2.161 Signal Processing: Continuous and Discrete

Fall 2008

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING

2.161 Signal Processing - Continuous and Discrete
Fall Term 2008

Solution of Problem Set 8: FIR Linear Filters

Assigned: November 6, 2008 **Due:** November 18, 2008

Problem 1:
The standard Hilbert transformer is defined as:

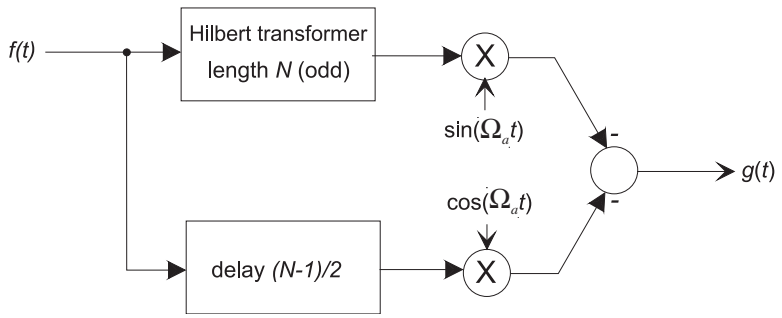
$$H(j\Omega) = \begin{cases} -j & \Omega > 0 \\ +j & \Omega < 0 \end{cases}$$

MATLAB uses the same definition and this can be verified by finding the phase of $\omega \to 0^+$. The implemented filter's phase has a linear component added to the $-\pi/2$ value (for $\omega > 0$).

Our filter has $-\pi/2$ phase shift and this corresponds to:

$$\begin{aligned} A\sin\left((\Omega_a - \Omega)\,t\right) &= A\left(\sin(\Omega_a t)\cos(\Omega t) - \cos(\Omega_a t)\sin(\Omega t)\right) \\ &= -A\sin(\Omega_a t)\sin(\Omega t - \pi/2) - A\cos(\Omega_a t)\sin(\Omega t) \end{aligned}$$

We have to be careful in implementing the scrambler. The scrambler diagram corresponds to the below figure. However, note that the audio play is not affected by sign of output (i.e. a signal is played the same as its negative version).



(a) The delay in implementation does not affect this discussion. So we ignore it for this analysis. We consider a sinusoidal component of input as $f(t) = \sin(\Omega_o t)$. The output of filter would be $|H(j\Omega_o)|\sin(\Omega_o t + \angle H(j\Omega_o)) = (1+\delta)\sin(\Omega_o t - \frac{\pi}{2})$. We follow this through above diagram to compute $g(t)$:

$$\begin{aligned} g(t) &= -(1+\delta)\sin(\Omega_o t - \pi/2)\sin(\Omega_a t) - \cos(\Omega_a t)\sin(\Omega_o t) \\ &= +(1+\delta)\cos(\Omega_o t)\sin(\Omega_a t) - \cos(\Omega_a t)\sin(\Omega_o t) \\ &= \sin((\Omega_a - \Omega_o)t) + \delta\sin(\Omega_a t)\cos(\Omega_o t) \\ &= \sin((\Omega_a - \Omega_o)t) + \frac{\delta}{2}(\sin((\Omega_a - \Omega_o)t) + \sin((\Omega_a + \Omega_o)t)) \end{aligned}$$

$$= \quad (1+\frac{\delta}{2})\sin((\Omega_a-\Omega_o)t)+\frac{\delta}{2}\sin((\Omega_a+\Omega_o)t)$$

As a result due to practical implementation, some "spurious" components are introduced at $\Omega_a+\Omega_o$.

The original signal has a (desired) audio frequency content of $\Omega_o = 300-3000$ Hz. In the beginning, we can pre-process the audio file with a band-pass filter passing $300-3000$ Hz contents and then follow it with the scrambling process.
We use a Hilbert transformer with a pass-band equal to 300 to $(F_s/2-300)$ Hz, where F_s is the sampling frequency . The extended pass-band of the filter, is because a high-order (low-ripple) "firpm-built" Hilbert filter requires a symmetric pass-band (see MATLAB documentation). After passing the audio file through our transformer with $\Omega_a = 3500$ Hz, the output will have a large (desired) spectrum content at $500-3200$ Hz and a spurious content at $3800-6500$ Hz (which could be folded to some lower frequency). If δ is not small enough, we might wish to post-process the filter output with a pass-band filter passing $500-3200$ Hz contents and then save it as scrambled signal. This post processing step of scrambling, can be considered as a pre-processing step of desrcambling procedure.
Theoretically, descrambling is the same as scramblibg. However, the practical realization of the Hilbert filter is bandwidth limited (while the ideal Hilbert filter is an all-pass filter). The Hilbert Filter for desrcambling, requires a pass-band equal to 500 to $(F_s/2-500)$ Hz. However, if we pre-process the srcambled signal with a pass-band filter passing $500-3200$ Hz contents, then we can use the same Hilbert filter for scrambling and descrambling. The output of desrcambling can be further post-processed with a pass-band filter passing audio contents $(300-300$ Hz).

(b) This is not the complete solution to all parts of the problem. This simply serves to show the approach. To study the full solution start with attached Main.m file and follow other scripts. Note that the encoding and decoding functions can be identical (if we use proper pre/post filters). Here is the encoding script:

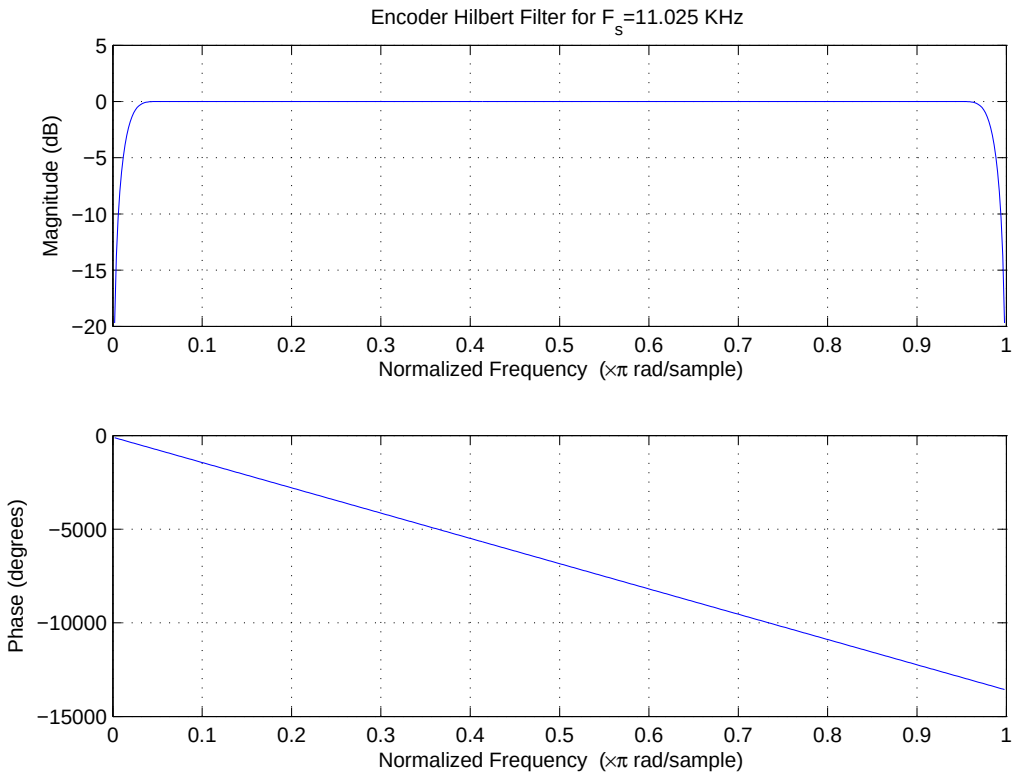
```
function fout = encode(fin,Fs,Fc)

%% Design the Hilber transformer
% Note:  firpm() seems to need the band edges to be symmetric about 0.5
h1=firpm(150,[300*2/Fs 1-300*2/Fs],[1 1 ],'Hilbert');
% h1=firpm(150,[500*2/Fs 1-500*2/Fs],[1 1 ],'Hilbert'); % change in the decode file
y1=fftfilt(h1,fin);

%% Delay: Approach 1
N=length(h1); %N is odd
y2=zeros(size(fin));y2((N-1)/2+1:end)=fin(1:end-(N-1)/2);
%% Delay: Design the delay filter, Approach 2
% h2 = zeros(1,length(h1));
% h2((N-1)/2+1) = 1;
% Filter the data
% y2=fftfilt(h2,fin);
```

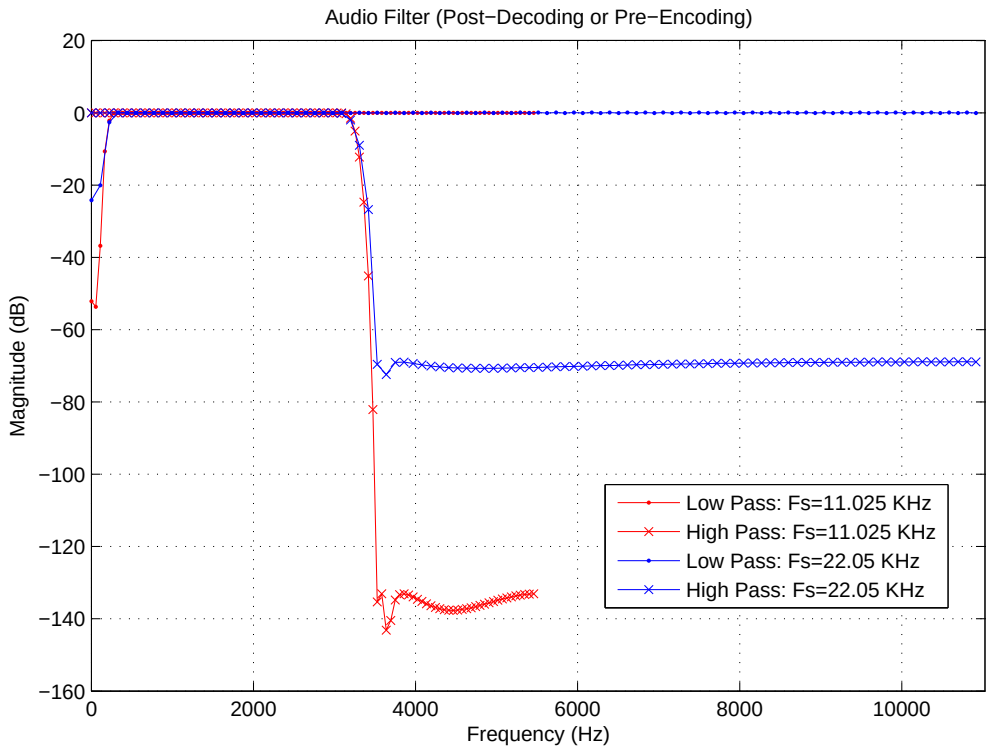
```
%% Filter Output
t=(0:length(fin)-1)*(1/Fs);
y3=-y1.*sin(2*pi*Fc*t');
y4=-y2.*cos(2*pi*Fc*t');
fout= y3+y4;
```

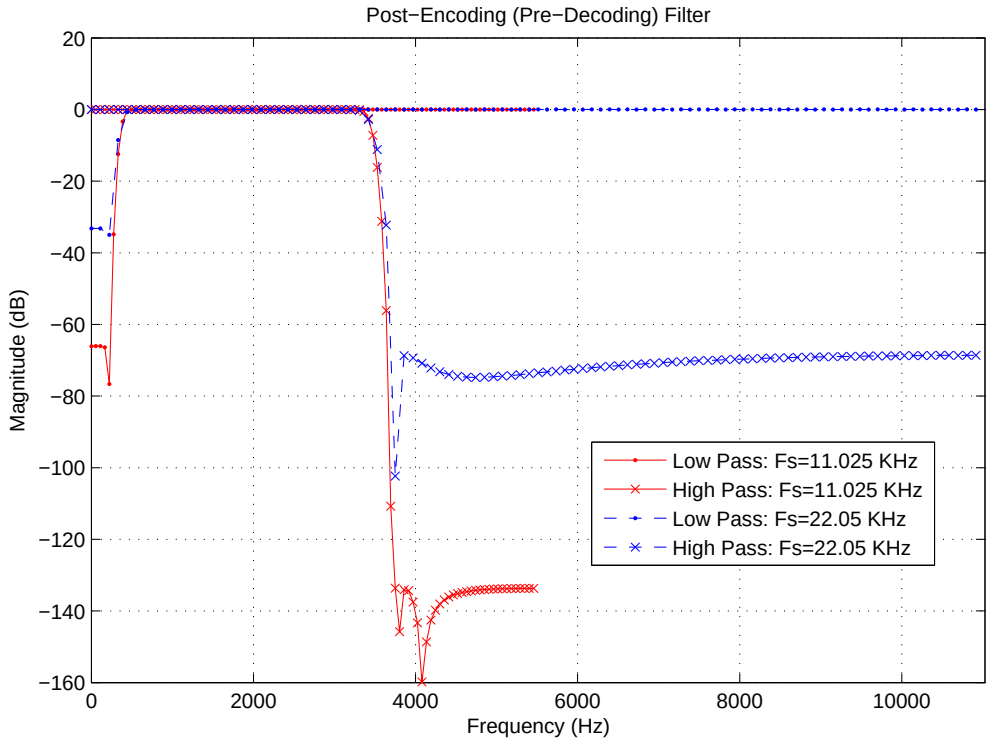
The following graph shows one the encoder's Hilbert transformer (order 150).



