Digital Filter Design Using Matlab

By Timothy J. Schlichter

EE 4000 Introduction to Digital Filtering

5/2/99

Submitted to: Dr. Joseph Picone
Mississippi State University
Department of Electrical and Computer Engineering

EXECUTIVE SUMMARY

A fundamental aspect of signal processing is filtering. Filtering involves the manipulation of the spectrum of a signal by passing or blocking certain portions of the spectrum, depending on the frequency of those portions. Filters are designed according to what kind of manipulation of the signal is required for a particular application. Digital filters are implemented using three fundamental building blocks: an adder, a multiplier, and a delay element.

The design process of a digital filter is long and tedious if done by hand. With the aid of computer programs performing filter design algorithms, designing and optimizing filters can be done relatively quickly. This paper discusses the use of Matlab, a mathematical software package, to design, manipulate, and analyze digital filters.

The design options in Matlab allow the user to either create a code for designing filters that calls built-in functions, or to design filters in Sptool, a graphical user interface. Each of these methods are examined in this paper. The strengths and weaknesses of each are detailed in the following discussion.

This paper concludes with a discussion of how the data given by Matlab for various filters can be used to implement filters on real digital signal processors. Matlab provides all the information necessary for building a hardware replica of the filter designed in software.

TABLE OF CONTENTS

Abstract

Matlab provides different options for digital filter design, which include function calls to filter algorithms and a graphical user interface called Sptool. A variety of filter design algorithms are available in Matlab for both IIR and FIR filters. This paper discusses the different options in Matlab and gives examples of lowpass, highpass, and bandpass filter designs.

Results show that the graphical user interface Sptool is a quicker and simpler option than the option of making function calls to the filter algorithms. Sptool has a more user-friendly environment since the spectrum of the filter is immediately displayed to the user, and the user can quickly zoom in and examine particular areas of interest in the spectrum (i.e. the passband). However, the shortcoming of Sptool is that it only displays the magnitude response of the filter, not the phase response.

Introduction

A key element in processing digital signals is the filter. Filters perform direct manipulations on the spectra of signals. To completely describe digital filters, three basic elements (or building blocks) are needed: an adder, a multiplier, and a delay element. The adder has two inputs and one output, and it simply adds the two inputs together. The multiplier is a gain element, and it multiplies the input signal by a constant. The delay element delays the incoming signal by one sample. Digital filters can be implemented using either a block diagram or a signal flow graph. Figure 1 shows the three basic elements in block diagram form, and Figure 2 shows them in signal flow graph form.

Figure 1: Block Diagram of Filter Elements

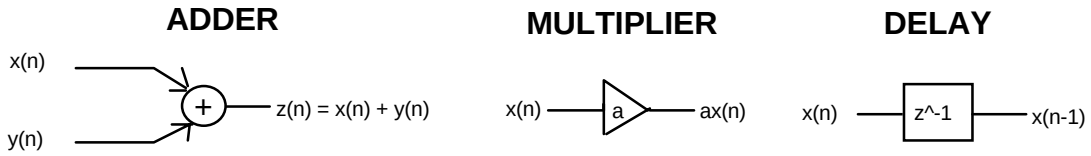
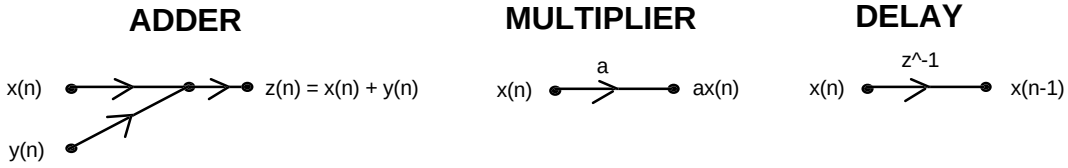


Figure 2: Signal Flow Graph of Filter Elements



With the basic building blocks at hand, the two different filter structures can easily be implemented. These two structures are Infinite Impulse Response (IIR) and Finite Impulse Response (FIR), depending on the form of the system's response to a unit pulse input. IIR filters are commonly implemented using a feedback (recursive) structure, while FIR filters usually require no feedback (non-recursive).

In the design of IIR filters, a commonly used approach is called the bilinear transformation. This design begins with the transfer function of an analog filter, then performs a mapping from the s-domain to the z-domain. Using differential equations, it can be shown (Proakis 677) that the mapping from the s-plane to the z-plane is

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

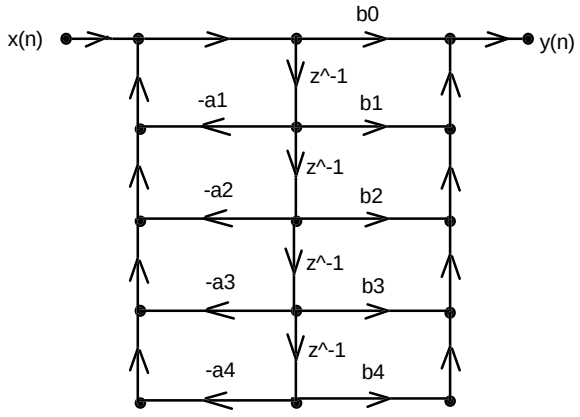
This mapping results in a general form for an IIR filter with an arbitrary number of poles and zeros. The system response and the difference equation for this filter is as follows:

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{n=0}^{M} b_n z^{-n}}{\sum_{n=0}^{N} a_n z^{-n}} = \frac{b_0 + b_1 z^{-1} + \ldots\ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots\ldots + a_N z^{-N}}, a_0 = 1 \qquad \text{(Ingle 183)}$$

$$y(n) = \sum_{m=0}^{M} b_m x(n-m) - \sum_{n=0}^{N} a_m y(n-m)$$

This system response can be easily realized using a signal flow graph.

