EN2063 – SIGNALS AND SYSTEMS

Project

Digital Filters

Index No: 200285E

Name: Nuwan Kannangara

The objective of the project is to provide experience in the design of FIR and IIR digital filters for prescribed specifications. For FIR filters, the windowing method (in conjunction with the Kaiser window) is used whereas, for IIR filters, the bilinear transformation method is used.

The specification of the digital filter

Parameter	**Value**
Maximum passband ripple	0.12 dB
Minimum stopband attenuation	58 dB
Lower passband edge	900 rad/s
Upper passband edge	1400 rad/s
Lower stopband edge	600 rad/s
Upper stopband edge	1600 rad/s
Sampling frequency	4000 rad/s

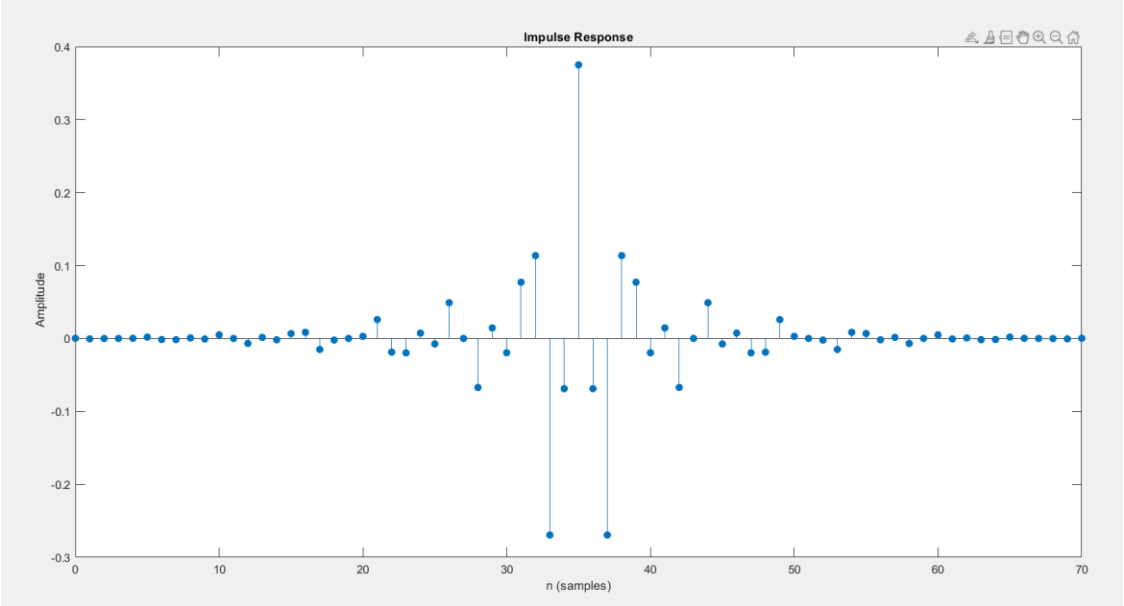
Design of FIR bandpass digital filter

FIR filter is an impulse response with finite duration. Here we used windowing method in conjunction with the Kaiser window to design FIR bandpass digital filter. Windowing is the simplest technique for designing FIR filters because of its conceptual simplicity and ease of implementation. Designing FIR filters by windowing takes the inverse FFT of the desired magnitude response and applies a smoothing window to the result. The smoothing window is a time domain window. To design an FIR filter by windowing, we must first decide on an ideal frequency response and calculate the impulse response of the ideal frequency response. Then truncate the impulse response to produce a finite number of coefficients. To meet the linear-phase constraint, maintain symmetry about the center point of the coefficients. Finally, we must apply a symmetric smoothing window. Kaiser window is a type of window which is mostly used in FIR filter design because it contains an adjustable parameter with which main lobe width and corresponding minimum stop band attenuation of designed filter can be controlled.

