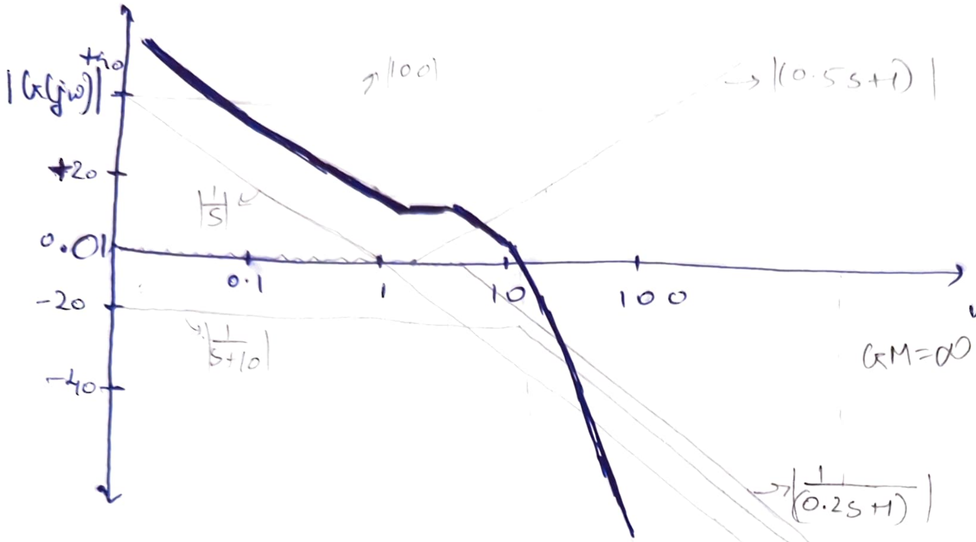
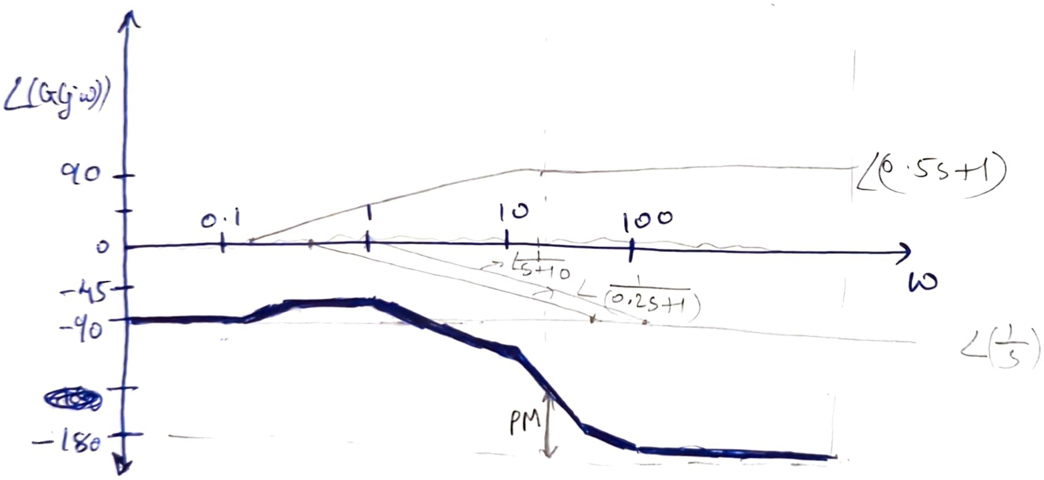
Q1 (a) $L(s) = \dfrac{100(0.5s+1)}{s(0.2s+1)(s+10)}$

Collaborated with
→ Ruchita Sinha
→ Aditya Rathi

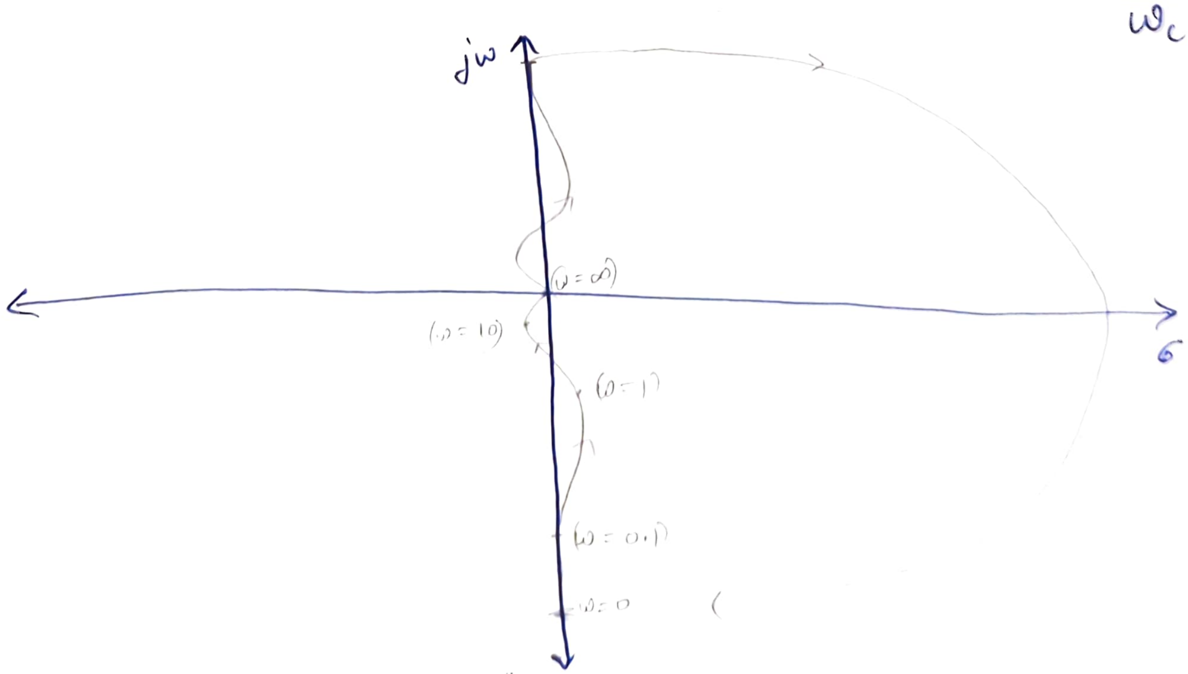




(b)
For gain margin,
the phase plot asymptotically
reaches $-180°$ at $\omega = \infty$.
Gain at $\omega = \infty$, $= \infty$
$\therefore$ GM $= \infty$