Figure 3: Signal Flow Graph of IIR Filter



An FIR filter has a difference equation of

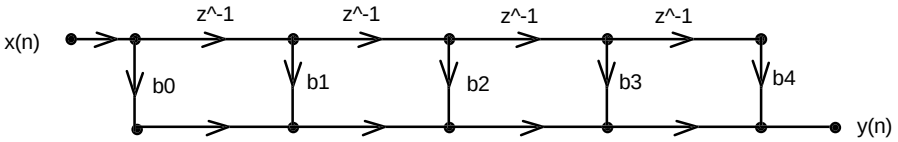
$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k) \qquad \text{(Proakis 620)}$$

By taking the z-transform, the system response is

$$H(z) = b_0 + b_1 z^{-1} + \ldots\ldots + b_{M-1} z^{1-M} = \sum_{k=0}^{M-1} b_k z^{-k} \qquad \text{(Ingle 197)}$$

The realization of an FIR filter using a signal flow graph is straightforward.

Figure 4: Signal Flow Graph of FIR Filter



Matlab has several design algorithms that can be used to create and analyze both IIR and FIR digital filters. The IIR filters that can be created in Matlab are Butterworth, Chebyshev type 1 and 2, and elliptic. The FIR filter algorithms in Matlab are equiripple, least squares, and Kaiser window. The Matlab code required to implement these filters involves bilinear transformations and function calls to analog prototype filters. The following sections give examples of Matlab implementation of the IIR filters listed above.

Lowpass Filter Design

Using Matlab, a lowpass digital filter is designed using various analog prototypes: Chebyshev, Butterworth, and Elliptic. The optimum filter type is chosen on the basis of implementation complexity, magnitude response, and phase response. The design specifications for the filter are as follows:

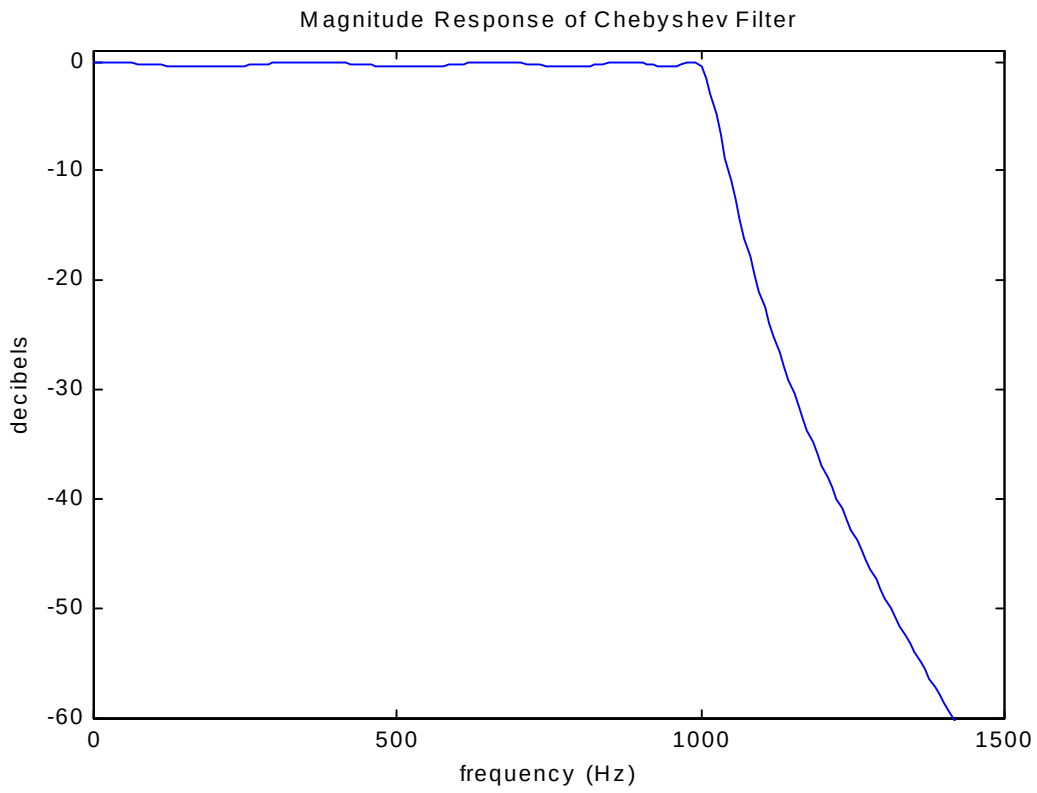
- Cutoff frequency = 1000Hz
- Sample frequency = 8000Hz
- Passband ripple = 0.5dB
- Stopband attn. = 60dB
- Transition band = 100Hz

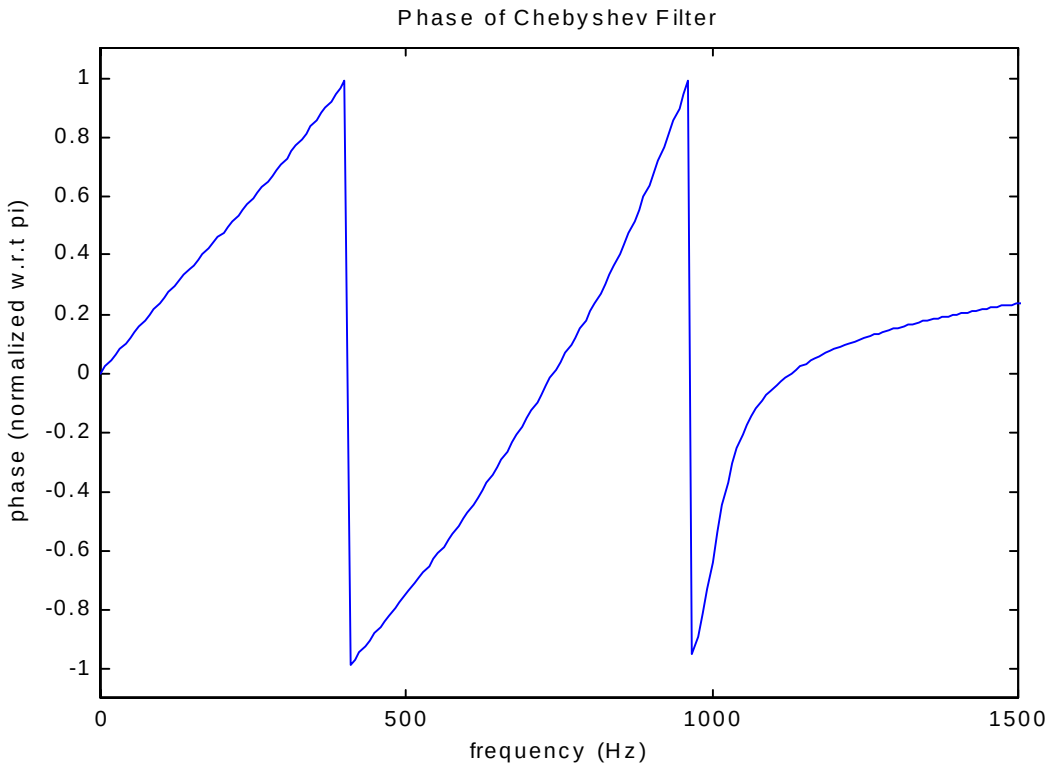
Matlab Code (Chebyshev):

