ECEN 4632
Digital Signal Processing Project

Brent Hyman

December 17, 2015

Methods for designing a digital band-stop filter

Project Objective

The objective of this project is to analyze the use of the following IIR filters:

→Butterworth
→Chebyshev Type I
→Chebyshev Type II
→Elliptic

and following FIR filters:

→Kaiser
→Parks-McClellan

and they will be used to design a band-stop filter. This filter (defined as H(z) in this report) will remove unwanted noise from an audio file. It will need to have the following specifications for each design method:

$$\begin{cases} -\frac{1}{2}dB \leq |H(e^{j\omega})|_{dB} \leq \frac{1}{2}dB & : \omega \in [0-1400]Hz \cup [1900-5512.5]Hz \\ |H(e^{j\omega})|_{dB} \leq -100dB & : \omega \in [1600-1700]Hz \end{cases}$$

The magnitude ($|H(e^{j\omega})|$) in the pass-bands should ideally be of unity gain. In other words, the amplitude of the frequency response ($|H(e^{j\omega})|_{dB}$) should be at 0 dB since

$$20 * log_{10}(1) = 0dB$$

So basically we're tolerating a $\pm\frac{1}{2}$ dB deviation in the pass-bands. Of coarse, we can't necessarily expect a constant amplitude in the stop-band either but as long as the filter restricts the amplitude from getting any bigger than -100 dB, then we're good. One of the goals of this project is to see how the different implementations can affect the magnitude response in both its pass and stop bands.

The name of the game in digital filter design is in the determining of the system function coefficients such that the filter meets the required specifications. In this case, the specifications require the uses of a band-stop filter. The system function is also known as a transfer function of the filter which is in the z-domain for a discrete time system. The rational transfer function of a filter takes on the following form:

$$H(z) = \frac{\Sigma_{i=0}^{P} b_i z^{-i}}{\Sigma_{j=0}^{Q} a_j z^{-j}} = \frac{Y(z)}{X(z)} \tag{1}$$

This means that

$$\Sigma_{j=0}^{Q} a_j z^{-j} Y(z) = \Sigma_{i=0}^{P} b_i z^{-i} X(z) \tag{2}$$

Using the linearity and time delay properties, the inverse z-transform yields the following linear constant-coefficient difference equation which represents the input-output relationship of the filter:

$$y[n] = \frac{1}{a_0}(\Sigma_{i=0}^{P} b_i x[n-i] - \Sigma_{j=1}^{Q} a_j y[n-j]) \tag{3}$$

The filter can then be implemented by its difference equation. There are multiple algorithms and structures that can be used for the implementation. But this includes material from a whole other field within DSP that this report will not attempt to cover. Built in Matlab functions will be used for the implementations. This report is purely focused on the design.

Project Layout

The conclusion and the developed Matlab code can also be found at the very end of this report.

1 IIR Filter Design

The big idea behind designing an IIR filter is to "discretize" an analog filter. A band-stop filter can be obtained by first designing a corresponding prototype low pass filter. Once the minimum order of this filter is calculated such that it meets the required specifications, it can be transformed into its band-stop form and then the cut off frequencies can be determined. And then the filter gets transformed into the z-domain and the coefficients for eqn (1) are obtained. Of course this is all done underneath the hood of Matlab. The users part in design basically comes down to deciding what kind of filter to use.

1.1 Butterworth

A Buterworth filter has characteristics such that:

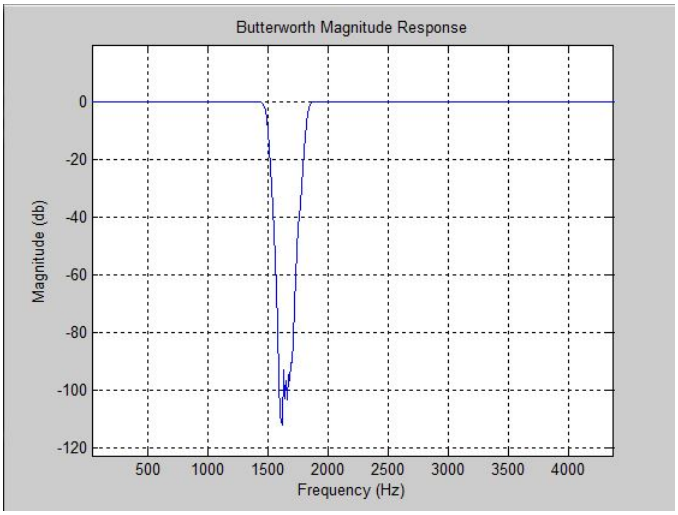
- Its magnitude response is:
  - maximally flat in the pass-band
  - monotonic in both the pass-band and the stop-band
- the low-pass version of this analog filter has a magnitude-squared frequency response defined as

$$|H_B(j\omega)|^2 = \frac{1}{1 + (\frac{\Omega}{\Omega_c})^{2n}}$$

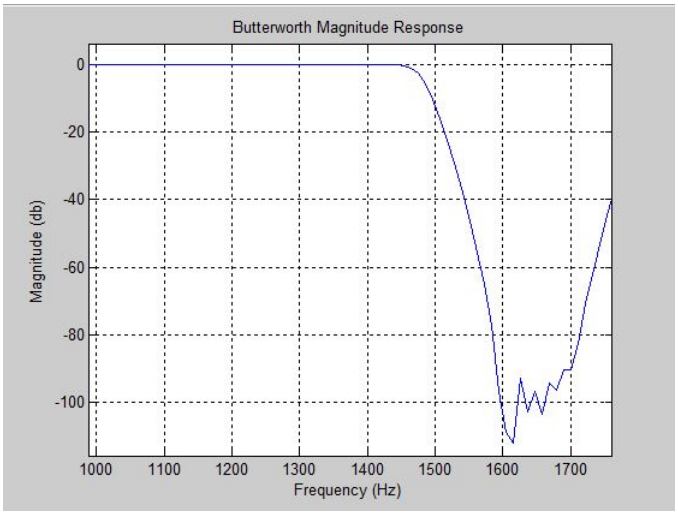
Using the Butterworth design, the results for this filter were as follows:

