# ECSE 4440: Final Project

Due on Tuesday, December 15, 2015

*Prof. Wencen Wu 4:00 PM - 5:20 PM*

**Andrew Cunningham & Sabbir Rashid**

December 1, 2015

## Contents

## Problem 1

**Approach:** This part of the project focused on using root locus methods to regulate generator output voltage. The power system provided was expressed in state space and a epow.m was provided to obtain transfer function for individual output states. Using epow.m, open loop transfer functions were obtained for $V$ and $\omega_f$. These transfer functions were then included in the Simulink model below.

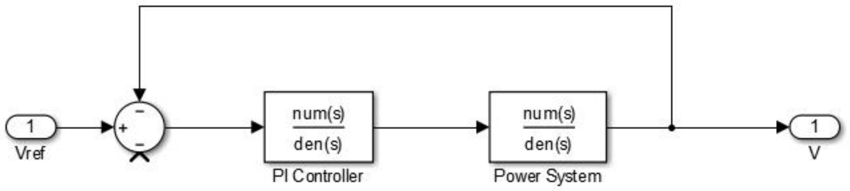


Figure 1: Simulink Voltage Control Model

Throughout the rest of the project, the numerator and the denominator of the controllers to be designed (PI Controller and Damping Controller) were set to be obtained from the Matlab workspace. The controller was then designed in phases. The first stage involved designing the PI controller for the voltage control loop. The second stage involved designing the damping controller to close the damping loop, as shown below.

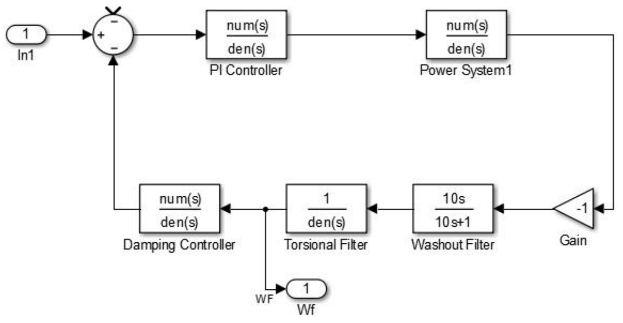


Figure 2: Simulink Damping Control Model

Lastly, the combined model includes both the PI controller and the damping controller.

Problem 1 continued on next page. . .

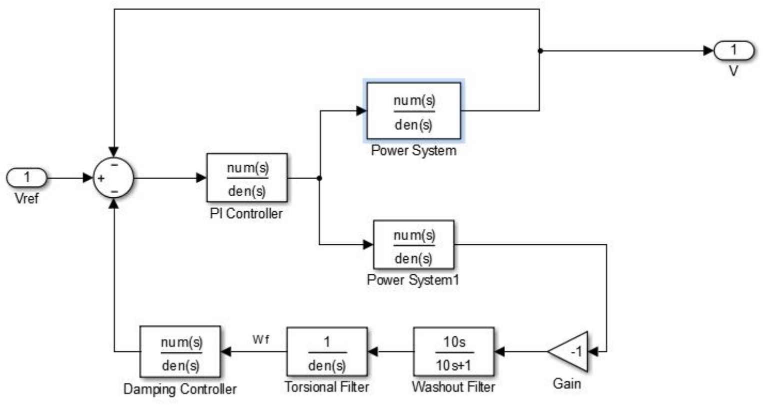


Figure 3: Simulink Combined Model

```
Power system state-space model in arrays A B C D
    outputs are Vterm, w(speed), P
Power system transfer function
    den is denominator
    num1 is numerator from u to Vterm
    num2 is numerator from u to w(speed)
    num3 is numerator from u to P
The system data is saved in power.mat
```

## TASK 1

Develop the linear power system model in MATLAB using V as the output variable. Plot the poles and zeros of the open-loop system and list them in a table.

```
% Form sys1 to be the transfer function from u to V
sysOpenLoop = tf(numV1,denV1)
pzmap(sysOpenLoop)

% Open Loop System Poles and Zeros
pOpen = pole(sysOpenLoop)
zOpen = zero(sysOpenLoop)
```

```
sysOpenLoop =
4.804 s^5 + 436.3 s^4 + 9614 s^3 + 7.257e04 s^2 + 1.024e06 s + 2.3e06
-----------------------------------------------------------------------
s^7 + 181 s^6 + 8903 s^5 + 1.585e05 s^4 + 1.233e06 s^3 + 1.212e07 s^2
                                                 + 3.052e07 s + 3.081e06
```

Problem 1 continued on next page. . .

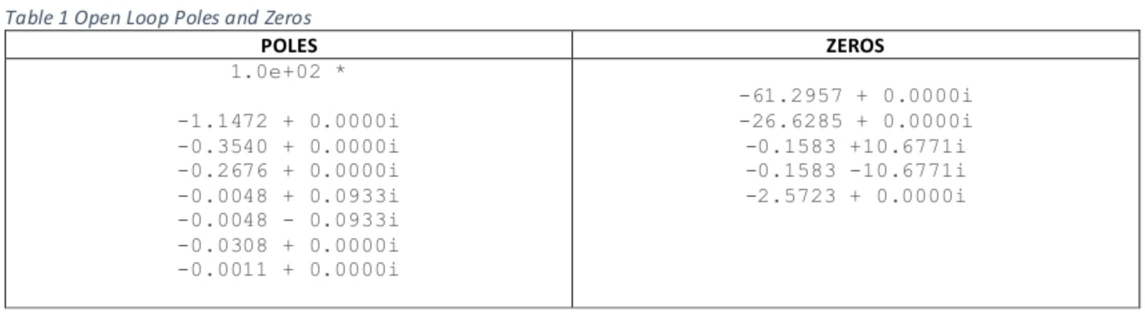


*Table 1 Open Loop Poles and Zeros*

| POLES | ZEROS |
|---|---|
| 1.0e+02 * | |
| | -61.2957 + 0.0000i |
| -1.1472 + 0.0000i | -26.6285 + 0.0000i |
| -0.3540 + 0.0000i | -0.1583 +10.6771i |
| -0.2676 + 0.0000i | -0.1583 -10.6771i |
| -0.0048 + 0.0933i | -2.5723 + 0.0000i |
| -0.0048 - 0.0933i | |
| -0.0308 + 0.0000i | |
| -0.0011 + 0.0000i | |

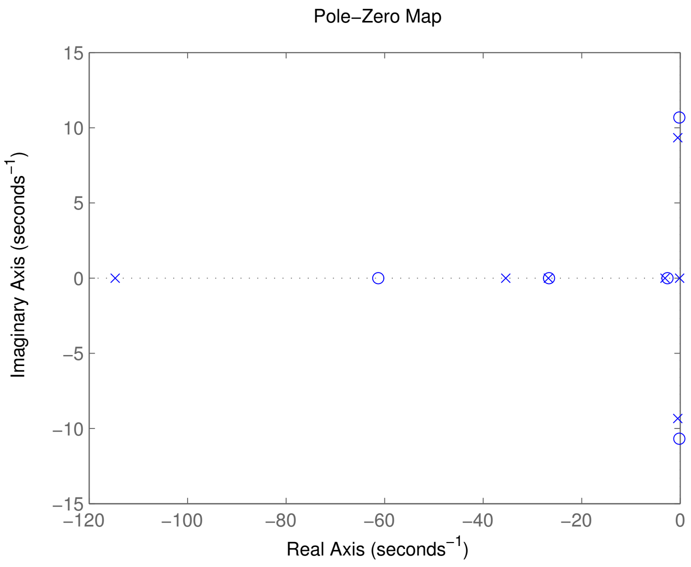


Figure 4: Pole Zero Map

## TASK 2

Simulate and plot the response of the open-loop system for a 0.1 pu step input.

```
step(sysOpenLoop*.1)
```

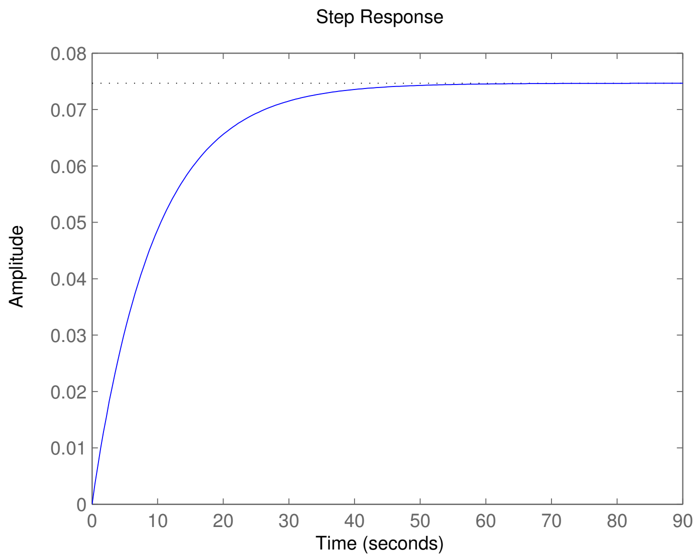
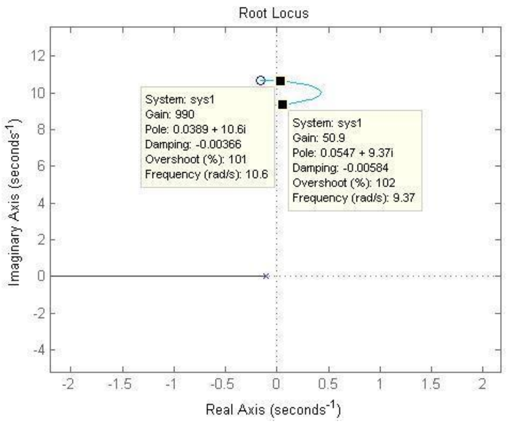


Figure 5: Open-loop Step Response

## TASK 3

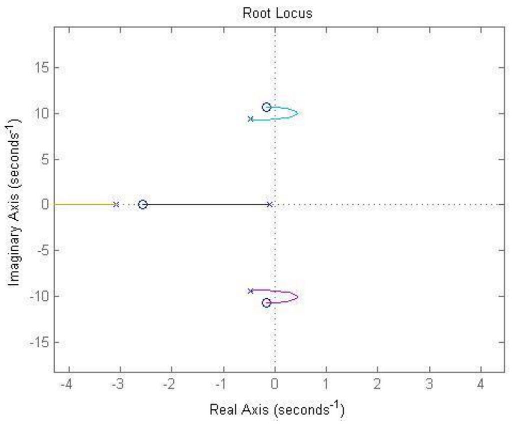
Perform a root-locus analysis of the system using a P (proportional gain) controller $K(s) = K_p$. Plot the root locus. You may need several plots to show all the details. Find approximately the gain $K_u$ when the lightly damping EM mode becomes unstable. The MATLAB data marker feature is useful for finding $K_u$. List all the poles of the closed-loop system for $K(s) = K_u$ in the table of Task 1.

```
figure()
rlocus(sysOpenLoop);
% Using the data marker reveals that:
ku1 = 50.9;
ku2 = 990;

polesku1 = pole(feedback(ku1*sysOpenLoop,1))
polesku2 = pole(feedback(ku2*sysOpenLoop,1))
```



System: sys1
Gain: 990
Pole: 0.0389 + 10.6i
Damping: -0.00366
Overshoot (%): 101
Frequency (rad/s): 10.6

System: sys1
Gain: 50.9
Pole: 0.0547 + 9.37i
Damping: -0.00584
Overshoot (%): 102
Frequency (rad/s): 9.37



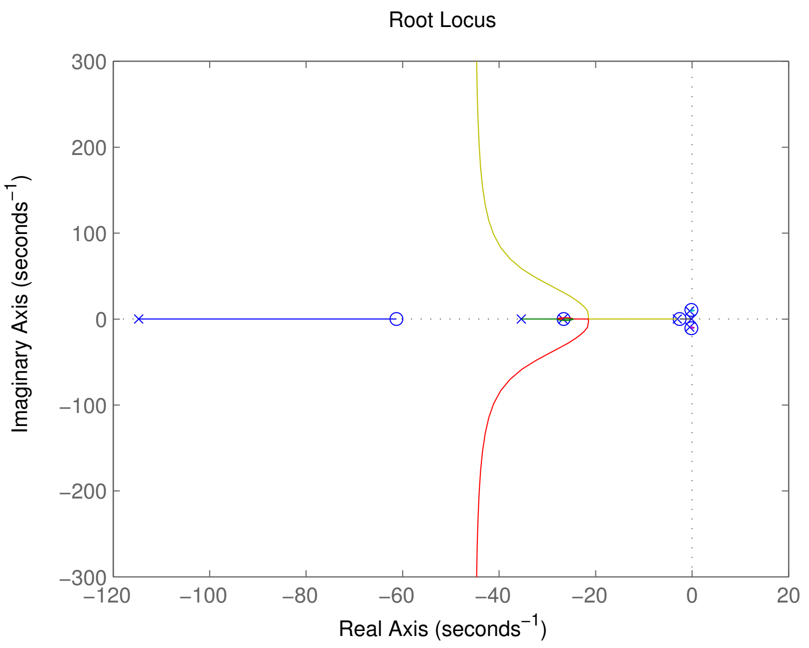
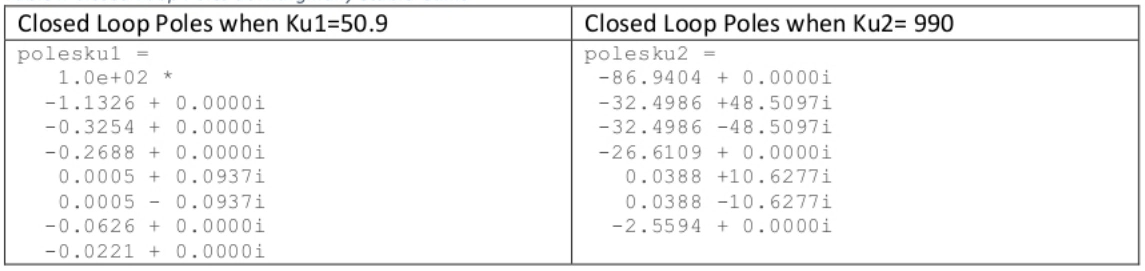


