University of Moratuwa

Department of Electronic and Telecommunication Engineering

EN2570 - Digital Signal Processing

Project Report

Finite Impulse Response Filter Designing

R.T.N.Rathnayake.
180538F

Abstract

This report gives the detailed description about Finite Impulse Response (FIR) Bandpass Digital Filter designing for prescribed specifications using the windowing method in conjunction with the Kaiser window. Theories and methods are discussed initially and then the filter is implemented using the MATLAB R2018a software. Then an excitation signal which is consisted with several sinusoids is sent through the filter and observe the output signal to confirm the correctness of the filter.

Introduction

Final goal of this project is to design and implement an FIR bandpass digital filter under given specifications. A bandpass filter attenuates all the frequencies except a band of frequencies which we are wanted to obtain in the output. Digital filters are used almost in everywhere in telecommunication field. Although there are many methods to design digital filters with various accuracies, here we are only look at the Fourier Series method with Kaiser window which is a direct method of filter designing. In this process, the infinite impulse response of the ideal filter is truncated with the Kaiser window in time domain. The filter is implemented using MATLAB R2018a and plots are used for the analyzing purposes. Filter Visualization Tool ('fvtool') in MATLAB is very useful in plotting results. Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) algorithms have been used to analyze the filter output for the input excitation signal.

Basic Theory and Method

- Index number = 180538F
- A = 5
- B = 3
- C = 8
- • = F

Parameter	Value
Maximum passband ripple, $\tilde{A}_p$	0.08 dB
Minimum stopband attenuation, $\tilde{A}_a$	48dB
Lower passband edge, Ω_{p1}	1100 rad/s
Upper passband edge, Ω_{p2}	1500 rad/s
Lower stopband edge, Ω_{a1}	950 rad/s
Upper stopband edge, Ω_{a2}	1600 rad/s
Sampling frequency, Ω_s	4000 rad/s

Some parameters should be calculated directly using above specification. For the above bandpass specifications, the design must be based on the narrower of the two transition bands.

- Sampling Period $T = 2\pi/\Omega_s = 0.0016$ s
- Critical Transition Width $B_t = \min\{(\Omega_{p2} - \Omega_{a1}), (\Omega_{a2} - \Omega_{p2})\} = 100$ rad/s
- Lower Cutoff Frequency $\Omega_{c1} = \Omega_{p1} - B_t/2 = 1050$ rad/s
- Upper Cutoff Frequency $\Omega_{c2} = \Omega_{p2} + B_t/2 = 1550$ rad/s

If we consider the normalized frequency, the values respective to each frequency parameter are shown as below.

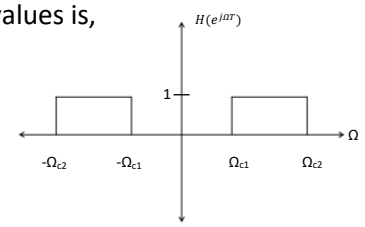
$$\omega = \Omega T$$

Parameter	Value (rad/sample)
Lower passband edge, ω_{p1}	0.55π
Upper passband edge, ω_{p2}	0.75π
Lower stopband edge, ω_{a1}	0.475π
Upper stopband edge, ω_{a2}	0.8π
Sampling frequency, ω_s	2π
Lower cutoff frequency, ω_{c1}	0.525π
Upper cutoff frequency, ω_{c2}	0.775π

Since Ω is closer to the reality than ω, we are going to use Ω for the following filter designing procedure. By using ω also we can get the same filter.

- Frequency response of the ideal bandpass filter for above cutoff values is,

$$H(e^{j\Omega T}) = \begin{cases} 1 & ; for \ -\Omega_{c2} \le \Omega \le -\Omega_{c1} \\ 1 & ; for \quad \Omega_{c1} \le \Omega \le \Omega_{c2} \\ 0 & ; otherwise \end{cases}$$



- We can obtain the impulse response of the ideal bandpass filter by taking inverse Fourier transform.

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

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

- Choosing δ such that $A_p \leq \tilde{A}_p$ and $A_a \geq \tilde{A}_a$ where Ap is the actual passband ripple and Aa is the actual stopband attenuation.

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

where, $\delta_p = \frac{10^{0.05\tilde{A}_p} - 1}{10^{0.05\tilde{A}_p} + 1}$ and $\delta_a = 10^{-10^{0.05\tilde{A}_a}}$

- With the required δ defined, the actual stopband attenuation A_a can be calculated.

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

- Choosing parameter α as,

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

- Choosing parameter D as,

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

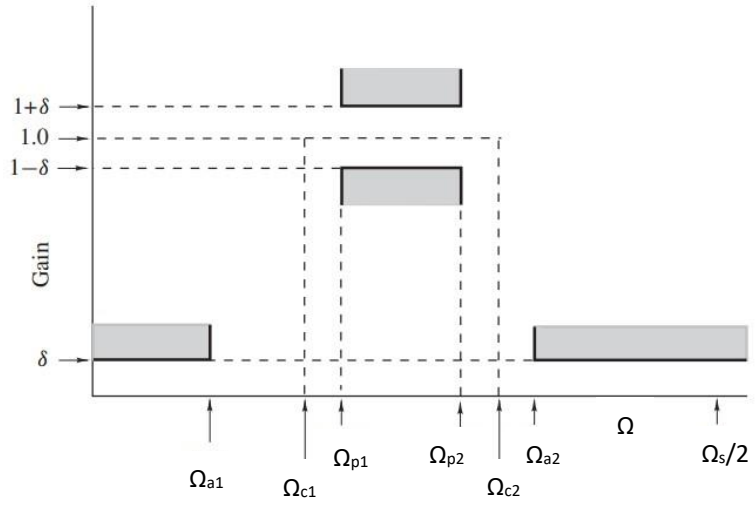
- Then choosing the lowest odd value for N that would satisfy the inequality.