(a) Filter Efficiency:
→order 18
→37 add/multiply ops per input cycle

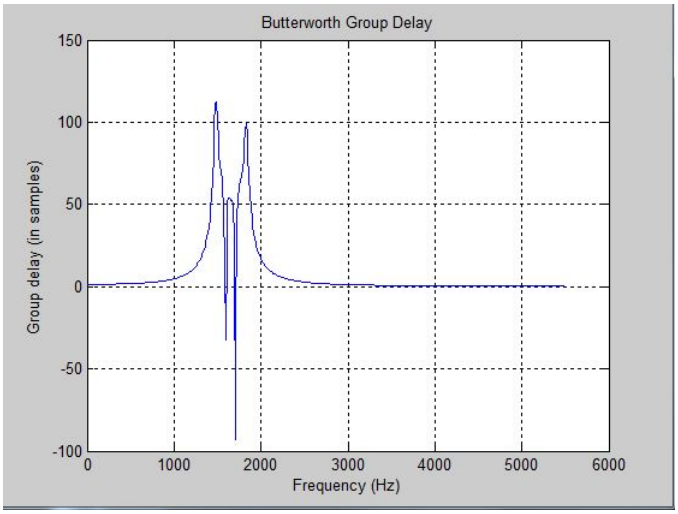
(b) Magnitude Response



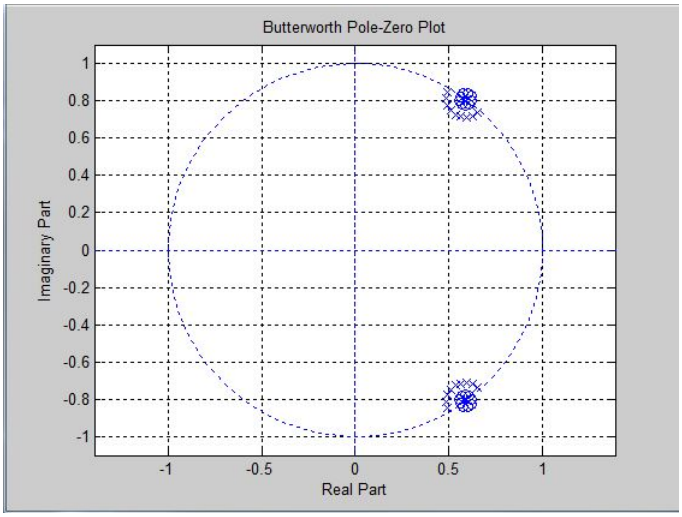
(c) Magnified Magnitude Response



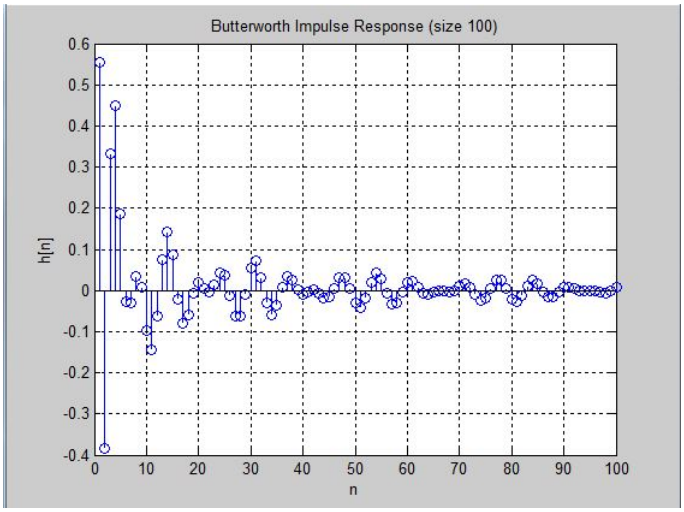
(d) Group Delay



(e) Pole-Zero Diagram



(f) Impulse Response



1.2 Chebyshev Type I

A Chebyshev Type I filter has properties such that:

- its magnitude response is:
  - equiripple in the pass-band
  - monotonic in the stop-band
- the low-pass version of this filter has a magnitude-squared frequency response defined as

$$|H_B(j\omega)|^2 = \frac{1}{1 + [\epsilon^2 V_N^2(\frac{\Omega}{\Omega_c})]^{-1}}$$

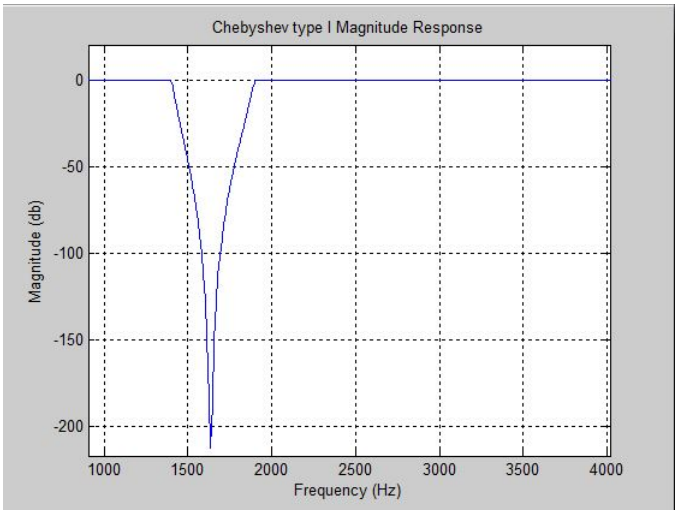
  where $V_N(x)$ represents the Nth order Chebyshev polynomial defined as

$$V_N(x) = cos(Ncos^{-1}(x))$$

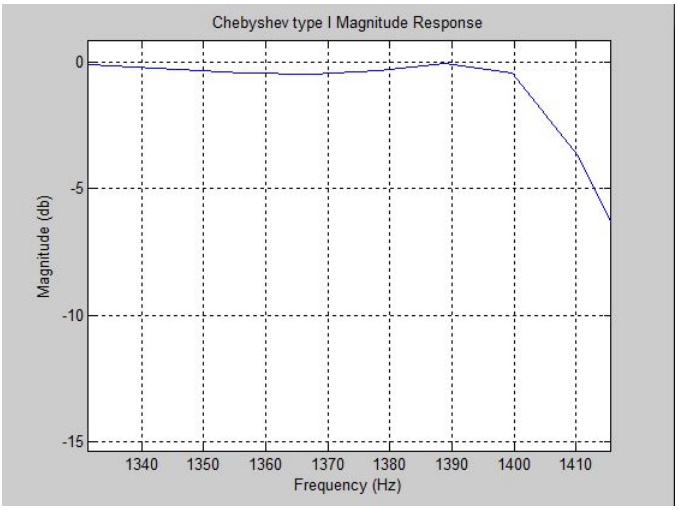
Using the Chebyshev type I design, the results for this filter were as follows:

(a) Filter Efficiency:
→order 12
→25 add/multiply ops per input cycle

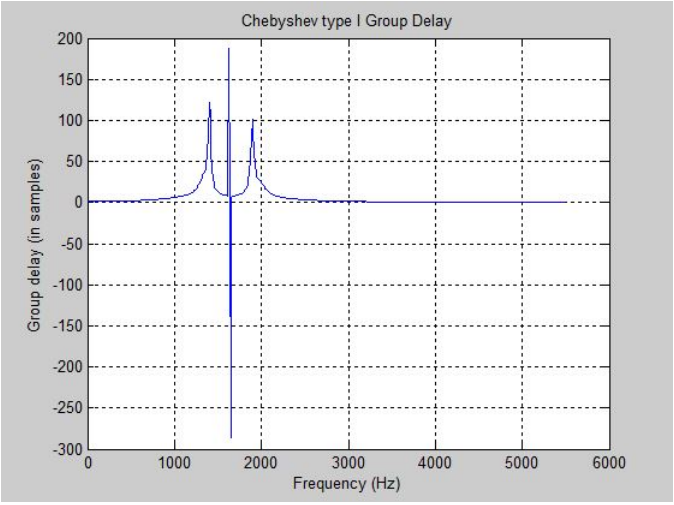
(b) Magnitude Response



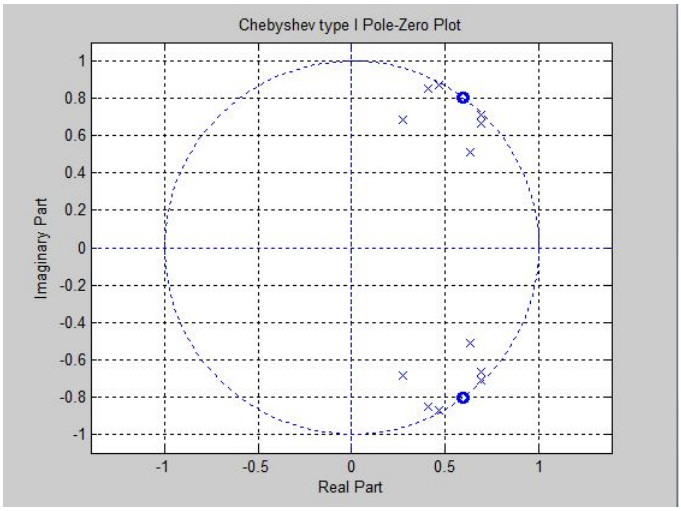
(c) Magnified Magnitude Response



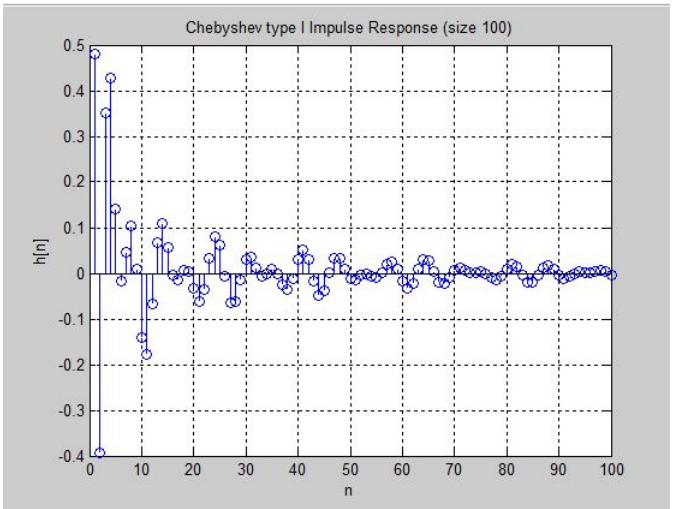
(d) Group Delay



(e) Pole-Zero Diagram



(f) Impulse Response



1.3 Chebyshev Type II

A Chebyshev Type II filter has properties such that:

- its magnitude response is:
  - monotonic in the pass-band
  - equiripple in the stop-band
- the low-pass version of this filter has a magnitude-squared frequency response defined as

$$|H_B(j\omega)|^2 = \frac{1}{1 + \epsilon^2 V_N^2(\frac{\Omega}{\Omega_c})}$$

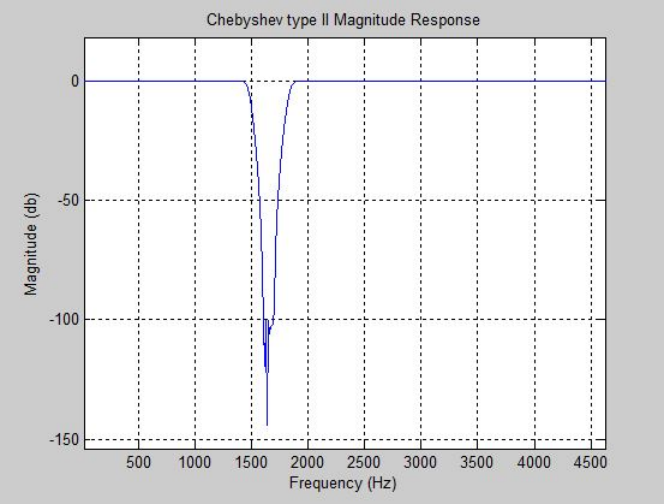
  where $V_N(x)$ represents the Nth order Chebyshev polynomial defined as

$$V_N(x) = cos(Ncos^{-1}(x))$$

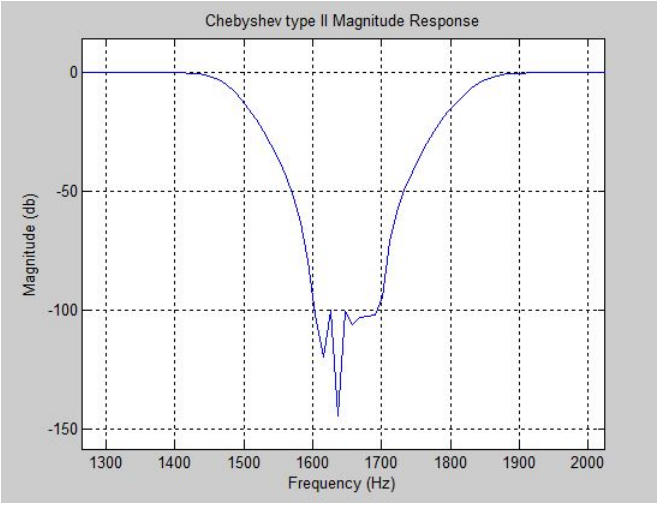
Using the Chebyshev type II design, the results for this filter were as follows:

(a) Filter Efficiency:
→order 12
→25 add/multiply ops per input cycle

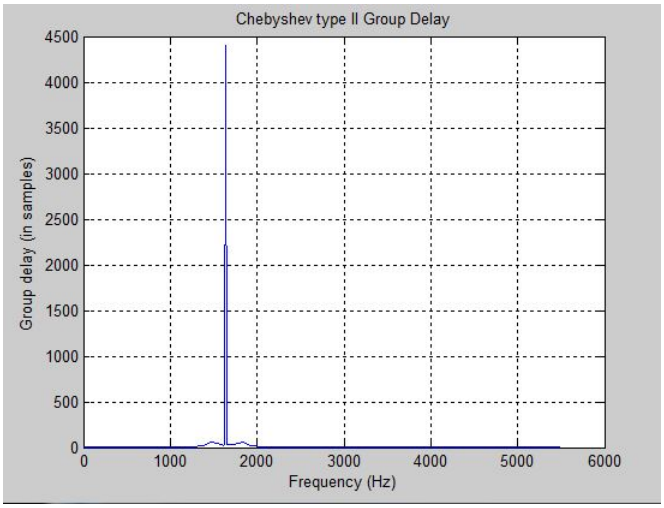
(b) Magnitude Response



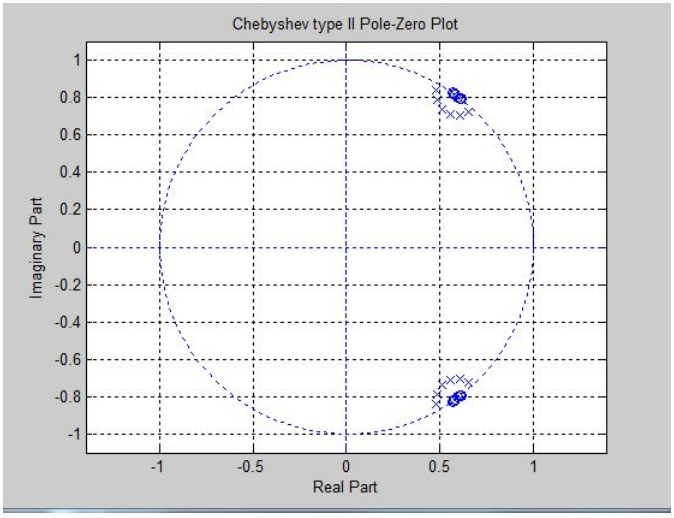
(c) Magnified Magnitude Response



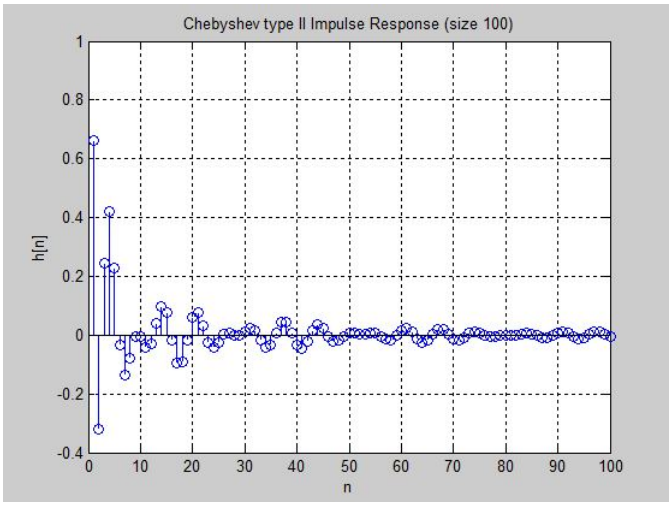
(d) Group Delay



(e) Pole-Zero Diagram



(f) Impulse Response



1.4 Elliptic

An Elliptic filter has properties such that:

- its magnitude response has:
  - equiripple error in both the pass-band and stop-band
- the low-pass version of this filter has a magnitude-squared frequency response defined as

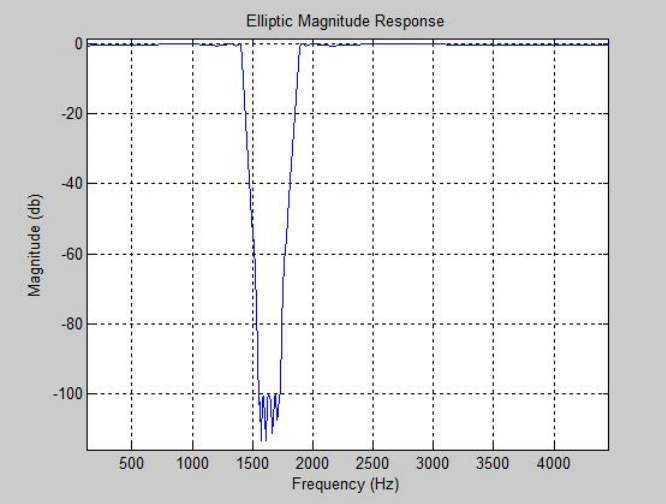
$$|H_B(j\omega)|^2 = \frac{1}{1+\epsilon^2 U_N^2(\Omega)} \tag{4}$$

