FIR Filter Design for BandPass Filter

Sandeepa H.K.C.A.
dept. of Electronic and Telecommunication Engineering,
University of Moratuwa.

***Abstract*—In this document we try to implement FIR filter design using windowing method in conjunction with the Kaiser window. And also, there are several specifications to show. In addition to that accuracy of this design will be checked using given function and resultant plots in both time and frequency domain. The software is used for project is MATLABR2018a.**

I. Introduction

This project is used to implement FIR (Finite Duration Impulse Response) filter design using given specification in the project description. When we try to implement filter design using windowing method this is also call as Fourier series method, there are more techniques but in here we use most famous windowing method call as Kaiser windowing method. Using MATLABR2018a software and building functions these calculations for designing were implemented.

II. Basic Theory

When we design the filters there are two main techniques. Closed form design technique and Optimized design technique. In here we use closed form technique.

Since the frequency response is periodic, we can represent it as a Fourier Series as below.

$$H(e^{j\omega T}) = \sum_{n=-\infty}^{n=\infty} h(nT)e^{-j\omega nT}$$

where,

$$h(nT) = \frac{1}{\omega_s}\int_{-\omega_s}^{\omega_s} H(e^{j\omega T})\, e^{j\omega nT} d\omega$$

Define the impulse response of the Filter as $h(nT)$ and then by substituting $z = e^{j\omega T}$ we obtain the Transfer function of the filter as follows.

$$H(z) = \sum_{n=-\infty}^{n=\infty} h(nT)z^{-n}$$

In the windowing method we truncate the impulse response using multiplication with window function in time domain as follows,

$$h_w(nT) = h(nT)w(nT)$$

Hence, the frequency response of the modified filter is given by the convolution integral.

$$H_w(e^{j\omega T}) = \frac{T}{2\pi}\int_0^{\frac{2\pi}{T}} H(e^{j\omega' T})\, W(e^{j(\omega-\omega')T})d\omega'$$

With these theory aspects filtering can be done by using several windows such as Rectangular, Hann, Hamming, Blackman, Dolph-Chebyshev and Kaiser. If Rectangular window is easy it creates more ripple ratio. In order to reduce that effect and good approximation Kaiser window is selected

Kaiser window function is,

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

where,

$$\beta = \alpha\sqrt{1-\left(\frac{2n}{N-1}\right)^2}$$

and

$$I_0(x) = 1 + \sum_{k=1}^{\infty}\left[\frac{1}{k!}\left(\frac{x}{2}\right)^k\right]^2$$

Using that knowledge try to implement Band Pass filter using Kaiser Window.

III. Implementation

A. *Finding Parameters for given Index(Your Index)*

In the project description it mentions how can we select the basic parameters according to your index.

For given index number 180564F, the parameters are,

Maximum passband ripple, Ãp	0.08 dB
Minimum stopband attenuation, Ãa	51dB
Lower passband edge, Ωp1	700 rad/s
Upper passband edge, Ωp2	1100 rad/s
Lower stopband edge, Ωa1	550 rad/s
Upper stopband edge, Ωa2	1200 rad/s
Sampling frequency, Ωs	3200 rad/s

B. *Calculations*

Parameters are defined as follows,

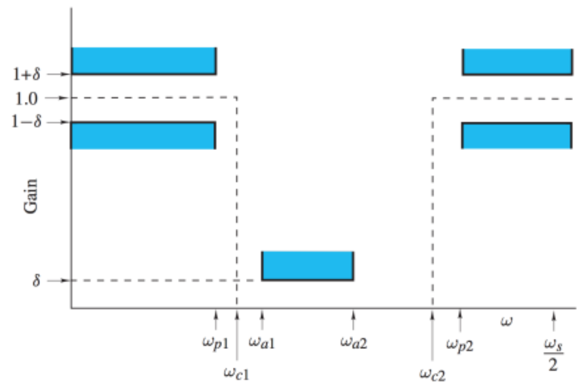

Figure 1: Design Specification

Critical transition width can be calculated as follows,

$$B_t = \min\left[\left(\omega_{p1} - \omega_{a1}\right), \left(\omega_{a2} - \omega_{p2}\right)\right]$$