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

- Below table has calculated parameters of Kaiser window

Parameter	Value
δ_p	1.0184
δ_a	0.0040
δ	0.0040
A_a	48 dB
A_p	0.0692 dB
α	4.3125
D	2.7890
N	113

- Graphical view of the specifications is shown in below.



- Now we can obtain the Kaiser Window function. ($w_k(nT)$)

$$w_K(nT)\begin{cases}\dfrac{I_o(\beta)}{I_o(\alpha)} & ;for\ |n| \le \dfrac{N-1}{2} \\ 0 & ;otherwise\end{cases}$$

- Where α is an independent parameter and,

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

- The impulse response of digital bandpass filter can be obtained using ideal filter and Kaiser window as follows. ($h_w(nT)$)

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

- This is the final bandpass filter in time domain.
- For frequency analysis, z transform should be taken as follows.

$$H_w(z) = \mathcal{Z}[h_w(nT)]$$

- For causality,

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

Testing

- Below excitation signal is sent through the designed filter to check the operation of the filter.

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

Ω_1	$\Omega_{a1}/2$	475 rad/s
Ω_2	$\Omega_{p1} + (\Omega_{p2} - \Omega_{p1})/2$	1300 rad/s
Ω_3	$\Omega_s/2 - (\Omega_s/2 - \Omega_{a2})/2$	1800 rad/s

The signal is consisted with three sinusoids as two frequencies are out of the passband and one frequency is within the passband. Therefore, expected output should have only one frequency that is within the passband.

- Expected (ideal) output = $\boldsymbol{\sin(\Omega_2 nT)}$

For simplicity of the computation, we avoid the convolution in time domain and instead of convolution, we do some transformation manipulations to get the output. First, we get the frequency domain representation of the filter and input signal by Discrete Fourier Transform (DFT) and multiply them and then convert it to the time domain again by taking the inverse. FFT and IFFT algorithms in MATLAB are used to those formations.

Results

Characteristics and performances of the filter have been illustrated using below plots. First there are plots about the filter function in time domain and frequency domain and then there are plots describing the functional situation for that given excitation input signal. Consider that some plots are obtained using the 'fvtool' function in MATLAB and those are in normalized frequency domain (ω).

