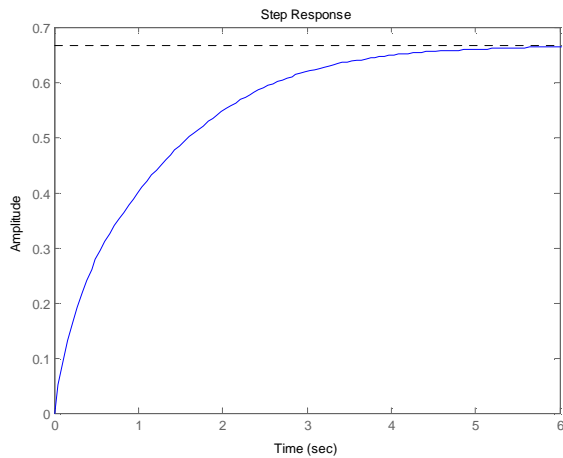


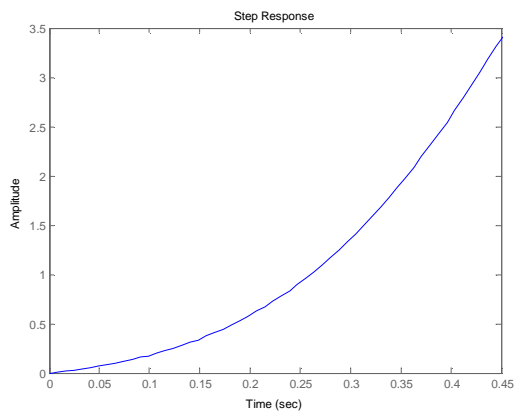
EE422G Solution to Homework #8

1. MATLAB

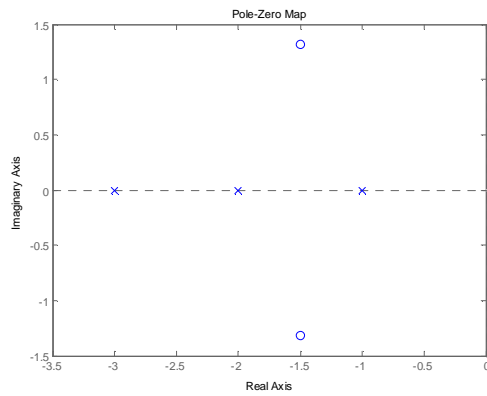
```
>> H1 = tf([1 3 4],[1 6 11 6]);  
>> H2 = tf([1 2 3 2],[1 -10 35 50 24]);  
>> step(H1)
```



```
>> step(H2)
```

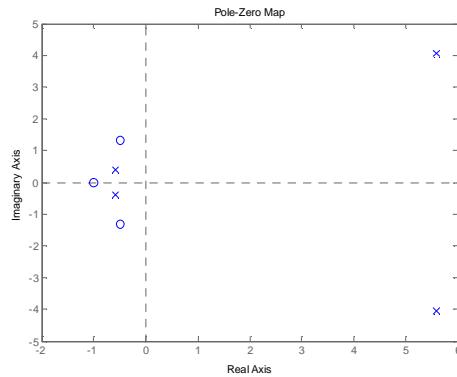


```
>> pzmap(H1)
```



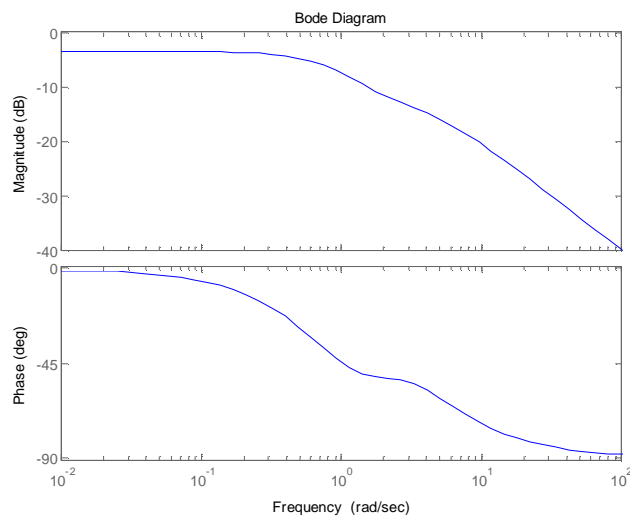
All LHP poles $\Rightarrow H_1$ is BIBO stable