For phase margin,
the cross-over frequency
is slightly >10,
In the phase plot,
we have a slope of
$-45°$/dec between $\omega \in (1, 10)$
and $-90°$/dec just after
$\omega = 10$.
The phase then at
cross-over frequency
would be $\sim -135°$
$\therefore$ PM $= |-180° - (-135°)|$
$= 45°$

(C) Delay Margin
$= \dfrac{PM}{\omega_c} = \dfrac{45}{15} \times \dfrac{2\pi}{180}$
~~$\approx 2/\pi$ seconds~~
$= \dfrac{\pi}{30}$ seconds



Q1 (C)

```
s = tf('s');
```

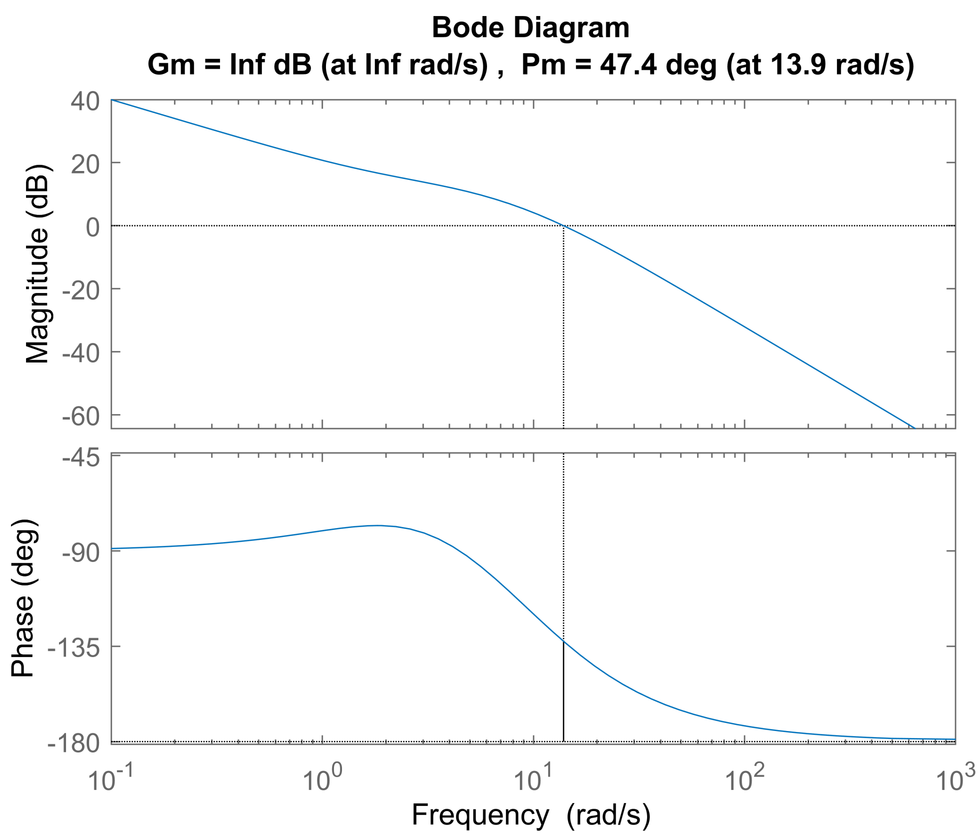
Plant transfer funciton

```
G = (100*(0.5*s+1))/(s*(0.2*s+1)*(s+10))
```

```
G =

        50 s + 100
  ----------------------
  0.2 s^3 + 3 s^2 + 10 s

Continuous-time transfer function.
```

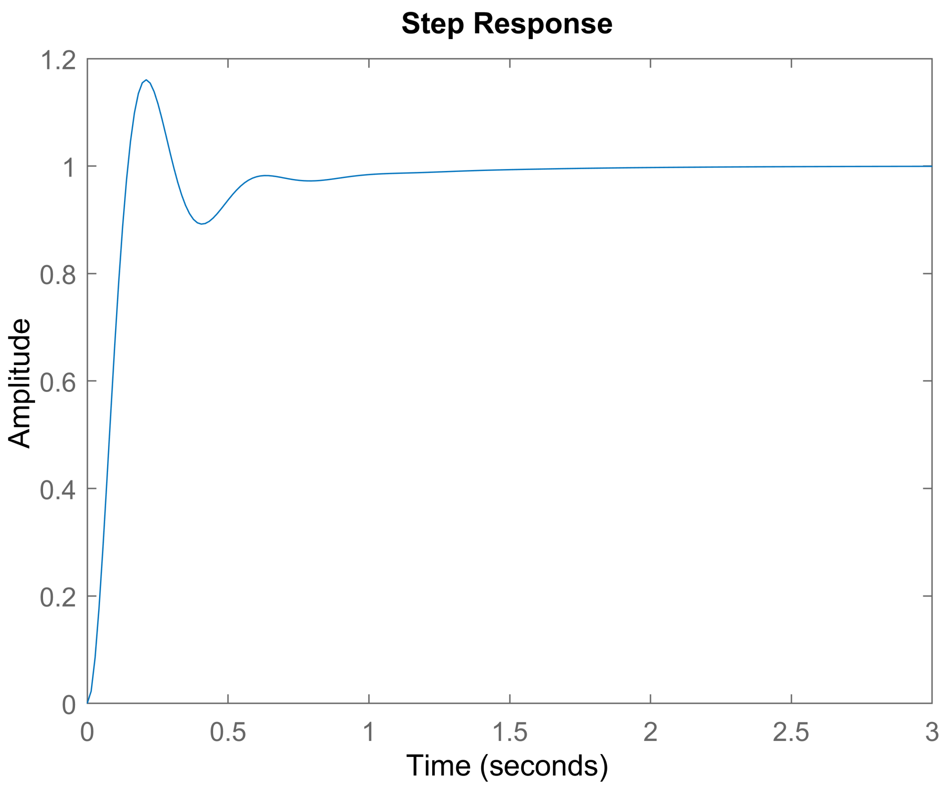
Compute the margins of the transfer funciton.

```
margin(G)
```