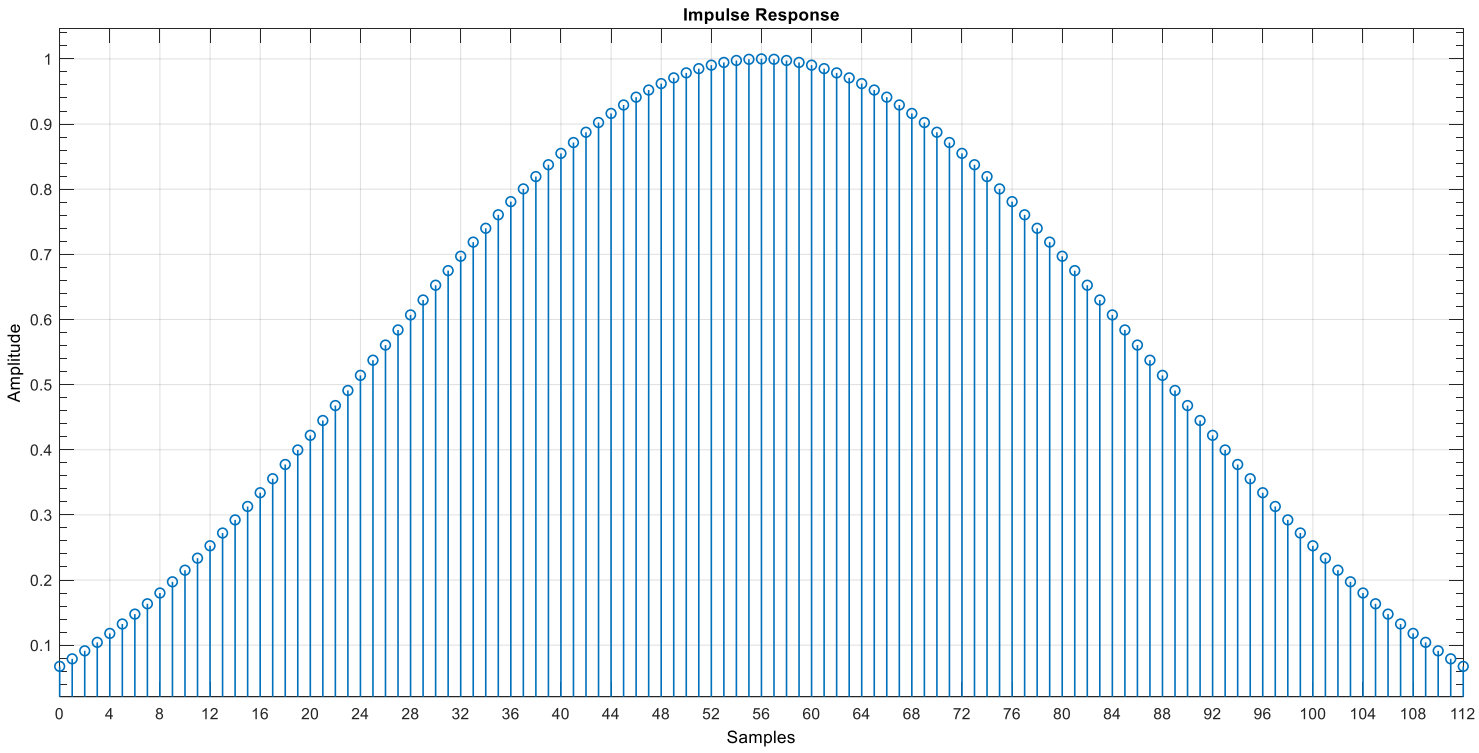


Figure 1 – impulse response of the Kaiser window

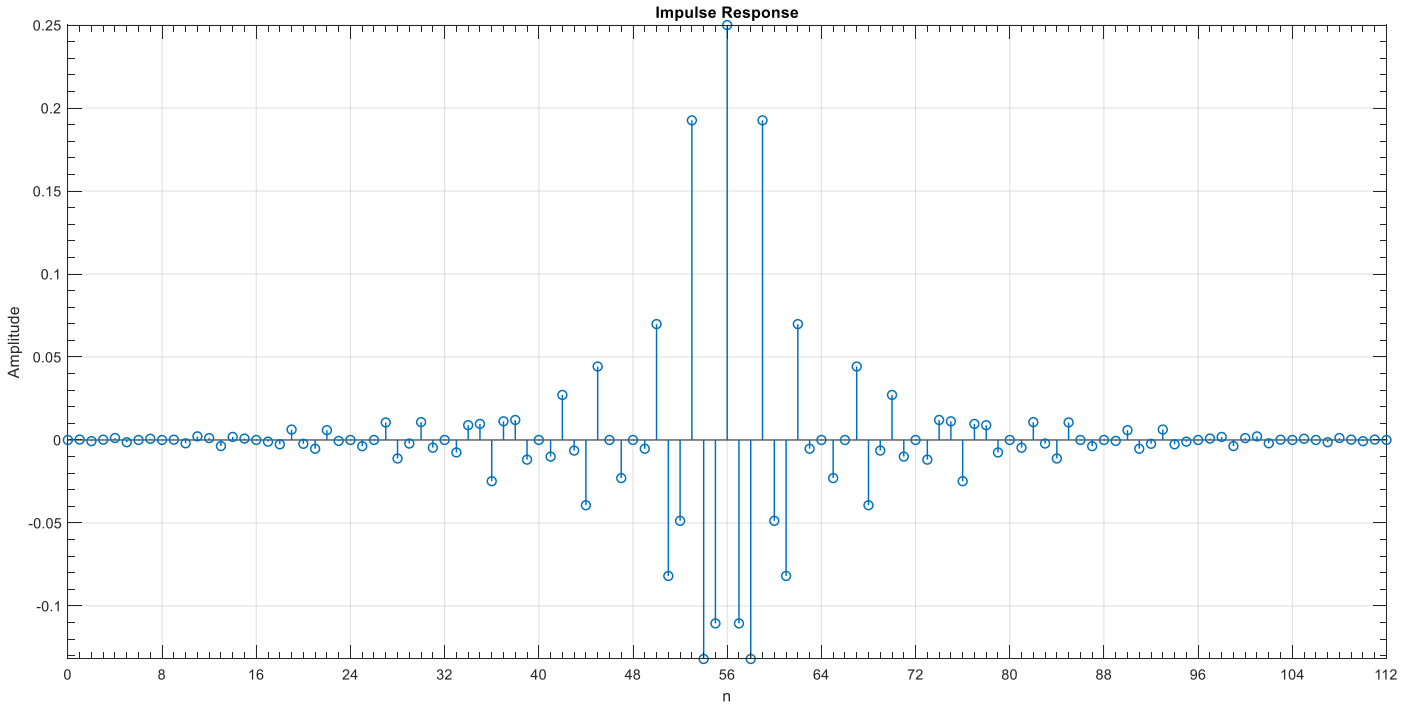


Figure 2 – impulse response of the filter

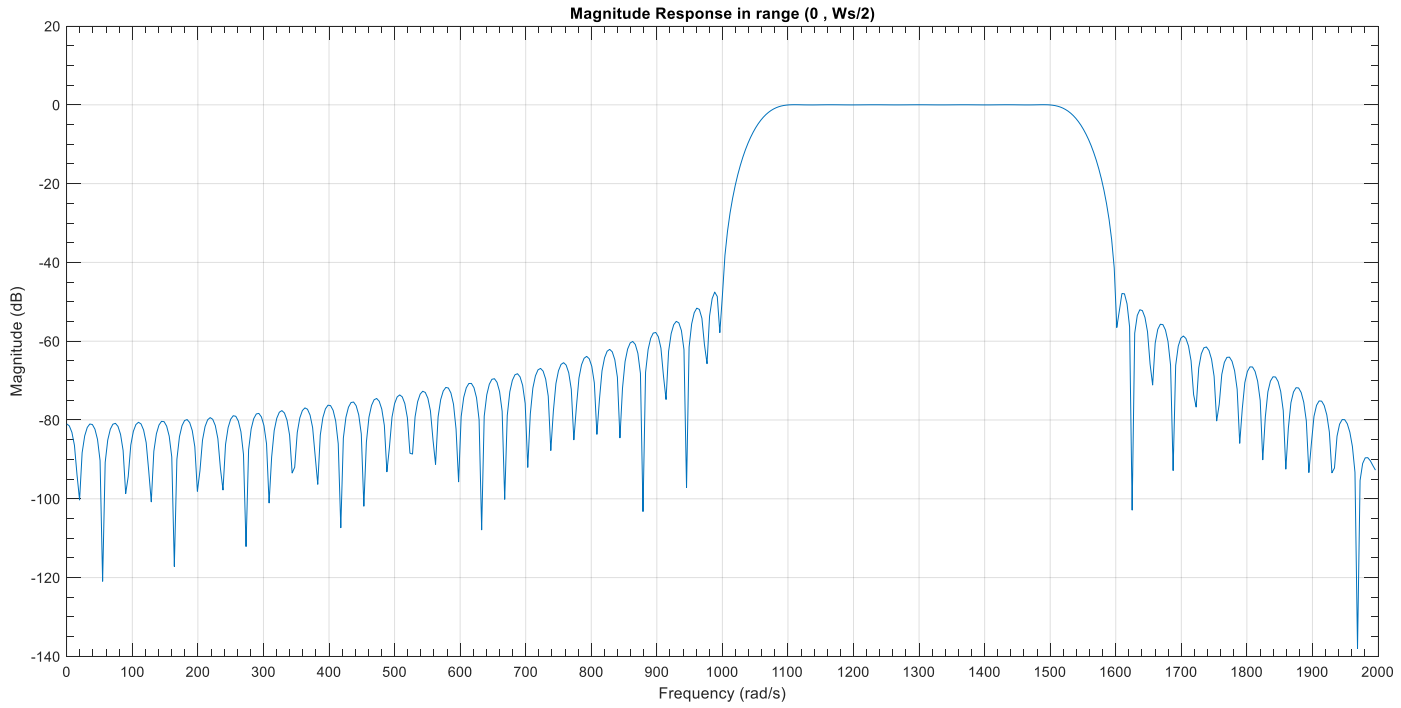