`>> pzmap(H2)`



Two RHP poles $\Rightarrow H_2$ is unstable

`>> bode(H1)`

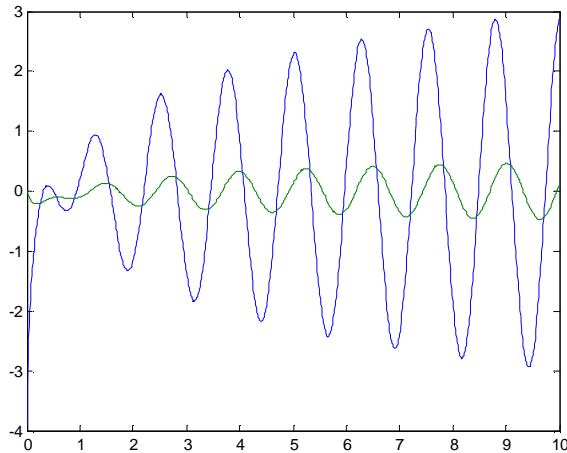


Even you can use MATLAB to show the Bode plot of H_2 , it does not exist because H_2 has two RHP poles. Thus, we should not blindly trust the results of MATLAB.

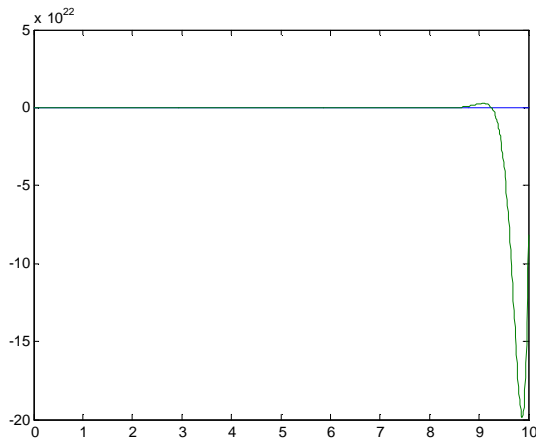
`>> t=0.01:0.01:10;`

```
>> x = log(2*t).*cos(5*t);

>> plot(t,x,t,lsim(H1,x,t))
Warning: Simulation will start at the nonzero initial
time T(1).
> In lti.lsim at 91
```



```
>> plot(t,x,t,lsim(H2,x,t))
Warning: Simulation will start at the nonzero initial
time T(1).
> In lti.lsim at 91
```



Note that the output explodes due to its instability!

- (6 points) To realize her “American Idol” dream, a friend of yours has recently bought a microphone for her own recording studio. Unfortunately, the microphone is inexpensive and the quality is poor. She has recorded a sample sound file which you can find in `distortedSound.wav` from the homework webpage. Armed with your recent knowledge of MATLAB, you try to help her out to post-process the recording. After some research on the specifications, you find out that the transfer function of the microphone is as follows:

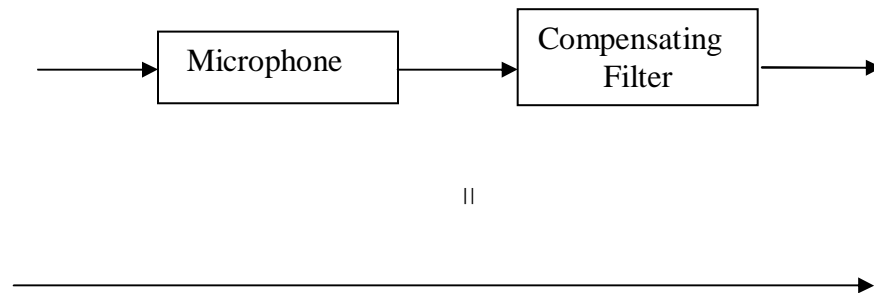
$$H(s) = 0.04 \frac{s^2 + 2 \times 10^2 s + 1.0001 \times 10^8}{s^2 + 1.52 \times 10^3 s + 2.92 \times 10^6}$$