Figure 6: Task 3 Root Locus Plots

*Table 2 Closed Loop Poles at Marginally Stable Gains*



| Closed Loop Poles when Ku1=50.9 | Closed Loop Poles when Ku2= 990 |
|---|---|
| `polesku1 =` | `polesku2 =` |
| `1.0e+02 *` | `-86.9404 + 0.0000i` |
| `-1.1326 + 0.0000i` | `-32.4986 +48.5097i` |
| `-0.3254 + 0.0000i` | `-32.4986 -48.5097i` |
| `-0.2688 + 0.0000i` | `-26.6109 + 0.0000i` |
| `0.0005 + 0.0937i` | `0.0388 +10.6277i` |
| `0.0005 - 0.0937i` | `0.0388 -10.6277i` |
| `-0.0626 + 0.0000i` | `-2.5594 + 0.0000i` |
| `-0.0221 + 0.0000i` | |

## TASK 4

For each of the following values $K_p$ = 10, 20, 30, 40, 50 simulate and plot the response of the closed-loop system and the control signal u for a 0.1 pu step change in $V_{ref}$. Put all the output responses on the same plot, and all the control signals on a separate plot. Be sure to label all the curves. (You can use the text command to put text on a plot.)

```
sys1 = feedback(10*sysOpenLoop,1);
sys2 = feedback(20*sysOpenLoop,1);
sys3 = feedback(30*sysOpenLoop,1);
sys4 = feedback(40*sysOpenLoop,1);
sys5 = feedback(50*sysOpenLoop,1);

figure()
step(.1*sys1,.1*sys2,.1*sys3,.1*sys4,.1*sys5,20)
title('Step Responses of Closed Loop Systems with Specified Gains')
legend('Kp = 10','Kp = 20','Kp = 30','Kp = 40','Kp = 50')
```

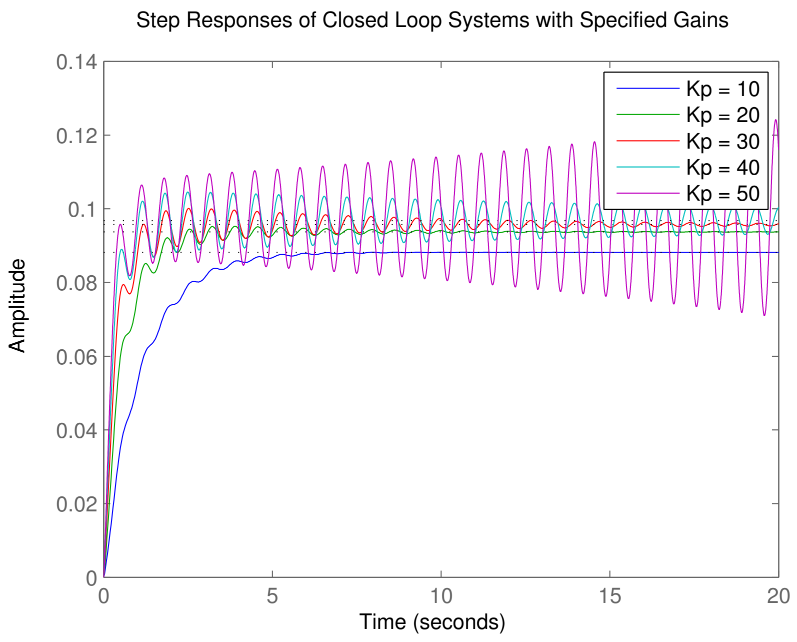


Figure 7: Task 4 Closed-Loop System with Specified Gains

## TASK 5

Design a PI controller

$$K(s) = K_p\left(1 + \frac{K_I}{s}\right) \tag{1}$$

for the voltage control loop such that for a 0.1 pu Vref step input, the rise time of the voltage (V) response satisfies $t_r \leq 0.45$ s and the over-shoot satisfies $M_p \leq 8\%$. In your design, select several sets of $K_p$ and $K_I$ and perform step response simulations. Reasonable ranges of these parameters are $0 < K_p < K_u$ and $0.1 < K_I < 10$. Then from the plots select the most appropriate $K_p$ and $K_I$. Verify your design by simulating a 0.1 pu $V_ref$ step input. The `tstats` function in the tools directory can be readily used to compute the rise time and overshoot. Please note that these two specifications need to be coordinated with the damping control to be designed later. The damping control will increase the rise time and reduce the 4 overshoot. (Why?) So planning ahead, you would need some margin in the rise time and not have to worry too much if the overshoot is somewhat higher than 8%.

To report on your design, list several of your selections of $K_p$ and $K_I$ in a table. Include in the table the resulting rise time and overshoot from the time simulation. Provide an explanation on how you find the desired set of gains $K_p$ and $K_I$. Show the step response of the closed-loop system using the desired gains. Also list the numerical values of the dominant closed-loop poles in a table.

Problem 1 continued on next page. . .

Problem 1

$\boxed{\text{Task 5}}$

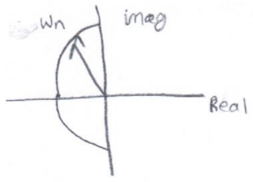
Rise time $\leq .45$

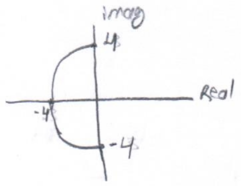
Rise time $\sim \frac{1.8}{\omega_n}$

$\frac{1.8}{\omega_n} \leq .45$

$\frac{\omega_n}{1.8} \geq 2.22$

$\omega_n \geq 4$



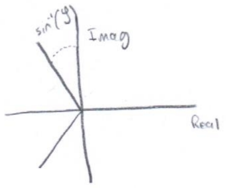


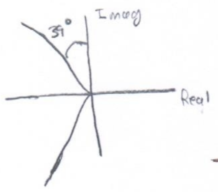
Overshoot $\leq .08$

$e^{-(\pi \zeta)/(\sqrt{1-\zeta^2})} < .08$

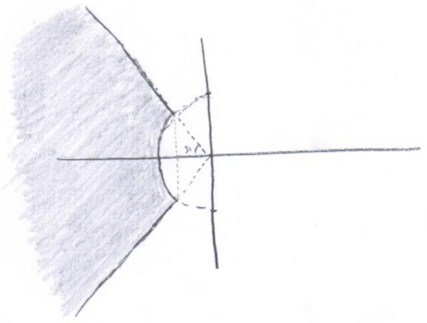
$\zeta \quad \sqrt{\frac{k}{k+1}} \quad K = \left(\frac{\ln(M_P)}{\pi}\right)^2$

$\zeta \geq .626$



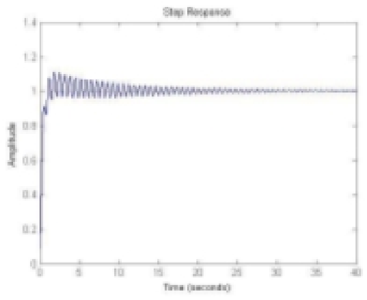
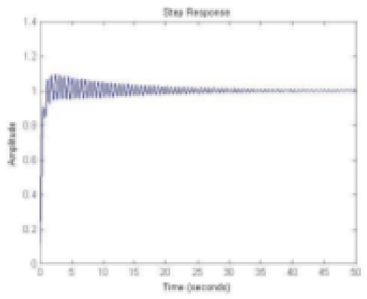
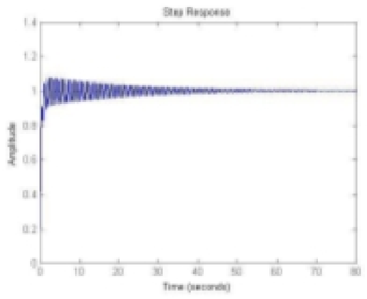


Combined constraints



(Roughly)
All poles must reside in shaded region to meet the constraints

Table 3 Design of damping controller

| KP | KI | Overshoot | Rise Time | Step Response | Closed Loop Poles |
|---|---|---|---|---|---|
| 37.5 | .2 | 9.26% | .461 |  | 1.0e+02 * <br><br> -1.1365 + 0.0000i <br> -0.3341 + 0.0000i <br> -0.2683 + 0.0000i <br> -0.0007 + 0.0931i <br> -0.0007 - 0.0931i <br> -0.0486 + 0.0000i <br> -0.0194 + 0.0000i <br> -0.0021 + 0.0000i |
| 36.3 | .303 | 11% | .4483 |  | 1.0e+02 * <br><br> -1.1368 + 0.0000i <br> -0.3349 + 0.0000i <br> -0.2683 + 0.0000i <br> -0.0008 + 0.0930i <br> -0.0008 - 0.0930i <br> -0.0469 + 0.0000i <br> -0.0186 + 0.0000i <br> -0.0033 + 0.0000i |
| 40.086 | .0959 | 7.6% | .4398 |  | 1.0e+02 * <br><br> -1.1357 + 0.0000i <br> -0.3324 + 0.0000i <br> -0.2684 + 0.0000i <br> -0.0004 + 0.0932i <br> -0.0004 - 0.0932i <br> -0.0516 + 0.0000i <br> -0.0204 + 0.0000i <br> -0.0010 + 0.0000i |

The first two sets of $K_p$ and $K_i$ were obtained by using `rltool`. The structure of the desired controller is equivalent to a lag compensator. A lag compensator was added using `rltool` and then the values of the gains and locations of zeros were tinkered with until a reasonable result was generated. To find the set of $K_p$ and $K_i$ that satisfied the constraint, a simple randomized brute force searcher was created:

```
% RANDOM SEARCHER
Overshoot = 10;
RiseTime = 1;
while Overshoot > 8 || RiseTime > .45 || isnan(Overshoot) || isnan(RiseTime)
    KI = rand*.5;
    KP = rand * 50;
    D = KP*KI*((s/KI + 1))/s;
    testSys = feedback(D*sysOpenLoop,1);
    test = stepinfo(testSys);
    Overshoot = test.Overshoot
    RiseTime = test.RiseTime
end;
```

## TASK 6

Add the PI controller $K(s)$ and the washout filter to your model. Leave the damping controller $K_d(s)$ out for now. Generate the model (or transfer function) from $V_{ref}$ to $\omega_f$. Plot the zeros and poles of the transfer function from $V_{ref}$ to $\omega_f$ with $K(s)$ in the loop.

In order to examine the transfer function from $V_{ref}$ to $\omega_f$, the numerator and denominator of the Damping Controller were set to 1.

```
% Reference voltage to w
[tempNum,tempDen,sysTask6]=simulinkToTF('problem1DampingControlLoop');
% Transfer function with washout filter and K
sysTask6

pzmap(sysTask6)
```

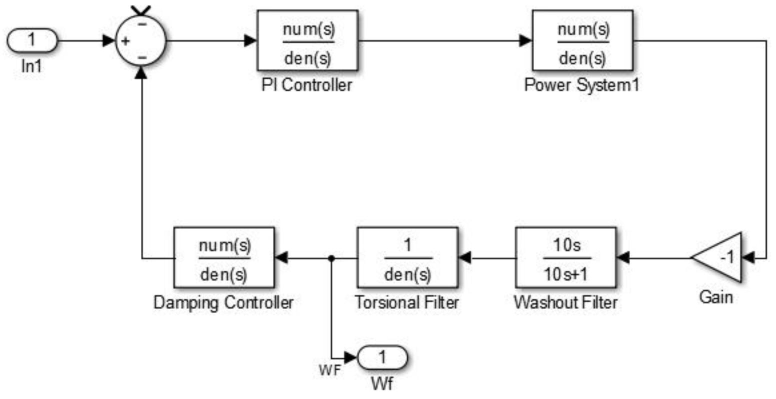


Figure 8: problem1DampingControlLoop.slx

```
sysTask6 =
4.237e04 s^6 + 3.872e06 s^5 + 8.187e07 s^4 + 2.299e08 s^3 + 2.13e07 s^2
+ 2.205e-07 s + 1.358e-09
-----------------------------------------------------------------------
s^11 + 209.4 s^10 + 1.44e04 s^9 + 4.783e05 s^8 + 9.051e06 s^7
+ 1.066e08 s^6 + 8.436e08 s^5 + 5.518e09 s^4 + 1.215e10 s^3
+ 2.301e09 s^2 + 1.141e08 s + 6.68e-07
```

Problem 1 continued on next page. . .