Step response of the closed-loop system

```
step(G/(1+G))
```

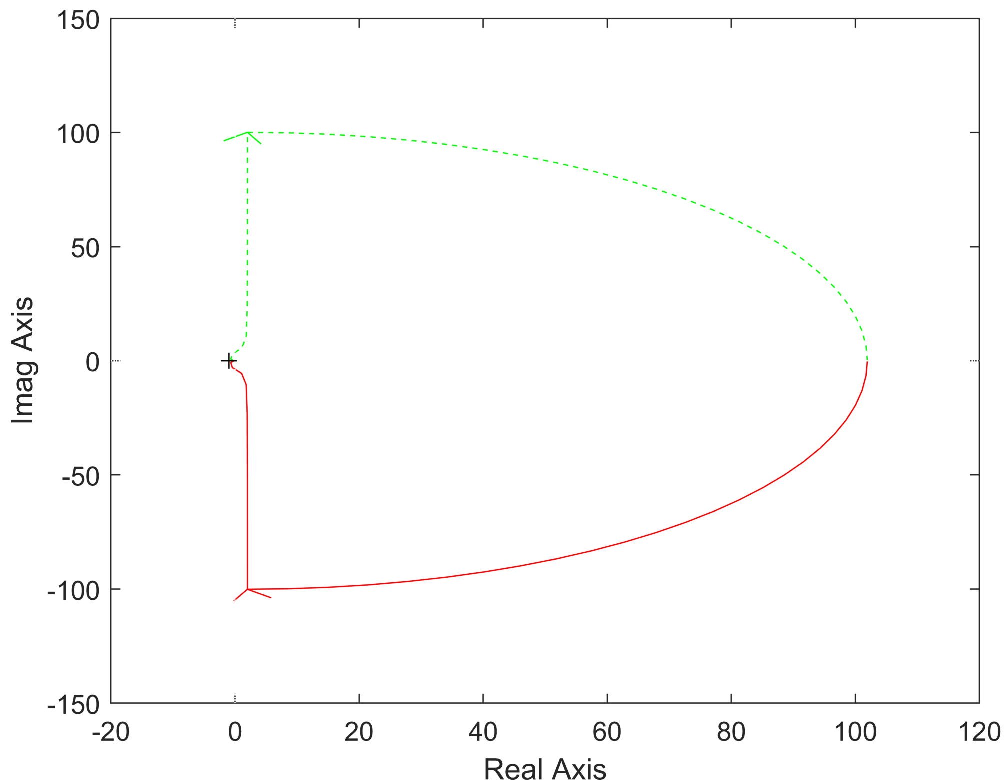



```
[Gm,Pm,Wcg,Wcp] = margin(G)
```

```
Gm = Inf
Pm = 47.3637
Wcg = Inf
Wcp = 13.8869
```

The margins obtained using MATLAB are close to the ones obtained using the bode plots drawn by hand.

```
nyquist1(G)
```


150
100
50
0
-50
-100
-150
-20
0
20
40
60
80
100
120
Imag Axis
Real Axis

```
s = tf('s')
```

```
s =

  s

Continuous-time transfer function.
```

Plant transfer function.

```
G = 100/(s*(s+1)*(s+10))
```

```
G =

          100
  -------------------
  s^3 + 11 s^2 + 10 s

Continuous-time transfer function.
```

Calculate the phase of the system at 5 rad/s to add a lead compensator.

```
[mag,phase,wout,sdmag,sdphase] = bode(G,5)
```

```
mag = 0.3508
phase = -195.2551
wout = 5
sdmag =

     []


sdphase =

     []
```

Lead compensator transfer function is now obtained which will provide the desired phase margin.

```
K = lead(75.2251,5)
```

```
K =

  s + 0.6483
  ----------
  s + 38.56

Continuous-time transfer function.
```

New transfer function is now obtained which has the desired phase margin.

```
L = K*G/0.0455
```

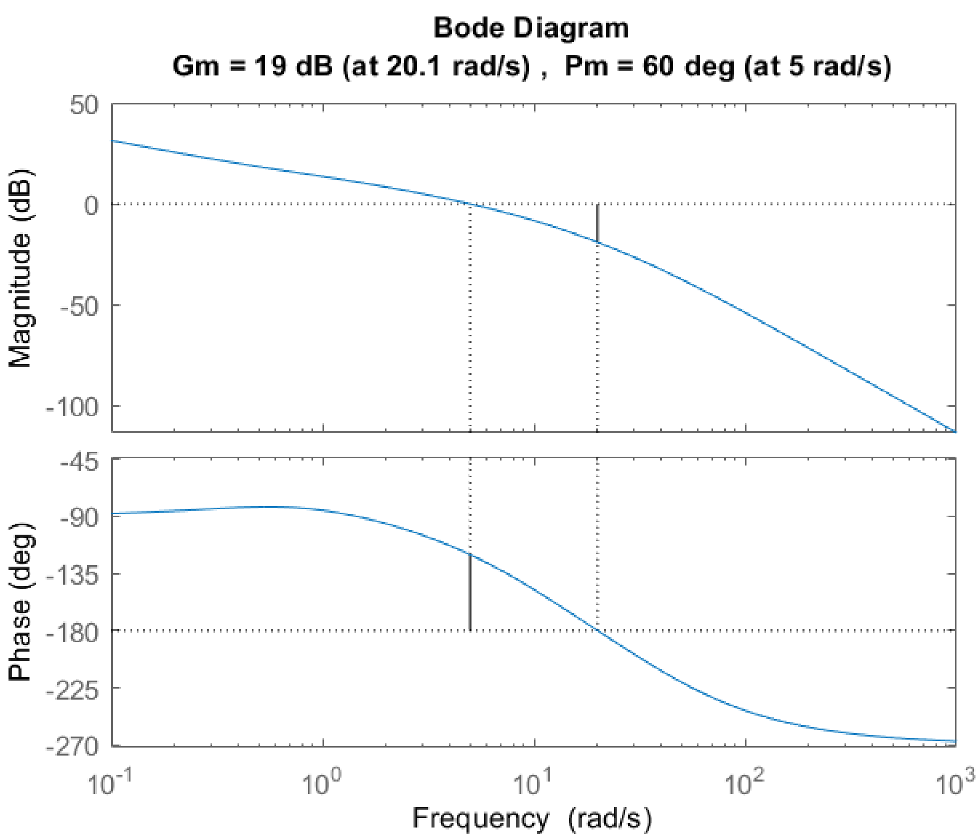
```
L =

                 100 s + 64.83
  --------------------------------------------
  0.0455 s^4 + 2.255 s^3 + 19.76 s^2 + 17.55 s

Continuous-time transfer function.
```