Figure 3 – magnitude response in frequency domain of the filter

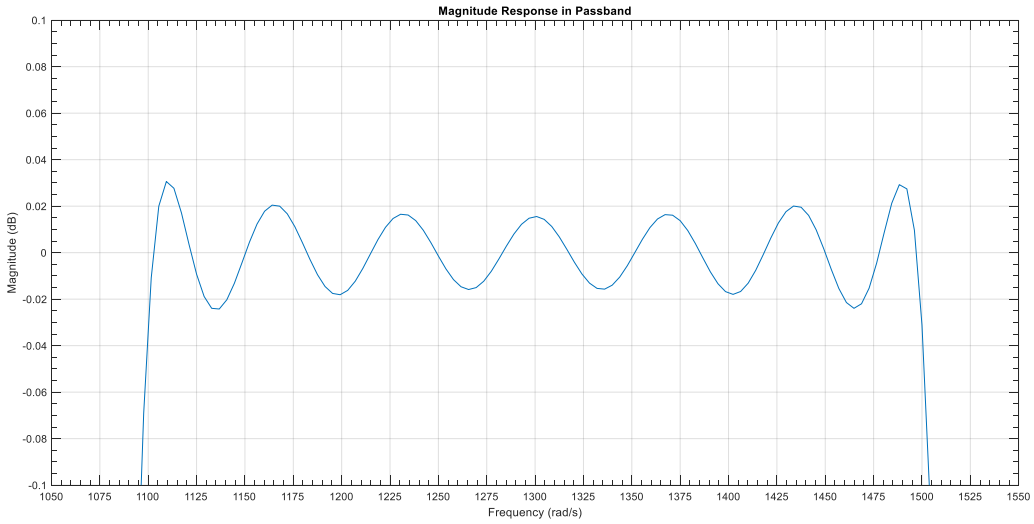


Figure 4 – magnitude response in frequency domain of the filter in the passband