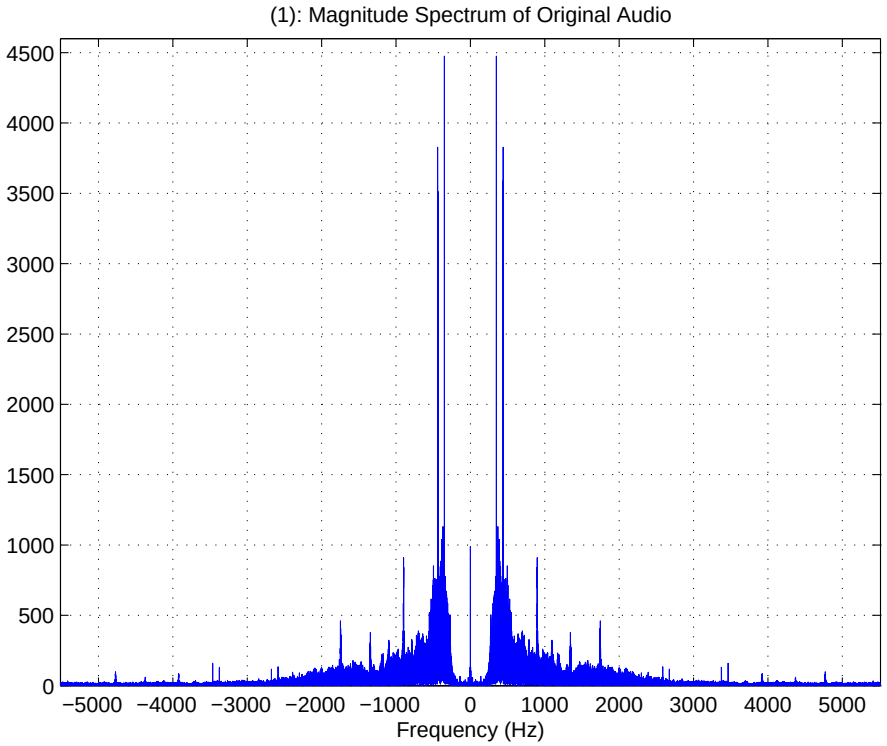
We passed the input original audio through a band-pass filter (300-3000 Hz) to minimize the folding of high frequencies. We used the same filter for the decoder output. We also used a band pass filter (500-3200 Hz) for the encoder output or for the decoder input. These pass-band filters are implemented as a series of low and high pass filters (all order 200). Otherwise, we could not build a nice pass band filter in a single step. All these filters are shown on the next page

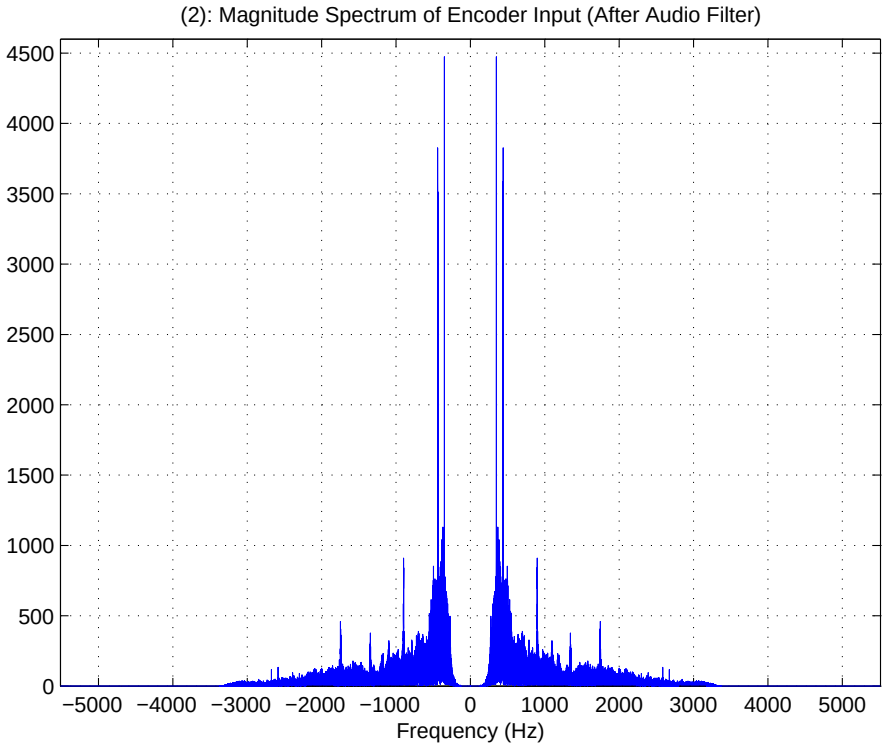
Note that following the original signal through all filters (audio filter, Hilbert, pre-decode, Hilbert and audio filter), results in a delay equal to 750 samples ($3 * (200/2 + 200/2) + 2 * (150/2)$). Although 750 might be a large number, but even for our lowest $F_s = 11.02$ KHz, this means less then 0.07 second delay. This small delay is hardly noticeable by our hearing system and hence our filter's orders are not too large.