- Could you design a linear system to compensate the distortion caused by the microphone? Please submit the MATLAB code for both the design of your “compensating filter” and the experiments.
- Due to your success in part a., your friend wants to further cut the cost by buying an even cheaper microphone. The sample sound file is stored in `distortedSound2.wav` and the transfer function is almost the same as before except for a sign change in the numerator and a small change in gain:

$$H(s) = 0.03 \frac{s^2 - 2 \times 10^2 s + 1.0001 \times 10^8}{s^2 + 1.52 \times 10^3 s + 2.92 \times 10^6}$$

Can you use the same approach as in part a) to compensate for the distortion in this case? I highly recommend plotting the output first before attempting to play it with your computer speaker. Also try the compensating filter you obtained from part a). Please comment your results.

The idea is to create a compensating filter in such a way that the overall system is as close as possible to a wire (no loss of information):

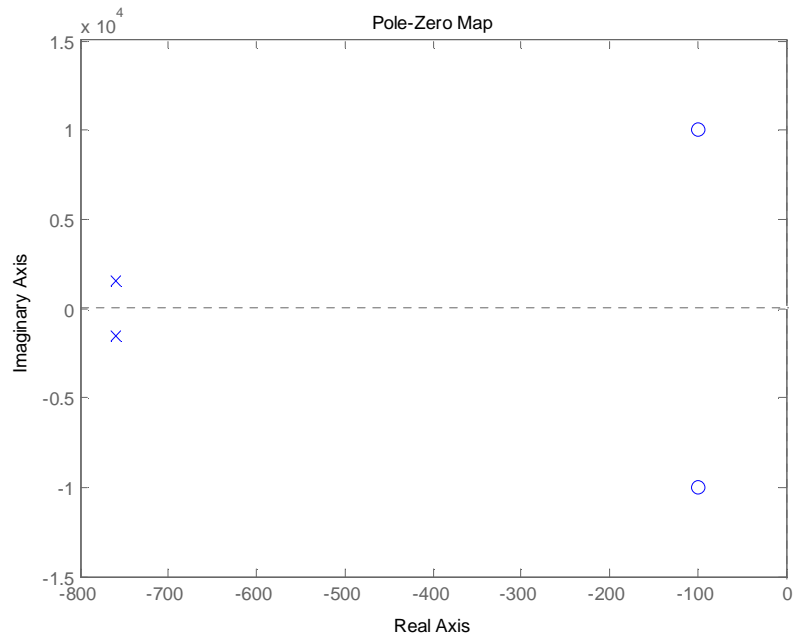


Given the microphone has a transfer function $H(s)$, it is natural to use $G(s) = k/H(s)$ as the compensating filter. The poles of $H(s)$ then become the zeros of $G(s)$ and the zeros of $H(s)$ become the poles of $G(s)$. Thus, even if $H(s)$ is an (asymptotically) stable system, $G(s)$ may not be. To guarantee the stability of both $H(s)$ AND $G(s)$, we want all the poles AND zeros of $H(s)$ to be on the open left half plane. Such a system is called a *minimum phase* system. To check whether our first microphone is minimum phase, we check the pole zero map of $H(s)$:

```
>> H1 = tf(0.04*[1 2e2 1.0001e8],[1 1.52e3 2.92e6])
```

```
Transfer function:
0.04 s^2 + 8 s + 4e006
-----
s^2 + 1520 s + 2.92e006
```

```
>> pzmap(H1)
```



Indeed, all the zeros and poles are on the open left half plane (the $j\omega$ -axis is the right boundary of the figure.) As a result, you can apply $G(s) = 1/H(s)$ to the sound and you should get a much clearer voice:

```
>> [numH1, denH1] = tfdata(H1,'v');  
>> G1 = tf(denH1, numH1)
```

```
Transfer function:  
s^2 + 1520 s + 2.92e006  
-----  
0.04 s^2 + 8 s + 4e006
```