1)

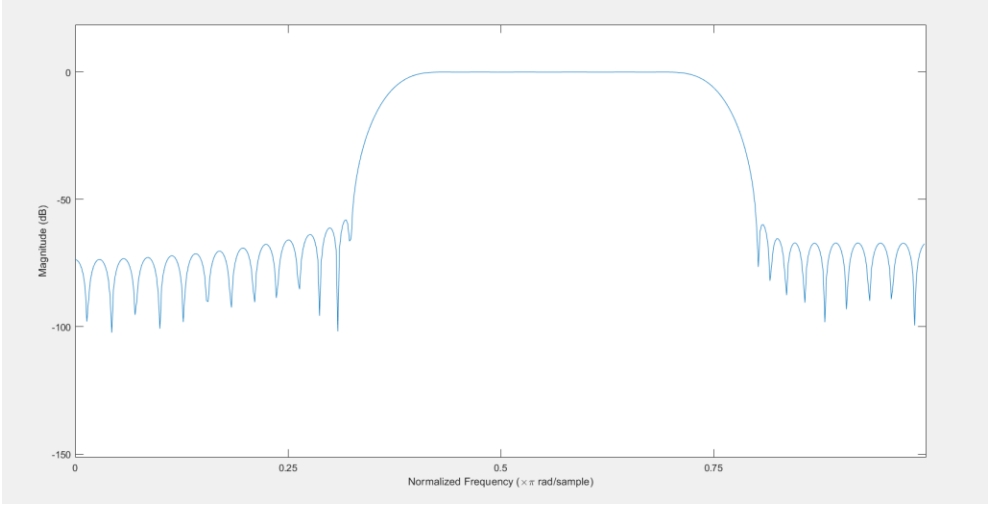
a) Impulse response

Here, the initial impulse response is determined by the idealized frequency response of bandpass filter.



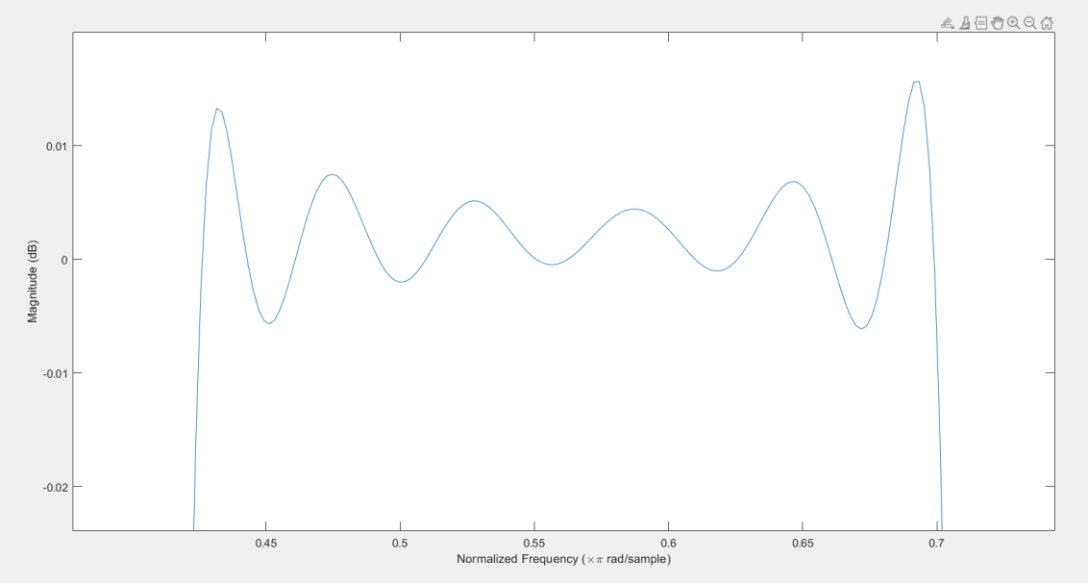
b) Magnitude response of the digital filter

Here we can see that the gain is 1 during the frequency range (900-1400) Hz. Also, during other frequencies, the gain is approximately equal to 0.



c) Magnitude response for wp1<w<wp2

Small oscillation can be seen in this band which is less than 0.02 dB.



Design of an IIR bandpass digital filter

IIR filter is an impulse response with infinite duration. Here we used bilinear transformation method to design IIR bandpass digital filter. Bilinear transformation method (BZT) is a mapping from analog S plane to digital Z plane. This conversion maps analog poles to digital poles and analog zeros to digital zeros. Thus, all poles and zeros are mapped. So, first we have to design an appropriate analog filter. Then the required digital filter should be obtained by applying the bilinear transform to the transfer function of the analog filter. Prewarping of frequencies is essential in order to obtain the required digital filter. Here we used the Chebyshev approximation method.

