Design of Bandpass Finite Duration Impulse Response Filter using Kaiser window method

Sauranga H.W.C (Undergraduate project, EN2570, University of Moratuwa)

Abstract

The performance of a digital control system is heavily dependant on the interference mixed with the desired input signal. Therefore, processing of the input signal before feeding into the system is mandatory to ensure a reliable output. Finite duration Impulse Response (FIR) filters play an important role in the processing of a digital signal. In this project, a Bandpass FIR filter is designed using the Kaiser window function to behave according to a pre specified set of requirements. The filter is designed and studied using Matlab software package (without using the inbuilt filter design functions) and its performance is analyzed using a test signal.

1.Introduction

The digital filter can be implemented both using hardware and software and its design also varies with the application. Digital filters are Linear Time Invariant (LTI) systems characterized by unit impulse response and have discrete time sequence as both input and output. Out of the 02 types, FIR and IIR, FIR filters have a greater flexibility in shaping their magnitude response according to the application. Digital filters can also be classified as follows using their frequency characteristics.

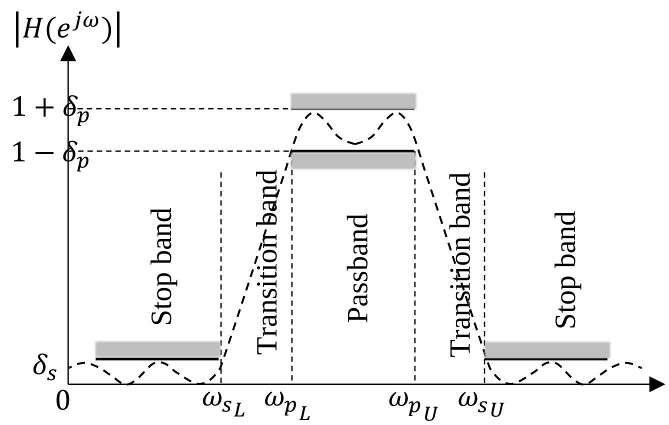
1. Lowpass filter
2. Highpass filter
3. Bandpass filter
4. Bandstop filter

In this paper, we will be focusing only on the bandpass filter and its design realization using windowing technique; Kaiser window to be precise. Bandpass filters are used to reject all except the required range of frequencies from a signal thus making it a significant tool in noise reduction, frequency boosting etc.

2.Filter Specifications

The first step in the design of an FIR bandpass filter is to iron out the requirements based on the needs from the filter as well as the practical feasibility of the design.

Eventhough the requirement would state a passband of, say f_1 to f_2 it is not practically feasible to get a sharp cut off at those frequencies. Hence, the design should specify a transition band from the passband edge to the stopband edge. These requirements can be given by a tolerance scheme.



ω_{s_L} – Lower stopband frequency

ω_{p_L} – Lower passband frequency

ω_{p_U} – Upper passband frequency

ω_{s_U} – Upper stopband frequency

δ_p – Maximum Passband ripple

δ_s – Maximum Stopband gain

3.Windowing Method

The ideal desired frequency response ($H_d(e^{j\omega})$) can be represented as

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d[n]e^{-j\omega n}$$

where $h_d[n]$ is the impulse response sequence of the desired filter.

These responses are non-causal and infinitely long. Hence, we truncate the impulse response by the process known as windowing.

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

This can be explained simply as follows.

If,

$$w[n] = \begin{cases} 1, & 0 \leq n \leq M, \\ 0, & otherwise \end{cases}$$

Then,

$$h[n] = h_d[n]w[n]$$

The window functions can be of two types.

Fixed window functions
- Rectangular
- von Hann
- Hamming
- Blackman

Adjustable window functions
- Dolph-Chebyshev
- Kaiser
- Ultraspherical

Factors such as main-lobe width and side-lobe area have to be considered in selecting the optimal window function to be used in the design.

In this paper, we will be discussing on the steps of using a Kaiser window function in our design of the bandpass filter.

4.Designing a bandpass filter using Kaiser window

The Kaiser window is defined as

$$w[n] = \begin{cases} I_0\left[\beta(1-[(n-\alpha/\alpha)]^2)^{\frac{1}{2}}\right], & 0 \leq n \leq M \\ 0, & otherwise \end{cases}$$

where

$$\alpha = \frac{M}{2}$$

$I_0(x)$ is the zeroth order modified Bessel function of the first kind.

The two parameters used here are the length of the window ($M + 1$) and the shape parameter β.