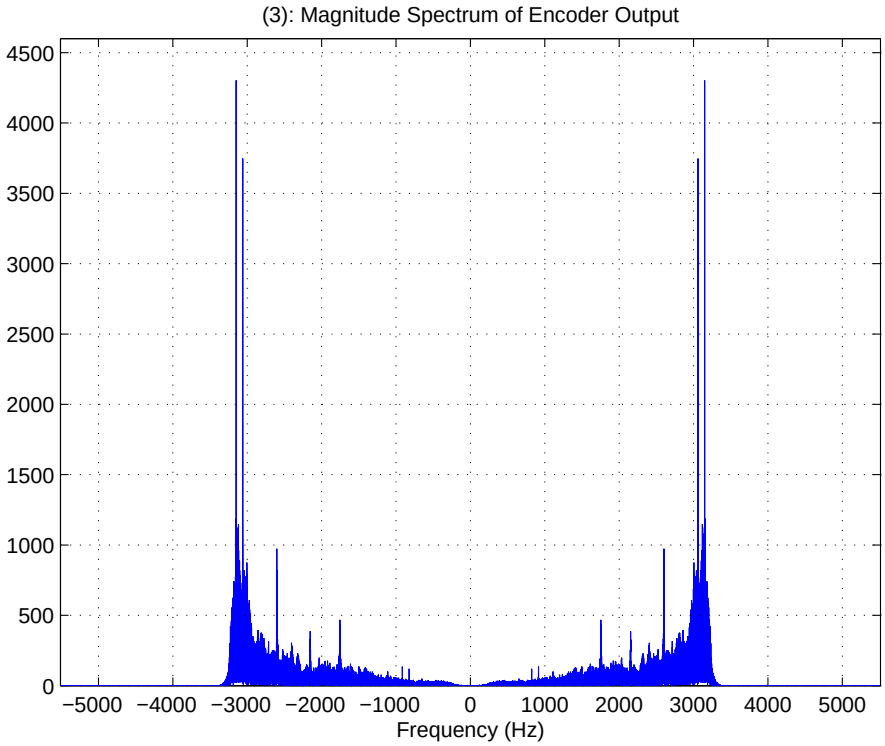


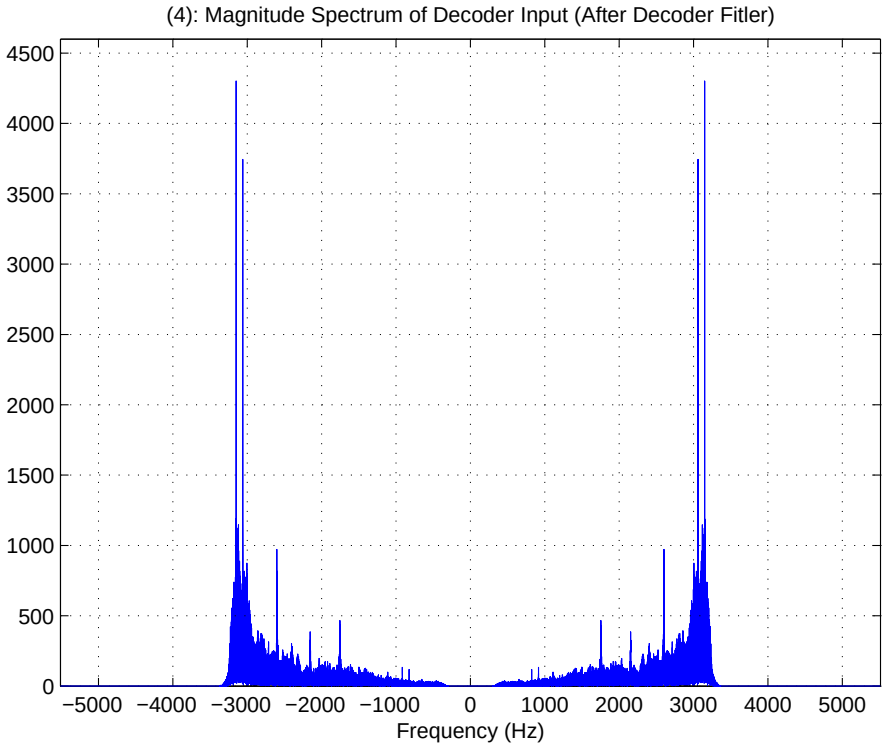


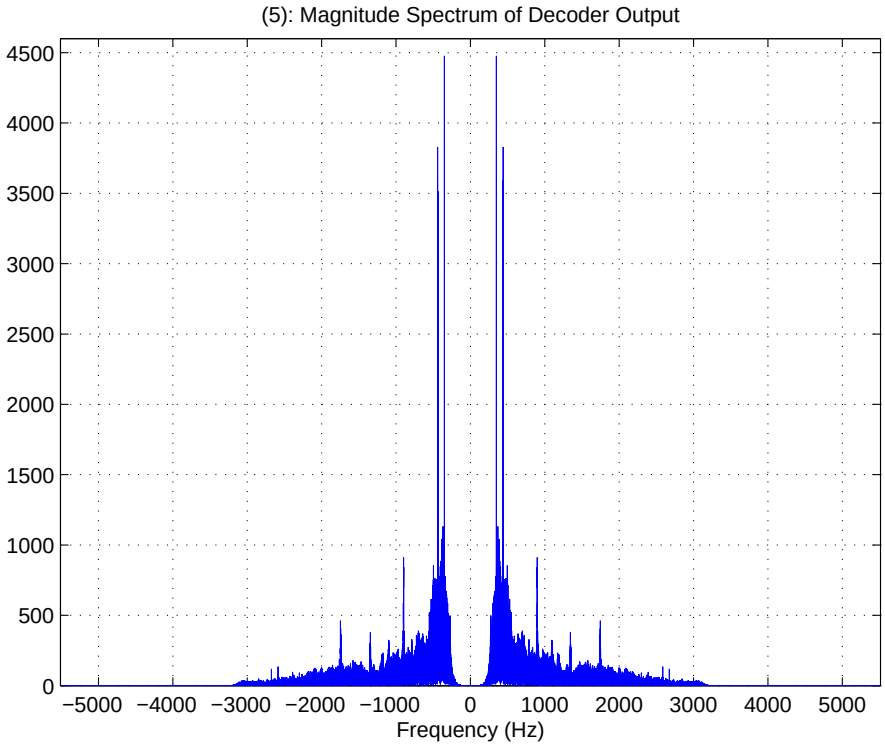
The following plots resulted from working on PS8Raw.wav:

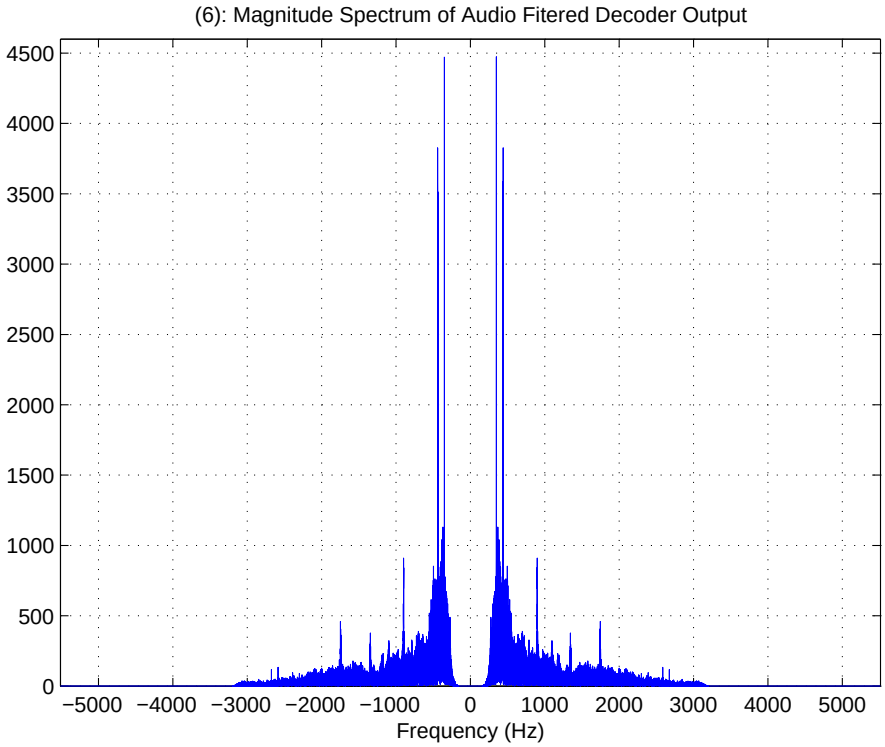

(1): Magnitude Spectrum of Original Audio
4500
4000
3500
3000
2500
2000
1500
1000
500
0
−5000 −4000 −3000 −2000 −1000 0 1000 2000 3000 4000 5000
Frequency (Hz)


(2): Magnitude Spectrum of Encoder Input (After Audio Filter)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
−5000 −4000 −3000 −2000 −1000 0 1000 2000 3000 4000 5000
Frequency (Hz)


(3): Magnitude Spectrum of Encoder Output
4500
4000
3500
3000
2500
2000
1500
1000
500
0
−5000 −4000 −3000 −2000 −1000 0 1000 2000 3000 4000 5000
Frequency (Hz)


(4): Magnitude Spectrum of Decoder Input (After Decoder Fitler)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
−5000 −4000 −3000 −2000 −1000 0 1000 2000 3000 4000 5000
Frequency (Hz)





Comparing the input and decoded spectra shows them to be the same. In your solutions, we will be looking for stray spectral components that may result from ripple in your Hilbert transformer response etc.

