0:1:samples;
n2 = 0:0.1:samples;

x     = cos(Omega1.*n1*T) + cos(Omega2.*n1*T) + cos(Omega3.*n1*T);
X_env = cos(Omega1.*n2*T) + cos(Omega2.*n2*T) + cos(Omega3.*n2*T);

R = cos(Omega1.*n1*T) + cos(Omega3.*n1*T);


stem(n1(1:100),x(1:100))
xlabel('n')
```

Appendix

```
ylabel('Amplitude')
title(strcat(['Input signal x[n]',' ','- Time Domain']));
hold on
plot(n2(1:1000),X_env(1:1000),'-.')
legend('Input signal','Input Signal Continous Time Shape');


figure;
L = length(x);
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(x,NFFT)/L;
f = Omega_s/(2)*linspace(0,1,NFFT/2+1);

% Plot single-sided amplitude spectrum.
plot(f,2*abs(Y(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of x(nT)')
xlabel('Frequency (rad/s)')
ylabel('|X(w)|')
saveas(gcf,'Xw','epsc')


figure
L=1400;
X = fft(x,L);
H_w = fft(h_w,L);
filtered = ifft(X.*H_w);
stem(filtered(1:100));
grid on;
title('Filtered Input Signal x(nT)')
saveas(gcf,'Xw','epsc')


figure
plot(filtered(42:42+85))
hold on
out = conv(h_w,x,'same'); %% 'same' is there to crop the filtered
 signal
plot(out(4:85))
title('Filtered Input Signal Using conv()')

figure;
L = length(out);
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(out,NFFT)/L;
f = Omega_s/(2)*linspace(0,1,NFFT/2+1);

plot(f,2*abs(Y(1:NFFT/2+1)))    % Plot single-sided amplitude
 spectrum.
title('Single-Sided Amplitude Spectrum of the Filtered Signal')
xlabel('Frequency (rad/s)')
ylabel('|Y(f)|')
```

Appendix

Expected Signal in the Freq domain

```
figure;
L = length(R);
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(R,NFFT)/L;
f = Omega_s/(2)*linspace(0,1,NFFT/2+1);

plot(f,2*abs(Y(1:NFFT/2+1)))    % Plot single-sided amplitude
 spectrum.
title('Single-Sided Amplitude Spectrum of the Expected Filtered
 Signal')
xlabel('Frequency (rad/s)')
ylabel('|Y(f)|')
saveas(gcf,'efs','epsc')
```

Published with MATLAB® R2019b

5