4.1 Filter Specifications

The specifications of the bandpass filter designed according to the parameters in the index number are as follows.

Maximum passband ripple	= 0.08 dB
Minimum stopband attenuation	= 52 dB
Lower passband edge	= 700 rad/s
Upper passband edge	= 1100 rad/s
Lower stopband edge	= 550 rad/s
Upper stopband edge	= 1200 rad/s
Sampling frequency	= 3200 rad/s

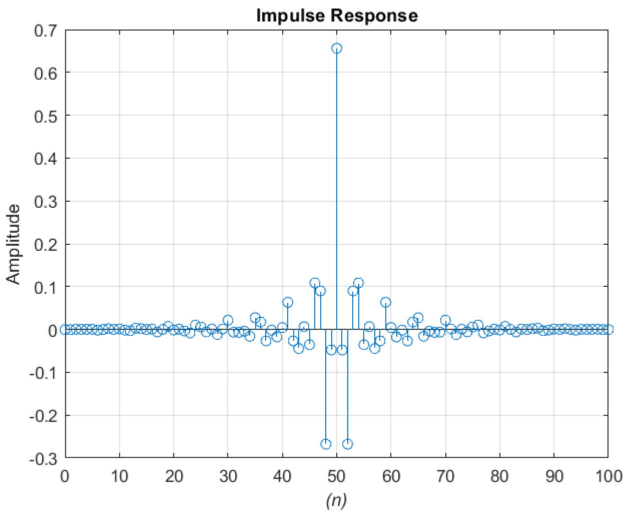
With the calculations pertaining to Kaiser window design algorithm, the following results were obtained. The relevant Matlab codes can be found in the appendix

Lower cutoff frequency	= 600 rad/s
Upper cutoff frequency	= 1150 rad/s

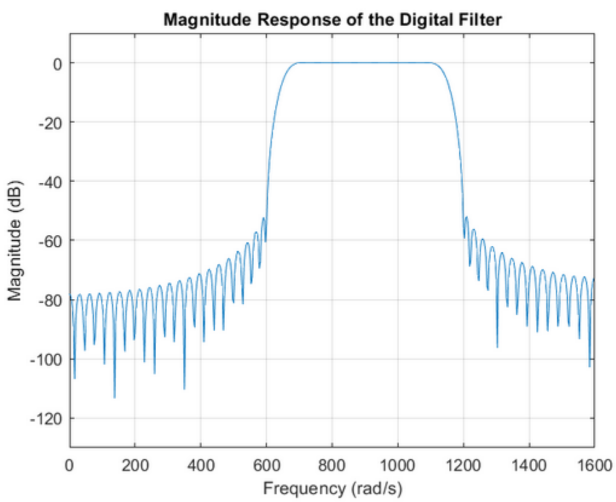
Achieved max passband ripple	= 0.043636 dB
Achieved min stopband attenuation	= 52 dB