Next, we calculate the magnitude at the desired cross-over frequency to provide an offset to the magnitude plot to achive the desired preformance.

```
[mag,phase,wout,sdmag,sdphase] = bode(L,5)
```

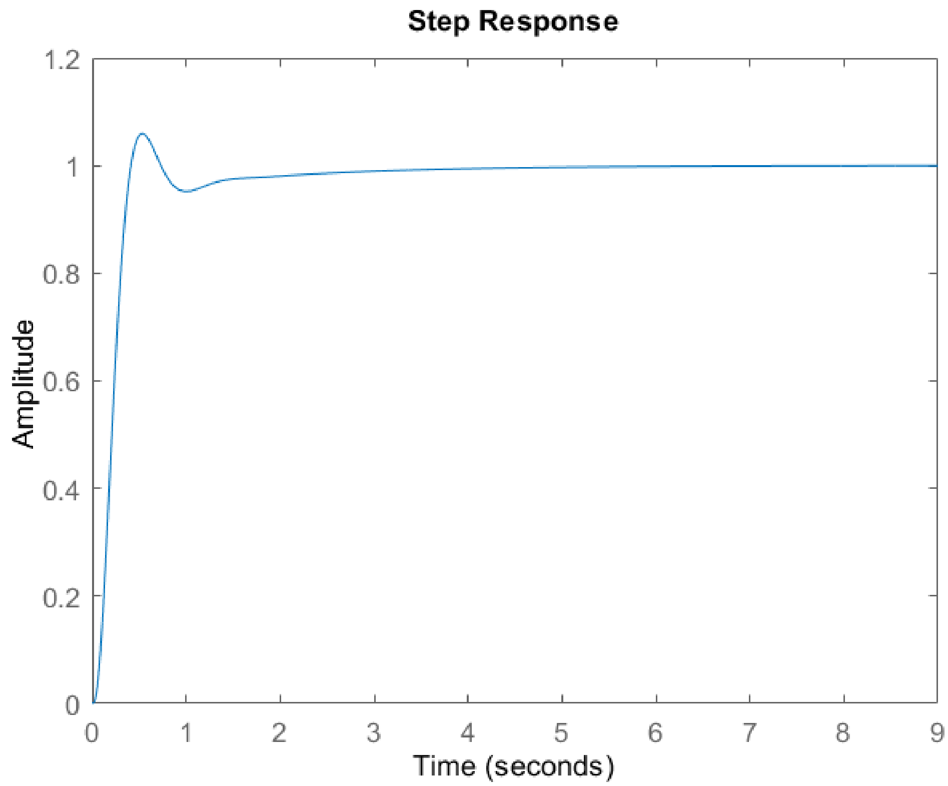
```
mag = 0.9997
phase = -120.0300
wout = 5
sdmag =

     []


sdphase =

     []
```

```
margin(L)
```



Step response of the closed loop system.

```
step(L/(L+1))
```


Step Response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1
2
3
4
5
6
7
8
9

```
clear all
clc
s = tf('s');
```

Plant transfer function.

```
G = 100/(s*(s+1)*(s+10))
```

```
G =

         100
  -------------------
  s^3 + 11 s^2 + 10 s

Continuous-time transfer function.
```

I chose canonical form from all the available opitions to get the state-space form of the system.

```
csys = canon(G,'companion')
```

```
csys =

  A =
        x1   x2   x3
   x1    0    0    0
   x2    1    0  -10
   x3    0    1  -11

  B =
       u1
   x1   1
   x2   0
   x3   0

  C =
        x1   x2   x3
   y1    0    0  100

  D =
       u1
   y1   0

Continuous-time state-space model.
```

```
A = csys.A;
B = csys.B;
C = csys.C;
```

Compute the desirerd poles using the second-order approximation of the system. The plan is to place the third pole far away as compared to the two poles obtained from the calculation so that the third pole does not have much impact on the system.

```
MO = 20;
ts = 6;
```

Compute the damping ratio

```
z = -log(MO/100)/sqrt(pi^2 + (log(MO/100))^2)
```

```
z = 0.4559
```

Compute natuaral frequency

```
w = 4/(ts*z)
```

```
w = 1.4621
```

Finding desired poles of the system from the characteristic equation.

```
p = roots([1 2*z*w w^2]);
poles = [p(1) p(2) -10];
```

Place the poles at the desired location.

```
K = place(A,B,poles)
```

```
K = 1×3
    0.3333    1.8045   -1.8045
```

Set the observer poles far away from the controller poles so that the observer does not interfere with the system response.

```
l = [-80,-81,-82];
```

Place observer poles.

```
L  = place(A',C',l)
```

```
L = 1×3
10^3 ×
    5.3136    0.1967    0.0023
```

```
L = L';
```

Create a regulator for the plant using the obtained K and L values.

```
feedback = reg(csys,K,L);
AA = feedback.A;
BB = feedback.B;
CC = feedback.C;
DD = feedback.D;
```

Convert state-space to transfer funciton for the reguator.

```
[Ns,Ds] = ss2tf(AA,BB,CC,DD);
S = [s^3,s^2,s,1];
tf = (S*Ns')/(S*Ds')
```

```
tf =
 
     -2122 s^2 - 3.258e04 s - 1.136e05
  ---------------------------------------
  s^3 + 243.3 s^2 + 1.976e04 s + 5.384e05
```

```
Continuous-time transfer function.
```

Compute the closed loop tranfer funciton.

```
TF = 100/((s*(s+1)*(s+10))-100*tf)
```

```
TF =
 
                    100 s^3 + 2.433e04 s^2 + 1.976e06 s + 5.384e07
  ------------------------------------------------------------------------------------
  s^6 + 254.3 s^5 + 2.245e04 s^4 + 7.582e05 s^3 + 6.332e06 s^2 + 8.642e06 s + 1.136e07
 
Continuous-time transfer function.
```

Convert the closed loop system to state-space canonical form.

```
sys = canon(TF,'companion')
```