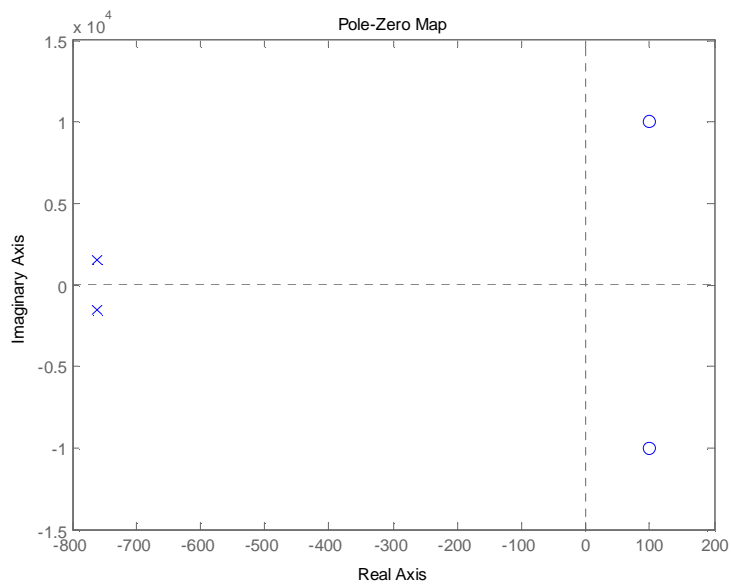
```
>> s1 = wavread('distortedSound.wav');  
>> tsteps = 1/44100*(0:1:(length(s1)-1)); % sampled @ 44.1kHz  
>> y1 = lsim(G1,s1,tsteps);  
>> sound(y1,44100)
```

For part b), once again let's consider whether the microphone transfer function is a minimum phase system:

```
>> H2 = tf(0.03*[1 -2e2 1.0001e8],[1 1.52e3 2.92e6])
```

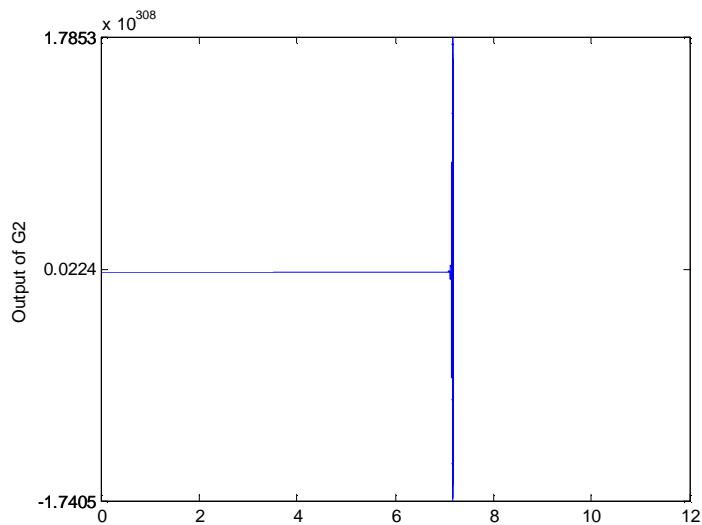
```
Transfer function:  
0.03 s^2 - 6 s + 3e006  
-----  
s^2 + 1520 s + 2.92e006
```

```
>> pzmap(H2)
```



The zeros are on the right half plane and thus it is not a minimum phase system – the inverse system $G(z) = 1/H(z)$ will have poles on the right half plane and become unstable. If you simulate $G(z)$ and plot the output, you will get something like this.

```
>> [numH2, denH2] = tfdata(H2,'v');
>> G2 = tf(denH2, numH2);
>> s2 = wavread('distortedSound2.wav');
>> y2 = lsim(G2,s2,tsteps);
>> plot(tsteps,y2)
```



I hope dearly that you didn't play this to your speaker! On the other hand, we can build an approximate inverse system by using positive feedback system.