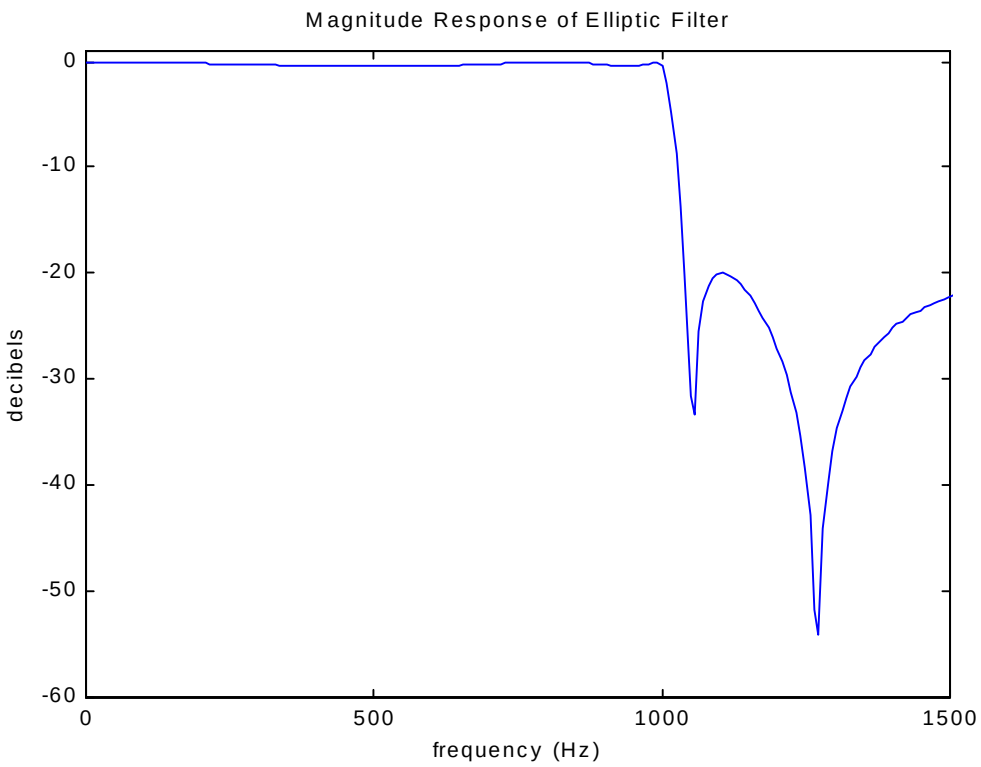
```
% Lowpass digital filter with Chebyshev-I analog prototype
%
% Digital Filter Specifications:
wp = 0.125*2*pi;  % digital passband frequency in Hz (normalized)
ws = 0.1375*2*pi; % digital stopband frequency in Hz (normalized)
Rp = 0.5;         % passband ripple in dB
As = 20;          % stopband attenuation in dB

% Analog Prototype Specifications:
Fs = 1; T = 1/Fs;
OmegaP = (2/T)*tan(wp/2);  % prewarp prototype passband frequency
OmegaS = (2/T)*tan(ws/2);  % prewarp prototype stopband frequency

% Analog Chebyshev-1 Prototype Filter Calculation:
[c, d] = chb1(OmegaP, OmegaS, Rp, As);

% Bilinear Transformation:
[b, a] = bilinear(cs, ds, Fs);
%
[db,mag,pha,grd,w] = freqz(b,a);
plot(w*8000/2/pi,db);
xlabel('frequency (Hz)'); ylabel('decibels'); title('Magnitude in
dB');
```

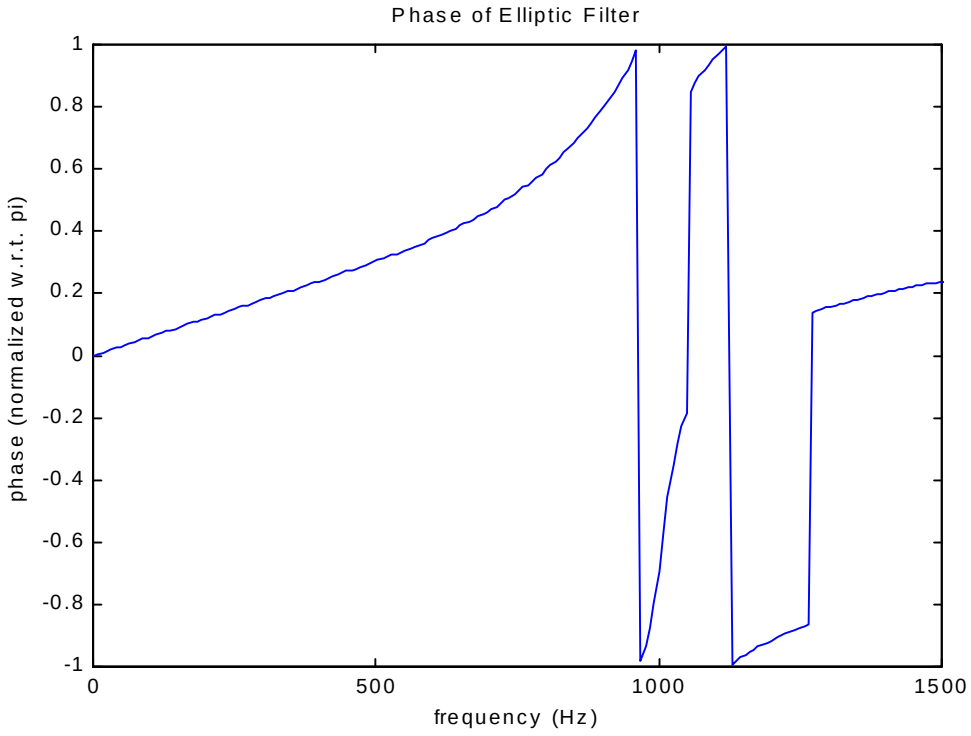
This exact code is also used for the elliptic and Butterworth designs. The only change is in the filter calculations of each type. Instead of calling **chb1**(), the elliptic filter design calls a function "**elliptic**()" and the Butterworth design calls a function "**butterworth**()". See the appendix for the Matlab code of the function **chb1**().