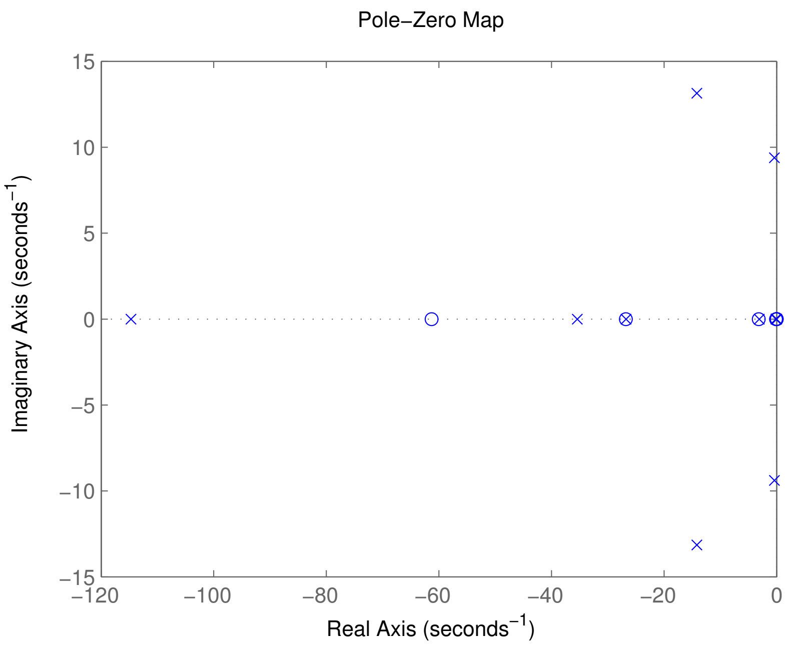


Figure 9: Damping Loop pzmap

## TASK 7

Let the damping controller $K_d(s)$ be a proportional gain controller and perform a rootlocus analysis. Plot the root locus and measure the angle of departure $\phi_{dep}$ from the EM mode.

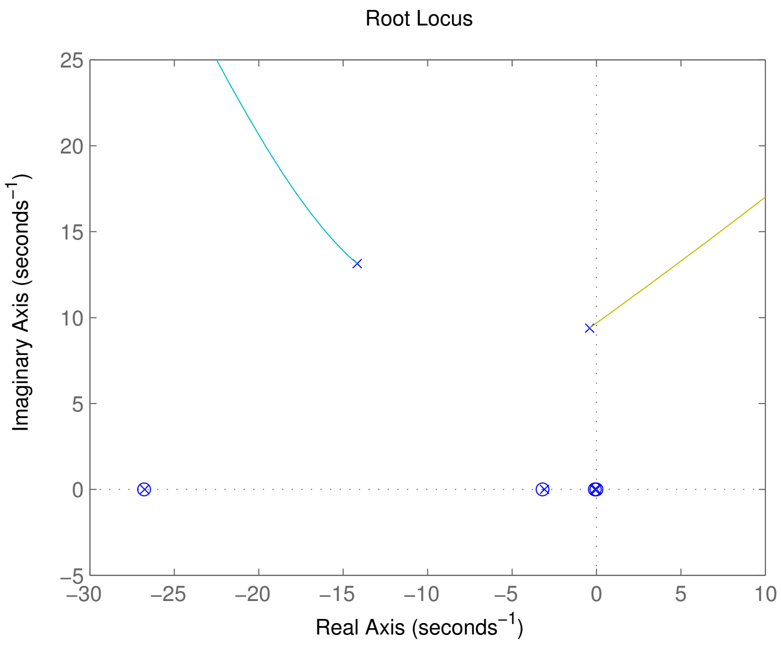
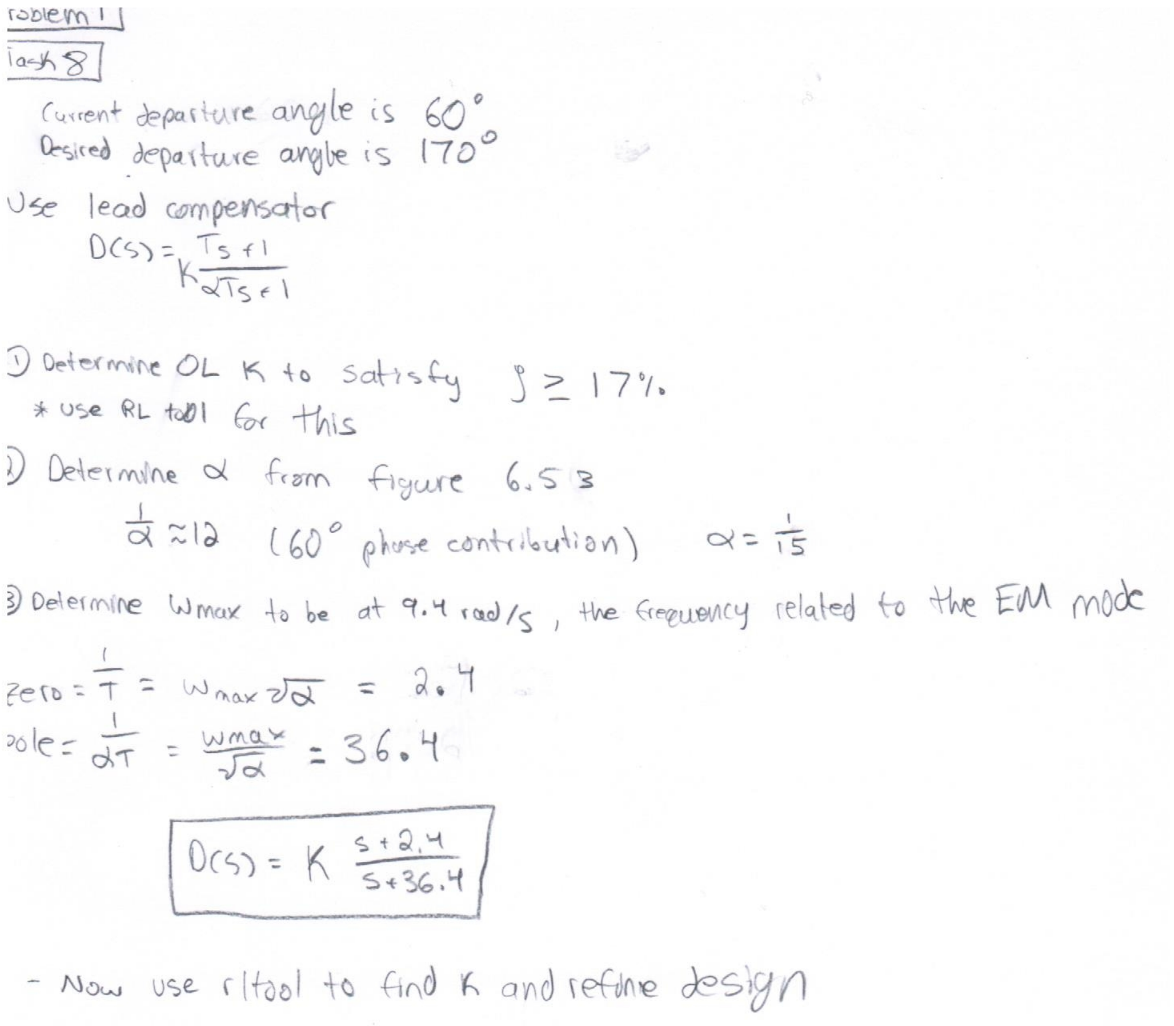


Figure 10: Root Locus of Damping Controller in Damping Loop

The departure angle is approximately 60°.

## TASK 8

Based on $\phi_{dep}$, design a phase (lead or lag, whichever is appropriate) compensation as your damping controller $K_d(s)$. Use the chart on phase compensation to select your design such that for the compensated system, the angle of departure from the swing mode with the positive imaginary part is 170°. The compensation you need may be higher than first order. Perform a root-locus analysis. Plot the root-locus and select the gain of $K_d(s)$ to achieve a damping ratio $\zeta \geq 17\%$ for the EM mode. Mark on the root-locus plots the gain selection and the location of the closed-loop poles. Also list the numerical values of the dominant closed-loop poles in the table of Task 5.



roblem 1

Task 8

Current departure angle is $60^\circ$
Desired departure angle is $170^\circ$

Use lead compensator

$$D(s) = K\frac{Ts+1}{\alpha Ts+1}$$

1) Determine OL K to satisfy $\zeta \geq 17\%$
* use RL tool for this

2) Determine $\alpha$ from figure 6.53

$\frac{1}{\alpha} \approx 12$ ($60^\circ$ phase contribution) $\alpha = \frac{1}{15}$

3) Determine $\omega_{max}$ to be at 9.4 rad/s, the frequency related to the EM mode

$$\text{zero} = \frac{1}{T} = \omega_{max}\sqrt{\alpha} = 2.4$$
$$\text{pole} = \frac{1}{\alpha T} = \frac{\omega_{max}}{\sqrt{\alpha}} = 36.4$$

$$\boxed{D(s) = K\frac{s+2.4}{s+36.4}}$$

- Now use rltool to find K and refine design

The steps above were followed to find the starting specifications for a compensator. Furthermore, the compensator used is second order, formed by cascading two of the first order compensators. `rltool` was then used to refine this design and obtain the following controller:

```
% Finding Damping Controller, we need to compensate 110 degrees to get to
% 170 degree departure angle
% rltool(sysTask6)

DampingControllerSys = .1*[14*(s+2.94)/(s+34.9)]^2;
[Damping_Controller_Num,Damping_Controller_Den] = tfdata(DampingControllerSys,'v');
[tempNum,tempDen,finalSys]=simulinkToTF('problem1');
```

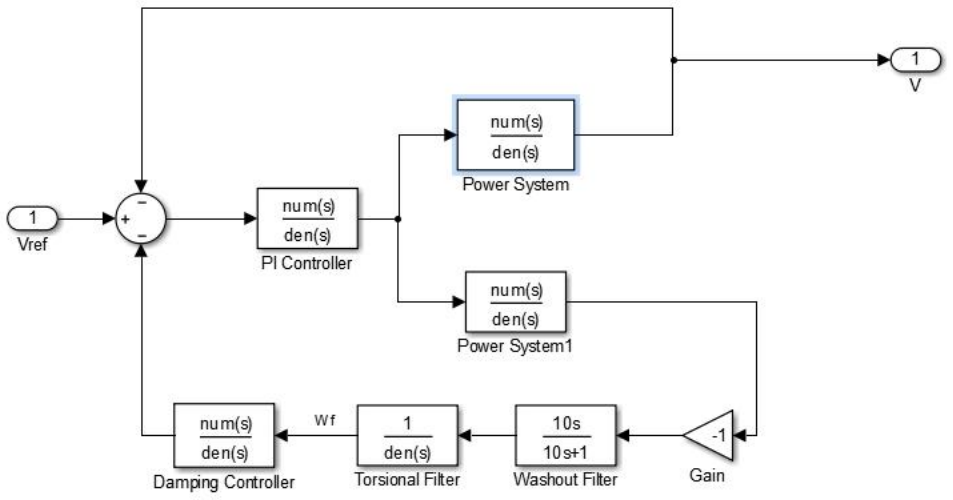


Figure 11: Simulink Combined Model - problem1.slx

## TASK 9

Add the damping controller $K_d(s)$ to complete the model. To demonstrate your design, apply a 0.1 pu $V_{ref}$ step input and check to see if the specifications $t_r \leq 0.45$ s and $M_p \leq 8\%$ have been satisfied. Plot $V$ and the output signals of $K(s)$ and $K_d(s)$.

Figure 11 shows the Simulink model used to obtain the transfer function from $V_{ref}$ to $V$. The PI compensator and damping controller were imported into the Simulink model by setting the numerator and denominator of those blocks to their corresponding values in the Matlab workspace.

```
stepinfo(finalSys)
step(finalSys*.1)
```

```
ans =
    RiseTime: 0.4461
    SettlingTime: 8.9486
    SettlingMin: 0.8373
    SettlingMax: 1.0563
    Overshoot: 5.6300
    Undershoot: 0
    Peak: 1.0563
    PeakTime: 1.8253
```

Problem 1 continued on next page. . .