  where $U_N(\Omega)$ represents the Jacobian Elliptic function

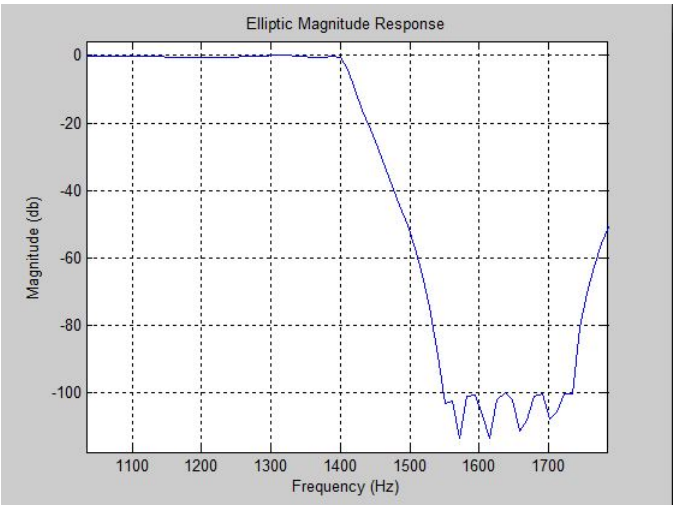
Using the Elliptic design, the results for this filter were as follows:

(a) Filter Efficiency:
→order 12
→25 add/multiply ops per input cycle

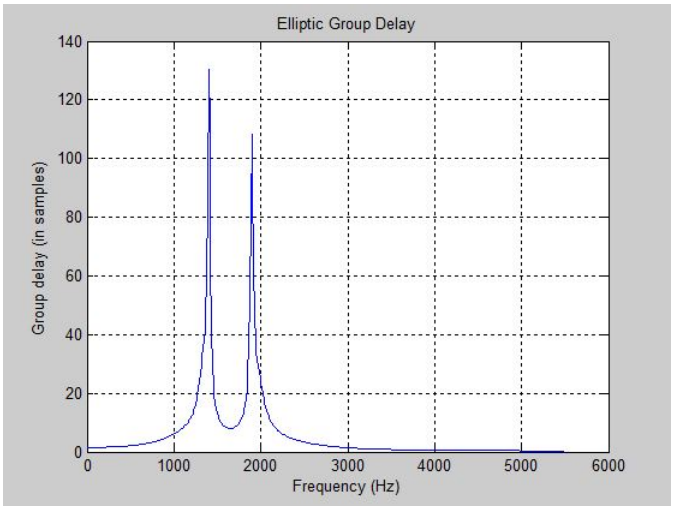
(b) Magnitude Response



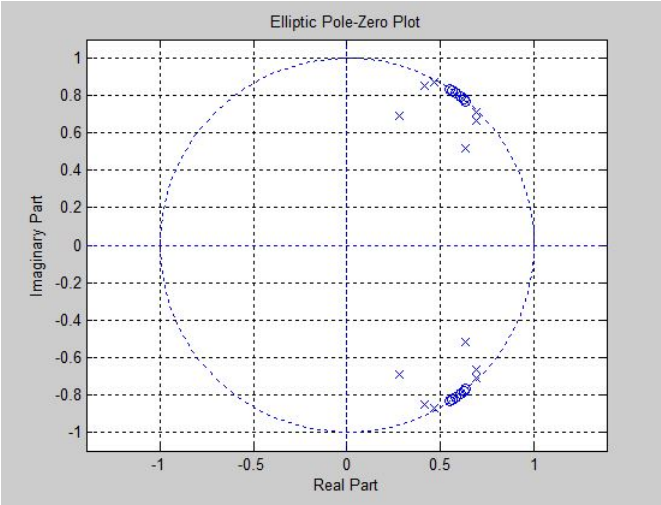
(c) Magnified Magnitude Response



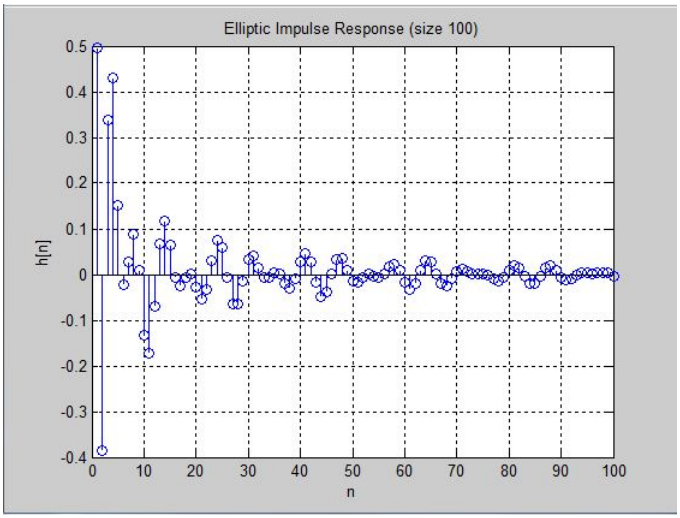
(d) Group Delay



(e) Pole-Zero Diagram



(f) Impulse Response



2 FIR Filter Design

The idea behind FIR design is to try and approximate the frequency response of a desired filter. The difference equation of an FIR filter also does not have any feedback components. It takes on the following form:

$$y[n] = \Sigma_{i=0}^{P} b_i x[n-i] \tag{5}$$

and the impulse response takes the form of:

$$h[n] = \Sigma_{i=0}^{P} b_i \delta[n-i] \tag{6}$$

Therefore, FIR filters are always stable. FIR filter design itself can also be done in various ways. Either windowing methods can be used or a numerical method for approximating functions can be used. This report will look at both approaches.

