HACETTEPE UNIVERSITY - COMPUTER ENGINEERING

Advanced Robust Control MMÜ 749

Homework-3

Student Name Ayça KULA

Student ID: 202237285

Spring 2022

1 Problem 1:

The design specifications for this problem is:

- Unit DC gain from r to y
- Max %16 overshoot
- $\zeta = 0.5$ and $\omega_n = 0.5$. Therefore, the desired pole locations are

$$s = \zeta\omega_n \pm \omega_n\sqrt{1-\zeta^2}j = 0.4 \pm 0.69282j$$

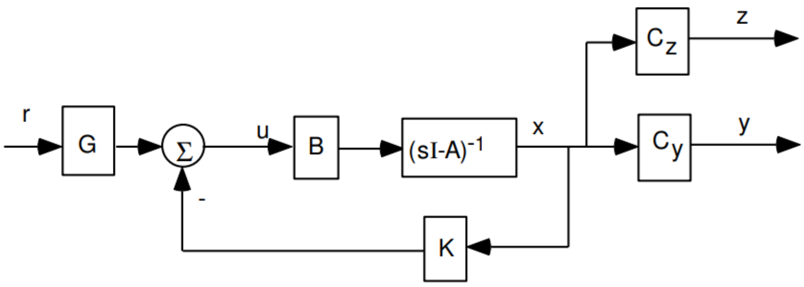

Figure 1: State Feedback with Two Sets of Outputs

And consider the transfer function as:

$$P_y(s) = \frac{1}{(0.5s+1)(s^2+2(0.01)(0.8)s+(0.8)^2)} \tag{1}$$

1.1 Part A: LQR-1

For each part, you can examine the codes in Appendix and you can see each formulation for every question.

1.1.1 State space representation for the TF:

State-space is found by using the command "tf2ss" in MATLAB.

$$A = \begin{bmatrix} -2.0160 & -0.6720 & -1.2800 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

$$C = \begin{bmatrix} 0 & 0 & 2 \end{bmatrix} \tag{4}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{5}$$

1.1.2 Determine Q and R:

The MATLAB function lqr allows to use R and Q, which will balance the relative importance of the control effort (u) and error (deviation from 0), respectively, in the cost function that you are trying to optimize. The simplest case is to assume control weight has a from of $R = \rho R_0$ and state weighting matrix is $Q = C^T C$. In order to calculate the R matrix, I have used ρ values as given in $\rho = 0.6, 0.5, 0.4, 0.3, 0.2$. I have not chosen the ρ value as bigger than 1 due to overshoot problem and less than 0.1 due to cheap control situation.The LQR gains are shown in 1 for each ρ value. Since the lower ρ leads to cheap control and it is hard to control and big rho leads to overshoot, I took the ρ value as 0.5.

1.1.3 Adding precompensation:

We need to compute what the steady-state value of the states should be, multiply that by the chosen gain K, and use a new value as our "reference" for computing the input. This can be done by adding a constant gain G after the reference[1].We can find this G factor by employing the used-defined function rscale.m [2].After adding a pre-compensator the step responses are obtained for each ρ value as seen in 2. For each ρ, pre-compensator design was made and shown in 1.

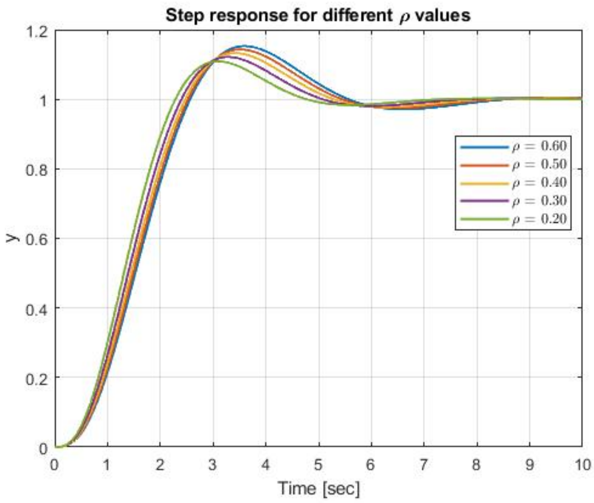

Figure 2: Step response for a system with pre-compensator

1.1.4 Draw SRL:

The symmetric root locus can be seen in figure 3. The SRL is obtained using the formulation in our lecture notes as 1. And, it can be gone through the the closed loop poles that satisfy the design requirements. This can be also seen in part-C figure 8.

```
%% Symmetric root locus plot
A_srl = [A zeros(size(A));-C'*C -A'];
B_srl = [B;-C'*D];
C_srl = [D'*C B'];
D_srl = D'*D;
sys_srl = ss(A_srl,B_srl,C_srl,D_srl);
figure;
rlocus(sys_srl)
grid
legend('SRL')
```