Problem 2:

(a) If $H(s) = 1/s$, then the bilinear transform (using Tustin's approximation) gives:

$$H(z) = \frac{1}{\frac{2}{T}\frac{z-1}{z+1}} = \frac{T}{2}\frac{z+1}{z-1} = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}$$

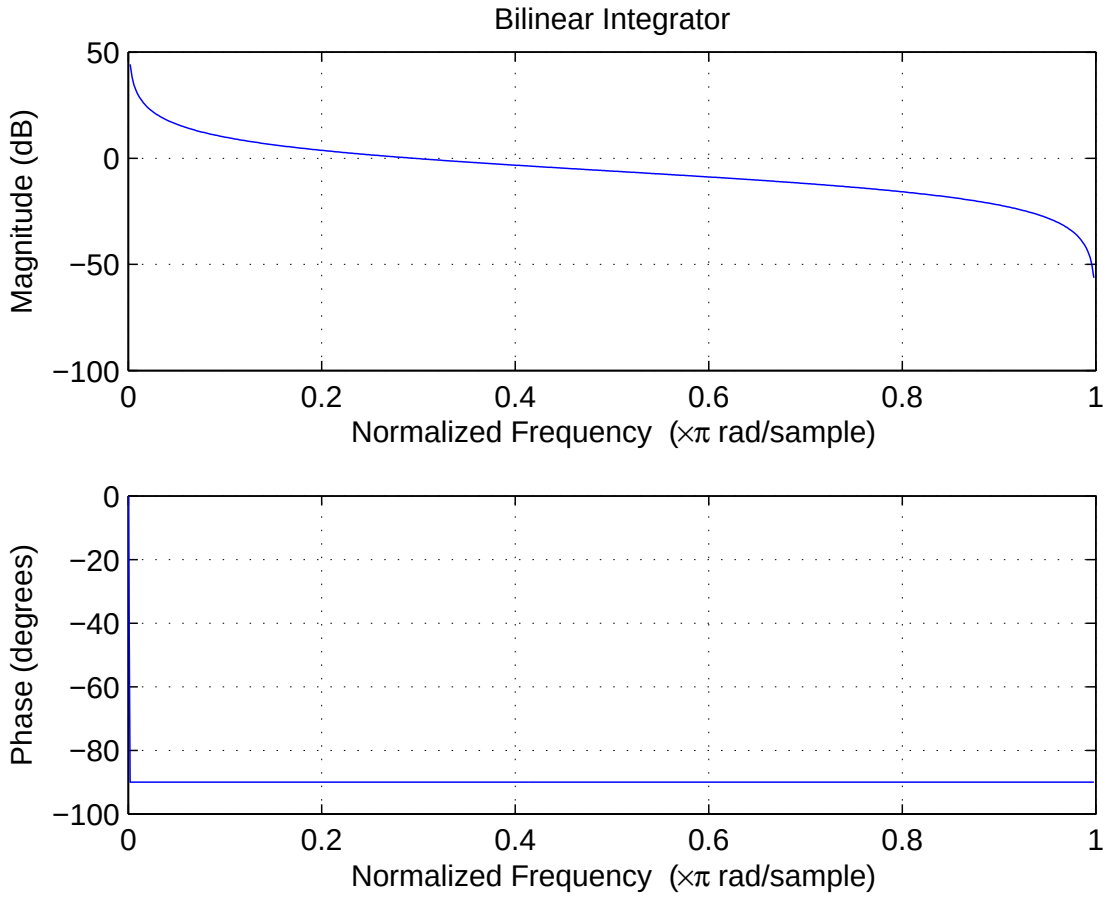
(b) From above

$$y_n = y_{n-1} + \frac{T}{2}\left(f_n + f_{n-1}\right)$$

which is the trapezoidal numerical integration rule.

The following MATLAB commands generate the frequency response:

```
b = [0.5 0.5];
a = [1 -1];
freqz(b,a)
title('Bilinear Integrator')
```



Note that while the phase is constant at $-\pi/2$, the magnitude plot does not show a constant slope of -20 dB/decade as does the analog integrator.

Problem 3:

(a) The plant

$$H(s) = \frac{s}{s^2 + 2s + 1}$$

has (1) a zero at $s = 0$, and (2) a pair of coincident poles at $s = -1$. The root-matching (matched z-transform) discrete-time system is

$$H(z) = K\frac{z - e^{0T}}{(z - e^{-T})(z - e^{-T})} = K\frac{z - 1}{z^2 - 2e^{-T}z + e^{-2T}}$$

and with $T = 0.1$ sec.,

$$H(z) = K\frac{z - 1}{z^2 - 1.809z + 0.8187}$$

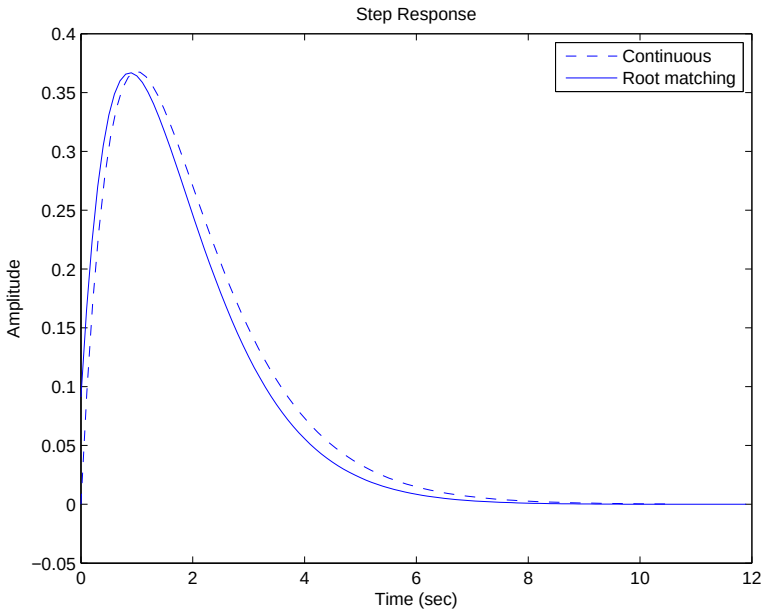
To create a minimum delay filter, make the order of the numerator and denominator equal by adding a zero at the origin:

$$H(z) = K\frac{z^2 - z}{z^2 - 1.809z + 0.8187}$$

The gain factor K must be determined empirically. This is normally done using the final-value theorem, but in this case $\lim_{s\to 0} H(s) = 0$ and the final values cannot be compared. Some other amplitude criterion must be used; in this case the *peak value* of the two step responses were compared and found to be 0.367 for the continuous system $H(s)$ and 4.018 for the discrete system $H(z)$. Then $K = 0.367/4.018 = 0.0913$.