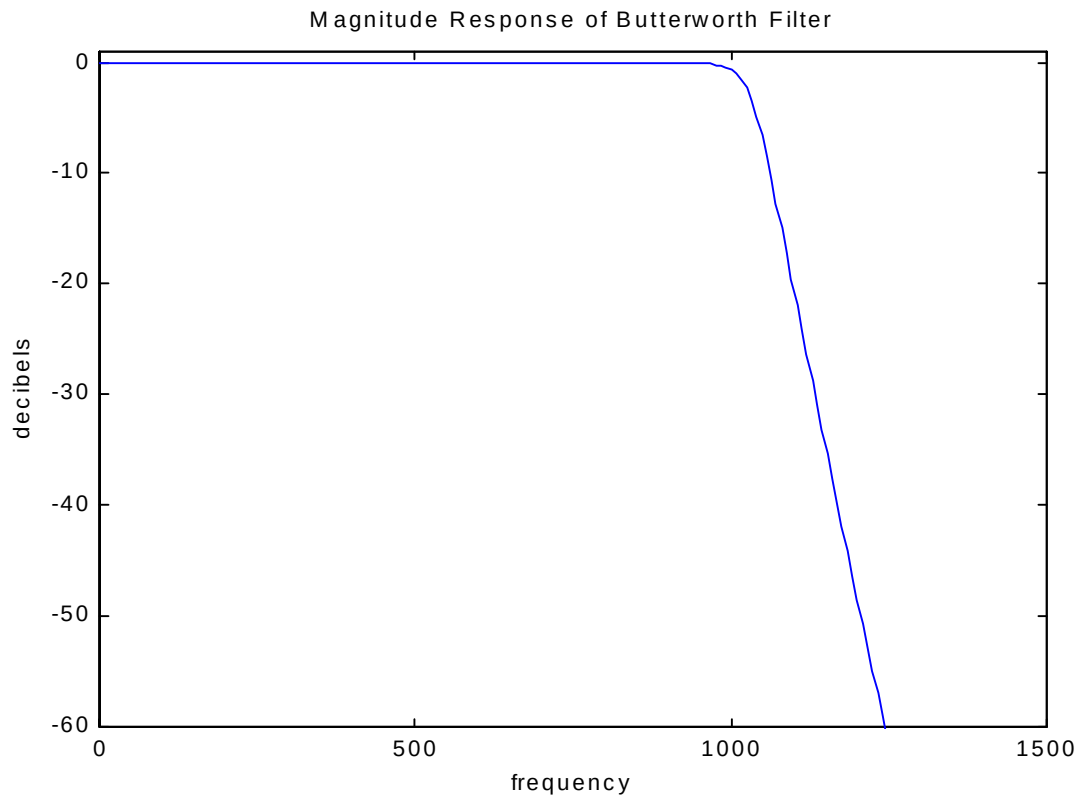
The following figures show the magnitude and phase responses of each type of filter.