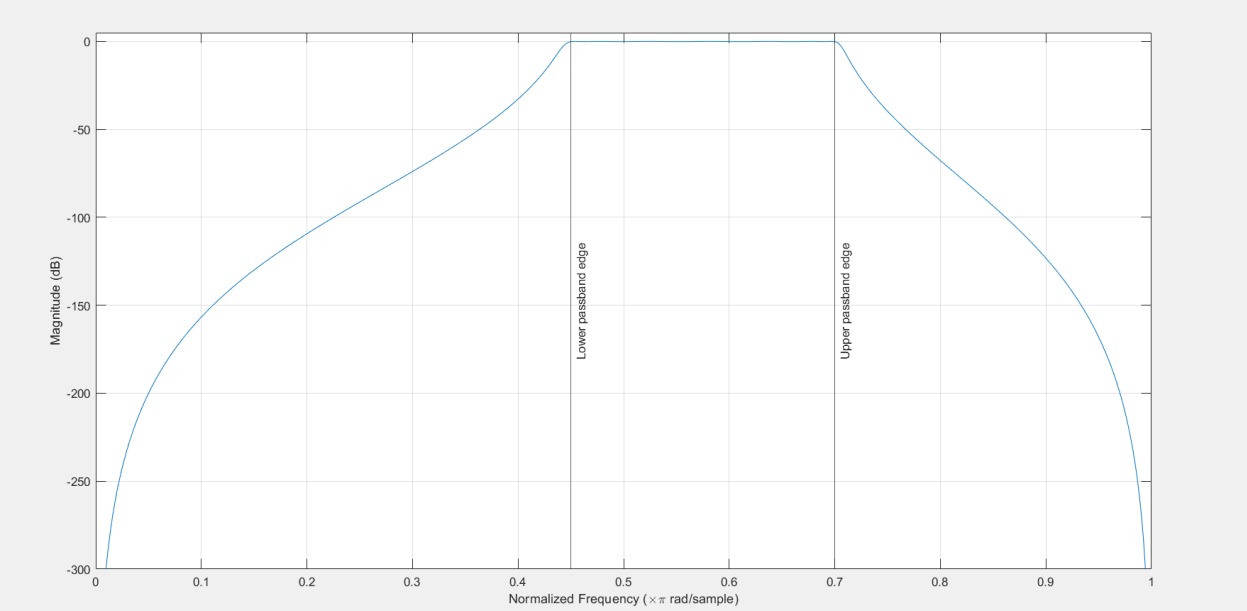
2)

a) Coefficients of the transfer function of IIR filter

Numerator_Coefficients	Denominator_Coefficients
1	0.037271
0.95972	0.2609
1.6582	0.78269
0.53205	1.3045
0.62408	1.3045
-0.056725	0.78269
0.091177	0.2609
-0.037807	0.037271

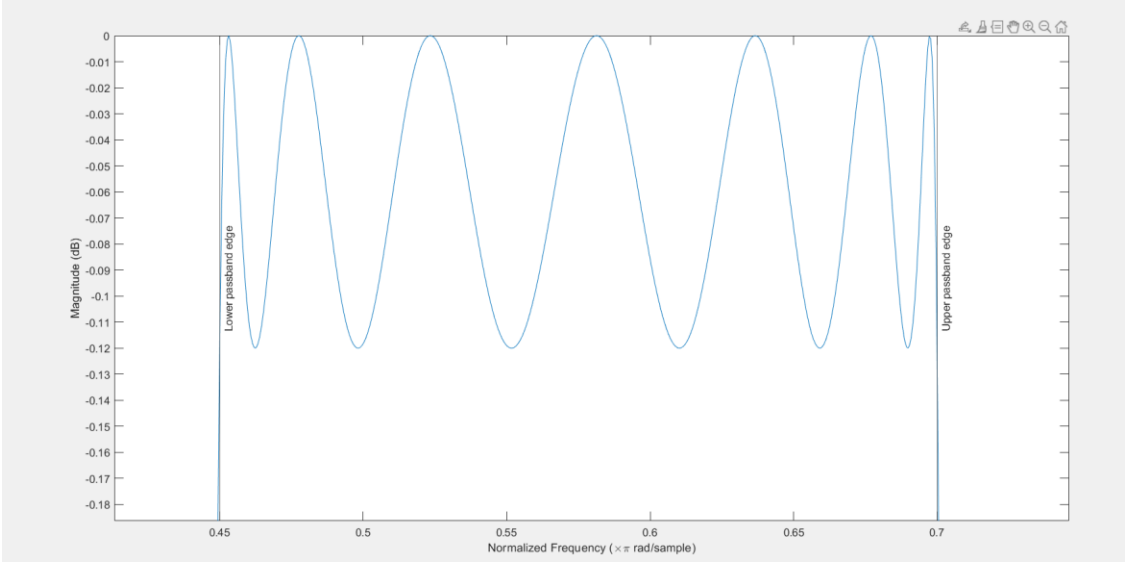
b) Magnitude response of the digital filter

Here we can see the gain is 1 between the frequency range (900-1400) Hz while before 900 Hz and after 1400 Hz the gain gradually decreases.



c) Magnitude response for wp1<w<wp2

Here we can see that oscillation with much higher frequency than FIR filter for same deviation.