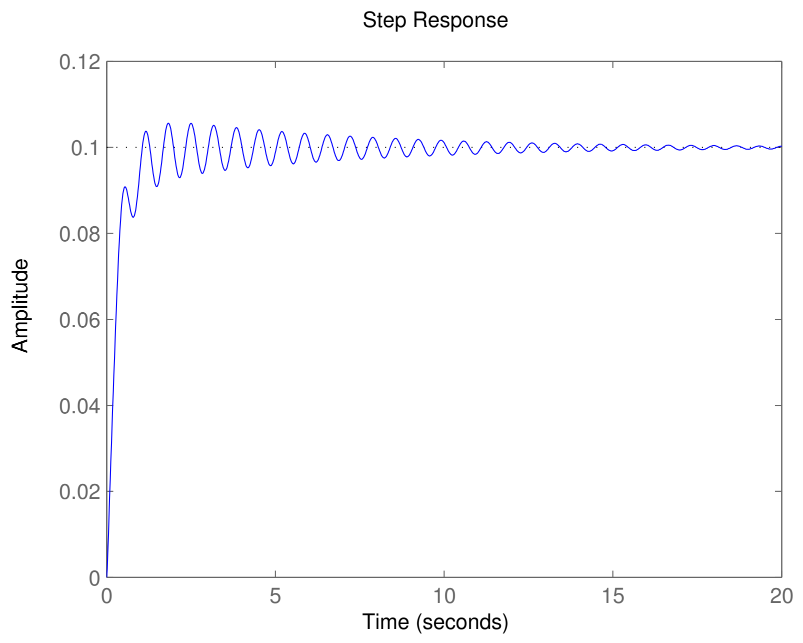


Figure 12: Success! Step Response of Final System

## Problem 2

### TASK 1

To develop a state-space form of the linear power system model using $V$ as the output variable, we consider only the top row of the given $\mathbf{C}$ matrix. The resulting state-space equations are as follows,

$$\dot{x} = \mathbf{A}x + \mathbf{B}u \ , \tag{2}$$
$$y = \mathbf{C}x \ , \tag{3}$$

where

$$x = \begin{bmatrix} \delta & \omega & E'_q & \psi_d & E'_d & \psi_q & V_R \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 377.0 & 0 & 0 & 0 & 0 & 0 \\ -0.246 & -0.156 & -0.137 & -0.123 & -0.0124 & -0.05456 & 0 \\ 0.109 & 0.262 & -2.17 & 2.30 & -0.0171 & -0.0753 & 1.27 \\ -4.58 & 0 & 30 & -34.3 & 0 & 0 & 0 \\ -0.161 & 0 & 0 & 0 & -8.44 & 6.33 & 0 \\ -1.70 & 0 & 0 & 0 & 15.2 & -21.5 & 0 \\ -33.9 & -23.1 & -6.86 & -59.5 & 1.5 & 6.63 & -114.0 \end{bmatrix} ,$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 16.4 \end{bmatrix} , \ \mathbf{C} = \begin{bmatrix} -0.123 & 1.05 & 0.23 & 0.207 & -0.105 & -0.46 & 0 \end{bmatrix} .$$

The frequency response of the system from $u$ to $V$ is shown in the figure below.

```
% Develop a state-space form of the linear power system model using V as
% the output variable.
u2V=tf(num1,den1);
% Plot the frequency response from u to V
bode(u2V);
```

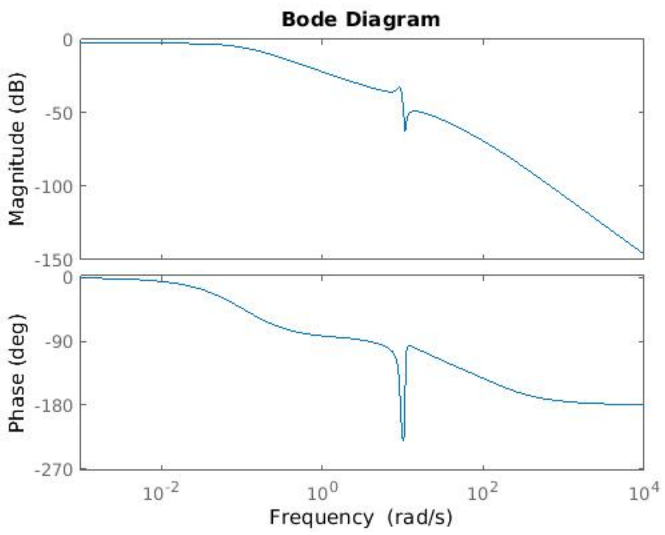


Figure 13: Frequency Response of the Power System from $u$ to $V$

The DC gain for the system is -2.54 dB, as shown in the figure below. In terms of magnitude, this corresponds to a DC gain of 0.7464.

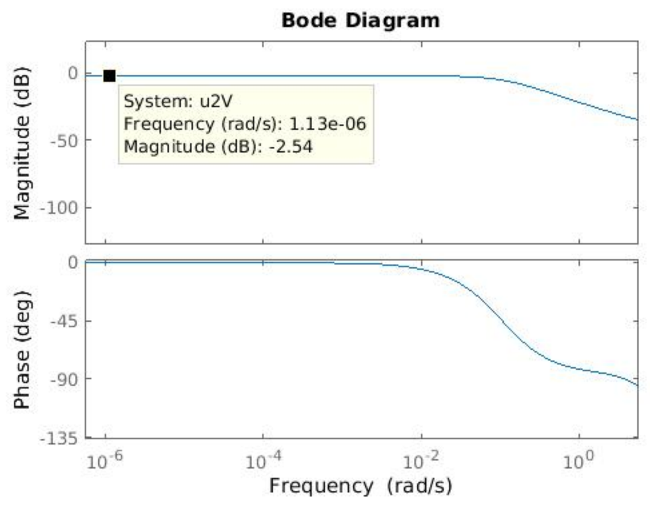


Figure 14: DC Gain of the Power System from $u$ to $V$

As is apparent in the Bode plot, the system never crosses the 0 dB line. Therefore, the gain crossover frequency, $\omega_g$ and the 0 dB crossover frequency, $\omega_c$ are undefined in this case. Therefore, there is no corresponding Phase Margin. Nevertheless, from the Bode plot diagram with labeled stability margins indicates a Gain Margin of 33 dB (44.67 in magnitude) at the phase crossover frequency of $\omega_p = 9.34$ rad/sec, as shown in the figure below.

Problem 2 continued on next page. . .

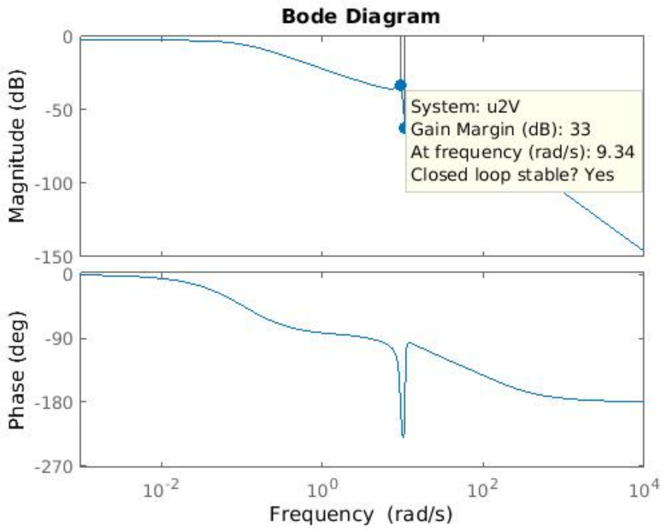


Figure 15: Gain Margin of the Power System from $u$ to $V$

The Matlab **margin** command can also be used to find the Gain Margin.

```
% Crossover values and margins
[Gm,Pm,Wgm,Wpm]=margin(u2V);
```

## TASK 2

For the next task, we wish to design a phase-lag controller $K_l(s)$ such that the steady-state error is less than 0.01 pu and the phase margin is at least 80°. From the steady-state error requirement, we can determine the minimum 0 dB gain as follows.

$$e_{ss} = \frac{1}{1+K_0} \leq 0.01 \ , \tag{4}$$

$$K_0 \geq 99 \ . \tag{5}$$

Since the lag-compensator is of the form,

$$D_C(s) = \alpha \frac{Ts+1}{\alpha Ts+1} \ , \tag{6}$$

the $K_0$ requirement sets a minimum value for $\alpha$. We choose $\alpha$ as follows,

$$\alpha = \frac{K_0}{\text{low frequency gain}} = \frac{99}{0.7464} = 133 \ . \tag{7}$$

In order to have a PM of at least 80°, as well as an included additional margin of 10°, we determine the expected gain crossover frequency from the original plot, where the corresponding PM is 90°. We find

$$\omega_c = 3.7 \text{ rad/sec} \ . \tag{8}$$

The corner frequency, $1/T$, is chosen the be 5 times slower than the expected gain crossover frequency,

$$\frac{1}{T} = \frac{3.7}{5} = 0.74 \ , \tag{9}$$

$$T = 1.35 \ . \tag{10}$$

Problem 2 continued on next page. . .

Therefore, our lag-compensator will be

$$D_C(s) = 133\frac{1.35s + 1}{179.73s + 1}\ . \tag{11}$$

```
% lag-compensator
D_C=133*(1.35*s+1)/(179.1*s+1);
figure;
bode(D_C*u2V)
```

The resulting Bode plot of the compensated system is shown below.

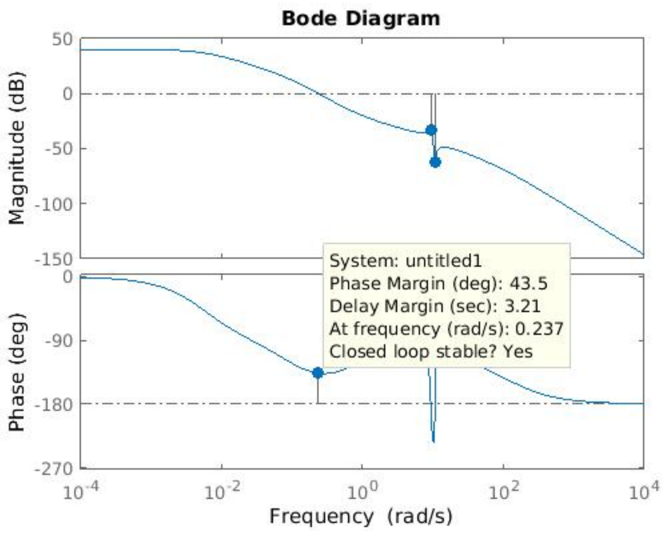


Figure 16: Added Lag-Compensator with Corner Frequency 5 times slower than $\omega_c$

We find that the phase-margin specifications are not met. To improve of this, we choose the corner frequency to be even smaller, at 25 times slower than the Crossover frequency.

```
% lag-compensator
%D_C=133*(1.35*s+1)/(179.1*s+1);
D_C=133*(6.75*s+1)/(900*s+1);
figure;
bode(D_C*u2V)
```

The resulting compensator is of the form

$$D_C(s) = 133\frac{6.75s + 1}{900s + 1}\ . \tag{12}$$

The Bode plot is shown below, with the requirements met.

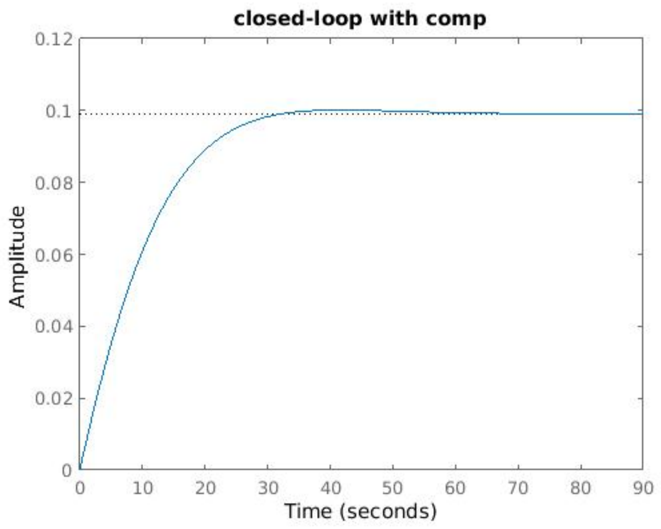


Figure 18: Step-Response of Closed-Loop System with Compensator

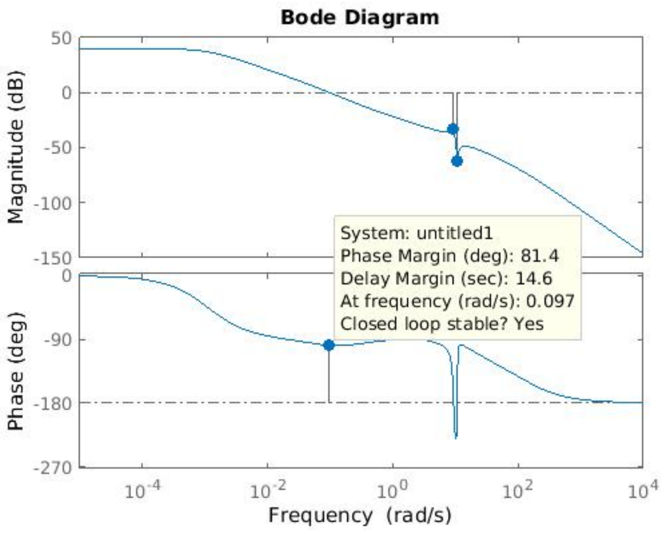


Figure 17: Added Lag-Compensator with Corner Frequency 25 times slower than $\omega_c$

## TASK 3

The response of the closed-loop system with a 0.1 pu step-input is shown below.

```
figure;
u2VDfb=feedback(D_C*u2V,1);
step(0.1*u2VDfb);
title('closed-loop with comp');
```

We can use the Matlab `stepinfo` command to find the characteristics of the curve, listed below.

```
stepinfo(u2VDfb);
```

```
RiseTime: 18.7888
SettlingTime: 27.5469
SettlingMin: 0.8970
SettlingMax: 1.0012
Overshoot: 1.1284
Undershoot: 0
Peak: 1.0012
PeakTime: 41.8753
```

The performance of this compensator fell short of the performance of the PI compensator in Problem 1 in both overshoot and rise time. We can find the poles using the `pole` command.

```
% poles of closed-loop system with compensator
polesDC = pole(feedback(D_C*u2V,1))
```

The dominant $K_l$ closed-loop poles are listed below,

$$\begin{aligned} -114.69 \ &, \\ -35.35 \ &, \\ -26.76 \ &, \\ -0.47 \pm 9.33j \ &, \\ -3.1 \ &, \\ -0.09 \pm 0.06j \ &. \end{aligned}$$

## TASK 4

The next task involved generating a state-space model $G_\omega(s)$ from $V_{ref}$ to $\omega$ and using it to generate a model $Q(s)$ by taking a portion of the A matrix. Then we connect Q(s) in series with the washout and torsional filters to form $F(s)$ and plot the frequency response of $F(s)$, using a frequency range of 1 to 100 rad/s.

```
u2w = tf(num2,den2);

testSys = feedback(D_C*u2V,1);
testSys = minreal(testSys, .01)
originalPoles = pole(u2V);
newPoles = pole(testSys);
zpk(testSys);

K = place(A,B,newPoles);

A1 = (A-B*K);
% State Space Model
a_23=A1(2,3:7);
a_31=A1(3:7,1);
a_32=A1(3:7,2);
a_33=A1(3:7,3:7);

[num_a,den_a]=ss2tf(a_33,a_31,a_23,0);
sysQ=tf(num_a,den_a);
sysF=sysQ*Gwo*Gtor;
```

```

bode(sysF,{1,100})
```

The resulting frequency response of $F(s)$ is shown in the figure below.