$$H(z) = \frac{0.0913(z^2 - z)}{z^2 - 1.809z + 0.8187}$$

Alternatively you might compare the response of the analog system to a ramp input $r(t) = t$ with the analogous discrete response to a ramp $r_n = 0.1n$ and use their final values to match the gains resulting in $K = 0.097$. The step responses with $K = 0.0913$ are compared in the plot below:



Note the slight delay in the digital response.

(b) The system has repeated poles at $s = -1$. Then $H(s)$ may be expressed in partial fractions as

$$H(s) = \frac{1}{s+1} - \frac{1}{(s+1)^2}$$

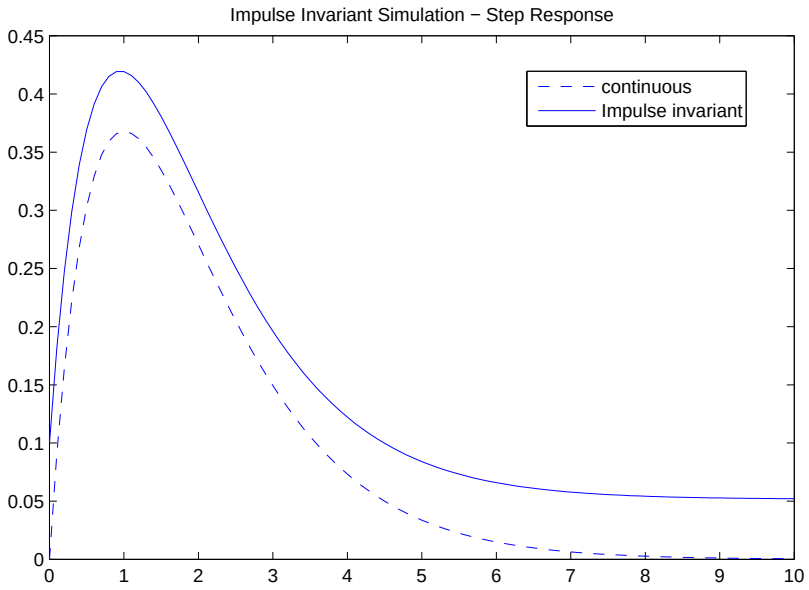
and

$$\begin{aligned} H(z) &= \mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{1}{s+1}\right\}\right\} - \mathcal{Z}\left\{\mathcal{L}^{-1}\left\{\frac{1}{(s+1)^2}\right\}\right\} \\ &= \frac{z}{z-e^{-T}} - \frac{Te^{-T}}{(z-e^{-T})^2} \qquad \text{(from tables)} \\ &= \frac{z(z-e^{-T}(1+T))}{z^2 - 2e^{-T}z + e^{-2T}} \\ &= \frac{z^2 - 0.9953z}{z^2 - 1.8097z + 0.8187} \end{aligned}$$

As we discussed in class, this form of impulse invariant simulation suffers from a gain error, and a correction

$$H'(z) = TH(z) = 0.1\frac{z^2 - 0.9953z}{z^2 - 1.8097z + 0.8187}$$

is more frequently used. The step responses of $H(s)$ and $H'(z)$ are compared in the following plot.



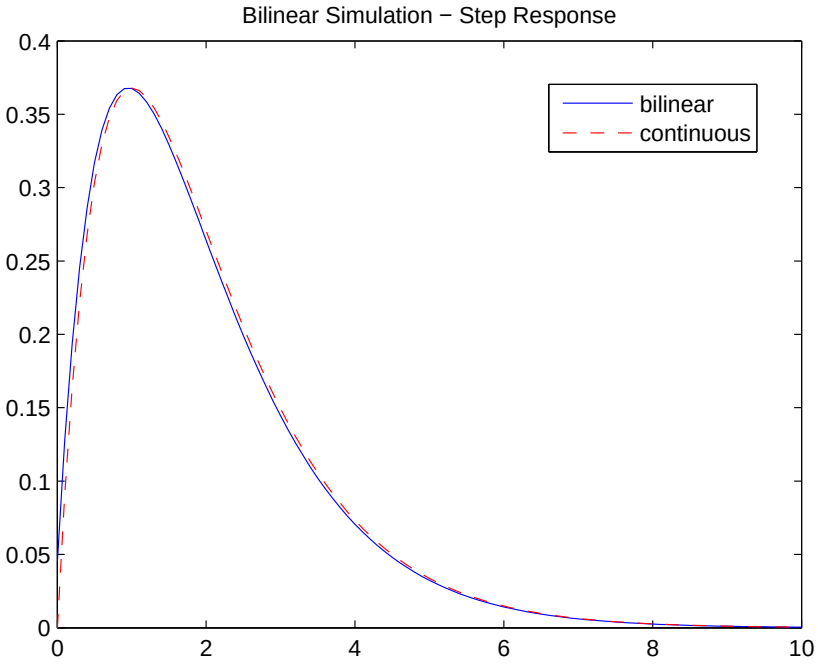
and we note:

- The final value is incorrect (due to aliasing in the transfer function), and
- Further empirical gain adjustment is necessary to match the peak responses.