Algorithm 1: Obtain SRL

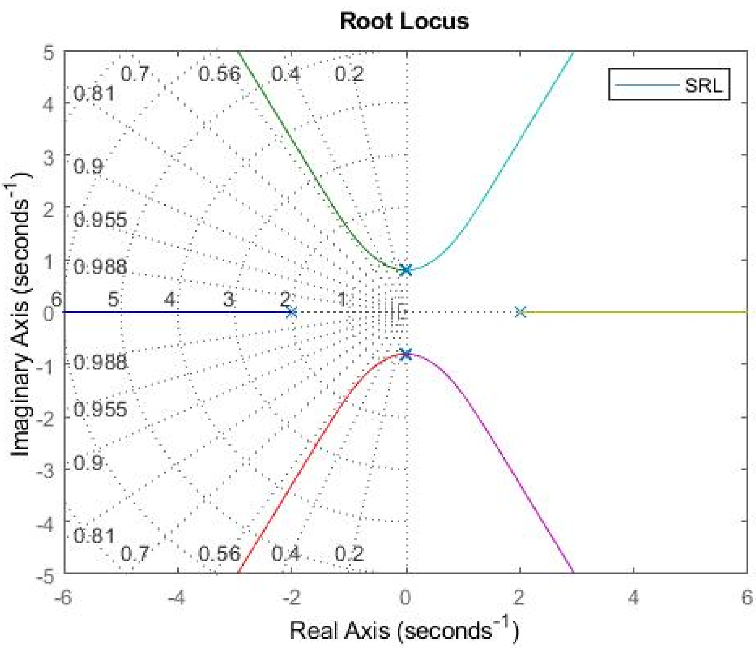

Figure 3: Symmetric Root Locus

1.1.5 Determine the loop gain L_{sf}:

The loop gain L_{sf} is found as:

$$L_{sf}(s) = K(sI - A)^{-1}B \tag{6}$$

And for each ρ value I have determined the L_{sf} which is shown in 1. It can be seen that increasing ρ value will increase the phase margin.

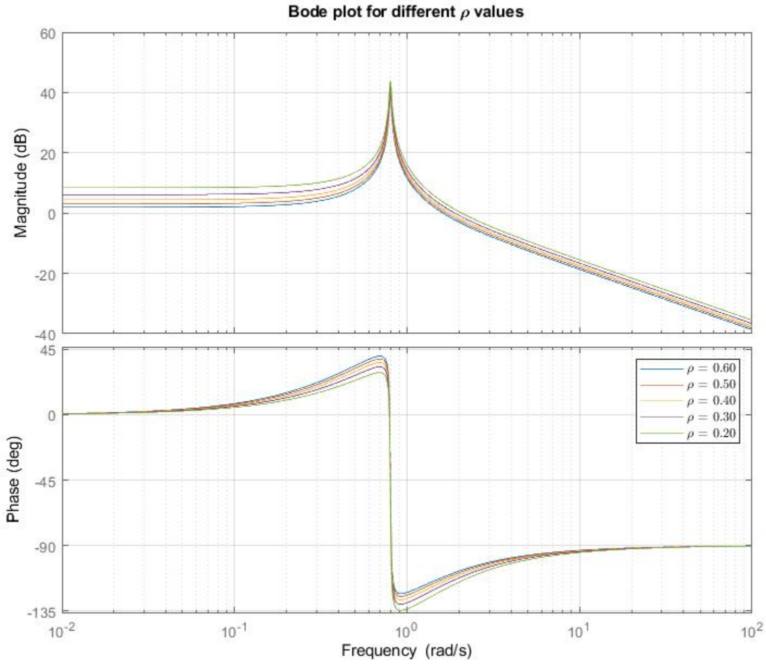

Figure 4: Bode plot

Table 1: The results for each ρ value

rho	LQR Gain	Pre-Compensator	Gain Margin	Phase Margin
0.6	1.1616, 3.0166, 1.6019	1.4409	-	68.2063
0.5	1.2389, 3.2651, 1.8246	1.5523	-	67.7044
0.4	1.3375, 3.5908, 2.1315	1.7058	-	67.1349
0.3	1.4711, 4.0477, 2.5893	1.9347	-	66.4782
0.2	1.6721, 4.7688, 3.3717	2.3259	-	65.6797

1.2 Part B: LQR-2

1.2.1 State space representation for the TF:

State-space is found by using the command "tf2ss" in MATLAB.

$$A = \begin{bmatrix} -2.0160 & -0.6720 & -1.2800 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{7}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

$$C = \begin{bmatrix} 2 & 1.6 & 1.28 \end{bmatrix} \tag{9}$$

$$D = \begin{bmatrix} 0 \end{bmatrix} \tag{10}$$

1.2.2 Determine Q and R:

The MATLAB function lqr allows to use R and Q, which will balance the relative importance of the control effort (u) and error (deviation from 0), respectively, in the cost function that you are trying to optimize. The simplest case is to assume control weight has a from of $R = \rho R_0$ and state weighting matrix is $Q = C^T C$. In order to calculate the R matrix, I have used ρ values as given in $\rho = 0.6, 0.5, 0.4, 0.3, 0.2$. I have not chosen the ρ value as bigger than 1 due to overshoot problem and less than 0.1 due to cheap control situation.The LQR gains are shown in 2 for each ρ value. Since the lower ρ leads to cheap control and it is hard to control and big rho leads to overshoot, I took the ρ value as 0.5.