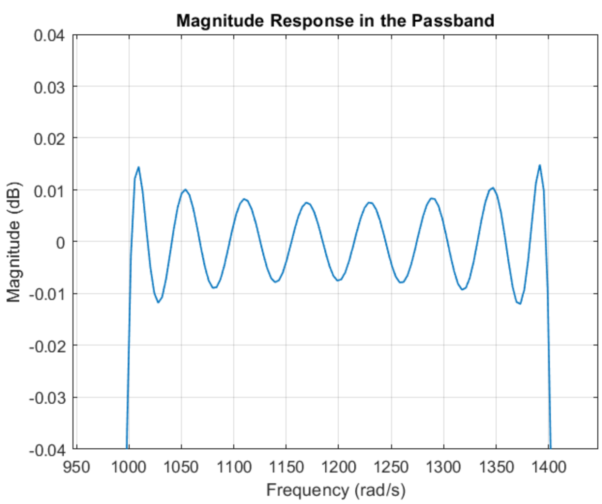
Filter order	= 100

4.2 Causal Impulse Response



4.3 Magnitude Response

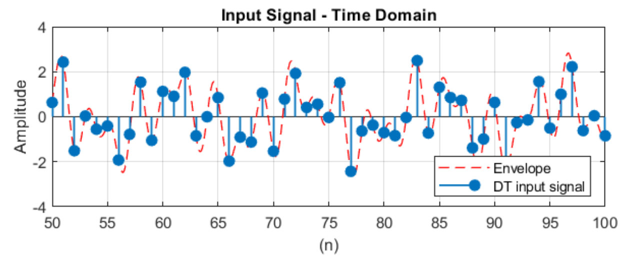


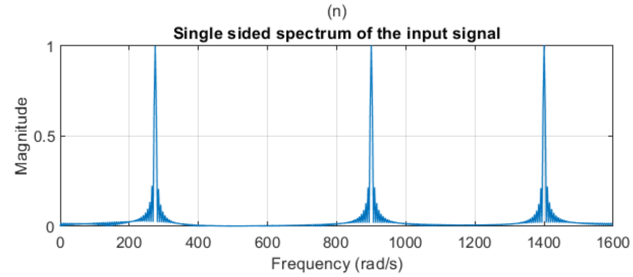


4.4 Input Signal

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

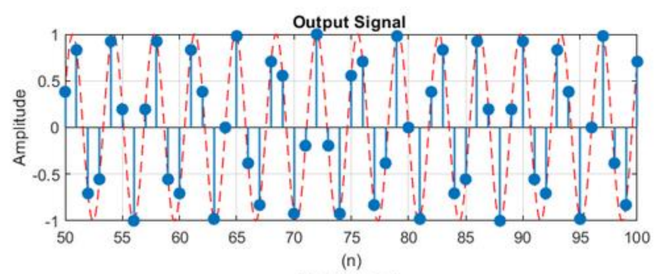
where
Ω_1 = middle frequency of the lower stopband
Ω_2 = middle frequency of the passband
Ω_3 = middle frequency of the upper stopband

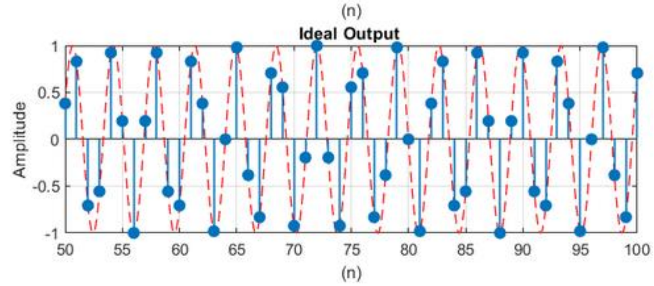




4.5 Output Signal

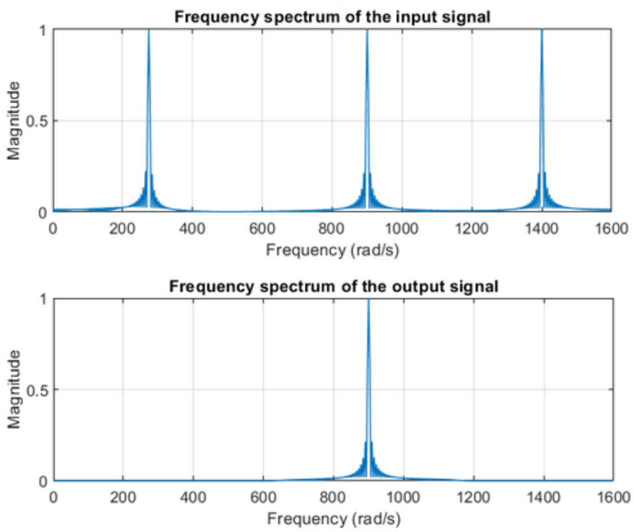
The aforementioned input signal is now filtered by using the bandpass filter we have designed using the Kaiser window.
The expected signal if an ideal bandpass filter is used is also provided alongside for comparison.





4.6 Input signal and the Output signal

Upon observation of the frequency domain plots of the input signal and the output signal, it can be clearly seen that the bandpass filter attenuates the stopband frequency components successfully while passing the passband frequencies.



5.Conclusion

The FIR bandpass filter designed here using the Kaiser window shows near ideal behaviour. Therefore we can verify that the design algorithm used here provides reliable results even under some approximations. And also, Matlab can be used as a platform to perform these designs efficiently for simulation purposes.

6. References

[1] Oppenheim, A., 2013. *Discrete-Time Signal Processing*. Harlow: Pearson, pp.493-553.

[2] Anthoniou, A., 2005. *Digital Signal Processing: Signals, Systems, and Filters*. pp.9.3, 9.4.