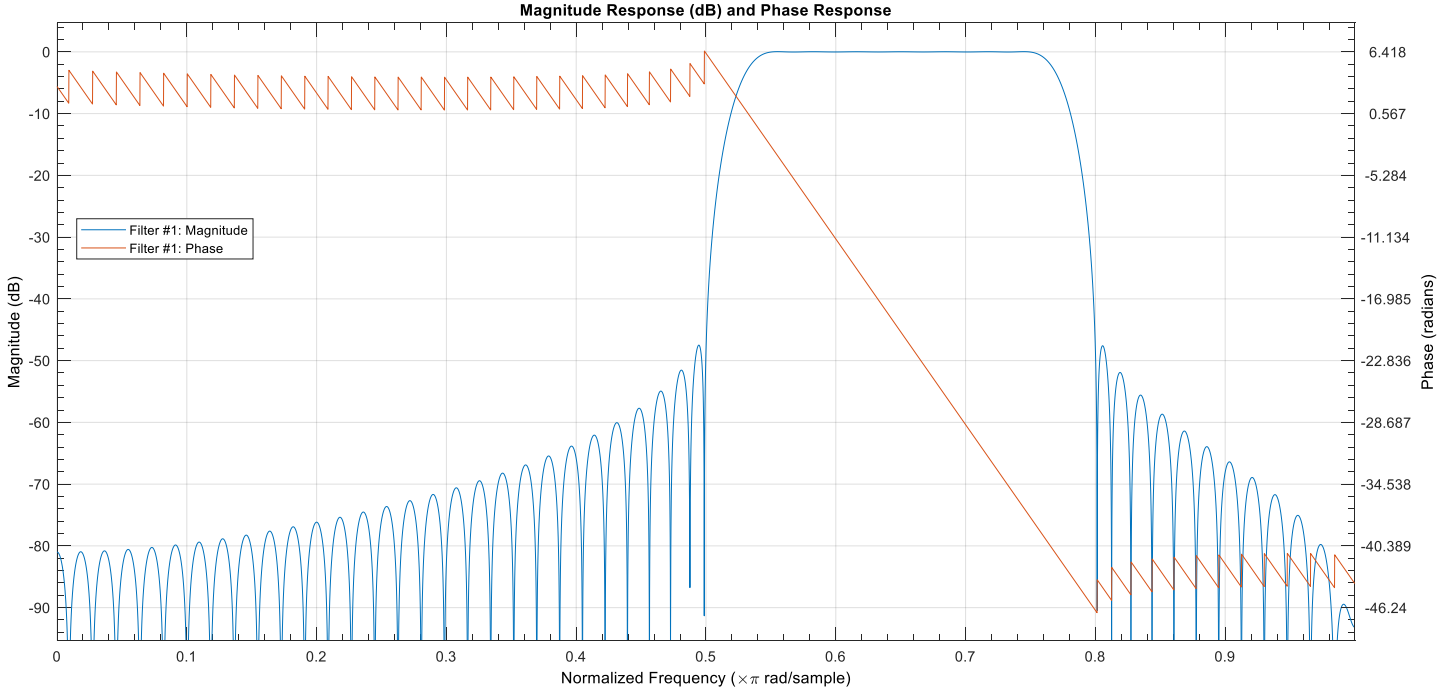


Figure 5 – magnitude and phase responses in normalized frequency domain of the filter

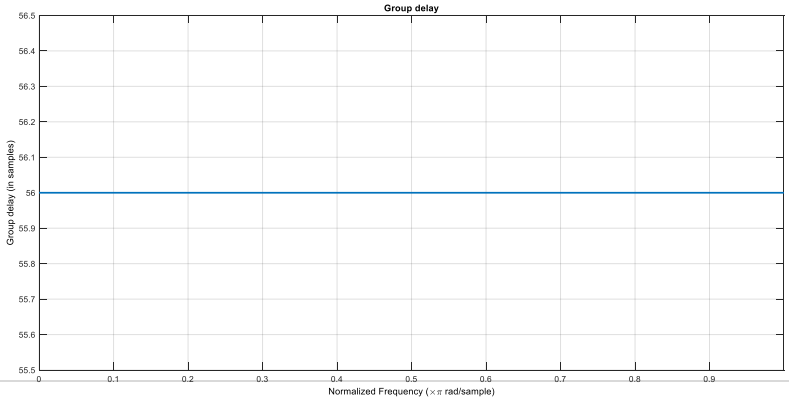


Figure 6 – group delay of the filter ('fvtool')