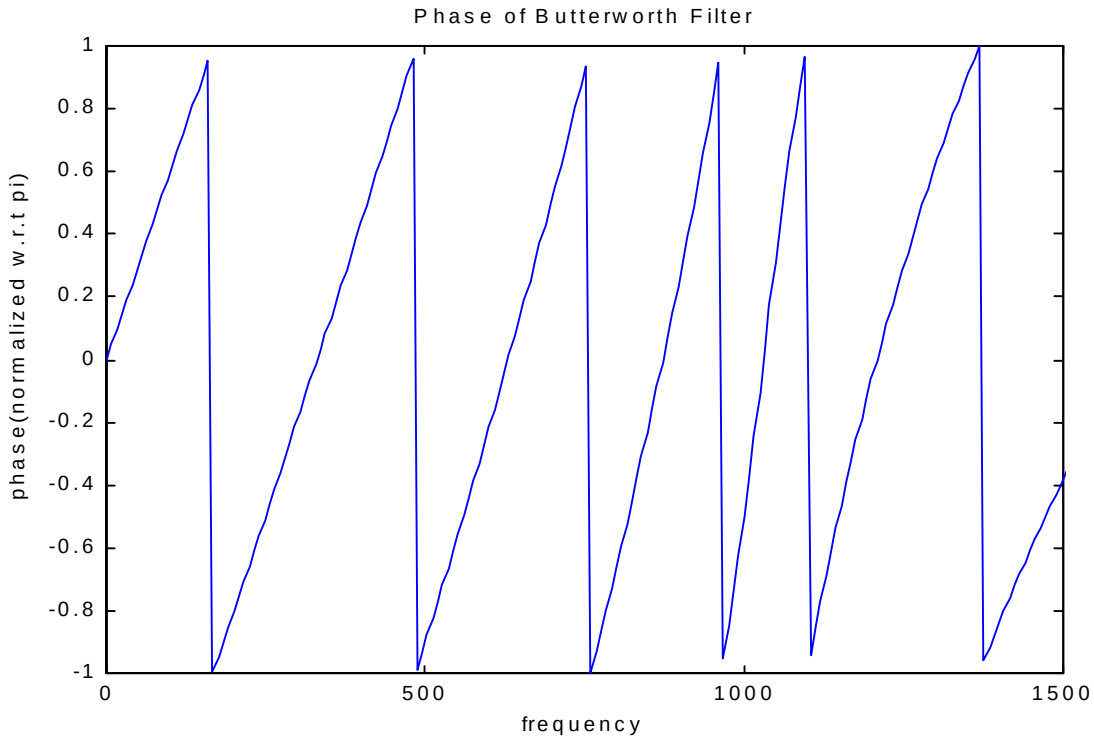

Magnitude Response of Chebyshev Filter
0
-10
-20
-30
-40
-50
-60
decibels
0
500
1000
1500
frequency (Hz)


Phase of Chebyshev Filter
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
phase (normalized w.r.t pi)
0
500
1000
1500
frequency (Hz)


Magnitude Response of Elliptic Filter
0
-10
-20
-30
-40
-50
-60
decibels
0
500
1000
1500
frequency (Hz)


Phase of Elliptic Filter
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
phase (normalized w.r.t. pi)
0
500
1000
1500
frequency (Hz)


Magnitude Response of Butterworth Filter
0
-10
-20
-30
-40
-50
-60
decibels
0
500
1000
1500
frequency


Phase of Butterworth Filter
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
phase(normalized w.r.t pi)
0
500
1000
1500
frequency

The Matlab code outputs the filter order and the filter coefficients. For this example, the Chebyshev filter order was nine. The elliptic filter had an order of five, and the Butterworth filter order was thirty-two.

Several conclusions can be drawn about these low-pass filter designs from this simple example. First, in general, for a given set of design constraints, the elliptic filter design algorithm will result in the simplest filter (in terms of complexity). The most complex filter is the Butterworth filter with an order of thirty-two. In terms of passband ripple, the Butterworth filter gives the optimum response. In the passband, there is almost no ripple (monotonic). The elliptic and Chebyshev filters both have much more ripple in the passband. So, there is a tradeoff between these three different types of filters. In terms of magnitude response and complexity, the elliptic ripple is most likely the optimum choice. However, the elliptic ripple has a phase response that is more nonlinear than the Chebyshev and Butterworth filters. Therefore, if a sharp cutoff and relatively low complexity is required, the choice would be the elliptic filter. If the phase response would need to be more linear, a Chebyshev or Butterworth filter should be chosen over the elliptic filter.

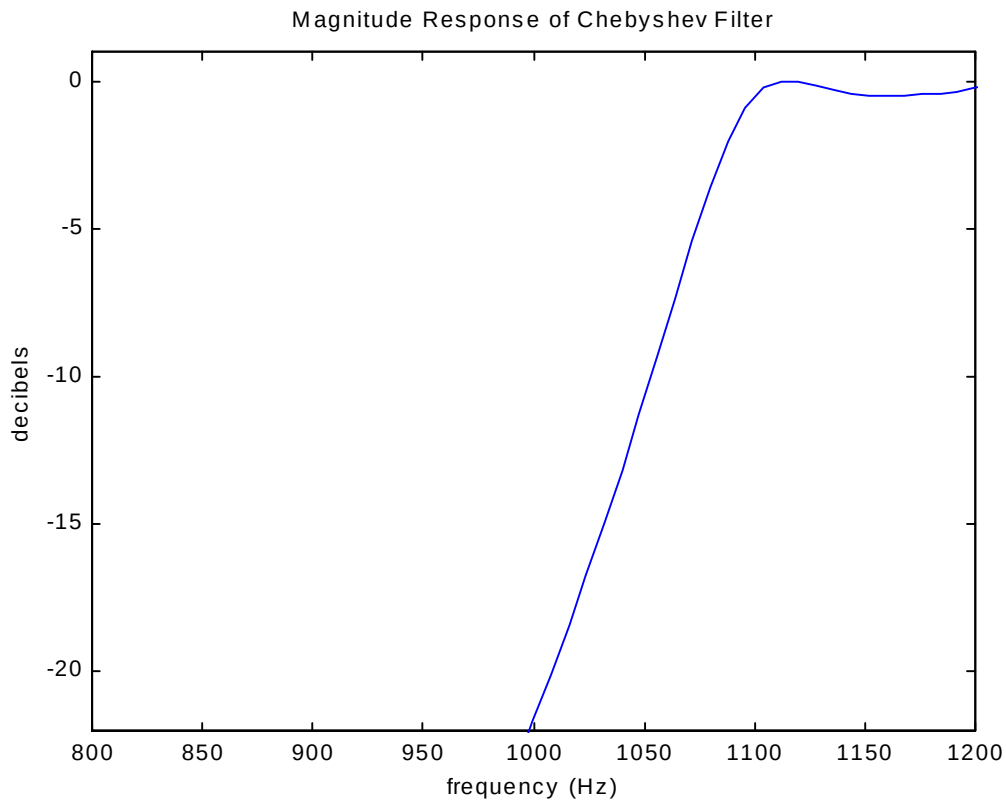
Highpass and Bandpass Filter Design

Matlab provides functions for implementing lowpass-to-highpass and lowpass-to-bandpass conversions. By providing a filter order, the passband ripple, and the 3dB cutoff frequency to the function **cheby1**(), a highpass filter can be designed. The filter order is found using the function **chebord**(). For a Butterworth prototype, the functions are **butter**() and **buttord**(). For the elliptic prototype, the functions are **ellip**() and **ellipord**().