(c) With the bilinear transform

$$\begin{aligned} H(z) &= \frac{\frac{2}{T}\left(\frac{z-1}{z+1}\right)}{\left(\frac{2}{T}\left(\frac{z-1}{z+1}\right)\right)^2 + 2\frac{2}{T}\left(\frac{z-1}{z+1}\right) + 1} \\ &= \frac{2T\left(z^2-1\right)}{(4+4T+T^2)z^2 + (2T^2-8)z + (4-4T+T^2)} \\ &= \frac{0.0454(z^2-1)}{z^2 - 1.8095z + 0.8186} \end{aligned}$$

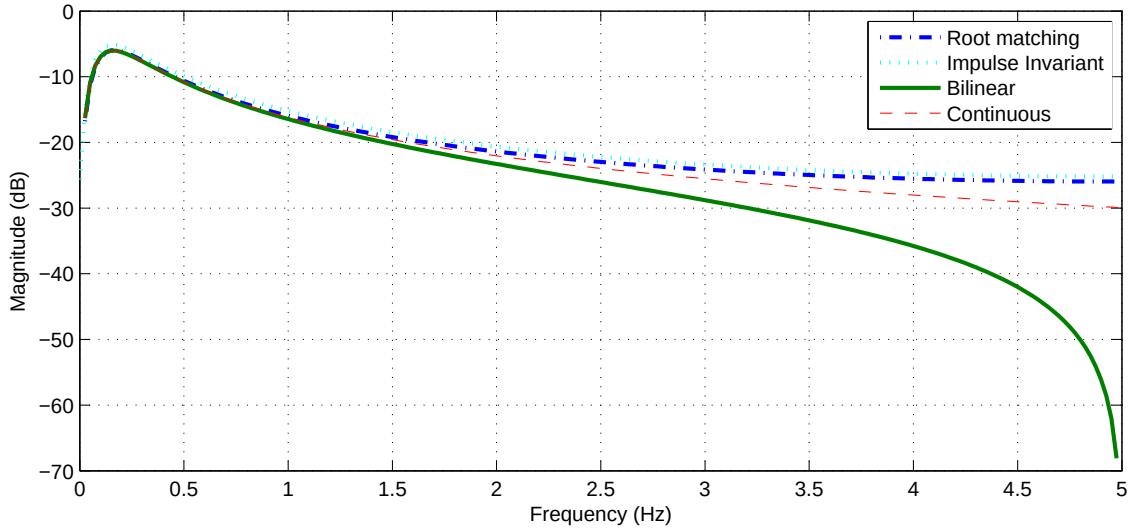
The step responses of $H(s)$ and $H(z)$ are compared in the following plot.

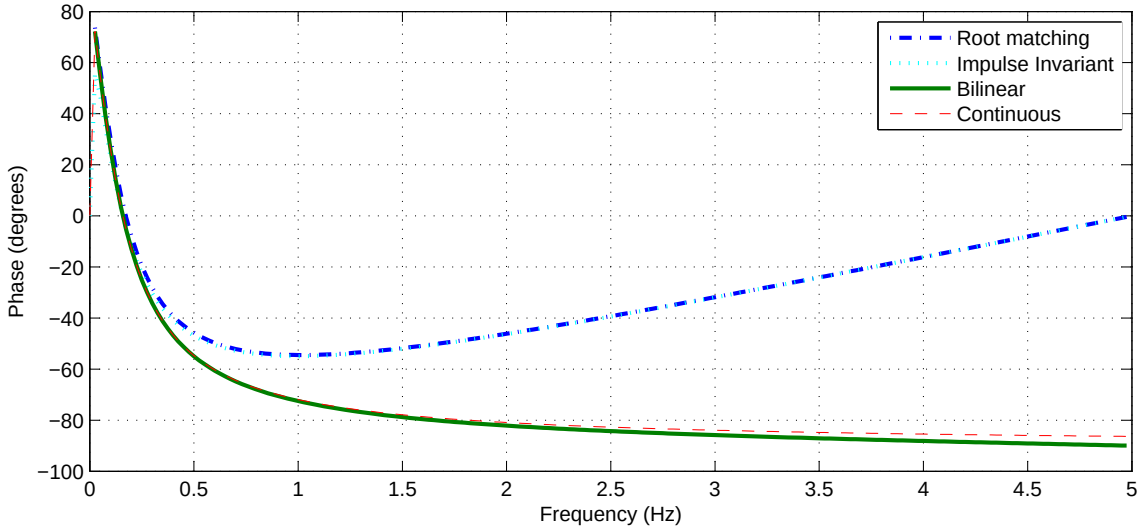


The frequency response of these three discrete approximations of our continuous filter, are plotted and compared to the frequency response of the continuous filter in the next page. In general, root matching and impulse invariant filters are approximately the same and show almost exactly the same curves.

The magnitude plot shows that below 1 Hz, all of the three filters match very well with the continuous filter. However as the frequency increases toward the Nyquist frequency the bilinear filters deviates strongly from the continuos filter and the two others almost follow the continuos curve.

On the other hand, in the phase plot, the bilinear filter almost exactly matches the continuos filter, while the two others deviate strongly from the continuous filter and only match it at very low frequencies (i.e. below 0.3 Hz).





Problem 4:

(a) From the specifications

$$\frac{1}{1+\epsilon^2} = 0.5 \quad \longrightarrow \quad \epsilon = 1$$
$$\frac{1}{1+\lambda^2} = 0.1 \quad \longrightarrow \quad \lambda = 3$$

For a continuous filter (no pre-warping)

$$N \geq \frac{\cosh^{-1}(\lambda/\epsilon)}{\cosh^{-1}(\omega_r/\omega_c)} = \frac{\cosh^{-1}(3/1)}{\cosh^{-1}(11.75/10)} = 3.022$$

Therefore take $N = 4$.

(b) In the pre-warped analog filter the critical frequencies will be

$$\begin{aligned} \omega_c' &= \frac{2}{T}\tan\left(\frac{\omega_c T}{2}\right) = 72654 \text{ rad/s} \\ \omega_r' &= \frac{2}{T}\tan\left(\frac{\omega_r T}{2}\right) = 90992 \text{ rad/s} \end{aligned}$$

and the order is given by

$$N \geq \frac{\cosh^{-1}(\lambda/\epsilon)}{\cosh^{-1}(\omega_r/\omega_c)} = \frac{\cosh^{-1}(3/1)}{\cosh^{-1}(.9099/.7265)} = 2.53$$

Therefore take $N = 3$.

(c) MATLAB gave me a lot of problems with ill-conditioning while trying to do this! I had to scale the sampling rate back – for example by a factor of 100, to 500 samples/sec.:

```
[b, a] = cheby1(3, 3, 2*pi*100, 's');
[bz,az] = bilinear(b, a, 500, 100)      %let bilinear() do the pre-warping
```

or

```
T = 1/500; wc = 2*pi*100;
wc_{p} = (2/T)*tan(wc*T/2);            %do the pre-warping directly.
[b, a] = cheby1(3, 3, wc_{p}, 's');
[bz,az] = bilinear(b, a, 500)
```