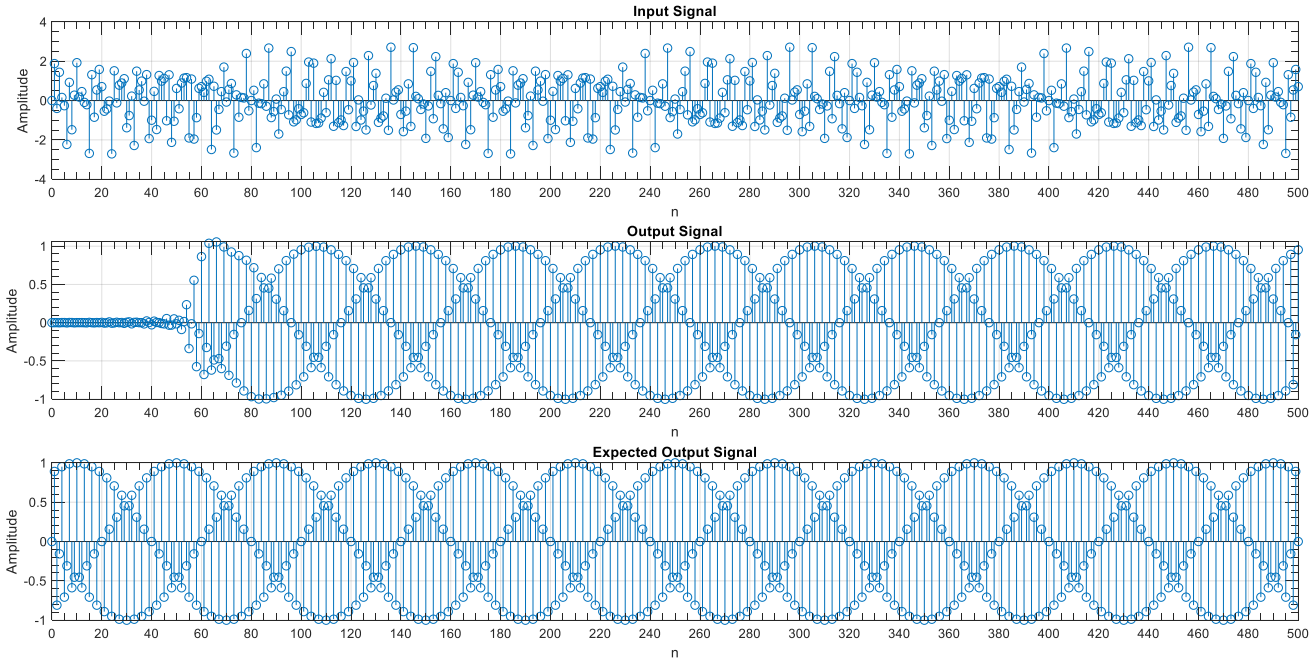


Figure 7

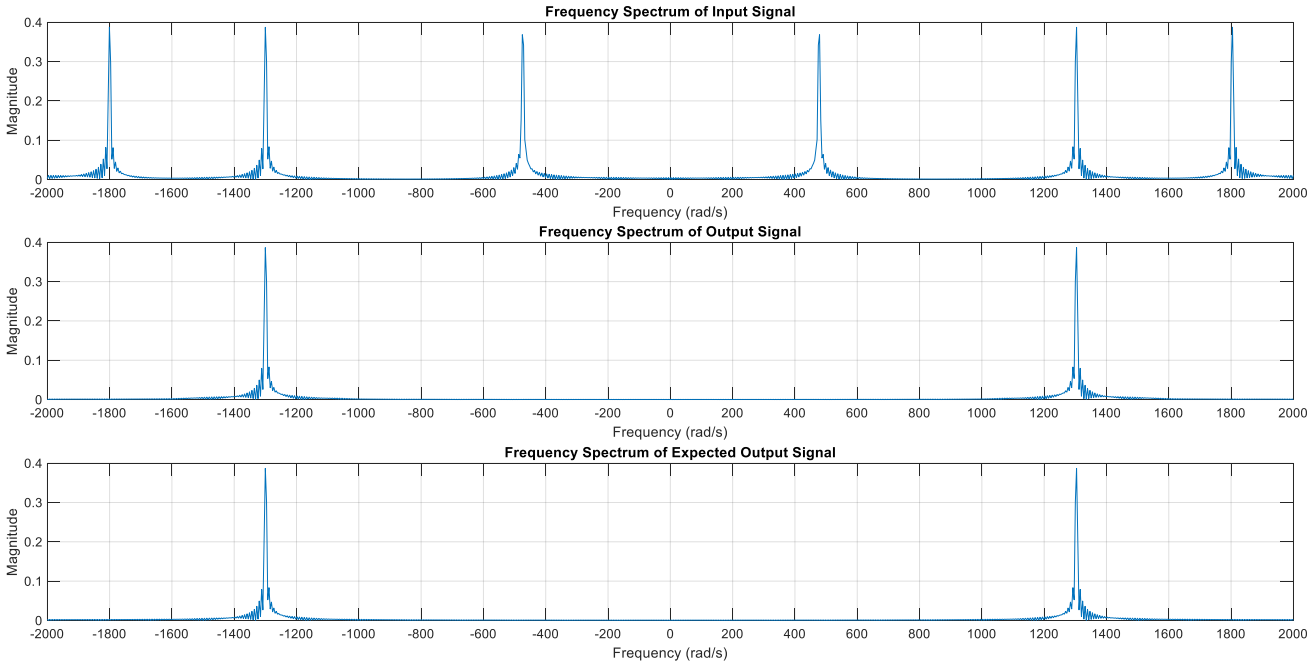


Figure 8

Discussion

According to information depicted by above figures, specifications for the digital filter has been achieved very well. Figure 1 shows the impulse response of the Kaiser window function and the length of that is 113 samples. It was used to truncate the ideal filter in time domain. As you can see, the window is not a rectangular but somewhat curvy shape. The causal impulse response of the designed bandpass filter is depicted in figure 2. We have to convolute the input signal with this impulse response to get the output signal in time domain. But there are more easy methods to get the output without doing convolution. In figure 3 has much more clear information to compare the filter specifications with filter performances. We can clearly see that frequencies (1100 – 1500 rad/s) in passband has almost unity gain (0dB) and other frequencies have been attenuated and those attenuations are less than 48dB. The transition width is about 100 rad/s as our specifications. Figure 4 has the closer view what happens to magnitudes in passband frequencies. The ripple is only about 0.07dB and it is well under our limit value 0.08dB in specifications. Next there is magnitude response and the phase response in normalized frequency domain in figure 5. The main point is the linear phase in passband. It is very important to maintain constant gain of magnitude and linear phase in passband to the the out without any major distortions. Most of cases, we care about constant group delay and here we have that property as depicted in figure 6. So far, we have designed a good filter which satisfy the given specification.