The following Matlab code is used to design a Chebyshev highpass digital filter with a passband at 1100Hz and a 100Hz transition band.

```
% Highpass Chebyshev Digital Filter

ws = 0.125*2*pi;     % digital stopband frequency in rad/s
wp = 0.1375*2*pi;    % digital passband frequency in rad/s
Rp = 0.5;            % passband ripple in dB
As = 20;

[N,wn] = cheb1ord(wp/pi,ws/pi,Rp,As);
[b,a] = cheby1(N, Rp, wn, 'high');

[db,mag,pha,grd,w] = freqz_m(b,a);
plot(w*8000/2/pi,db);
axis([800 1200 -22 1]);
```

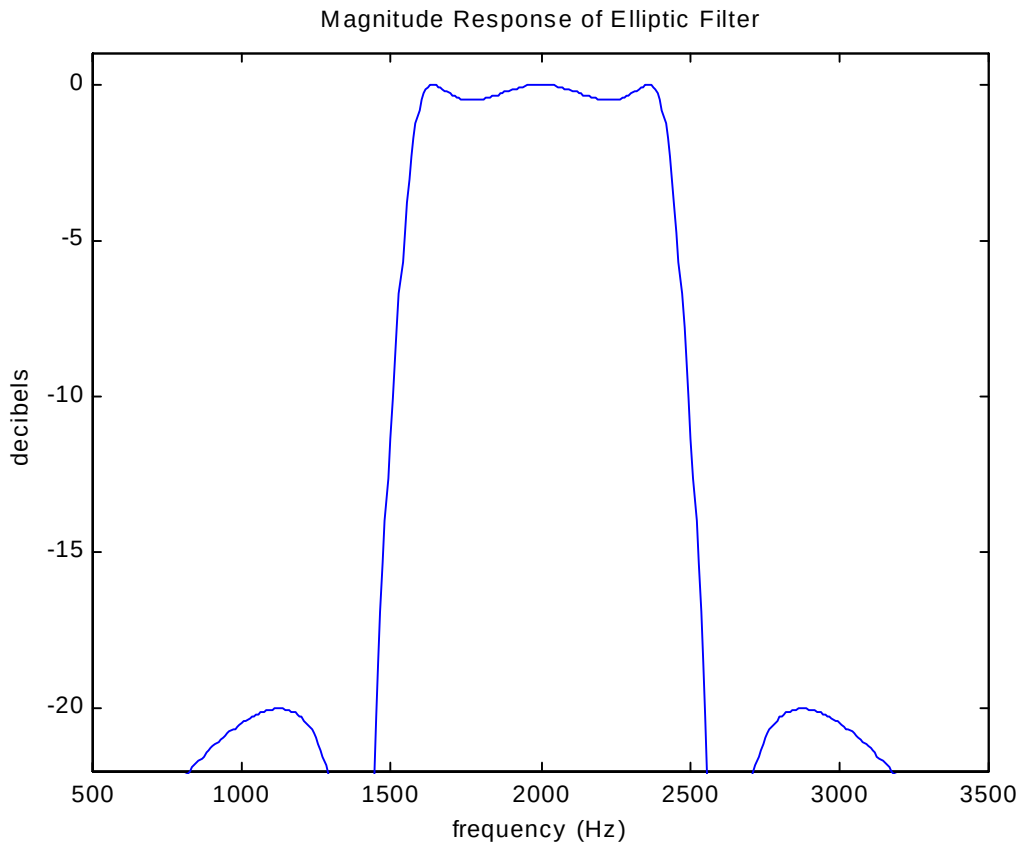
The following figure shows the magnitude response of the highpass filter.



Bandpass filters are found using these same two functions. However, with bandpass filters, the passband and stopband frequencies (w_p and w_s) are two-element vectors since there are two passband frequencies and two stopband frequencies. The Matlab code below shows the design of an elliptic digital bandpass filter.

```
% Bandpass Elliptic Digital Filter

ws = [0.3*pi 0.75*pi]            %Stopband edge frequency
wp = [0.4*pi 0.6*pi]             %Passband edge frequency
Rp = 0.5;                        %Passband ripple in dB
As = 20;                         %Stopband attenuation in dB

[N,wn] = ellipord(wp/pi,ws/pi,Rp,As);
[b,a] = ellip(N,Rp,As,wn);

[db,mag,pha,grd,w] = freqz_m(b,a);
plot(w*8000/2/pi,db);
axis([500 3500 -22 1]);
xlabel('frequency (Hz)'); ylabel('decibels'); title('Magnitude
Response of Elliptic Filter');
```