7.Appendix

```
function EN2570_FIR_design(index)

clc;
close all;

%getting numbers from index
A = floor(index/100);
B = floor((index-A*100)/10);
C = floor(index-(A*100+B*10));

Ap = 0.03+0.01*A;
%max passband ripple
Aa = 45+B;
%min stopband attenuation
wa1 = C*100+150;
%frequency of lower stopband edge
wa2 = C*100+800;
%frequency of upper stopband edge
wp1 = C*100+300;
%frequency of lower passband edge
wp2 = C*100+700;
%frequency of upper passband edge

ws = 2*(C*100+1200);
%sampling frequency
T = 2*pi/ws;
%sampling interval

disp(['Maximum passband ripple =
',num2str(Ap),' dB']);
disp(['Minimum stopband attenuation
= ',num2str(Aa),' dB']);
disp(['Lower passband edge =
',num2str(wp1),' rad/s']);
disp(['Upper passband edge =
',num2str(wp2),' rad/s']);
disp(['Lower stopband edge =
',num2str(wa1),' rad/s']);
disp(['Upper stopband edge =
',num2str(wa2),' rad/s']);
disp(['Sampling frequency =
',num2str(ws),' rad/s']);

%determining the cutoff frequencies
based on the narrowest transition
band

tran1 = wp1-wa1;
tran2 = wa2-wp2;
Bt = min(tran1,tran2);
%bandwidth of transition band
wc1 = wa1+(Bt/2);
%lower cutoff frequency
wc2 = wa2-(Bt/2);
%upper cutoff frequency
```

```
fprintf('\n');
disp(['Lower cutoff frequncy =
',num2str(wc1),' rad/s']);
disp(['Upper cutoff frequncy =
',num2str(wc2),' rad/s']);

%Determine the minimum delta value
and final Ap and Aa

delta_p = (10^(0.05*Ap)-
1)/(10^(0.05*Ap)+1);
delta_a = 10^(-0.05*Aa);
delta = min(delta_p,delta_a);
final_Ap = 20*log10((1+delta)/(1-
delta));   %final Ap
final_Aa = -20*log10(delta);
%final Aa

fprintf('\n');
disp(['delta_p =
',num2str(delta_p)]);
disp(['delta_a =
',num2str(delta_a)]);
disp(['achieved maximum passband
ripple = ',num2str(final_Ap),'
dB']);
disp(['achieved minimum stopband
attenuation = ',num2str(final_Aa),'
dB']);

%Determine alpha
if (final_Aa<= 21)
    alpha = 0;
elseif (final_Aa>21 &&
final_Aa<=50)
    alpha = 0.5842*((final_Aa-
21)^0.4) + 0.0786*(final_Aa-21);
else
    alpha = 0.1102*(final_Aa-8.7);
end

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

%Determine filter order

Nmin = ceil(((ws*D)/Bt)+1);
%minimum length of the filter
N = Nmin + 1 - rem(Nmin,2);
%correction to get an odd length

fprintf('\n');
disp(['alpha = ',num2str(alpha)]);
disp(['D = ',num2str(D)]);
disp(['Minimum filter order =
',num2str(Nmin-1)]);
disp(['filter order = ',num2str(N-
1)]);

%Determine values of the Kaiser
window function

nmax = (N-1)/2;
%maximum time step
n = 1:nmax;
%positive discrete-time vector

wn = zeros(nmax+1,1);
%vector to store the Kaiser window
hi = zeros(nmax+1,1);
%vector to store the ideal impulse
response

test=0;
I0_alpha = 1;
k=0;
while (test==0)
    k = k+1;
    bessel =
((1/factorial(k))*(alpha/2)^k)^2;
%term of bessel function
    I0_alpha = I0_alpha + bessel;
    if(bessel<10e-6)
        test = 1;
    end
end

for k = 1:nmax+1
    beta = alpha*sqrt(1-(((k-
1)/nmax).^2));
%calculate beta value

    test = 0;
%denominator of the Kaiser window
    I0_beta = 1;
%initialization of I0(beta)
    j = 0;
    while(test == 0)
        j = j+1;
        bessel =
((1/factorial(j))*(beta/2)^j)^2;
%term of Bessel function
        I0_beta = I0_beta + bessel;
        if (bessel<10e-6)
            test = 1;
        end
    end

    wn(k) = I0_beta/I0_alpha;
%value of the Kaiser window
```

```
end

%Determine the values of truncated
h(n)

hi(1) = 1-(2*(wc2-wc1)/ws);
hi(2:nmax+1) = ((sin(wc2*n*T)-
sin(wc1*n*T)))./(n*pi);