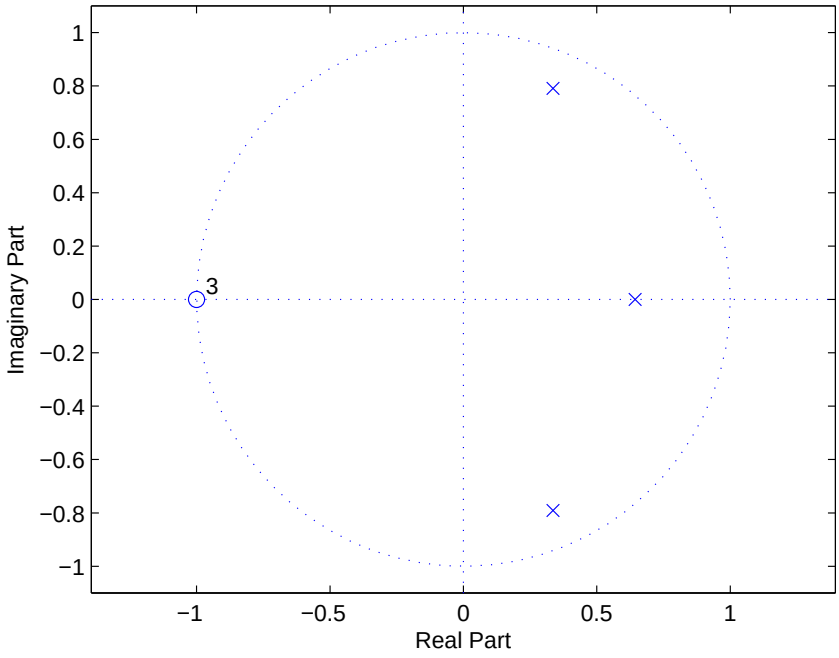
or

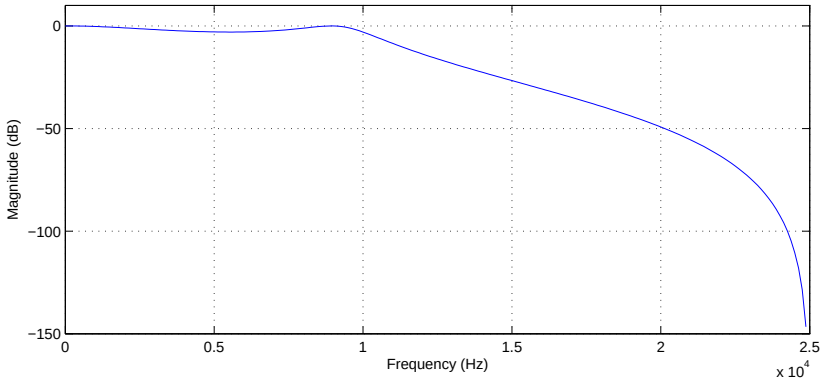
```
[b, a] = cheby1(3, 3, 0.4);
```

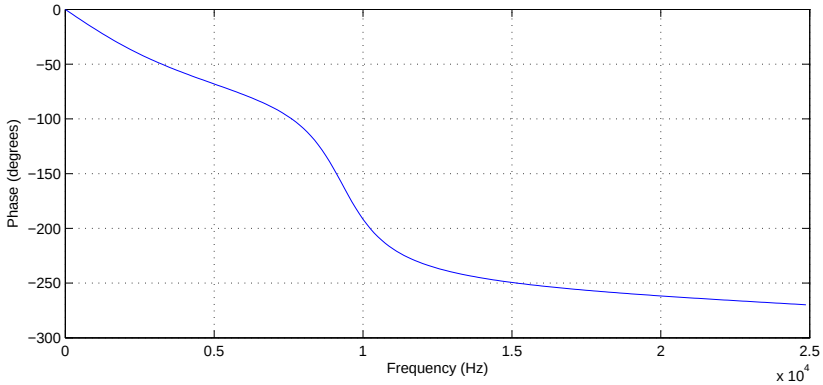
all give the same result:

$$H(z) = \frac{.04756z^3 + 0.14273z^2 + +0.14273z + 0.04756}{z^3 - 1.3146z^2 + 1.17043z - 0.47524}$$

(d) The pole-zero plot and frequency response plots are shown on the next page.

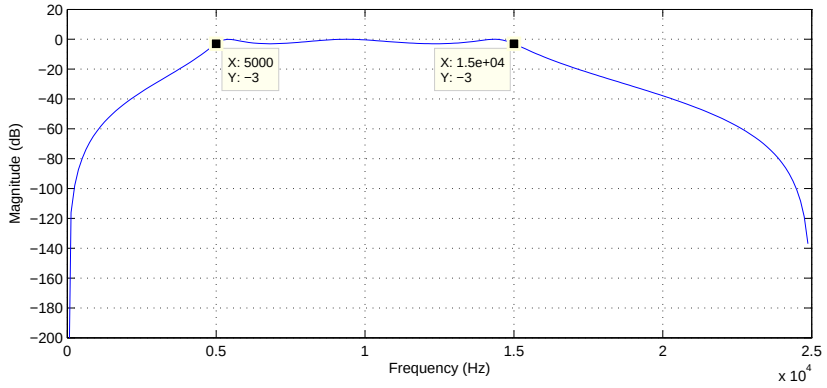

Imaginary Part
Real Part
3
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
−1
−0.5
0
0.5
1

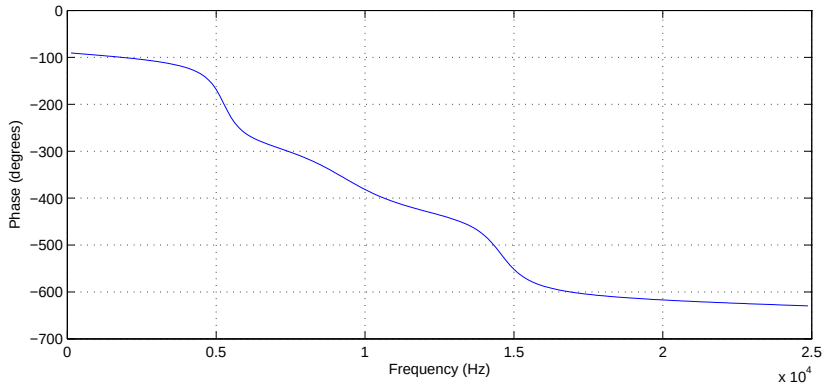

Magnitude (dB)
Frequency (Hz)
x 10^4
0
−50
−100
−150
0
0.5
1
1.5
2
2.5


Phase (degrees)
Frequency (Hz)
x 10^4
0
−50
−100
−150
−200
−250
−300
0
0.5
1
1.5
2
2.5

(e) We have to pre-warp the band limits to use them in converting the prototype continuous low pass filter (with cut-off frequency of 1 rad/sec) to our prototype continuous band pass filter.

$$H_{bp}(z) = \frac{0.04758z^6 - 0.14273z^4 + 0.14273z^2 - 0.04758}{z^6 - 1.6480z^5 + 2.3066z^4 - 2.1191z^3 + 1.9118z^2 - 0.9916z + 0.47524}$$






```
T = 1/500;
wc1 = 2*pi*50;
wc2 = 2*pi*150;

wc1_p = (2/T)*tan(wc1*T/2);
wc2_p = (2/T)*tan(wc2*T/2);

[b, a] = cheby1(3, 3, 1,'s'); % Design LP with wc=1 rad/sec
[b,a]=lp2bp(b,a,sqrt(wc1_p*wc2_p),(wc2_p-wc1_p)); %Change to BP
[bz,az] = bilinear(b, a, 500);
```