Idealized frequency response for a BandPass filter is given as,

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

where cutoff frequencies,

$$\omega_{c1} = \omega_{p1} - \left(\frac{B_t}{2}\right), \omega_{c2} = \omega_{p2} + \left(\frac{B_t}{2}\right)$$

By using inverse Fourier Transform impulse response can be obtained,

$$h[nT] = \begin{cases} \frac{2}{\omega_s}(\omega_{c1} - \omega_{c2}), & for\ n = 0 \\ \frac{1}{n\pi}(sin(\omega_{c2} nT) - \sin(\omega_{c1} nT)), & Otherwise \end{cases}$$

C. Determine δ value

The parameter δ is defined in a way that, actual passband ripple $A_p \leq \tilde{A}_p$
The actual minimum stopband attenuation, $A_a \geq \tilde{A}_a$

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

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

D. Actual Stopband Attenueation

After finding correct value of delta then move to find out Actual Attenuation at the stop band as follows,

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

E. Detaremine parameter D and α

Using A_a which is determined earlier.

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

and

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

The smallest, odd value which satisfies the following condition is chosen as N.

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

IV. Results

By going through the above defined equations, the parameters can be calculated as follows,

Parameter	Value
delta p	0.0046
delta a	0.0028
A_a	51.00
D	2.9979
α	4.6614
N	97

By doing calculation using MATLAB the results can be generated as figures. It helps to figure out our implementation.