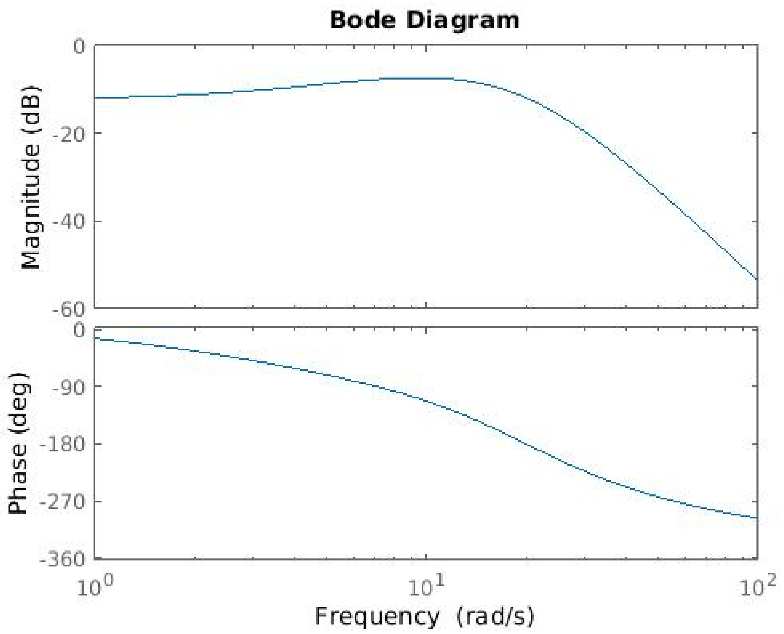


Figure 19: Frequency Response of $F(s)$

## TASK 5

Next we designed a phase-lead compensator $K_\phi(s)$.

```
Wmax = 20
theta = 56
theta = theta/180 * pi;
a = (1-sin(theta))/(1+sin(theta))
zero = Wmax*(a)^.5
pole = Wmax/(a)^.5
D_K = ((s/zero+1)/(s/pole+1))^2;
D_K = D_K*37;
bode(sysF*D_K,{1,100})
```

The frequency response of $F(s)K_\phi(s)$ is shown below.

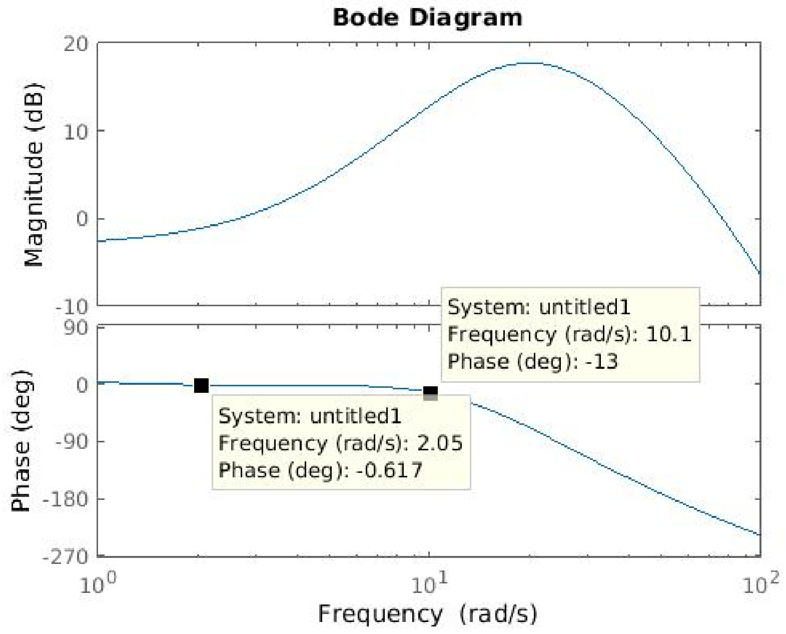


Figure 20: Frequency Response of $F(s)K_\phi(s)$

As is apparent from the figure, the specification that the phase is between 6° to $-15°$ from 2 to 10 rad/sec is met.

## TASK 6

The next task included performing a root-locus analysis to select the gain of $K_\phi(s)$ to achieve a damping ratio $\zeta = 15$ for the EM mode. Mark on the root-locus plot the gain selection and the location of the closed-loop poles.

```
figure()
rlocus(Gwo*Gtor*D_K*u2W*-1)
```

The resulting root-locus plot is shown below.

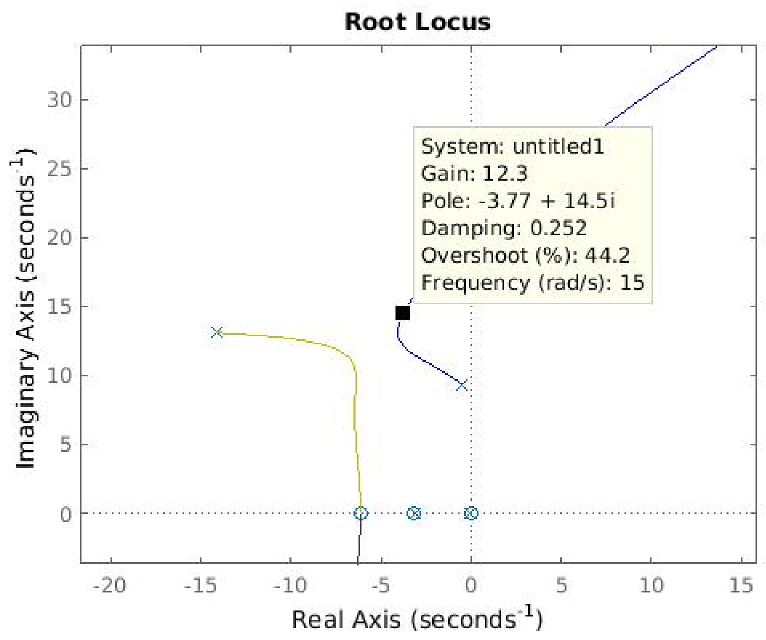


Figure 21: Root Locus with Gain Selection

We found that a gain of 13 maximized the damping ratio. In our code we divided by this gain to achieve the desired result.

```
% Apply gain
D_K= D_K/13
```

## TASK 7

For the final task in this part of the assignment, we added the damping controller to complete the system.

```

[Damping_Controller_Num,Damping_Controller_Den]=tfdata(D_K,'v');
[PI_Controller_Num,PI_Controller_Den]=tfdata(D_C,'v');
[tempNum,tempDen,tfSysFinal]=simulinkToTF('problem1');

stepinfo(.1*tfSysFinal)
```

The resulting characteristics using the `stepinfo` command are shown below.

```
ans =
        RiseTime: 0.4537
    SettlingTime: 4.5339
     SettlingMin: 0.0902
     SettlingMax: 0.1051
       Overshoot: 5.6324
      Undershoot: 0
            Peak: 0.1051
        PeakTime: 1.8475
```

The plots for $V$ and the output signals of $K_l(s)$ and $K_\phi(s)$ are shown below.

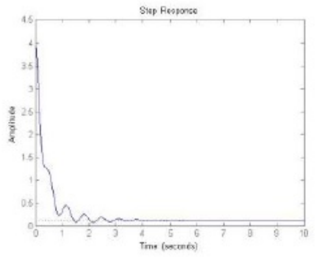
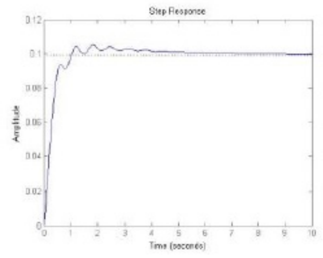
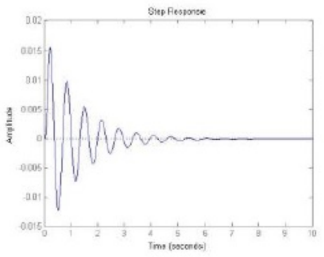
| **VOUT** | **VK$_\phi$** | **VK$_L$** |
|---|---|---|
|  |  |  |

Figure 22: Final Plots

## Problem 3

### SYSTEM DESCRIPTION

```
s = tf('s');
epow
Gwo = 10*s/(1+10*s);
Gtor = 1/(.0027*s^2 + .0762*s + 1);
Damping_Controller_Num = [0];
Damping_Controller_Den = [1];
PI_Controller_Den = [1];
PI_Controller_Num = [1];
```

```
Power system state-space model in arrays A B C D
outputs are Vterm, w(speed), P
Power system transfer function
den is denominator
num1 is numerator from u to Vterm
num2 is numerator from u to w(speed)
num3 is numerator from u to P
The system data is saved in power.mat
```

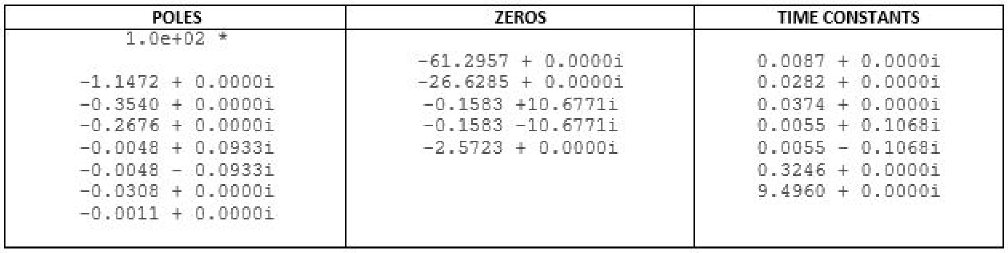
### TASK 1

Develop the state-space form of the linearized power system model in MATLAB or Simulink using V as the output variable. List the poles and the zeros of the single-input, single-output system. Find the time constants of the poles (the time constant is equal to the inverse of the negative of the real part of the pole).

```
% Get the system in control canonical form
tfSys = tf(num1,den1);
[A,B,C,D]=tf2ss(num1,den1)

poles = pole(tfSys);
zeros1 = zero(tfSys);
timeConstants = (-poles).^-1;
```

```
A =

1.0e+07 *

-0.0000   -0.0009   -0.0159   -0.1233   -1.2119   -3.0517   -0.3081
0.0000         0         0         0         0         0         0
0    0.0000         0         0         0         0         0
0         0    0.0000         0         0         0         0
0         0         0    0.0000         0         0         0
0         0         0         0    0.0000         0         0
0         0         0         0         0    0.0000         0
```

```
B =

1
0
0
0
0
0
0


C =

1.0e+06 *

0    0.0000    0.0004    0.0096    0.0726    1.0245    2.3001

D = 0
```

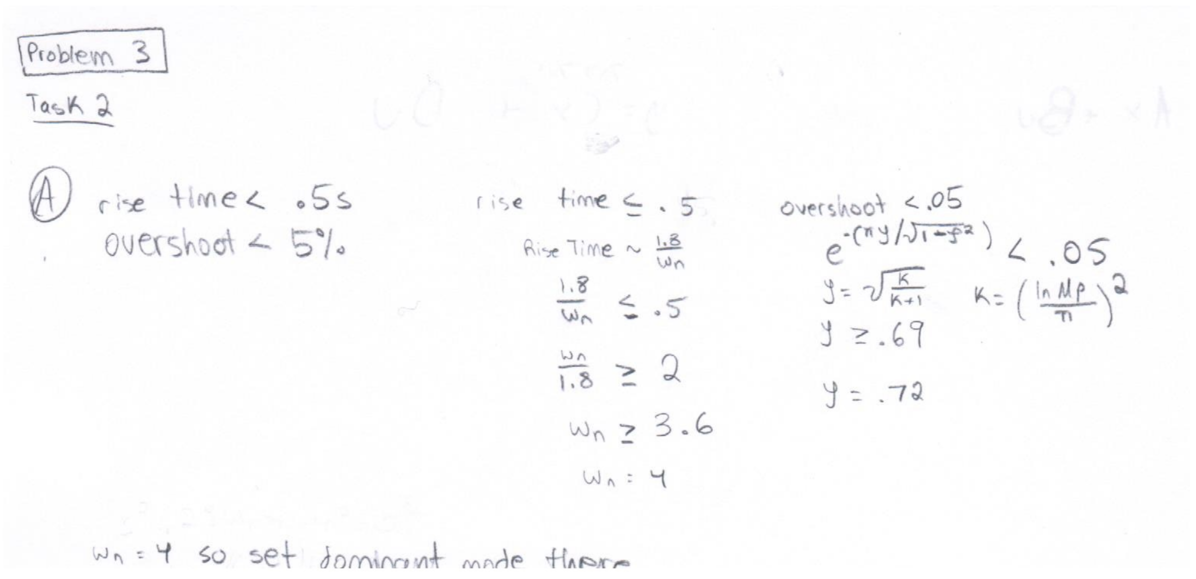


| POLES | ZEROS | TIME CONSTANTS |
|---|---|---|
| 1.0e+02 * | | |
| -1.1472 + 0.0000i | -61.2957 + 0.0000i | 0.0087 + 0.0000i |
| -0.3540 + 0.0000i | -26.6285 + 0.0000i | 0.0282 + 0.0000i |
| -0.2676 + 0.0000i | -0.1583 +10.6771i | 0.0374 + 0.0000i |
| -0.0048 + 0.0933i | -0.1583 -10.6771i | 0.0055 + 0.1068i |
| -0.0048 - 0.0933i | -2.5723 + 0.0000i | 0.0055 - 0.1068i |
| -0.0308 + 0.0000i | | 0.3246 + 0.0000i |
| -0.0011 + 0.0000i | | 9.4960 + 0.0000i |

## TASK 2

With the damping control loop open, design an observer-based controller $K_1(s)$ and the scalar gain N such there for a unit step input $V_{ref}$, the rise time is less than 0.5 s, the overshoot is less than 5%, and the steady-state voltage error is zero

Problem 3 continued on next page. . .