2.1 Kaiser

- This is a windowing method
- the impulse response of a desired filter gets windowed in an effort to make the closest approximation to the desired IIR system
- the kaiser window is defined as

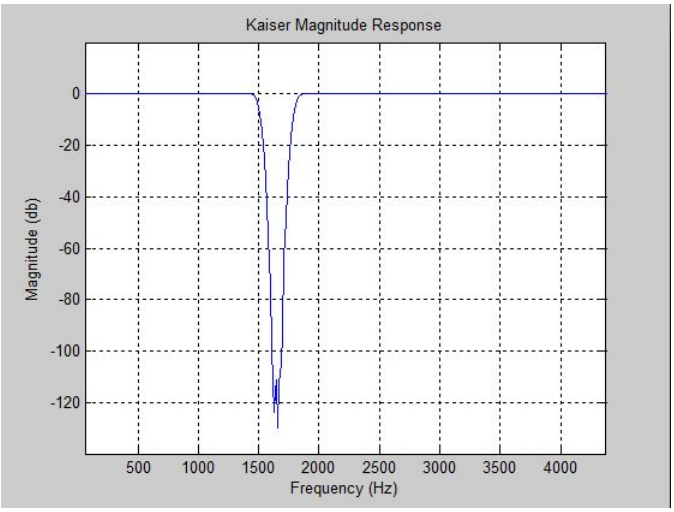
$$w[n] = \begin{cases} \frac{I_0[\beta(1-[(n-\alpha)/\alpha]^2)^{\frac{1}{2}}]}{I_0(\beta)} & : 0 \leq n \leq M \\ 0 & : else \end{cases}$$

  where $\alpha = M/2$ and I_0 is the 0th order modified Bessel Function.

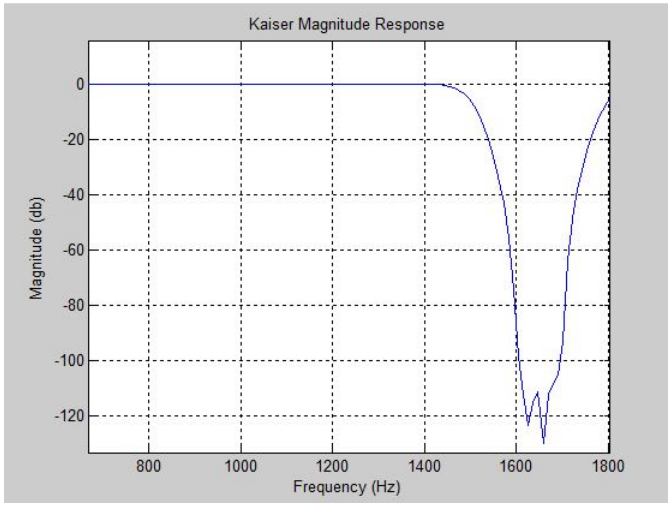
Using the Kaiser window approach toward the design, the results for this filter were as follows:

(a) Filter Efficiency:
→order 354
→709 add/multiply ops per input cycle

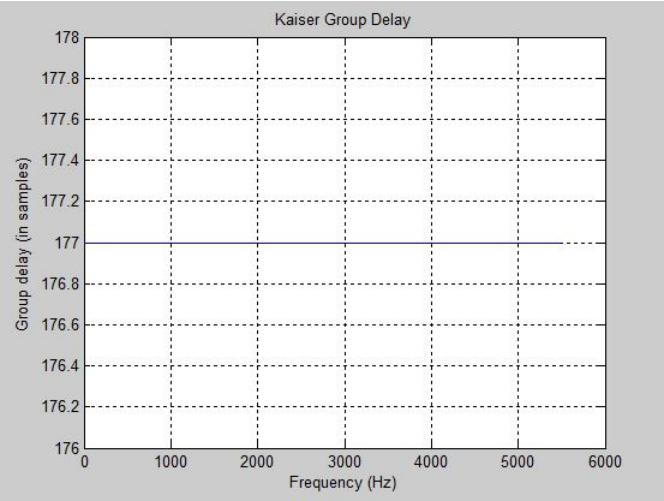
(b) Magnitude Response



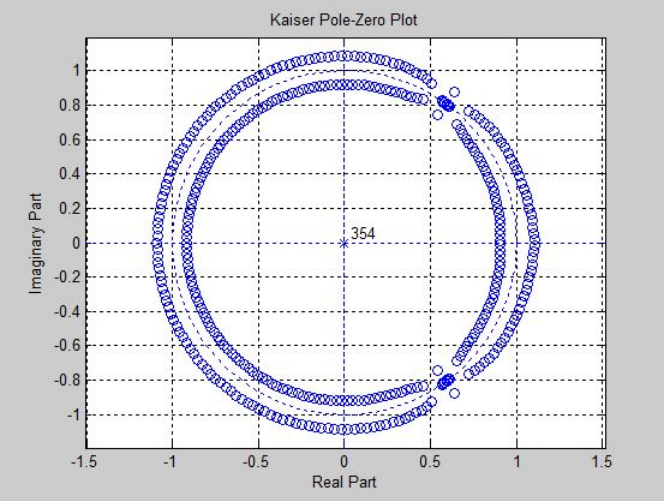
(c) Magnified Magnitude Response



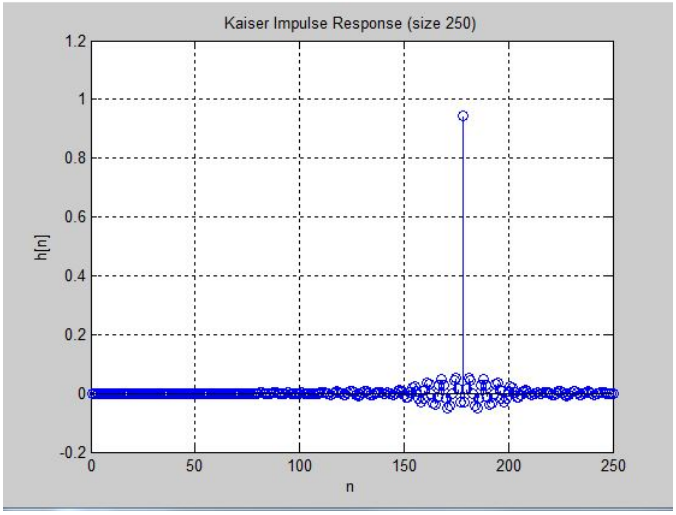
(d) Group Delay



(e) Pole-Zero Diagram



(f) Impulse Response



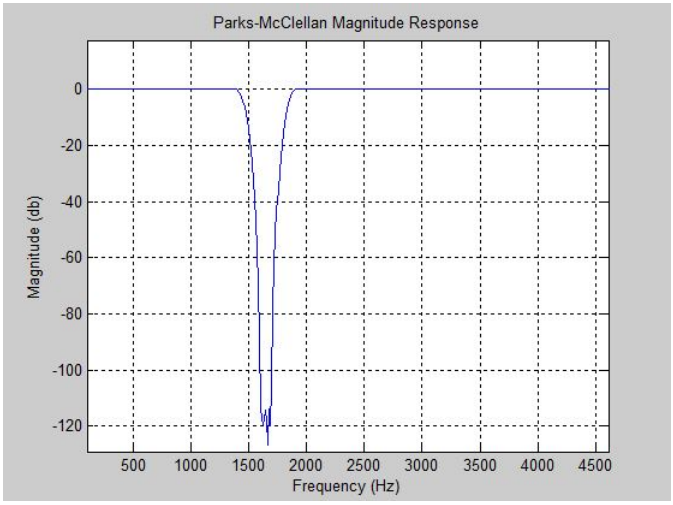
2.2 Parks-McClellan

- this is what's called an optimum approximation method
- Parks-McClellan is a numerical algorithm for approximating the best fit function to the ideal frequency response
- it's meant to be a real efficient algorithm that allows for a more efficient filter