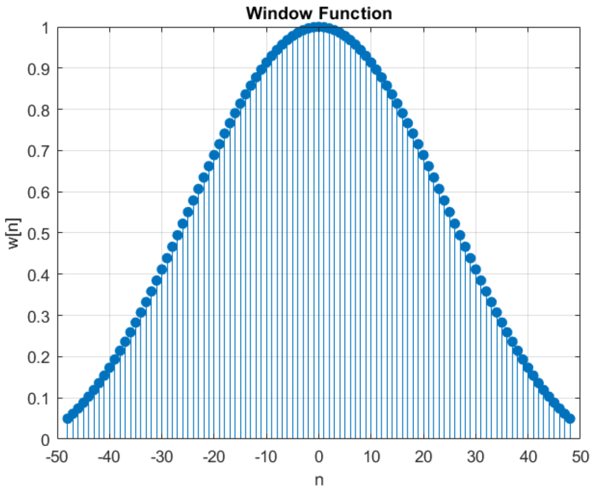

Figure 2: Impulse Response of Kaiser Window

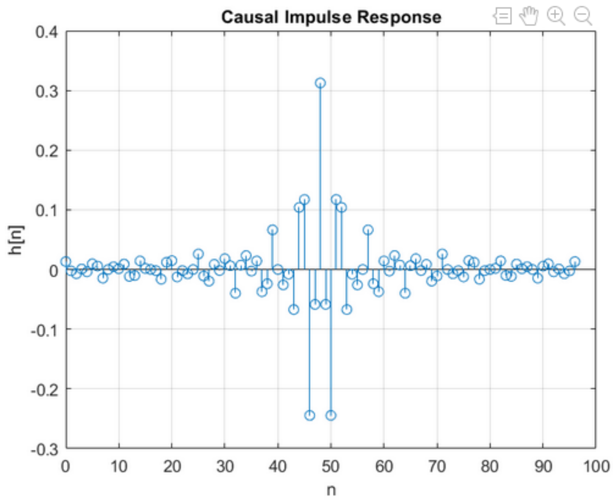

Figure 3: Causal Impulse Response

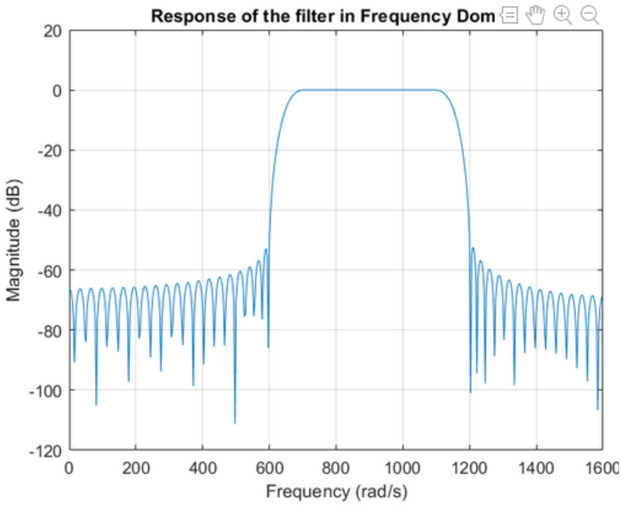

Figure 4: Magnitude Response of the Filter

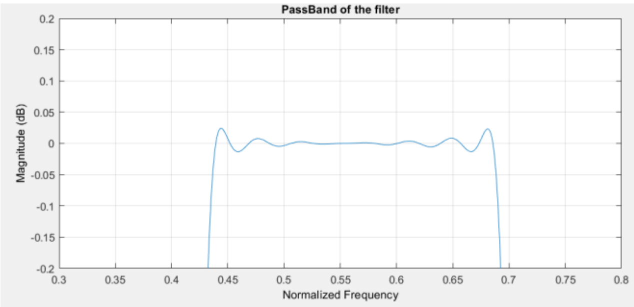

Figure 5: Magnitude Response of the passband

V. Validation of The Designed Filter

For validations following function is used and the omega values are mentioned below,

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

where,

ω_1	*275 rad/s*
ω_2	*900 rad/s*
ω_3	*1400 rad/s*

for the validations,

when we give the above input to our designed filter it should filter out the ω_2 because it defined using passband frequency.

The observations are in both time domain and frequency domain and also have the expectation vs ideal output results. Using these results, we can clearly say that this bandpass filter works properly.

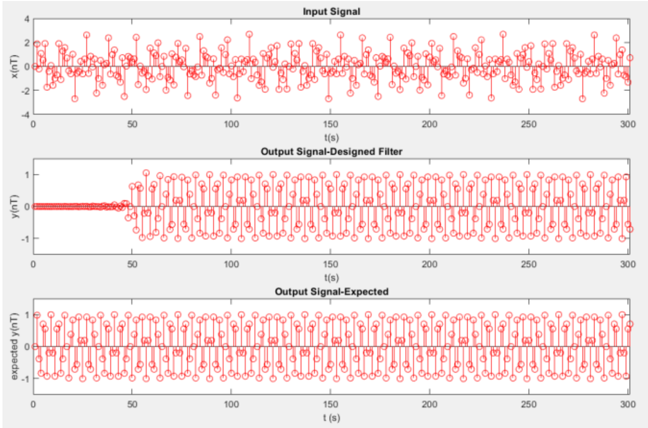

Figure 6: Time Domain Representation for input signal, designed filter's output and expected output

Expected output and filter output are almost same.

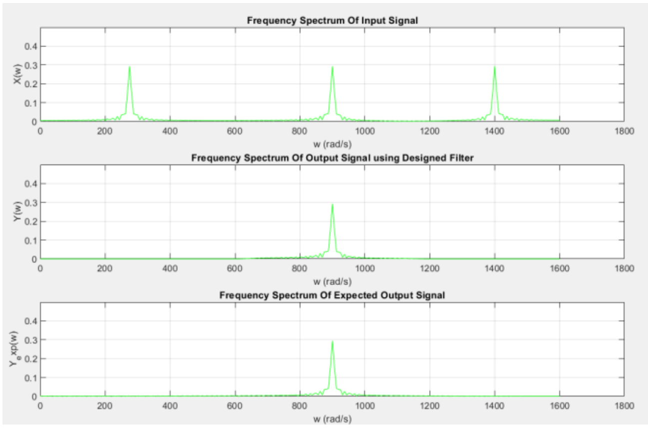

Figure 7: Frequency Domain Representation for input signal, designed filter's output and expected output

VI. Conclusion

The implementation of FIR Filter design is efficient way because it gives the almost same output as the ideal filter. Its ripple ratio is very small compared to the other windows. And also, the methods that we used to implement is very helpful for students and it can be easily implemented.