3) Comparison of FIR and IIR digital filter

Order of FIR digital filter = 70
Order of IIR digital filter = 7

The order of a filter refers to the number of previous input samples that are used to produce each output sample. A 70th order FIR digital filter designed using the Kaiser window method would use 70 previous input samples, while a 7th order IIR filter generated by a Chebyshev analog filter using bilinear transformation would use 7 previous input samples and 7 previous output samples.

In terms of the number of multiplications and additions required to process a sample, the 70th order FIR filter would require 70 multiplications and 69 additions, assuming symmetry of coefficients is exploited. The 7th order IIR filter would require 7 multiplications and 7 additions for the input samples, and 7 multiplications and 7 additions for the output samples, for a total of 14 multiplications and 14 additions.

In summary, the 70th order FIR filter would require more multiplications and additions than the 7th order IIR filter, but it would use more previous input samples to produce each output sample. However, FIR filter has the property of linear phase, which is very useful in applications such as audio processing and signal analysis, while IIR filter are more suited for applications that require sharper cutoff and lower computational cost.

APPENDIX

MATLAB Code for the FIR Filter

```
%%specification of the filter
fsamp = 4000;
fcuts = [600 900 1400 1600];
mags = [0 1 0];
devs = [0.001259 0.01372 0.001259];

%%Designing the Kaiser window
[n,Wn,beta,ftype] = kaiserord(fcuts,mags,devs,fsamp);
n = n + rem(n,2);
hh = fir1(n,Wn,ftype,kaiser(n+1,beta));

%%Magnitude response of the filter
[H,f] = freqz(hh,1);
plot(f/pi,20*log10(abs(H)))
ax = gca;
```

```
ax.YLim = [-150 20];
ax.XTick = 0:.25:1;
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')

%%impulse response of the filter
[n,Wn,beta,ftype] = kaiserord(fcuts,mags,devs,fsamp);
n = n + rem(n,2);
hh = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
impz(hh)


%%Magnitude response of the filter for wp1<w<wp2
[H,f] = freqz(hh,1);
plot(f/pi,20*log10(abs(H)))
ax = gca;
ax.XTick = 9/20:.05:7/10;
ax.YTick = -1:.010:1;
ax.XLim = [9/20 7/10];
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
```

MATLAB Code for the IIR Filter

```
%%Specification of the filter and Normalizinf frequencies
Wp = [900 1400]/2000;
Ws = [600 1600]/2000;
Rp = 0.12;
Rs = 58;

%%Transforming into digital filters
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs);
[z,p,k] = cheb1ap(n,0.12);
[A,B,C,D] = zp2ss(z,p,k);

%%Bilinear transformation
fs = 4e3;
f1 = 900; u1 = 2*fs*tan(f1*(2*pi/fs)/2);
f2 = 1400; u2 = 2*fs*tan(f2*(2*pi/fs)/2);
[At,Bt,Ct,Dt] = lp2bp(A,B,C,D,sqrt(u1*u2),u2-u1);
[Ad,Bd,Cd,Dd] = bilinear(At,Bt,Ct,Dt,fs);
[hd,fd] = freqz(ss2sos(Ad,Bd,Cd,Dd),2048,fs);

%%Magnitude response of filter
plot(fd/2000,mag2db(abs(hd)))
xline([f1/2000 f2/2000],"-",["Lower" "Upper"]+" passband edge",
...
```

```
    LabelVerticalAlignment="middle")
ax = gca;
ax.XTick = 9/20:.05:7/10;
ax.YTick = -1:.010:2;
ax.XLim = [9/20 7/10];
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')


%%Coeffients of the transfer function
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs);
[a,b] = cheby1(n,Rp,900/1400);
oef_table =
table(b',a','VariableNames',{'Numerator_Coefficients','Denominat
or_Coefficients'})


%%Magnitude response of filter
plot(fd/2000,mag2db(abs(hd)))
xline([f1/2000 f2/2000],"-",["Lower" "Upper"]+" passband edge",
...
    LabelVerticalAlignment="middle")
ax = gca;
ax.XTick = 9/20:.05:7/10;
ax.YTick = -1:.010:2;
ax.XLim = [9/20 7/10];
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
```