Problem 3

Task 2

A rise time < .5s

overshoot < 5%

rise time $\leq .5$

Rise Time $\sim \frac{1.8}{w_n}$

$\frac{1.8}{w_n} \leq .5$

$\frac{w_n}{1.8} \geq 2$

$w_n \geq 3.6$

$w_n = 4$

overshoot < .05

$e^{-(\pi y/\sqrt{1-y^2})} < .05$

$y = \sqrt{\frac{K}{K+1}}$ $K = \left(\frac{\ln Mp}{\pi}\right)^2$

$y \geq .69$

$y = .72$

$w_n = 4$ so set dominant mode there

Figure 23: Hand derivation of Pole Placement Locations

**SUBTASK A** Design a full-state feedback control to place the poles of the voltage closed-loop system at the desired locations to reduce the dominant time constant(s). There is no need to move the poles that do not significantly influence the rise time. In particular, do not move the EM modes.

Controller is of form $\dot{x} = \mathbf{A}x - \mathbf{BK}x$ which simplifies to $\dot{x} = (\mathbf{A} - \mathbf{BK})x$ for full state feedback. Place the poles of the system in the locations derived above to achieve the desired specifications. The only pole that needs to be moved is the dominant pole.

```
    p = poles(1:6)';
    p = [p -4];
    K = place(A,B,p);
    controllerPoles=eig(A-B*K)
```

```
controllerPoles =

1.0e+02 *

-1.1472 + 0.0000i
-0.3540 + 0.0000i
-0.2676 + 0.0000i
-0.0048 + 0.0933i
-0.0048 - 0.0933i
-0.0400 + 0.0000i
-0.0308 + 0.0000i
```

**SUBTASK B** Design an observer so that the states of the system can be estimated using the voltage measurement. You have to assign the observer poles such that their transients decay faster than the full-state feedback poles. Again, there is no need to move the observer poles that do not significantly influence the estimation transients.

Observer is of form $\dot{\hat{x}} = \mathbf{A}\hat{x} + \mathbf{B}u + \mathbf{L}(y - \mathbf{C}\hat{x})$. Find $\mathbf{L}$ to design observer. To design an observer that meets the specifications, place real part of dominant mode further out than in the controller:

```
p = poles(1:6)';
p = [p -7];

% Duality is our friend!
L = place(A',C',p)';
observerPoles=eig(A-L*C)
```

```
observerPoles =

1.0e+02 *

-1.1472 + 0.0000i
-0.3665 + 0.0000i
-0.2674 + 0.0000i
-0.0048 + 0.0933i
-0.0048 - 0.0933i
-0.0308 + 0.0000i
-0.0700 + 0.0000i
```

**SUBTASK C** Construct the observer-based controller K1(s) using the full-state feedback gain and the observer gain. Find the scaling N such that the controller will achieve zero steady-state error for a unit-step input

Combine the controller and observer generated in the previous parts by assigning $\mathbf{A} = (\mathbf{A} - \mathbf{BK} - \mathbf{LC})$, $\mathbf{B} = -\mathbf{L}$, $\mathbf{C} = -\mathbf{K}$ as follows:

```
K1 = ss(A-B*K-L*C, -L, -K, 0)
[numK1,denK1] = ss2tf(K1.a,K1.b,K1.c,K1.d);
DK1 = tf(numK1,denK1)
```

```
K1 =

a =
x1          x2          x3          x4          x5          x6
x1      -184.9       -9711  -2.025e+05  -2.053e+06  -1.841e+07  -9.918e+07
x2           1        2.06         187        4122   3.111e+04   4.392e+05
x3           0      0.9736      -2.401       -52.9      -399.3       -5637
x4           0  -0.0004395      0.9601     -0.8796       -6.64      -93.73
x5           0   5.657e-05    0.005137       1.113      0.8545       12.06
x6           0  -1.614e-06  -0.0001466   -0.003229      0.9756     -0.3441
x7           0  -1.547e-05   -0.001405    -0.03095     -0.2337      -2.298

x7
x1  -1.663e+08
x2   9.861e+05
```

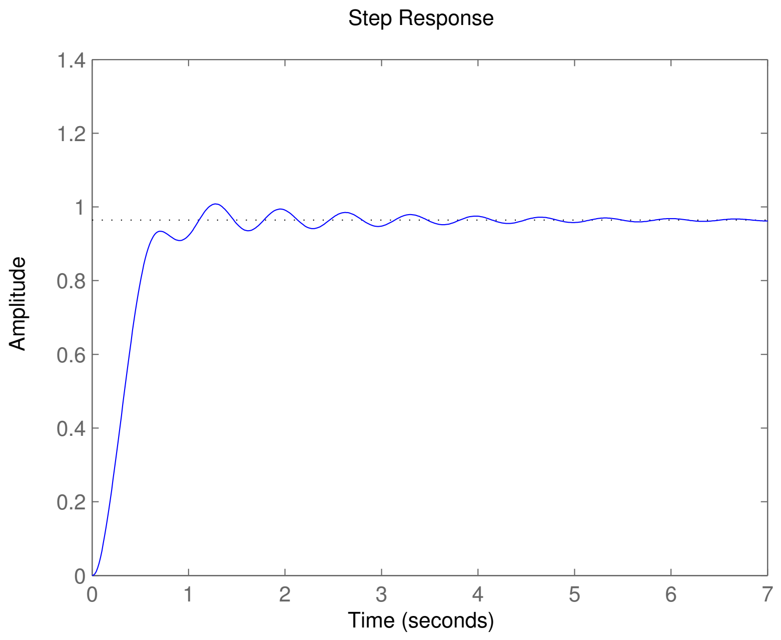
```
x3  -1.266e+04
x4      -210.4
x5       27.08
x6     -0.7726
x7      -7.406

b =
u1
x1      -21.44
x2      0.4287
x3   -0.005502
x4  -9.149e-05
x5   1.177e-05
x6  -3.359e-07
x7   -3.22e-06

c =
x1          x2          x3          x4          x5          x6
y1      -3.895      -704.6   -3.46e+04  -6.137e+05  -4.737e+06   -4.67e+07

x7
y1  -1.139e+08

d =
u1
y1   0

Continuous-time state-space model.


DK1 =


354.7 s^6 + 6.417e04 s^5 + 3.151e06 s^4 + 5.589e07 s^3 + 4.314e08 s^2

+ 4.253e09 s + 1.038e10

-----------------------------------------------------------------------

s^7 + 193 s^6 + 1.107e04 s^5 + 2.64e05 s^4 + 3.102e06 s^3 + 2.69e07 s^2

+ 1.727e08 s + 2.977e08


Continuous-time transfer function.
```

Where DK1 is the transfer function and K1 is the state space model of the observer-based controller.

**SUBTASK D** Apply the controller $K_1(s)$ to the voltage loop. Simulate the closed-loop system response to a 0.1 step in $V_{ref}$. Use `tstats` to verify that the controller meets all the specifications. Comment on the step response.

```
K1sysFB=feedback(1.04*tfSys*DK1,1);
stepinfo(K1sysFB)
figure()
step(K1sysFB)
```

```
ans =

RiseTime: 0.4366
SettlingTime: 2.6661
SettlingMin: 0.8734
SettlingMax: 1.0081
Overshoot: 4.5459
Undershoot: 0
Peak: 1.0081
PeakTime: 1.2754
```



The step response meets the rise time and overshoot requirements with a high margin.

**SUBTASK E** Find the gain and phase margins achieved by K1(s).

```
[GM,PM] = margin(K1sysFB)
```

```
GM = 1.4170

PM = Inf
```

## TASK 3

The controller $K_1(s)$ is 7th order, with many of its dynamics not important. We will now proceed to reduce its order

**SUBTASK A** Plot the frequency response of the 7th-order $K_1(s)$. What kind of controller is $K_1(s)$? Does your $K_1(s)$ exhibit a notch in the magnitude at about the EM mode frequency? Do you know what the purpose of this notch is? This notch is at a fixed frequency, implying it is less useful when the frequency of the EM mode changes

There is a notch-like low pass filter. The notch is there to reduce the controller's effect on the EM poles.

```
figure()
bode(DK1)
title('Frequency Response of Unreduced K1')
```

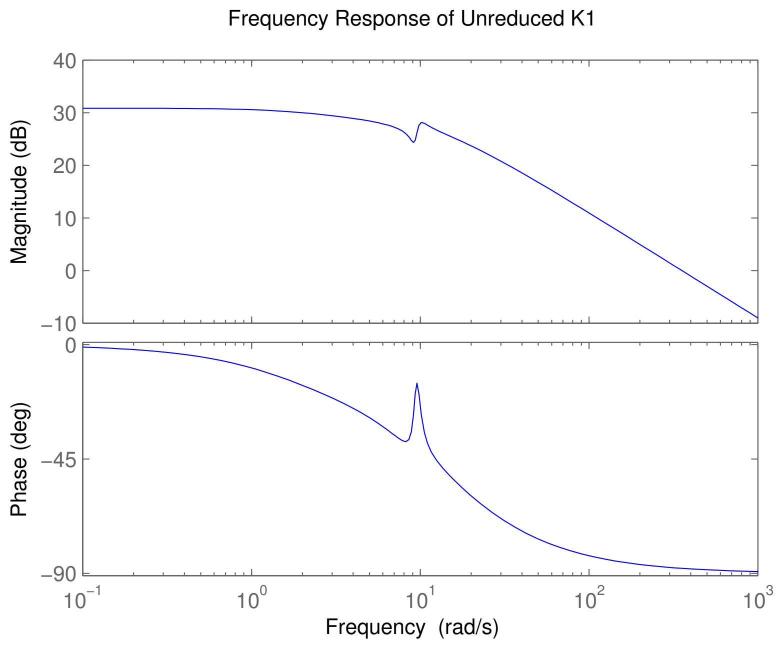


Figure 24: Frequency Response of Unreduced $K_1$

**SUBTASK B** Express $K_1(s)$ in ZPK form, showing all the factors of poles and zeros. You will notice that most of the poles have nearby zeros, except for one pole pd. Thus we can perform approximate pole-zero cancellations and retain only pd in the first-order reduced controller $K_1(s)$, which will have the form $K_1(s) = \frac{K_p}{s-p_d}$ where $K_p$ is selected so that the first-order $K_1(s)$ and the 7th-order $K_1(s)$ have the same DC gain.

The poles and zeros approximately cancel in most places

```
zpk(DK1)
DK1_Reduced = 415/(s+12.02)
```

```
ans =

354.67 (s+114.7) (s+35.4) (s+26.76) (s+3.081) (s^2 + 0.9588s + 87.36)
-----------------------------------------------------------------------
(s+114.6) (s+36.02) (s+26.76) (s+12.02) (s+2.438) (s^2 + 1.182s + 91.97)

Continuous-time zero/pole/gain model.


DK1_Reduced =

415
---------
s + 12.02

Continuous-time transfer function.
```

**SUBTASK C** Plot the frequency response of the first-order $K_1(s)$ versus the frequency response of the 7th-order $K_1(s)$. Comment on their similarities and differences.

```
figure()
bode(DK1_Reduced)
figure()
bode(DK1)
```

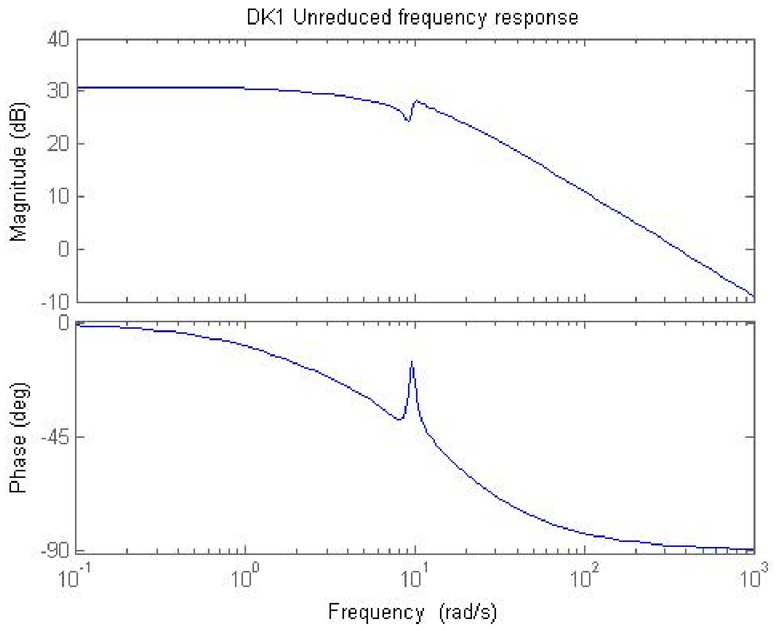


Figure 25: Frequency Response of 7th-order $K_1$

Problem 3 continued on next page...