Acknowledgment

I'd like to express my heartfelt gratitude to Dr. Chamira Edussooriya for her invaluable assistance in this project.

References

A. Antoniou, "Digital Signal Processing," 2005. [Online]. Available: www.ece.uvic.ca/~dsp. [Accessed 29 12 2016]

En.wikipedia.org. 2021. *Kaiser window*. [online] Available at: <https://en.wikipedia.org/wiki/Kaiser_window> [Accessed 5 March 2021].

Appendix

```
clc;
clear all;

output = calcMyFilterSpecs(5, 6, 4); % call the function to get values for the given index

Ws = output(7);    % set sample frequency value
T=2*pi/Ws;    % set the sampling period
delta_p = (10^(0.05*output(1)) - 1)/(10^(0.05*output(1)) + 1); % determine delta p value using Ap
delta_a = (10^((-0.05)*output(2))); % determine delta a value using Aa
delta = min(delta_a, delta_p); % find delta value
Actual_Aa = (-20)*log10(delta); % finding actual stopband attenuation
Bt = min((output(3) - output(5)), (output(6) - output(4)));   % create transition width
Wc1 = output(3) - Bt/2; % define the cutoff frequencies
Wc2 = output(4) + Bt/2;
valAlfa = getAlfa(Actual_Aa); % call the function to get value alpha
valD = getD(Actual_Aa);  % call the function to get value D


calcN = ceil((output(7)*valD)/Bt + 1); % calculate current N value and round it to the nearest integer
finalN = calcN + 1 - rem(calcN, 2); % get the correct N value


M = finalN - 1;
tau = M/2;
range1_n = -tau: 1: tau;   % define range of n value

%----------------------- Create Kaiser Window ---------------------------

valBeta = valAlfa*(1 - ((2*range1_n)/M).^2).^0.5; % Determine beta value
i0_alpha = bessel(valAlfa); % call the bessel function to fing window
i0_beta = bessel(valBeta);
win = i0_beta/i0_alpha; % find the window function for specified n values
stem(range1_n, win, "fill")
%ylim([0 1])
grid on;
xlabel("n");
ylabel("w[n]");
title("Window Function")

%-------------------- create impulse response ---------------------------

impulse_response = hnT(range1_n,Wc1,Wc2,Ws,T);  function for ideal filter
stem(range1_n, impulse_response)
xlabel("n"); ylabel("h[n]");
grid on
title ("Ideal Filter Response");
range2_n = 0:1:M;
stem(range2_n, impulse_response);  %causal response
title("Causal Impulse Response");
grid on;
```

```
% Determine the impulse response of the filter - with causal condition

imp = impulse_response.*win;
stem(range2_n, imp);
title("Impulse Response of the filter - Causal");
xlabel("n"); ylabel("h[n]")
grid on;


%------------------- compute the magnitude response -----------------------
[h, W] = freqz(imp);
W = W/T;
h = 20*log10(abs(h));
plot(W, h);
title("Response of the filter in Frequency Domain");
xlabel("Frequency (rad/s)"); ylabel("Magnitude (dB)")
grid on;

% -------------------------------------------------------------------------

fvtool(imp) % determine magnitude response using function

lower_limit = round(length(W)/(Ws/2)*Wc1);
upper_limit = round(length(W)/(Ws/2)*Wc2);
w_pass = W(lower_limit:upper_limit);
h_pass = h(lower_limit:upper_limit);
axis([-inf, inf, -0.2, 0.2]); %zooming version of pass band
grid on;
title("PassBand of the filter");
xlabel("Normalized Frequency"); ylabel("Magnitude (dB)");
plot(w_pass,h_pass);
sample = 300; %select an appropriate samples
nfft = 2^nextpow2(sample); % create length