Now it is time to check the filter by giving a known input signal and compare the output with ideal output. Figure 7 shows the time domain observations for input, output and expected output signals. As we can see, output and expected output is equivalent except that delay in the filter output. But in time domain, this is not a problem and it is obvious to have some delay between input and output. We can receive all the data that sent without any loss with a delay. Then there are frequency domain plots in figure 8. It is clear that, the frequency which was in the passband has been sent through the filter and other frequencies have been suppressed. Therefore, the filter output and expected output is almost similar which means the designed bandpass digital filter is working correctly according to the specifications.

Conclusion

In this project, a practical FIR bandpass digital filter was designed for a given set of specifications using Kaiser window method since the ideal filter cannot be implemented in practically. Results show that the designed filter was able to give the output as almost as the ideal output. The computational work in this method is not very complex and this is a very efficient way to design a filter. Moreover, we can design the filter for different types of specifications by choosing appropriate values for Kaiser window parameters. That gives us the flexibility of the design process. Even though this is not the ideal filter, (since there is ripples in passband and stopbands and has a transition width) in a practical scenario, it gives almost the best results we can get. Major drawback in this design technique is the higher order of the filter. Since the order of the filter is high, lots of hardware resources (delays, adders, multipliers) are needed to implement the filter in reality. Because of this reason, Kaiser window method is not the optimal way to design a digital filter. There are more optimal methods which give the same results with lower order. But computational complexity of those are more than the Kaiser window.

Acknowledgement

I would like to express my gratitude to Dr.C.U.S.Edussooriya, the project supervisor, for giving me the necessary knowledge, resources and the guidance to understand and complete this project successfully. I thank my colleagues for helping me by sharing the knowledge and experiences on this project.

References

- ✓ Andreas Antoniou," Digital Signal Processing - Signals Systems and Filters" (1st ed.). (2005). McGraw-Hill.
- ✓ Alan V. Oppenheim - Ronald W. Schafer," Discrete-Time Signal Processing" (3rd ed.). (2010). Pearson.
- ✓ https://www.mathworks.com/support.html?s_tid=gn_supp

Appendix I

MATLAB code

Contents


```
clc;
clear all;
close all;
format long;
```

Index number = 180538F

```
A = 5;
B = 3;
C = 8;

Ap_ = 0.03+(0.01*A);          % maximum passband ripple = 0.08 dB
Aa_ = 45+B;                   % minimum stopband attenuation = 48 dB
omiga_p1 = (C*100)+300;       % lower passband edge = 1100 rad/s (0.55pi)
omiga_p2 = (C*100)+700;       % upper passband edge = 1500 rad/s (0.75pi)
omiga_a1 = (C*100)+150;       % lower stopband edge = 950 rad/s (0.475pi)
omiga_a2 = (C*100)+800;       % upper stopband edge = 1600 rad/s (0.8pi)
omiga_s = 2*((C*100)+1200); % sampling frequency = 4000 rad/s
T = 2*pi/omiga_s;             % sampling period = 0.00157 s
```

```
Bt = min((omiga_p1-omiga_a1),(omiga_a2-omiga_p2)); % critical transition width = 100
rad/s
omiga_c1 = omiga_p1-Bt/2;                          % lower cutoff frequency = 1050
rad/s
omiga_c2 = omiga_p2+Bt/2;                          % upper cutoff frequency = 1550
rad/s
```

Choosing delta

```
delta_p = (10^(Ap_/20)-1/10^(Ap_/20)+1);
delta_a = 10^(-Aa_/20);
delta = min(delta_p,delta_a);

Ap = 20*log10((1+delta)/(1-delta));     % actual passband ripple
Aa = -20*log10(delta);                  % actual stopband attenuation
```

Choosing 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
```

Choosing D

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

Choosing order of the filter

```
N = ceil((omiga_s*D/Bt)+1);
if (mod(N,2)==0)    % setting N to an odd value
    N=N+1;
end
```

Generating Kaiser Window function

```
n = -(N-1)/2:1:(N-1)/2;                      % defining length of the filter
beta = alpha*((1-(2*(n)/(N-1)).^2).^0.5);    % generating beta
Io_alpha = bessel(alpha);                    % generating Io_alpha
Io_beta = zeros(1,N);                        % defining empty array for Io_beta

for (x = 1:N)
    Io_beta(x) = bessel(beta(x));
end

wk_nT = Io_beta/Io_alpha;   % generating Kaiser Window function
fvtool(wk_nT);  % visualization of Kaiser Window
```

Generating the impulse response of the ideal bandpass filter