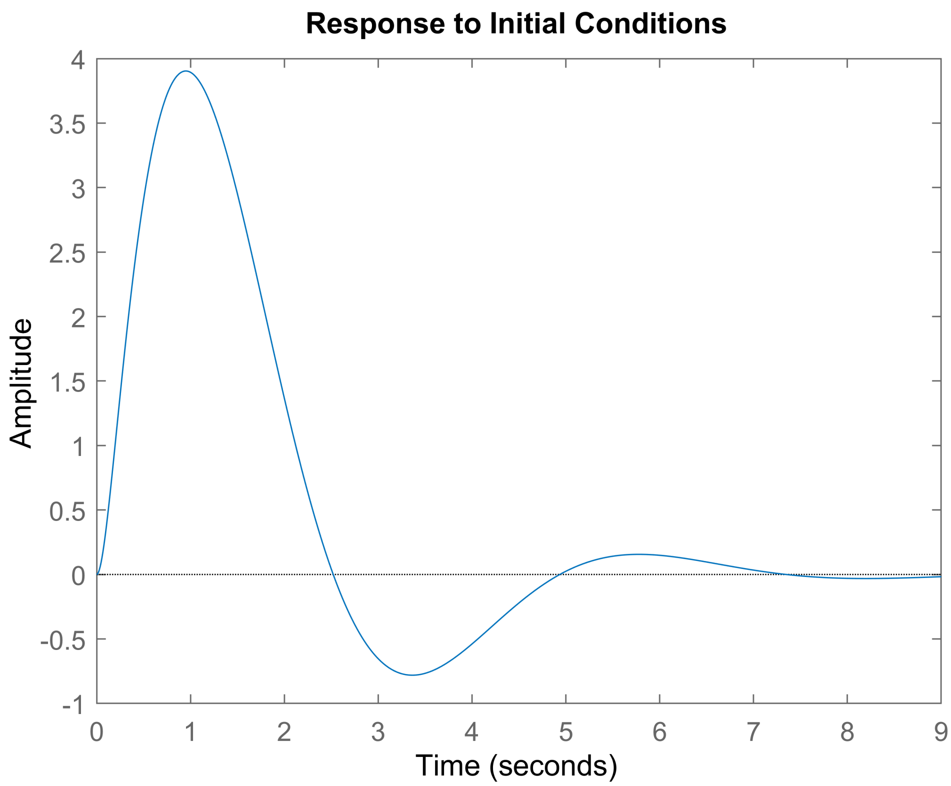
```
sys =
 
  A = 
              x1         x2         x3         x4         x5         x6
   x1          0          0          0          0          0  -1.136e+07
   x2          1          0          0          0          0  -8.642e+06
   x3          0          1          0          0          0  -6.332e+06
   x4          0          0          1          0          0  -7.582e+05
   x5          0          0          0          1          0  -2.245e+04
   x6          0          0          0          0          1      -254.3
 
  B = 
       u1
   x1   1
   x2   0
   x3   0
   x4   0
   x5   0
   x6   0
 
  C = 
              x1         x2         x3         x4         x5         x6
   y1          0          0        100      -1100    1.11e+04  -1.111e+05
 
  D = 
       u1
   y1   0
 
Continuous-time state-space model.
```

The poles of the combined system to identify the plant states:

```
pole(sys)
```

System resposnse for using the initial conditions.

```
initial(sys,[1 0 0 0 0 0])
```




```
ans = 6×1 complex
  -0.6667 + 1.3013i
  -0.6667 - 1.3013i
 -10.0000 + 0.0000i
 -80.0000 + 0.0000i
 -81.0000 + 0.0000i
 -82.0000 + 0.0000i
```

System performance

```
stepinfo(sys)
```

```
ans = struct with fields:
        RiseTime: 1.0822
    SettlingTime: 5.7951
     SettlingMin: 4.3202
     SettlingMax: 5.6748
       Overshoot: 19.7439
      Undershoot: 0
            Peak: 5.6748
        PeakTime: 2.4868
```

```
clear all
load('HDD_freqresp.mat');
```

PART (A)

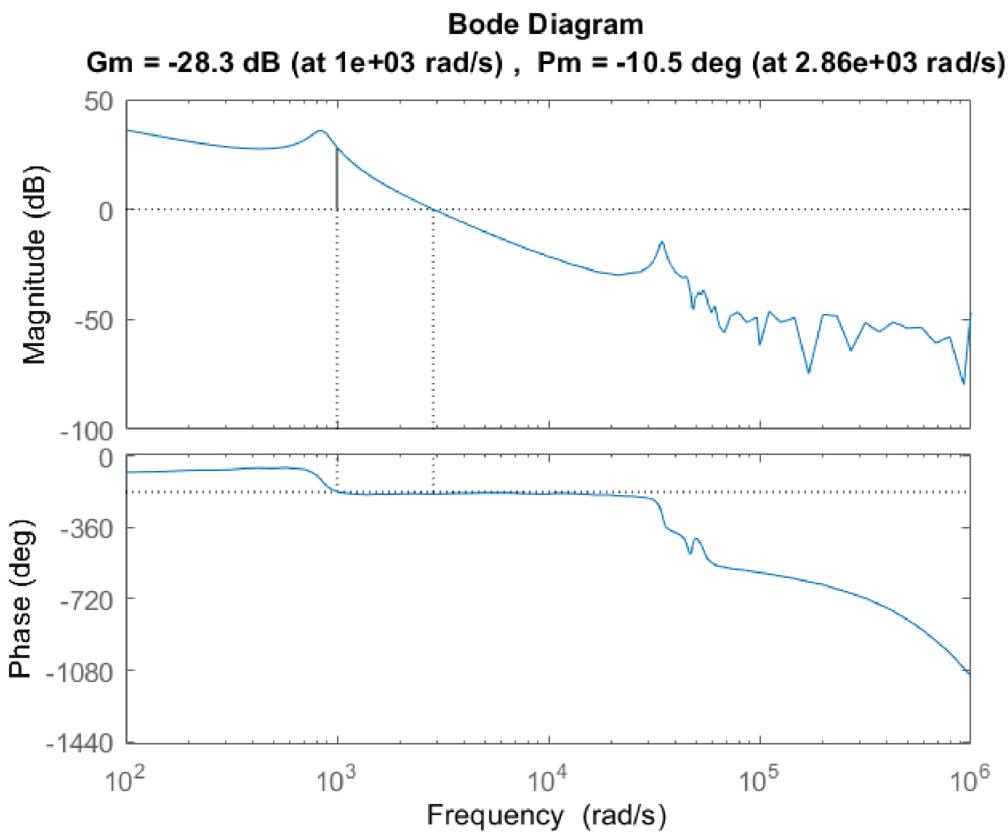
```
Ts = 1/50000;
s = tf('s');
G = HDD_freqresp;
```