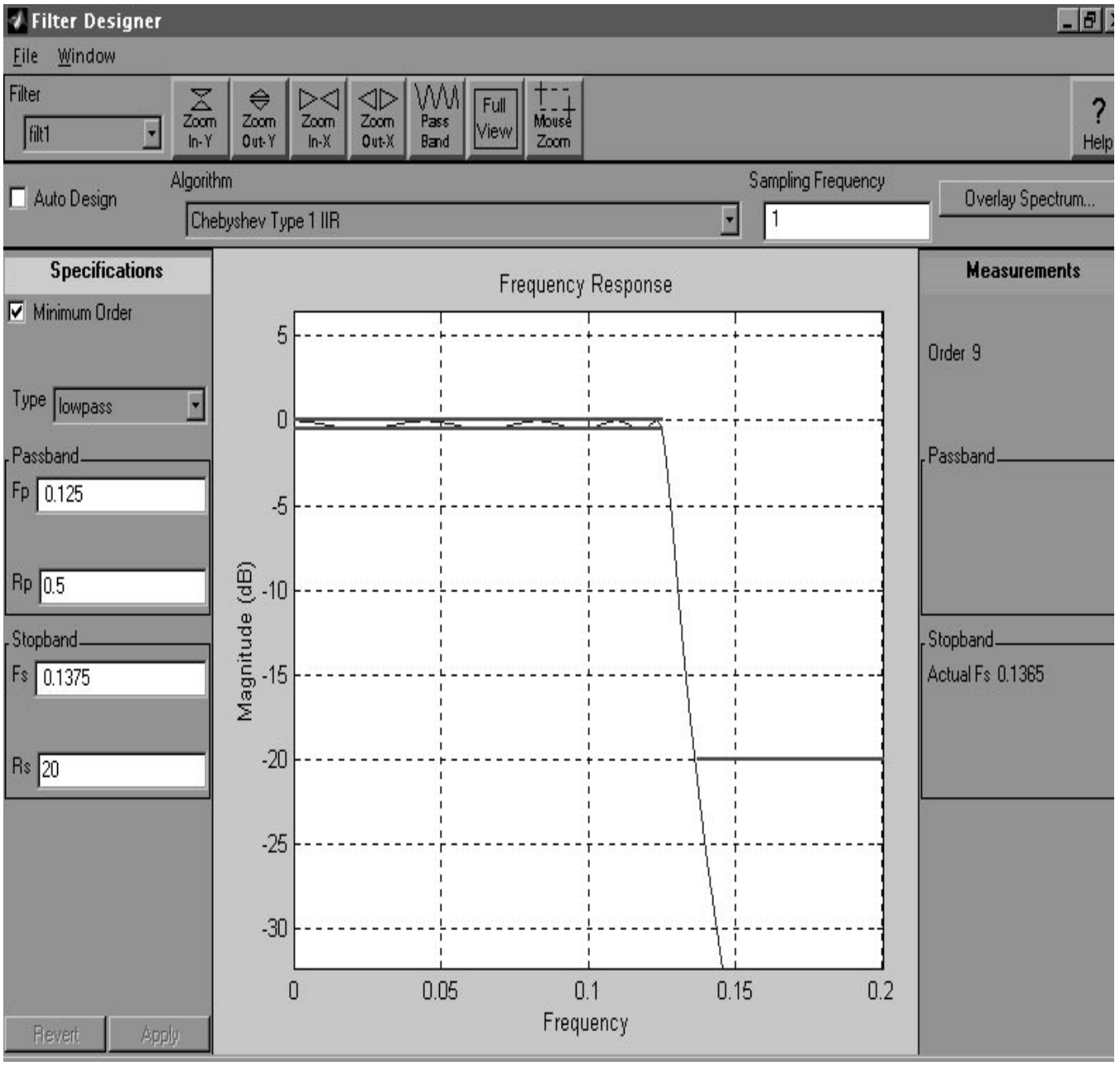
The following figure shows the magnitude response of the bandpass filter designed in the Matlab code above.

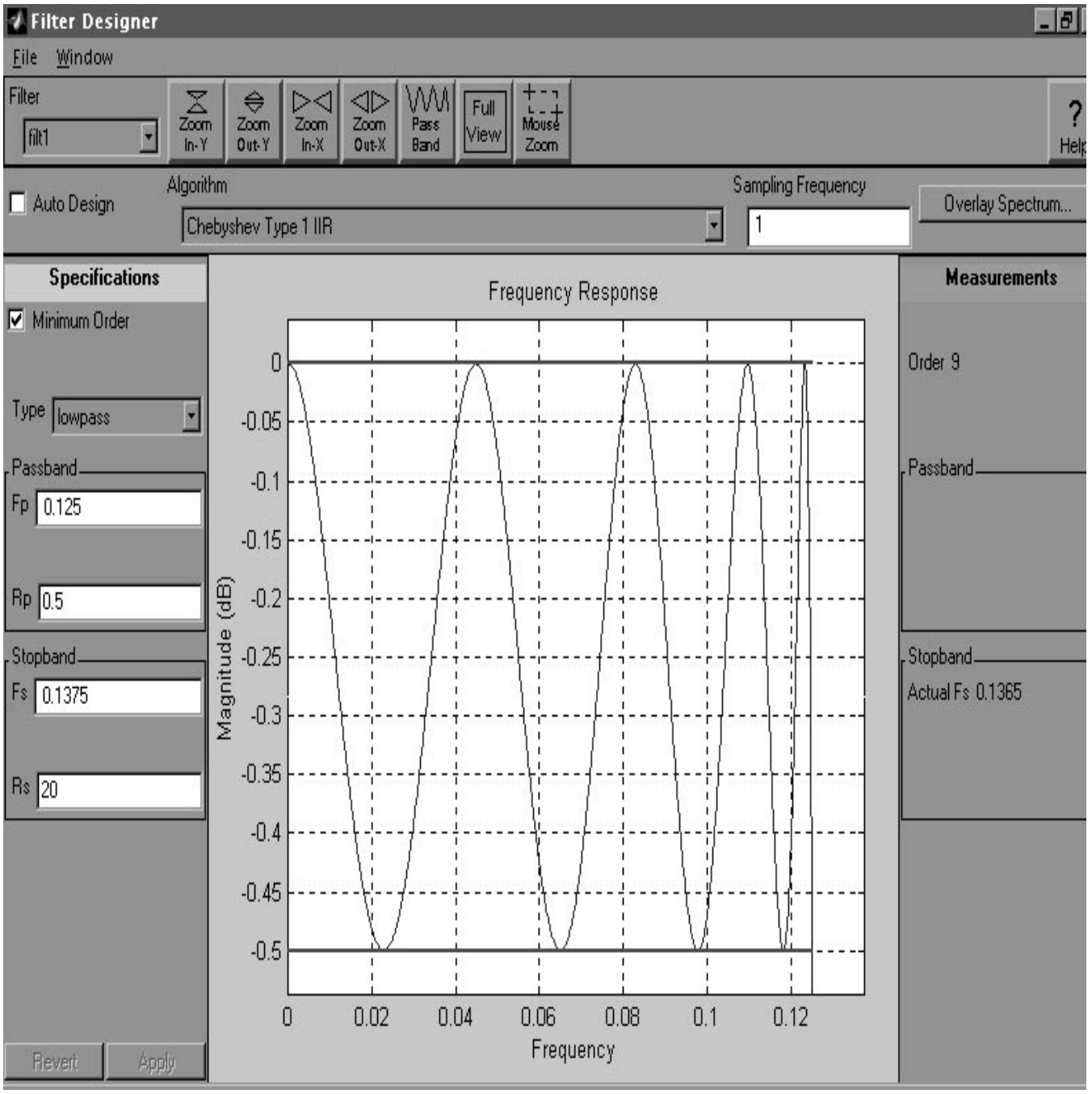


Sptool

Matlab has a very useful visualization tool for designing and analyzing digital filters called Signal Processing Tool, or Sptool. Sptool is a graphical user interface capable of analyzing and manipulating signals, filters, and spectra. For filter design, Sptool allows the user to select the filter design algorithm to use when creating the filter. The design algorithms included in Sptool are FIR filters (equiripple, least squares, Kaiser window) and IIR filters (Butterworth, Chebyshev type 1 and 2, elliptic). The user is also allowed to specify the type of filter (lowpass, bandpass, highpass, or bandstop). Sptool designs the filter and displays the magnitude response and the filter order.