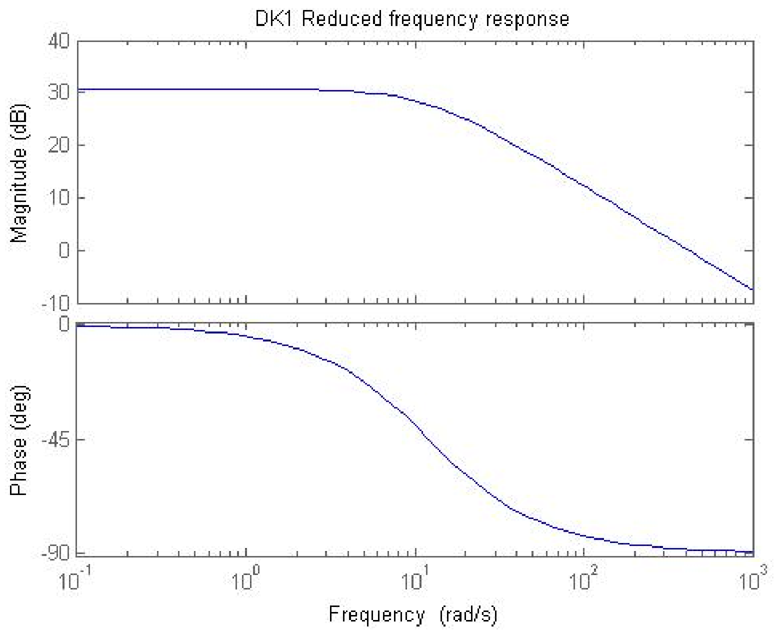


Figure 26: Frequency Response of first-order $K_1$

The reduced order controller loses the notch at the EM frequency. However, they are similar in their DC gain, and approximately the same in their phase contributions (with the notable exception of the notch in the full order controller).

**SUBTASK D** Close the voltage loop with this first-order controller and perform a step response with a 0.1 pu $V_{ref}$ step input. Comment on the time response.

```
    K1_ReducedsysFB=feedback(1.04*tfSys*DK1_Reduced,1);
    step(K1_ReducedsysFB*.1)
    stepinfo(K1_ReducedsysFB)
```

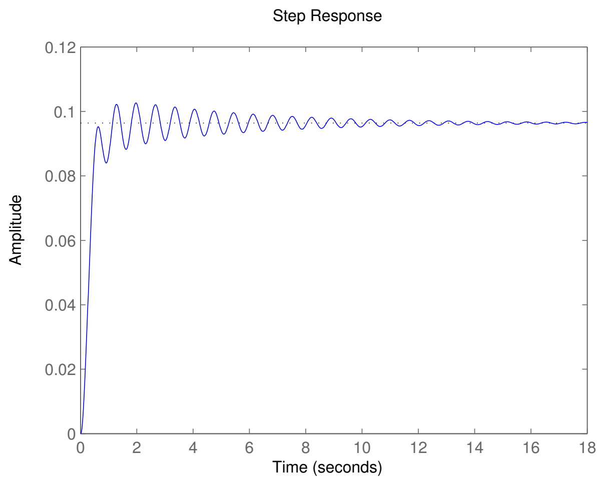


Figure 27: Step Response with 0.1 pu step input

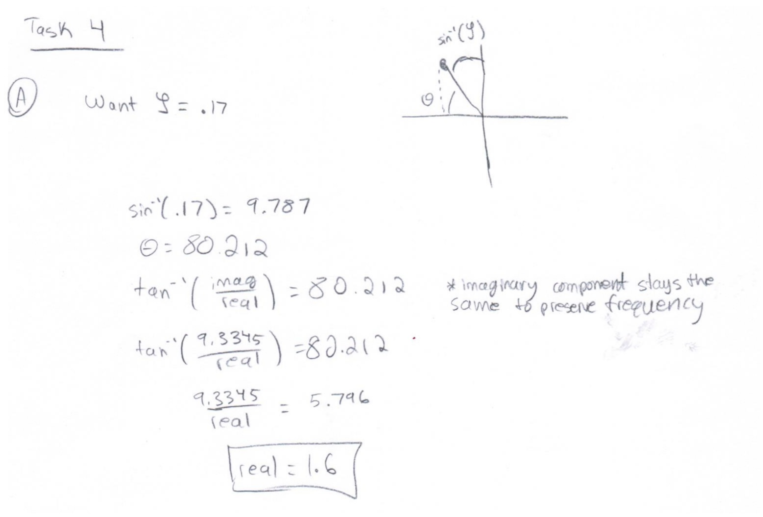
```
RiseTime: 0.3812
SettlingTime: 7.5606
SettlingMin: 0.8399
SettlingMax: 1.0264
Overshoot: 6.4837
Undershoot: 0
Peak: 1.0264
PeakTime: 1.9699
```

The response is pretty good in terms of rise time and overshoot relative to previous parts. However, there is a lot of oscillation in the output.

## TASK 4

Use Simulink or MATLAB to generate the state-space model $G_\omega(s)$ from $V_{ref}$ to $\omega_f$ with the voltage controller $K_1(s)$ loop closed. The damping controller $K_2(s)$ is then designed to improve the damping ratio of the EM mode to 0.17. The recommended steps to obtain $K_2(s)$ are contained in the following list

We obtain the desired pole locations from the specifications:



Task 4

(A) Want $\zeta = .17$

$\sin^{-1}(\zeta)$
$\theta$

$\sin^{-1}(.17) = 9.787$

$\theta = 80.212$

$\tan^{-1}\left(\frac{\text{imag}}{\text{real}}\right) = 80.212$ * imaginary component stays the same to preserve frequency

$\tan^{-1}\left(\frac{9.3345}{\text{real}}\right) = 80.212$

$\frac{9.3345}{\text{real}} = 5.796$

$\boxed{\text{real} = 1.6}$

We then use Simulink to obtain the transfer function from $V_{ref}$ to $\omega_f$ with the damping control disconnected.

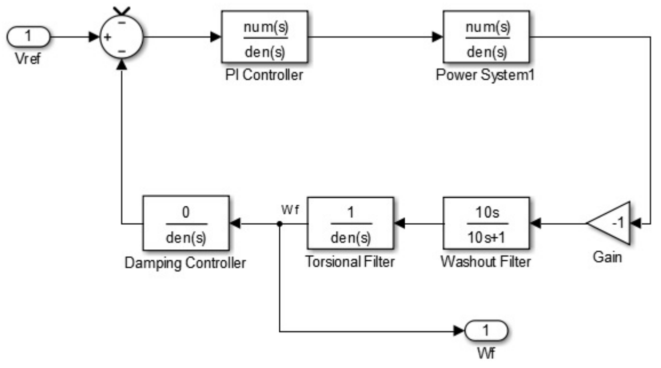


Figure 28: Simulink Model of $V_{ref}$ to $\omega_f$ (problem3.slx)

**SUBTASK A** Design a full-state feedback control to place the EM poles at their desired locations. There is no need to move the other poles.

Get the system in control canonical form and find the system's current poles. Once the poles are found, move the real part of the EM poles to the calculated location:

```
    [PI_Controller_Num,PI_Controller_Den] = tfdata(DK1_Reduced,'v');
    [tempNum,tempDen,tfSys2]=simulinkToTF('problem3');

    [num3,den3] = tfdata(tfSys2, 'v');
    [A2,B2,C2,D2]=tf2ss(num3,den3);

    poles = pole(tfSys2)
    p = poles(1:6)';
    p = [p -1.6+9.3345i -1.6-9.3345i poles(9:11)'];
    K2 = place(A2,B2,p);
    controllerPoles=eig(A2-B2*K2)
```

Problem 3 continued on next page. . .

```
poles =

1.0e+02 *

-1.1472 + 0.0000i
-0.3540 + 0.0000i
-0.2676 + 0.0000i
-0.1411 + 0.1309i
-0.1411 - 0.1309i
-0.1202 + 0.0000i
-0.0048 + 0.0933i
-0.0048 - 0.0933i
-0.0308 + 0.0000i
-0.0011 + 0.0000i
-0.0010 + 0.0000i


controllerPoles =

1.0e+02 *

-1.1472 + 0.0000i
-0.3540 + 0.0000i
-0.2676 + 0.0000i
-0.1411 + 0.1309i
-0.1411 - 0.1309i
-0.1202 + 0.0000i
-0.0160 + 0.0933i
-0.0160 - 0.0933i
-0.0308 + 0.0000i
-0.0011 + 0.0000i
-0.0010 + 0.0000i
```

**SUBTASK B** Design an observer so that the states of the system can be estimated using the torsional filter output. You have to assign the observer poles such that the EM mode is estimated well. Again, there is no need to move the observer poles that do not significantly influence the observer transients

Find L to place observer, place real part of dominant mode further out.

```
% Duality is our friend!
p = poles(1:6)';
p = [p -4.6+9.3345i -4.6-9.3345i poles(9:11)'];
L2 = place(A2',C2',p)';
eig(A2-L2*C2)
```

```
ans =

1.0e+02 *
```

```
-1.1472 + 0.0000i
-0.4338 + 0.0000i
-0.3588 + 0.0000i
-0.2677 + 0.0000i
-0.1771 + 0.1280i
-0.1771 - 0.1280i
-0.0321 + 0.1288i
-0.0321 - 0.1288i
-0.0367 + 0.0000i
-0.0011 + 0.0000i
-0.0010 + 0.0000i
```

**SUBTASK C** Construct the observer-based controller $K_2(s)$ using the full-state feedback gain and the observer gain. Plot and comment on its frequency response

Observer based-controller as in TASK 2C: $\mathbf{A} = (\mathbf{A} - \mathbf{BK} - \mathbf{LC})$, $\mathbf{B} = -\mathbf{L}$, $\mathbf{C} = -\mathbf{K}$

```
K2 = ss(A2-B2*K2-L2*C2, -L2, -K2, 0);
[numK2,denK2] = ss2tf(K2.a,K2.b,K2.c,K2.d);
DK2 = tf(numK2,denK2);
bode(DK2)
```

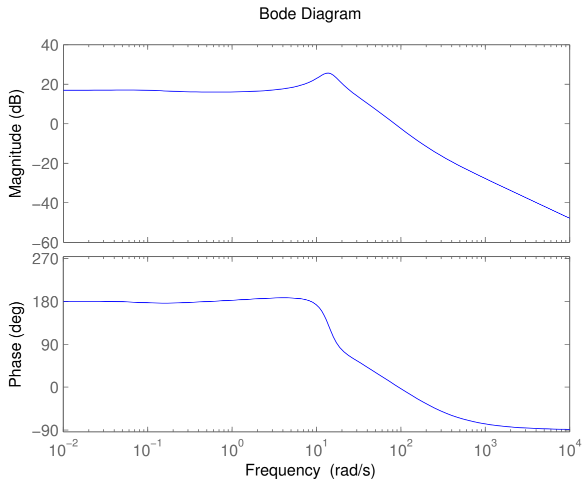


Figure 29: Frequency response of $K_2(s)$

In contrast to $K_1$, this controller specifically contributes to the EM poles. The net effect of this controller is that it changes the angle of departure of the EM poles (as will be seen in the next section).

**SUBTASK D** Do a root-locus analysis of the damping controller loop using $K_2(s)$. Comment on your observations

```
figure()
rlocus(tfSys2*DK2)
axis([-30 10 -25 25])
```

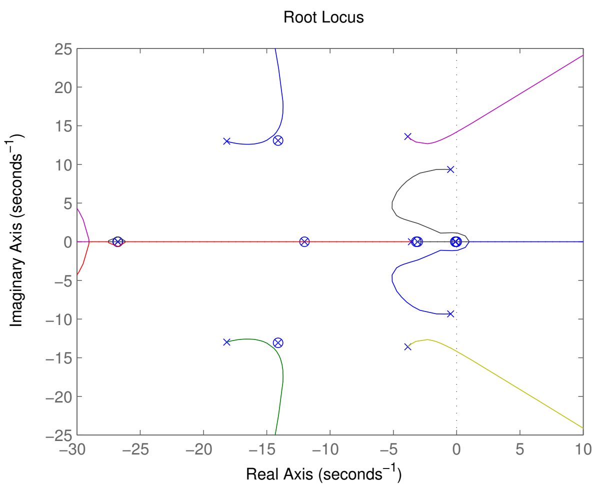


Figure 30: Root-Locus of Damping Controller Loop

The root locus now exhibits a departure angle of about 180 degrees which is similar to the compensated root locus of part 1. We now have damping on the EM poles.

**SUBTASK E** Apply the controller $K_2(s)$ to the damping control loop. Simulate the closed-loop system response to a 0.1 step in $V_{ref}$. Use `tstats` to assess the performance of the overall system, that is, the combined $K_1(s)$ and $K_2(s)$ controller. Comment on the step response.

```
[Damping_Controller_Num,Damping_Controller_Den]=tfdata(DK2,'v');
[PI_Controller_Num,PI_Controller_Den]=tfdata(DK1_Reduced,'v');
[tempNum,tempDen,tfSysFinal_unReduced]=simulinkToTF('problem1');
stepinfo(tfSysFinal_unReduced*.1)
figure()
step(.1*tfSysFinal_unReduced)
```

Problem 3 continued on next page. . .