Now adding time delay approximation,

```
sys1 = exp(-Ts*s/2);
```

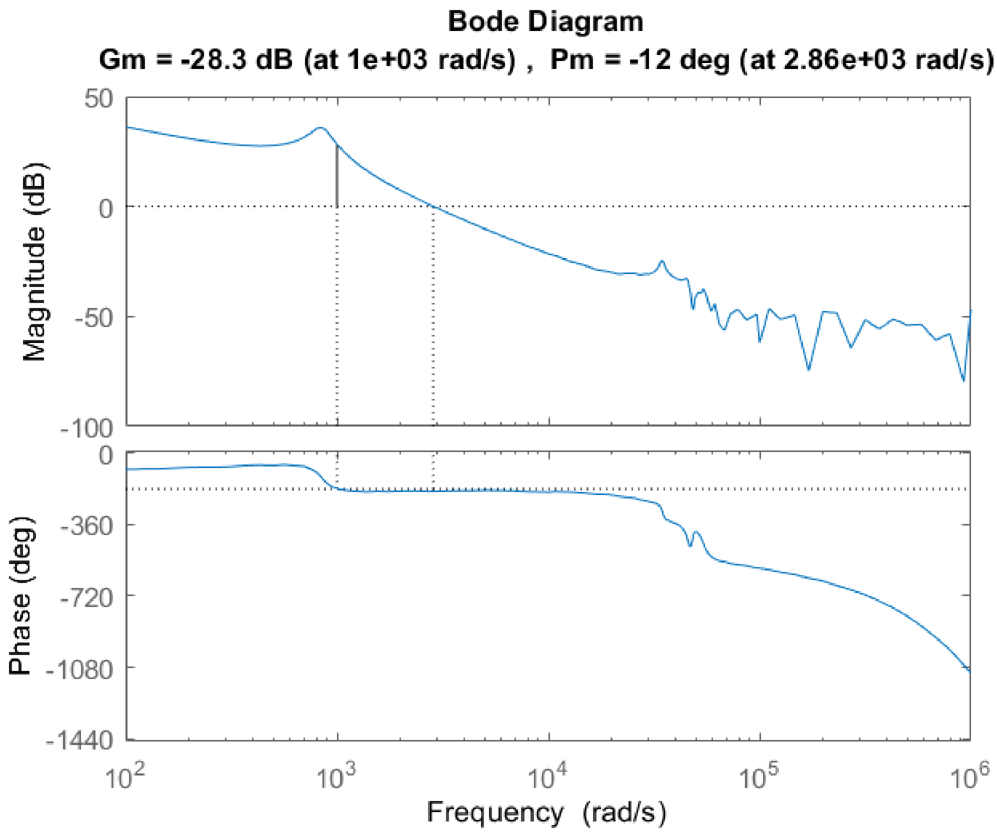
Add PI compensator. the zero is placed at 0.1 x wc = 2*pi*100 rad/s

```
tf_intx = G*sys1*(s+100*2*pi)/s;
margin(tf_intx)
```



Since the bode plot was crossing the x-axis 2 times, I added a notch filter at the second crossing to obtain a single crossing at w = 1 KHz or 2*pi*1000 rad/s.

```
n = notch(10,5000,34300);
tf_intx = tf_intx*n;
margin(tf_intx)
```



Calculate the phase of the system at 1 KHz to add a lead compensator.

```
[m,phase,wout] = bode(tf_intx,2*pi*1000)
```

```
m = 0.1970
phase = 171.0884
wout = 6.2832e+03
```

```
phase_req = -135-phase
```

```
phase_req = -306.0884
```

```
L = lead(phase_req,2*pi*1000)
```

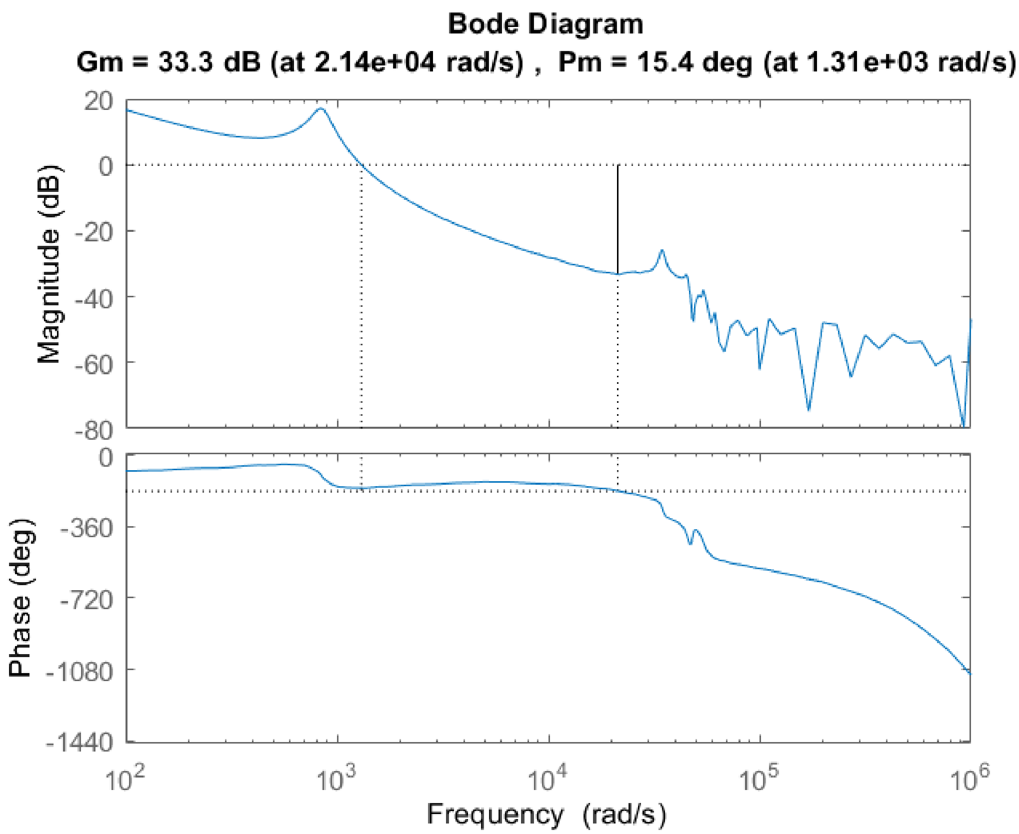
```
L =

    s + 2047
  ------------
  s + 1.929e04

Continuous-time transfer function.
```