The figures below show actual Sptool screenshots for a lowpass filter design using the specifications given above. The Chebyshev Type 1 algorithm was used for these screenshots. By using the zoom options, different parts of the spectrum can be analyzed quickly and easily with Sptool. The second screenshot is a windowed view of the passband of the spectrum contained in the first screenshot.


Filter Designer
File Window
Filter
filt1
Zoom In-Y
Zoom Out-Y
Zoom In-X
Zoom Out-X
Pass Band
Full View
Mouse Zoom
Help
Auto Design
Algorithm
Chebyshev Type 1 IIR
Sampling Frequency
1
Overlay Spectrum...
Specifications
Minimum Order
Type
lowpass
Passband
Fp
0.125
Rp
0.5
Stopband
Fs
0.1375
Rs
20
Revert
Apply
Frequency Response
Magnitude (dB)
5
0
-5
-10
-15
-20
-25
-30
0
0.05
0.1
0.15
0.2
Frequency
Measurements
Order 9
Passband
Stopband
Actual Fs 0.1365


Filter Designer
File Window
Filter
filt1
Zoom In-Y
Zoom Out-Y
Zoom In-X
Zoom Out-X
Pass Band
Full View
Mouse Zoom
Help
Auto Design
Algorithm
Chebyshev Type 1 IIR
Sampling Frequency
1
Overlay Spectrum...
Specifications
Minimum Order
Type
lowpass
Passband
Fp
0.125
Rp
0.5
Stopband
Fs
0.1375
Rs
20
Revert
Apply
Frequency Response
Magnitude (dB)
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
-0.45
-0.5
0
0.02
0.04
0.06
0.08
0.1
0.12
Frequency
Measurements
Order 9
Passband
Stopband
Actual Fs 0.1365

Future Directions

Digital filters can be quickly and easily designed in Matlab using the methods described above. Sptool offers a much quicker way of designing and comparing different filters than using the function calls to the filter algorithms. However, Sptool allows the user to view the magnitude response of the filter, not the phase response. For some applications, the phase response may be more important than the magnitude response, so in these cases Sptool would not be as useful in determining the optimum filter design. Also, Sptool does not give a direct output of the filter coefficients. With the Matlab code given above, the filter coefficient are displayed to the user.

The results from Matlab can be used directly to implement the digital filters on real DSPs. These results are all that is needed to draw a complete signal flow graph with adders, multipliers, and delay elements. The filter coefficients are used as the gain factors for the multipliers, and shift registers are used as the delay elements (for each z^{-n} factor).

Acknowledgments

Many thanks to Dr. Joseph Picone for his guidance, assistance, and time in the execution of this project.

References

- Hanselman, Duane, and Littlefield, Bruce. Mastering Matlab 5. Prentice Hall. Upper Saddle River, NJ, 1998.

- Ingle, Vinay K. and Proakis, John G. Digital Signal Processing Using Matlab. PWS Publishing Company, 1997.

- Proakis, John G. and Manolakis, Dimitris G. Digital Signal Processing: Principles, Algorithms, and Applications, 3rd Edition. Prentice Hall. Upper Saddle River, NJ, 1996.

- Ziemer, Rodger E., Tranter, William H., and Fannin, D. Ronald. Signals and Systems: Continuous and Discrete, 3rd Edition. Macmillan Publishing Company, 1993.

Appendix

```
function [b,a] = chb1(Wp, Ws, Rp, As);
% Analog Lowpass Filter Design:  Chebyshev-1
%
% [b,a] = chb1(Wp, Ws, Rp, As);
%  b = Numerator coefficients of Ha(s)
%  a = Denominator coefficients of Ha(s)
%  Wp = Passband edge frequency in rad/sec
%  Ws = Stopband edge frequency in rad/sec
%  Rp = Passband ripple in dB
%  As = Stopband attenuation in dB
%
if Wp <= 0
   error('Passband edge must be larger than 0')
end
if Ws <= Wp
   error('Stopband edge must be larger than Passband edge')
end
if (Rp <= 0) | (As < 0)
   error('PB ripple and/or SB attenuation must be larger than 0')
end
ep = sqrt(10^(Rp/10)-1);
A = 10^(As/20);
OmegaC = Wp;
OmegaR = Ws/Wp;
g = sqrt(A*A-1)/ep;
N = ceil(log10(g+sqrt(g*g-1))/log10(OmegaR+sqrt(OmegaR*OmegaR-1)));
fprintf('\n*** Chebyshev-1 Filter Order = %2.0f \n',N);
[b,a] = ap_chb1(N, Rp, OmegaC);


function [b,a] = ap_chb1(N, Rp, Omegac);
% Chebyshev-1 Analog Lowpass Filter Prototype
%
% [b,a] = ap_chb1(N, Rp, Omegac);
%  b = nemerator polynomial coefficients
%  a = denominator polynomial coefficients
%  N = Order of the elliptical filter
%  Rp = Passband Ripple in dB
%  Omegac = cutoff frequency in rad/sec
%
[z,p,k] = cheb1ap(N,Rp);
   a = real(poly(p));
   aNn = a(N+1);
   p = p*Omegac;
   a = real(poly(p));
   aNu = a(N+1);
   k = k*aNu/aNn;
   b0 = k;
   B = real(poly(z));
   b = k*B;
```