%Determine values of causal and
windowed h(n)

htemp = hi .* wn;
h = [flipud(htemp(2:end));htemp];

%Impulse response h(n)

n = 0:(2*nmax);
figure('Name','Impulse
Response','NumberTitle','off');
stem(n,h);
title('Impulse Response');
xlabel('\it(n)');
ylabel('Amplitude');
xlim([0 N-1]);
grid on;

%Frequency response of the filter

[fre_res,f] = freqz(h,1,2048);
%frequency response
w = f*(ws/2)/pi;
%frequency vector

%Amplitude response of the filter

M = 20*log10(abs(fre_res));

figure('Name','Amplitude
Response','NumberTitle','off');

subplot(2,2,1)
plot(w,M);
title('Amplitude response');
xlabel('frequency (rad/s)');
ylabel('M(\omega), dB');
axis([0 ws/2 -20 10]);
grid on;

subplot(2,2,2)
plot(w,M);
title('Amplitude response in the
lower stopband');
xlabel('frequency (rad/s)');
ylabel('M(\omega), dB');
axis([0 wa1 -10.25 -10]);
grid on;

subplot(2,2,3)
plot(w,M);
title('Amplitude response in the
upper stopband');
xlabel('frequency (rad/s)');
ylabel('M(\omega), dB');
axis([wa2 1500 -10.25 -10]);
grid on;

subplot(2,2,4)
plot(w,M);
title('Amplitude response in the
passband');
xlabel('frequency (rad/s)');
ylabel('M(\omega), dB');
axis([wp1 wp2 2.34 2.4]);
grid on;

%Defining the frequencies for the
input signal
omega_1 = (0+wa1)/2;
omega_2 = (wp1+wp2)/2;
omega_3 = (wa2+ws/2)/2;

samples = 500;

n1 = 0:1:samples;
n2 = 0:0.0001:samples;

%defining the input signal
x_nT =
sin(omega_1.*n1.*T)+sin(omega_2.*n1
.*T)+sin(omega_3.*n1.*T);
envelope =
sin(omega_1.*n2.*T)+sin(omega_2.*n2
.*T)+sin(omega_3.*n2.*T);

%plotting a sample of the input
signal in time domain
figure('Name','Input
Signal','NumberTitle','off');

subplot(2,1,1)
plot(n2,envelope,'r--
','LineWidth',0.75)
hold on
stem(n1,
x_nT,'filled','LineWidth',1)
xlabel('(n)')
ylabel('Amplitude')
title('Input Signal - Time Domain')
xlim([50,100])
grid on
legend('Envelope','DT input
signal','Location','best')
hold off
```

```
%Calculating FFT of the input
signal
numpoints =
2^(nextpow2(numel(x_nT)+numel(h)));
x_fft = fft(x_nT,numpoints);
magnitudes =
abs(x_fft/numel(x_nT));
magnitudes =
magnitudes(:,1:numpoints/2+1);
magnitudes(:,2:end-1) =
2*magnitudes(:,2:end-1);
freq = 0:(ws/numpoints):(ws/2)-
ws/numpoints;

%plotting the input signal in the
frequency domain
subplot(2,1,2)
plot(freq,magnitudes(1:numpoints/2)
,'LineWidth',0.75)
xlabel('Frequency (rad/s)')
ylabel('Magnitude')
title('Single sided spectrum of the
input signal')
grid on

%Filtering the input
h_fft = fft(h,numpoints);
y_fft = x_fft.*h_fft;
y_nT = ifft(y_fft,numpoints);
y_nT =
y_nT(floor(N/2)+1:numel(y_nT)-
floor(N/2));

%Ideal output
y_ideal = sin(omega_2.*n2.*T);

%Plotting the Output
figure('Name','Output
Signal','NumberTitle','off');

subplot(2,1,1)
plot(n2,y_ideal,'r--
','LineWidth',0.75)
hold on
stem(n1,
y_nT(1:samples+1),'filled','LineWid
th',1)
xlabel('(n)')
ylabel('Amplitude')
title('Output Signal - Time Domain
(n:50 to 100)')
xlim([50,100])
grid on
legend('Envelope','DT output
signal','Location','southeast')
hold off

%FFT parameters of the output
signal
magnitudes_out =
abs(y_fft/(samples+1));
magnitudes_out =
magnitudes_out(:,1:numpoints/2+1);
magnitudes_out(:,2:end-1) =
2*magnitudes_out(:,2:end-1);
freq_out = 0:(ws/numpoints):(ws/2)-
ws/numpoints;

%plotting the output signal in the
frequency domain
subplot(2,1,2)
plot(freq_out,magnitudes_out(1:nump
oints/2),'LineWidth',0.75)
xlabel('Frequency (rad/s)')
ylabel('Magnitude')
title('Single sided spectrum of the
output signal')
grid on
```