New transfer function is now obtained which has the desired phase margin.

```
TF = tf_intx*L;
[mag,phase,wout] = bode(TF,2*pi*1000)
```

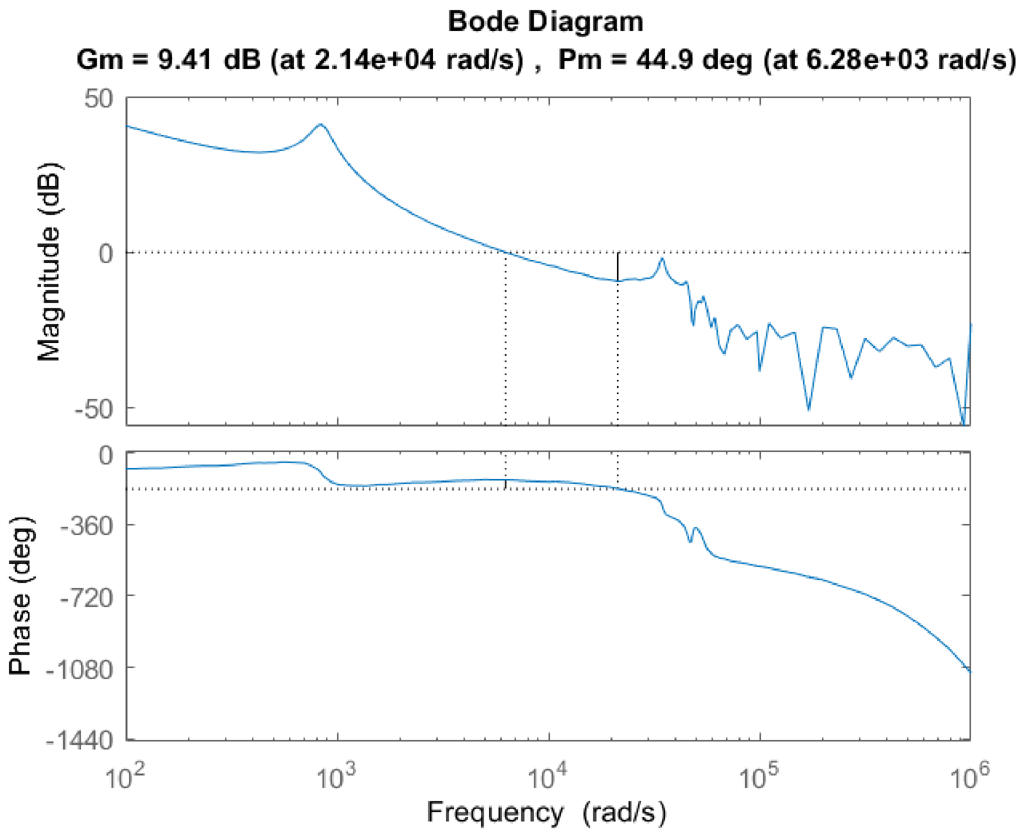
```
mag = 0.0642
phase = -135.0804
wout = 6.2832e+03
```

```
margin(TF)
```



Next, we calculate the magnitude at the desired cross-over frequency to provide an offset to the magnitude plot to achive the desired preformance.

```
tf_final = TF/mag;
margin(tf_final)
```



PART (B)

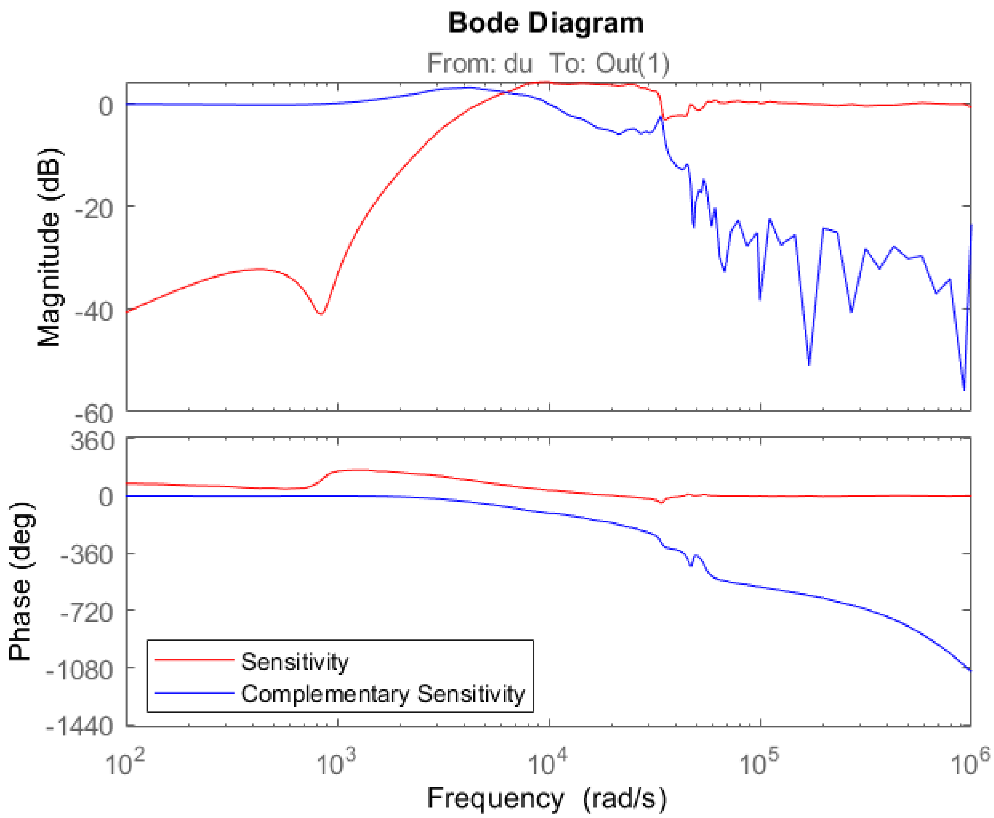
Compute the controller TF by multiplying all the above added compensators.

```
C = n*L*sys1*(s+100*2*pi)/(mag*s)
```

```
C =

                  s^4 + 7675 s^3 + 1.191e09 s^2 + 3.154e12 s + 1.513e15
  exp(-1e-05*s) * -----------------------------------------------------
                   0.06417 s^4 + 2252 s^3 + 9.507e07 s^2 + 1.456e12 s

Continuous-time transfer function.
```

Compute the Sensitivity and complementary sensitivity function using loopsens(). It can be alternatively done by the closed loop transfer funciton. S = I - CL_TF.

```
loops = loopsens(G,C);
bode(loops.Si,'r',loops.Ti,'b')
legend('Sensitivity','Complementary Sensitivity','Location','southwest')
```



Compute the peak gain of the Sensitivity funciton.

```
S_fn = loops.Si;
S_gpeak = getPeakGain(S_fn)
```

```
S_gpeak = 1.6440
```

Compute the peak gain of the Complementary Sensitivity funciton.