W_1 = output(5)/2;                        % define omega values
W_2 = (output(3) + output(4))/2;
W_3 = (output(6) + output(7)/2)/2;

nT = 0: T: sample*T;


%------------------------ Create an input Signal ------------------------

XnT = sin(W_1*nT) + sin(W_2*nT) + sin(W_3*nT);
X_W = fft(XnT, nfft);
X_1 = T*abs(X_W(1:nfft/2 + 1));

%--------------------- Output Signal of the filter ----------------------


HnT = fft(imp, nfft);
Y_W = HnT.*X_W;
yt = ifft(Y_W, nfft);
Y = T*abs(Y_W(1: nfft/2 + 1));
```

```
%-------------- Expected output of the filter (as the ideal one)---------

expected_ynT = sin(W_2*nT);
expected_YW = fft(expected_ynT, nfft);
Y_1 = T*abs(expected_YW(1: nfft/2 + 1));

%------------------ time domain plots for comparison --------------------

figure,
subplot(3, 1, 1);
stem(1:(sample+1),XnT(1:(sample+1)), 'r');
title('Input Signal');
xlabel('t(s)'); ylabel('x(nT)');
axis([0, (sample+1), -4, 4]);
subplot(3, 1, 2);
stem(1:(sample+1),yt(1:(sample+1)), 'r');
title('Output Signal-Designed Filter');xlabel('t(s)'); ylabel('y(nT)');
axis([0,(sample+1), -1.5, 1.5]);
subplot(3, 1, 3);
stem(1:(sample+1),expected_ynT(1:(sample+1)), 'r');
title('Output Signal-Expected');
xlabel('t (s)'); ylabel('expected y(nT)');
axis([0, (sample+1), -1.5, 1.5]);

%--------------- frequency domain plots for comparison ----------------

w = Ws*(0:1/nfft:1/2);
figure, subplot(3, 1, 1) ;
plot(w,X_1, "g");
title('Frequency Spectrum Of Input Signal');
xlabel('w (rad/s)'); ylabel('X(w)');
grid on;
axis([0, 1800, 0, 0.5]);
subplot(3, 1, 2);
plot(w, abs(Y'), "g");
title('Frequency Spectrum Of Output Signal using Designed Filter');
xlabel('w (rad/s)'); ylabel('Y(w)');
grid on;
axis([0, 1800, 0, 0.5]);
subplot(3, 1, 3);
plot(w, abs(Y_1'), "g");
title('Frequency Spectrum Of Expected Output Signal');
xlabel('w (rad/s)'); ylabel('Y_exp(w)');
grid on;
axis([0, 1800, 0, 0.5]);
```

```
%--------------------- choose parameter alfa --------------------------

function [valAlfa] = getAlfa(Attenuation)

if Attenuation <= 21
      valAlfa = 0;
else
if Attenuation > 50
      valAlfa = 0.1102*(Attenuation - 8.7);
else
      valAlfa = 0.5842*(Attenuation - 21)^0.4 + 0.07886*(Attenuation-
21);
end
end

end

%---------------------- choose parameter D as follows -----------------

function [valD] = getD(Attenuation)

if Attenuation <= 21
      valD = 0.9222;
else
      valD = (Attenuation - 7.95)/(14.36);
end

end

%-------------------------- window function --------------------------

function [i0_x] = bessel(x)
i0_x = 1;
for k = 1: 10
   i0_x = i0_x + (((x/2).^k)*(1/factorial(k))).^2;
end

end

%-------------------------- ideal filter function ---------------------

function [impulse_response] = hnT(n,Wc1,Wc2,Ws,T)
impulse_response=zeros(size(n));
for c1 = 1:length(n)
if n(c1)==0
      impulse_response(c1)=2*(Wc2-Wc1)/Ws;
else
      impulse_response(c1)=(sin(n(c1)*Wc2*T)-
sin(Wc1*n(c1)*T))/(n(c1)*pi);
end
end

end
```

```
%Function for take values the last integers of Index Number
function output= calcMyFilterSpecs(A, B, C)
Ap = 0.03 + (0.01 * A);
Aa = 45+B;
Wp1 = (C * 100) + 300;
Wp2 = (C * 100) + 700;
Wa1 = (C * 100) + 150;
Wa2 = (C * 100) + 800;
Ws =  2*((C * 100) + 1200);

output = [Ap, Aa, Wp1, Wp2, Wa1, Wa2, Ws]
end
```