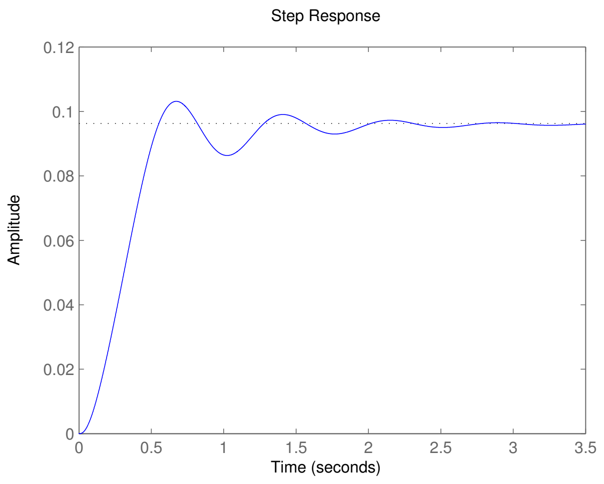


Figure 31: Step Response of Overall System

```
RiseTime: 0.3697
SettlingTime: 1.9025
SettlingMin: 0.0863
SettlingMax: 0.1031
Overshoot: 7.1407
Undershoot: 0
Peak: 0.1031
PeakTime: 0.6729
```

This is an awesome step response! It performs better than the control system in part 1 in rise time!

## TASK 5

The controller $K_2(s)$ is 11th order (why?), again with many of its dynamics not important. We will now proceed to reduce its order.

$K_2$ is of order 11 because the forward path of the damping loop is of order 11.

**SUBTASK A** Express $K_2(s)$ in ZPK form, showing all the factors of poles and zeros. You will notice that a few poles have nearby zeros. Instead of manually doing the polezero cancellation to reduce the order of $K2(s)$, this time we will use the MATLAB 12 `minreal` function (for minimal realization) to reduce the order of the controller. Use a tolerance of about 0.001 for `minreal` to obtain a controller of about 5th order. (Note that if the tolerance is set to zero, only uncontrollable and unobservable poles will be eliminated.) Plot the frequency response of this low-order controller and compared it to that of the 11th-order controller. (We could reduce further the order of $K_2(s)$ to perhaps a 2nd- or 3rd-order controller. This reduction will require more analysis and will not be attempted here.)

```
zpk(DK2)
```

```
ans =
40.386 (s-176.6) (s+114.7) (s+35.4) (s+26.76) (s+12.02) (s+3.081)

(s+0.1053) (s+0.1) (s^2 + 28.22s + 370.4)
--------------------------------------------------------------------------

(s+114.7) (s+43.33) (s+36.09) (s+26.77) (s+3.572) (s^2 + 0.1823s + 0.009341)

(s^2 + 36.33s + 498.5) (s^2 + 7.717s + 199.7)


Continuous-time zero/pole/gain model.
```

The controller used requires a more rough cancellation of poles/zeros to achieve a 5th order controller

```
      DK2_Reduced = minreal(DK2, .1)
```

```
DK2_Reduced =


40.39 s^5 - 5381 s^4 - 2.753e05 s^3 - 5.676e06 s^2 - 4.678e07 s

- 9.78e07

---------------------------------------------------------------------

s^6 + 90.95 s^5 + 3199 s^4 + 6.381e04 s^3 + 7.716e05 s^2 + 6.386e06 s

+ 1.541e07


Continuous-time transfer function.
```

**SUBTASK B** Close the damping control loop with this low-order controller and perform a step response of the overall closed-loop system with a 0.1 pu $V_{ref}$ step input. Comment on the time response.

```
 [Damping_Controller_Num,Damping_Controller_Den]=tfdata(DK2_Reduced,'v');
    [PI_Controller_Num,PI_Controller_Den]=tfdata(DK1_Reduced,'v');
    [tempNum,tempDen,tfSysFinal_Reduced]=simulinkToTF('problem1');
    stepinfo(tfSysFinal_Reduced*.1)
    figure()
    step(.1*tfSysFinal_Reduced)
    title('K1 & K2 reduced step response')
```

Problem 3 continued on next page. . .

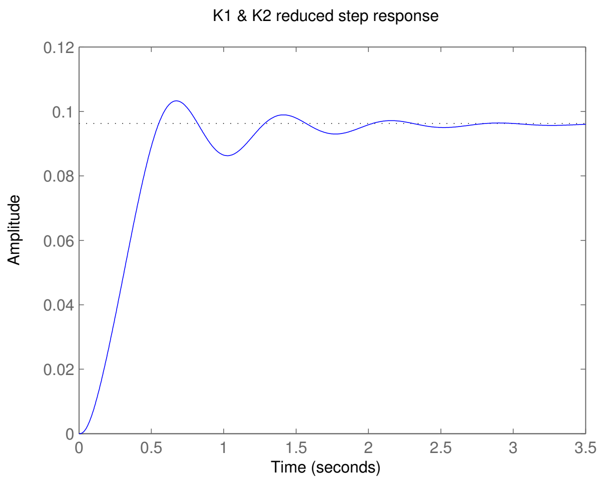


Figure 32: $K_1$ & $K_2$ Reduced Step Response

```
ans =

RiseTime: 0.3695
SettlingTime: 1.9091
SettlingMin: 0.0863
SettlingMax: 0.1033
Overshoot: 7.2927
Undershoot: 0
Peak: 0.1033
PeakTime: 0.6745
```

## COMPARING STATE SPACE RESULTS

```
figure()
step(.1*tfSysFinal_Reduced)
title('K1 & K2 reduced step response')

figure()
step(.1*tfSysFinal_unReduced)
title('K1 reduced K2 unreduced step response')

[Damping_Controller_Num,Damping_Controller_Den]=tfdata(DK2,'v');
[PI_Controller_Num,PI_Controller_Den]=tfdata(DK1,'v');
[tempNum,tempDen,tfSysFinal_NoneReduced]=simulinkToTF('problem1');
figure()
step(.1*tfSysFinal_NoneReduced)
title('K1 & K2 unreduced step response')
```

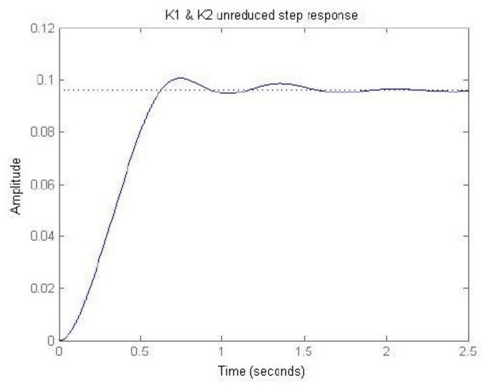
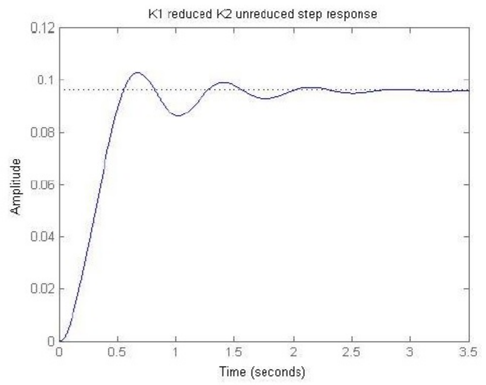
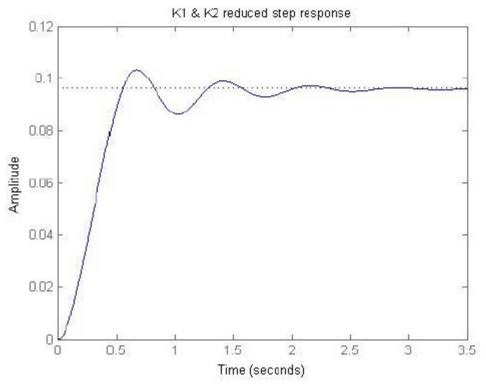
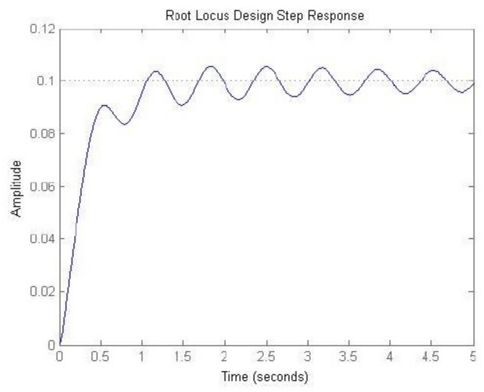
| STEP RESONSES | STEP INFO |
|---|---|
|  | RiseTime: 0.3695<br>SettlingTime: 1.9091<br>SettlingMin: 0.0863<br>SettlingMax: 0.1033<br>Overshoot: 7.2927<br>Undershoot: 0<br>Peak: 0.1033<br>PeakTime: 0.6745 |
|  | RiseTime: 0.3697<br>SettlingTime: 1.9025<br>SettlingMin: 0.0863<br>SettlingMax: 0.1031<br>Overshoot: 7.1407<br>Undershoot: 0<br>Peak: 0.1031<br>PeakTime: 0.6729 |
|  | RiseTime: 0.4153<br>SettlingTime: 1.4433<br>SettlingMin: 0.0868<br>SettlingMax: 0.1007<br>Overshoot: 4.5764<br>Undershoot: 0<br>Peak: 0.1007<br>PeakTime: 0.7452 |

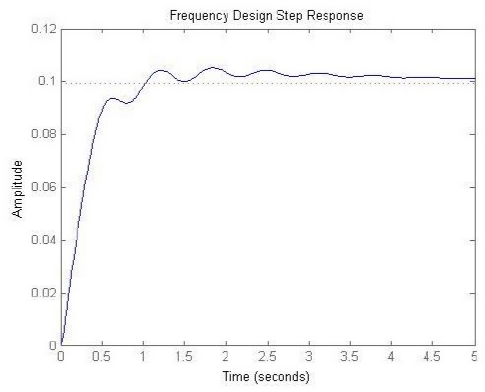
Figure 33: Comparison of State Space Results (reduced vs unreduced)

## Conclusion

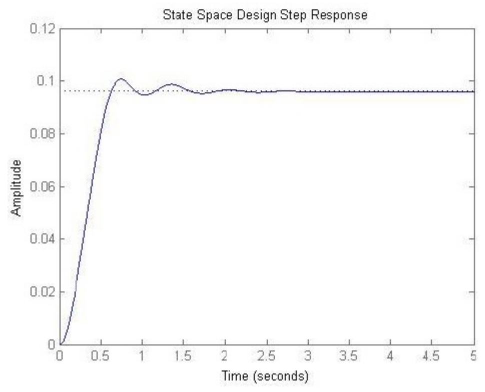
In this project, we designed controllers for a power system using Root Locus, Frequency Response and State Space techniques. The step responses from the controllers generated from each approach are listed below:



RiseTime: 0.4461
SettlingTime: 8.9486
SettlingMin: 0.8373
SettlingMax: 1.0563
Overshoot: 5.6300
Undershoot: 0
Peak: 1.0563
PeakTime: 1.8253

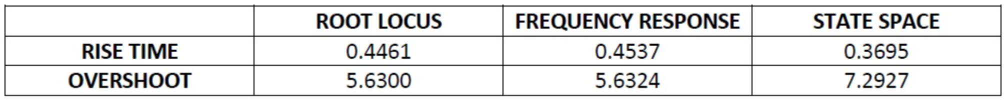


RiseTime: 0.4537
SettlingTime: 4.5339
SettlingMin: 0.0902
SettlingMax: 0.1051
Overshoot: 5.6324
Undershoot: 0
Peak: 0.1051
PeakTime: 1.8475



RiseTime: 0.3695
SettlingTime: 1.9091
SettlingMin: 0.0863
SettlingMax: 0.1033
Overshoot: 7.2927
Undershoot: 0
Peak: 0.1033
PeakTime: 0.6745

Figure 34: Step Comparisons of Results From Different Design Methodologies



| | ROOT LOCUS | FREQUENCY RESPONSE | STATE SPACE |
|---|---|---|---|
| RISE TIME | 0.4461 | 0.4537 | 0.3695 |
| OVERSHOOT | 5.6300 | 5.6324 | 7.2927 |

Figure 35: Comparison of Rise Time and Overshoot of Results from Different Methods

All of the approaches performed similarly with within reasonable margins of the rise time and overshoot specifications. The root locus and frequency response approaches performed quite similarly. The state space approach achieved the best rise time at the cost of a greater overshoot. From an observational stand point, the state space approach was the most successful at damping the EM poles.

## TEAM MEMBER PREFERENCES FOR DESIGN

Andrew Cunningham: I preferred the state space approach to designing the system. The ability to place poles in observer based controllers is powerful and simple. I particularly enjoyed the fact that the procedures for designing $K_1(s)$ and $K_2(s)$ were essentially identical. All one had to do was place the poles to meet the specifications. Furthermore, I really enjoyed the fact that state space design employs a procedural design process that starts from the specifications and ends (fairly quickly) in a controller. This stands in stark contrast to the root locus approach. I did not like the fact that I had to use rltool multiple times to get a single compensator that met the specifications.

Sabbir Rashid: I preferred the Frequency-Response design method mostly because I am the most comfortable in designing lead and lag compensators using this method. This was due to the length of time in class spent covering the related material to this approach. Nevertheless, I see that power and beauty of the state space approach, and as my abilities in Control Systems engineering improves, I believe this approach would be preferable.