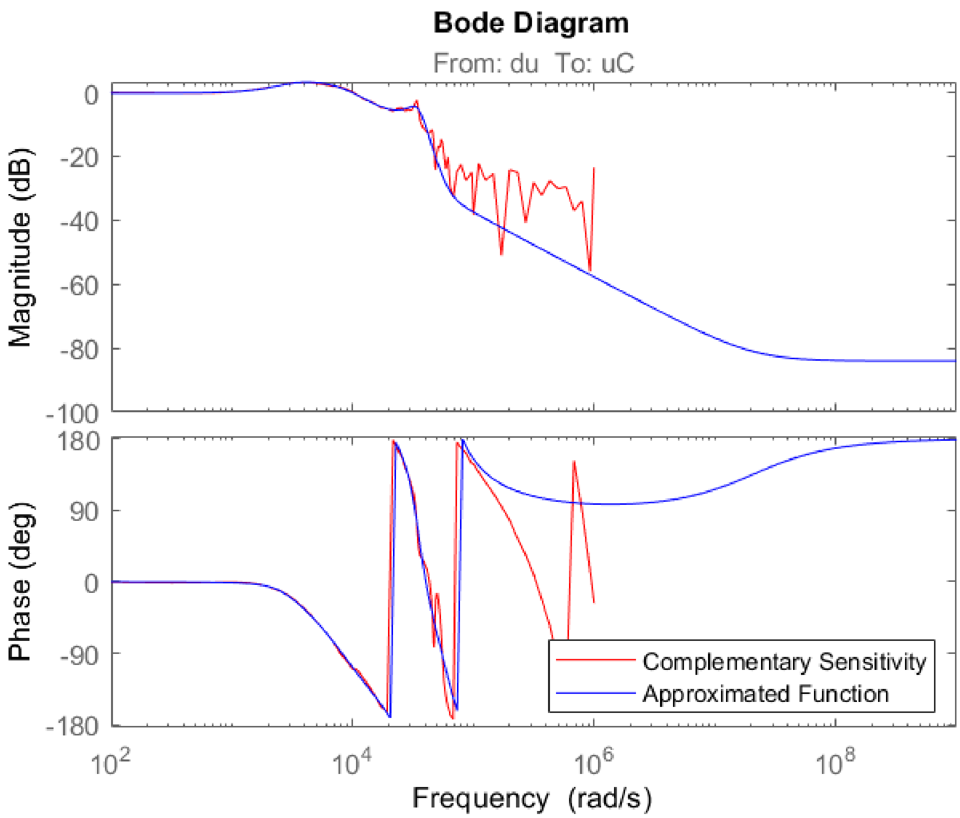
```
CS_fn = loops.Ti;
CS_gpeak = getPeakGain(CS_fn)
```

```
CS_gpeak = 1.4590
```

PART (C)

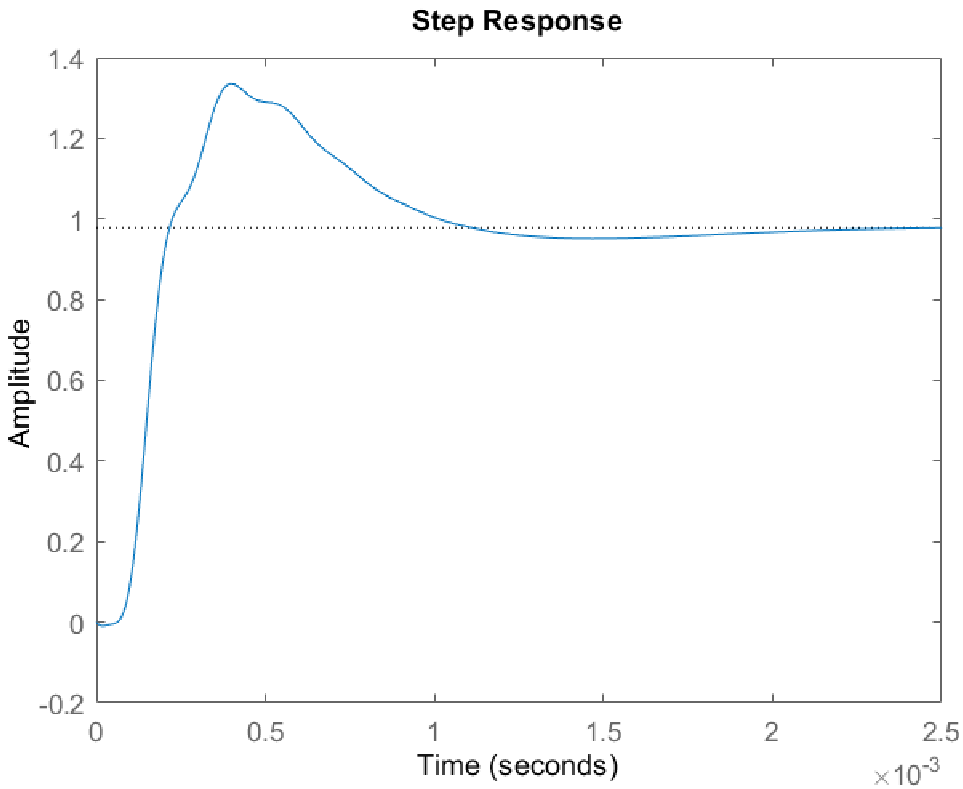
Assuming the number of states to be 8, the approximation of the FRD was obtained, and the bode plot is genereated.

```
N = 8;
B = fitfrd(CS_fn,N);
opts = bodeoptions('cstprefs');
opts.PhaseWrapping = 'on';
bode(CS_fn,'r',B,'b',opts)
legend('Complementary Sensitivity','Approximated Function','Location','southeast')
```



Step response of the approximated complementary system.

```
step(B)
```


Step Response
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
Amplitude
0
0.5
1
1.5
2
2.5
Time (seconds)
×10-3