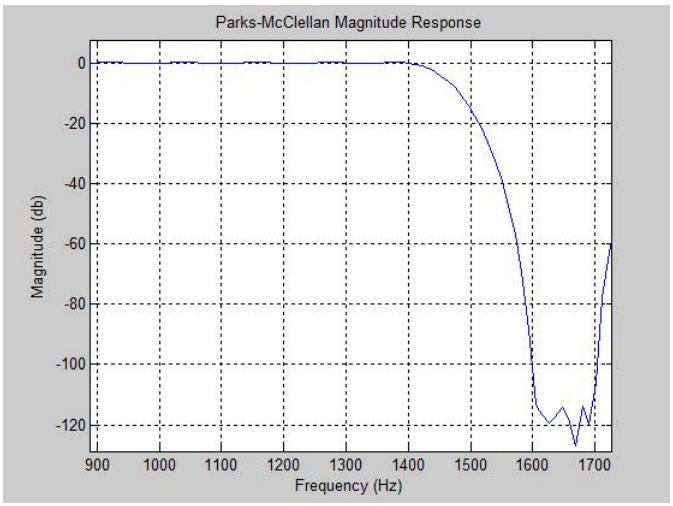
Using the Parks-McClellan optimum approximation approach toward the design, the results for this filter were as follows:

(a) Filter Efficiency:
→order 184
→369 add/multiply ops per input cycle

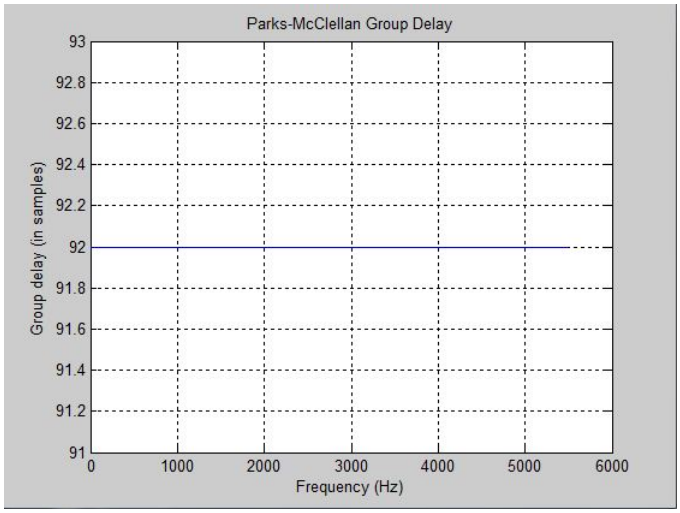
(b) Magnitude Response



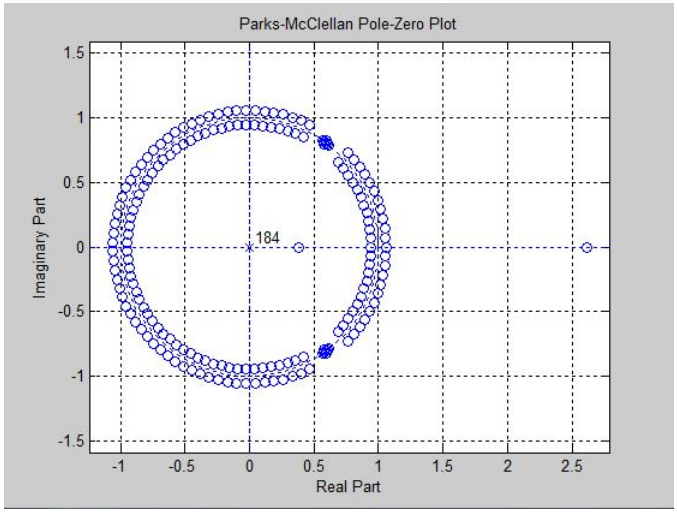
(c) Magnified Magnitude Response



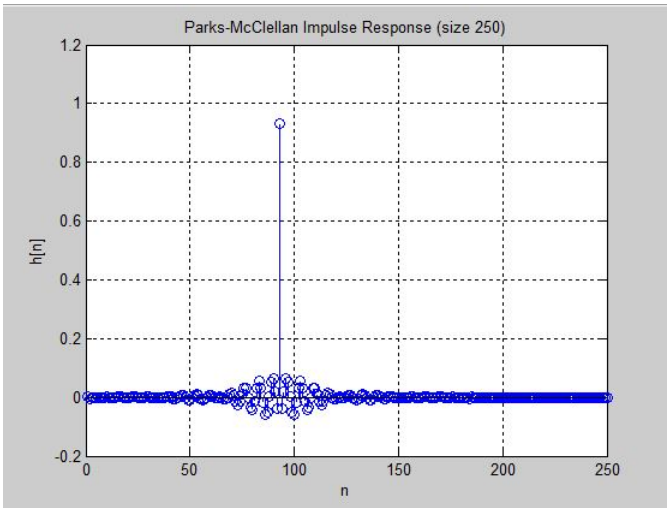
(d) Group Delay



(e) Pole-Zero Diagram



(f) Impulse Response



Conclusion

As these results show, IIR filters are typically more efficient, which was shown by their lower order and operation count. Although, based off of the group delay plots, FIR filters usually always have generalized linear phase. FIR filters also tend to be easier to design. The given audio file appeared to sound like a light soft song playing along with a single disturbing higher pitch tone. The Chebyshev type I and Elliptic filters seemed to have put out the best results. I couldn't hear the noise at all with those filters. The Butterworth, Chebyshev type II, and Parks-McClellan filters had output pretty clear sound but I was still able to hear an ever so slight noise in all three of their outputs. The kaiser filter was the worst. The audio file was filtered and sounded ok, but the noise was a lot more apparent in its output. All in all though, the project was a success. I was able to filter the audio with each filter and saw the advantages and disadvantages of each one.

```matlab
% noisyFilter.m (Matlab funciton file)
% Brent Hyman
% DSP project -> the filter
% December 17 2015
%=====================================================================
function [] = noisyFilter()

    clear all;

    %read in the audio file data
    [noisyAudio, Fs] = wavread('noisy.wav');
    nyqFreq = Fs/2;

    %define the band-stop filter specs for IIR filter design
    psBndRng = [1400 1900]/nyqFreq;
    stpBndRng = [1600 1700]/nyqFreq;
    psBndTol = 0.5;
    stpBndAttn = 100;

    %define the band-stop filter specs for FIR filter design
    bndEdgeFreqs = [1400 1600 1700 1900];
    stpBndMags = [1 0 1];
    rpplDevs = [10^(1/40)-1 10^-5 10^(1/40)-1];

    %display
    % -> give user opportunity to listen to the noisy.wav file
    % -> otherwise, they can go ahead and implement a filter
    disp('*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=')
    disp('Audio de-noising program')
    userInput = input('    type 1 to listen to noisy.wav or 0 to skip it: ');
    if userInput
        disp(' -> playing noisy.wav...')
        sound(noisyAudio,Fs)
    else
        disp(' ')
        disp(' -> noise filtering through the use of a digital band-stop filter')
        disp(' -> the noisy.wav file in this directory has already been read in')
        disp(' -> the different implementation methods available on this program:')
        disp('    1. Butterworth')
        disp('    2. Chebyshev type I')
        disp('    3. Chebyshev type II')
        disp('    4. Elliptic')
        disp('    5. Kaiser')
        disp('    6. Parks-McClellan')
        userInput = input('pick a method: ');
```

```
        %filter implementation options
        option = 1;
        switch userInput
            case 1
                disp(' Butterworth Implementation...')
                name = 'Butterworth'; impType = 'I';
                [n, Wn] = buttord(psBndRng,stpBndRng,psBndTol,stpBndAttn);
                [b,a] = butter(n,Wn,'stop');
            case 2
                disp(' Chebyshev type I Implementation...')
                name = 'Chebyshev type I'; impType = 'I';
                [n, Wn] = cheb1ord(psBndRng,stpBndRng,psBndTol,stpBndAttn);
                [b,a] = cheby1(n,psBndTol,Wn,'stop');
            case 3
                disp(' Chebyshev type II Implementation...')
                name = 'Chebyshev type II'; impType = 'I';
                [n, Wn] = cheb2ord(psBndRng,stpBndRng,psBndTol,stpBndAttn);
                [b,a] = cheby2(n,stpBndAttn,Wn,'stop');
            case 4
                disp(' Elliptic Implementation...')
                name = 'Elliptic'; impType = 'I';
                [n, Wn] = ellipord(psBndRng,stpBndRng,psBndTol,stpBndAttn);
                [b,a] = ellip(n,psBndTol,stpBndAttn,Wn,'stop');
            case 5
                disp(' Kaiser Implementation...')
                name = 'Kaiser'; impType = 'F';
                [n, Wn, beta, ftype] = kaiserord(bndEdgeFreqs, stpBndMags, rpplDevs, nyqFreq * 2);
                b = fir1(n, Wn, ftype, kaiser(n+1,beta)); a = 1;
            case 6
                disp(' Parks-McClellan Implementation...')
                name = 'Parks-McClellan'; impType = 'F';
                [n, fo, ao, w] = firpmord(bndEdgeFreqs, stpBndMags, rpplDevs, nyqFreq*2);
                b = firpm(n, fo, ao, w); a = 1;
            otherwise
                disp('NOT AN OPTION!!!')
                option = 0;
        end

        if option
            fltrdAudio = filter(b,a,noisyAudio);
            filterAnalysis(a,b,fltrdAudio,nyqFreq,name,n,impType);
            %sound(fltrdAudio,Fs)
        end
    end
end
```

```
% filterAnalysis.m (Matlab function file)
% Brent Hyman
% DSP project -> the analysis
% December 17 2015
%=========================================================================
function[] = filterAnalysis(a,b,fltrdAudio,nyqFreq,filterType,n,fImp)

    % -> Display the filter's efficiency
    if fImp == 'I'
        nstr = int2str(2*n);
        opCount = int2str(2*(2*n)+1);
    else
        nstr = int2str(n);
        opCount = int2str(2*n+1);
    end
    disp(' ')
    disp(['   filter is of order: ' nstr])
    disp(['   filter Op count: ' opCount ' add/multiply ops per input cycle'])
    disp(' ')

    % -> plot magnitude response in decibels
    [h,w1] = freqz(b,a,512);
    figure
    plot( (w1/pi)*nyqFreq,20*log10(abs(h)) )
    title([filterType ' Magnitude Response'])
    xlabel('Frequency (Hz)')
    ylabel('Magnitude (db)')
    grid on

    % -> plot the group delay (in samples)
    [gd,w2] = grpdelay(b,a);
    figure
    plot( (w2/pi)*nyqFreq,gd )
    title([filterType ' Group Delay'])
    xlabel('Frequency (Hz)')
    ylabel('Group delay (in samples)')
    grid on

    % -> plot pole-zero plot
    figure
    zplane(b,a);
    title([filterType ' Pole-Zero Plot'])
    grid on
```

```
    % -> plot the impulse response
    if fImp == 'I'
        hn = filter( b , a , [1 zeros(1,99)] );
        figure
        stem(1:100 , hn)
        title([filterType ' Impulse Response (size 100)'])
        xlabel('n')
        ylabel('h[n]')
        grid on
    else
        hn = filter( b , a , [1 zeros(1,249)] );
        figure
        stem(1:250 , hn)
        title([filterType ' Impulse Response (size 250)'])
        xlabel('n')
        ylabel('h[n]')
        grid on
    end

end
```