```
n_lower = -(N-1)/2:-1;
n_upper = 1:(N-1)/2;
h_nT_lower = (1./(n_lower*pi)).*(sin(omiga_c2*n_lower*T)-sin(omiga_c1*n_lower*T));
h_nT_upper = (1./(n_upper*pi)).*(sin(omiga_c2*n_upper*T)-sin(omiga_c1*n_upper*T));
h_0 = (2/omiga_s)*(omiga_c2-omiga_c1);
h_nT = [h_nT_lower,h_0,h_nT_upper];     % ideal bandpass filter

hw_nT = wk_nT.*h_nT;    % applying the window and generating the filter function
fvtool(hw_nT);  % visualization of the filter
```

Generating the frequency response of the filter

```
[Hw_omiga, w] = freqz(hw_nT);
omiga = (w*omiga_s)/(2*pi);
log_Hw_omiga = 20*log10(abs(Hw_omiga));
```

Plotting

```
figure,stem(n+floor(N/2),hw_nT);                                    % causal impulse
response of the filter
title('Impulse Response');
xlabel('n'); ylabel('Amplitude'); grid on; axis tight;
figure,plot(omiga,log_Hw_omiga);                          % magnitude response of the
filter
title('Magnitude Response in range (0 , Ws/2)');
xlabel('Frequency (rad/s)'); ylabel('Magnitude (dB)'); grid on;
figure,plot(omiga,log_Hw_omiga);                          % magnitude response of the
filter in the passband
```

```
title('Magnitude Response in Passband');
xlabel('Frequency (rad/s)'); ylabel('Magnitude (dB)');
xlim([omiga_c1,omiga_c2]); ylim([-0.1,0.1]); grid on;
```

Generating the input signal

```
omiga_1 = omiga_a1/2;
omiga_2 = omiga_p1 + ((omiga_p2-omiga_p1)/2);
omiga_3 = omiga_a2 + (((omiga_s/2)-omiga_a2)/2);
samples = 500;
n = 0:1:samples;
x_nT = sin(omiga_1*n.*T)+sin(omiga_2*n.*T)+sin(omiga_3*n.*T);    % input signal
```

Generating the output signal by applying the filter

```
N_point = length(x_nT)+length(hw_nT)-1;
x_fft = fft(x_nT,N_point);
hw_fft = fft(hw_nT,N_point);
y_fft = hw_fft.*x_fft;
y_nT = ifft(y_fft,N_point);
y_nT_delayed = y_nT(floor(N/2)+1:length(y_nT)-floor(N/2));  % delayed output signal
y_nT = y_nT(1:samples+1);       % output signal for first number of samples
y_nT_expected = sin(omiga_2*n.*T);     % expected output signal (if an ideal filter is
used)
```

Plotting

```
figure,subplot(3,1,1);stem(n,x_nT);     % input signal
title('Input Signal');
xlabel('n'); ylabel('Amplitude'); grid on;
subplot(3,1,2);stem(n,y_nT);            % output signal
title('Output Signal');
xlabel('n'); ylabel('Amplitude'); grid on;
subplot(3,1,3);stem(n,y_nT_expected);   % expected output signal
title('Expected Output Signal');
xlabel('n'); ylabel('Amplitude'); grid on;
```

Generating the frequency response of the input signal

```
N_point = 2^nextpow2(2*length(x_nT));
x_fft = fft(x_nT,N_point);
x_fft = fftshift(x_fft);
X_w = T*abs(x_fft);
w_x = omiga_s*linspace(0,1,N_point)-omiga_s/2;
```

Generating the frequency response of the output signal

```
N_point = 2^nextpow2(2*length(y_nT_delayed));
y_fft = fft(y_nT_delayed,N_point);
y_fft = fftshift(y_fft);
Y_w = T*abs(y_fft);
w_y = omiga_s*linspace(0,1,N_point)-omiga_s/2;
```

Generating the frequency response of the expected output signal

```
N_point = 2^nextpow2(2*length(y_nT_expected));
y_ex_fft = fft(y_nT_expected,N_point);
y_ex_fft = fftshift(y_ex_fft);
Y_ex_w = T*abs(y_ex_fft);
w_y_ex = omiga_s*linspace(0,1,N_point)-omiga_s/2;
```

Plotting

```
figure,subplot(3,1,1);plot(w_x,X_w);                    % magnitude of the frequency
response of input signal
title('Frequency Spectrum of Input Signal');
xlabel('Frequency (rad/s)'); ylabel('Magnitude'); grid on;
subplot(3,1,2);plot(w_y,Y_w);                           % magnitude of the frequency
response of output signal
title('Frequency Spectrum of Output Signal');
xlabel('Frequency (rad/s)'); ylabel('Magnitude'); grid on;
subplot(3,1,3);plot(w_y_ex,Y_ex_w);                     % magnitude of the frequency
response of expected output signal
title('Frequency Spectrum of Expected Output Signal');
xlabel('Frequency (rad/s)'); ylabel('Magnitude'); grid on;
```

Bessel function

```
function Io_x = bessel(x)
    Io_x = 1;
    k = 1;
    temp = 1;
    while (temp>10^-6)                          % ignore the components that are less
than 10^-6
        temp = ((1/factorial(k))*((x/2)^k))^2;
        Io_x = Io_x + temp;
        k = k+1;
    end
end
```

Published with MATLAB® R2018a