$$G_2(s) = \frac{A}{1 + AH_2(s)}$$

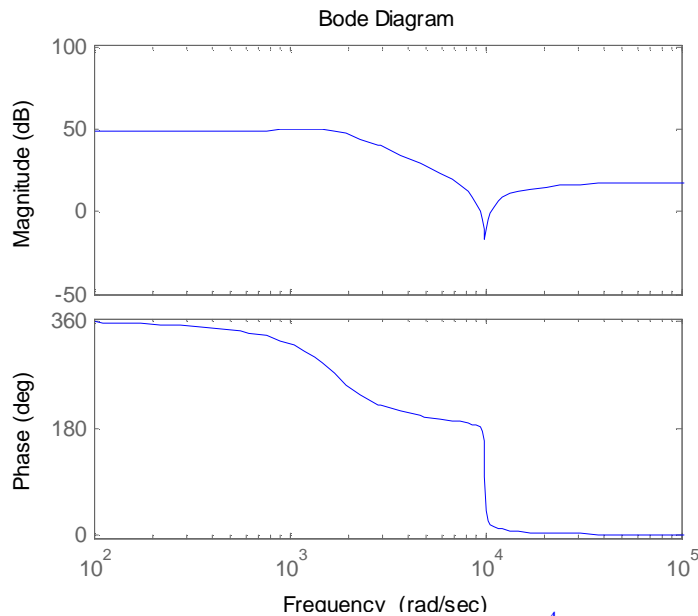
for a constant A such that $|AH_2(j\omega)| \gg 1$ is large enough within the operating range. Expanding $G_2(s)$ analytically, we have

$$G_2(s) = \frac{0.03As^2 - 6As + 3.0003 \times 10^6 A}{(1 + 0.03A)s^2 + (1.52 \times 10^3 - 6A)s + (2.92 + 3.003A) \times 10^3}$$

By using Routh Array, you can easily show that $G_2(s)$ does not have any RHP poles if $-0.97 < A < 253.3$. As we want A to be big, let's try $A = 250$.

First we want to check $|AH_2(j\omega)|$:

```
>> A = 250; trial = tf(A*numH2,denH2);
>> bode(trial)
```



Note that the $|AH_2(j\omega)|$ drops below 1 (0 dB) around 10^4 rad/second (or 1600 Hz). Human speech can contain frequency component up to 16000 Hz, thus we would expect that our inverse filter can help only in the low-frequency range.

```
>> G2 = tf(A*denH2,A*numH2+denH2);
>> y2 = lsim(G2,s2,tsteps);
>> sound(y2,44100)
```

How are you getting the desired results?

- (4 points) We have seen the aliasing effect visually during lecture. In this problem, you are asked to explore the aliasing effect in audio. Download the MATLAB script `aliasing_demo_audible.m` from the homework webpage and run it. You will hear six tones at different frequencies: 500Hz, 2kHz, 3kHz, 4.5kHz, 5.5kHz and 7kHz. All the signals are sampled at 5kHz. Explain what you hear using the Nyquist Theorem.

Aliasing occurs in the last four tones resulting in distortion on the pitch.

4. (4 points) The signal

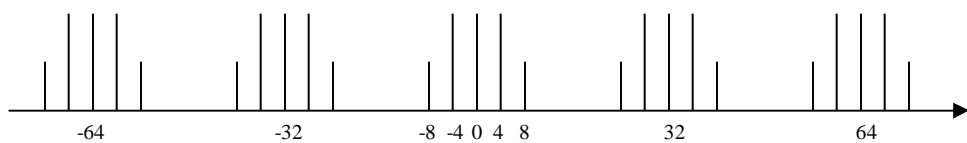
$$x(t) = 4 + 8 \cos(8\pi t) + 3 \cos(16\pi t)$$

is sampled at a rate of 32 samples per second. Plot the amplitude spectrum of the sampled signal showing the weight and the frequency of each component for $|f| < 80$ Hz. Show how the signal can be reconstructed from the samples.

The Fourier transform of the signal is

$$X(f) = 4\delta(f) + 4\delta(f - 4) + 4\delta(f + 4) + \frac{3}{2}\delta(f - 8) + \frac{3}{2}\delta(f - 8)$$

After sampling at 32 Hz, the spectrum looks like



Sampling frequency is 32 Hz > the Nyquist rate = $8 \times 2 = 16$. Thus, the signal can be reconstructed perfectly by a sinc filter.