1.2.3 Adding precompensation:

We need to compute what the steady-state value of the states should be, multiply that by the chosen gain K, and use a new value as our "reference" for computing the input. This can be done by adding a constant gain G after the reference[1].We can find this G factor by employing the used-defined function rscale.m [2].After adding a pre-compensator the step responses are obtained for each ρ value as seen in 5. For each ρ, pre-compensator design was made and shown in 2.

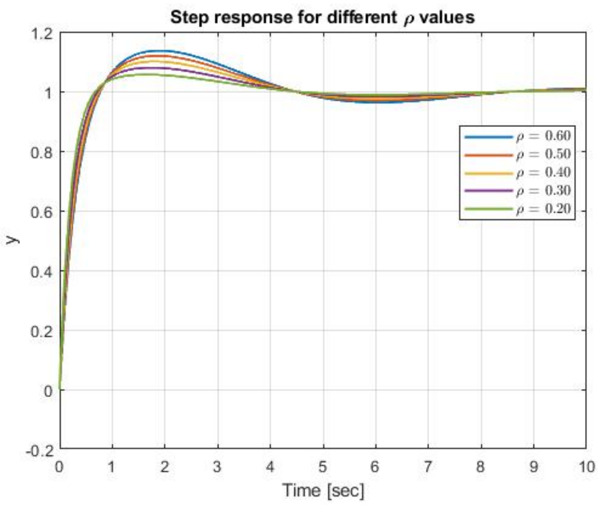

Figure 5: Step response for a system with pre-compensator

1.2.4 Draw SRL:

The symmetric root locus can be seen in figure 6. The SRL is obtained using the formulation in our lecture notes as 1. And, it can be gone through the the closed loop poles that satisfy the design requirements. This can be also seen in part-C figure 9. If we comapre the part-A and part-B SRL, I can say that in part-A the poles go to infinity and in part-B it goes to the zero locations.

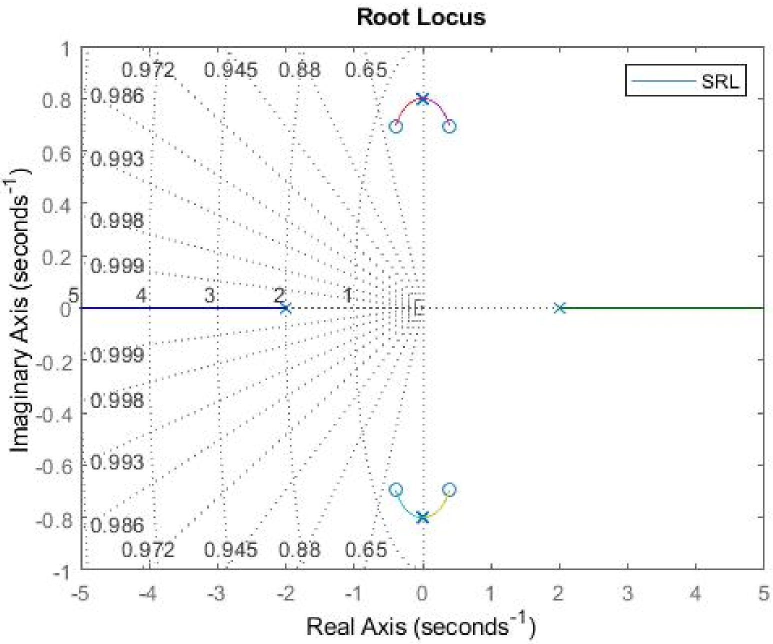

Figure 6: Symmetric Root Locus

Table 2: The results for each ρ value

rho	LQR Gain	Pre-Compensator	Gain Margin	Phase Margin
0.6	1.8177, 1.9833, 0.8102	1.6330	-	-133.9272, 102.6751
0.5	2.0370, 2.1814, 0.9370	1.7321	-	-141.2966, 103.8164
0.4	2.3388, 2.4501, 1.1147	1.8708	-	-153.7585, 104.8312
0.3	2.7887, 2.8440, 1.3845	2.0817	-	105.2576
0.2	3.5584, 3.5048, 1.8553	2.4495	-	104.3113

1.2.5 Determine the loop gain L_{sf}:

The loop gain L_{sf} is found as:

$$L_{sf}(s) = K(sI - A)^{-1}B \tag{11}$$

And for each ρ value I have determined the L_{sf} which is shown in 2. It can be seen that decreasing ρ value will increase the phase margin (look at only positive values of PM).The bode plot is shown in figure 7.

And, different controller designs are shown in table 2 for decreasing ρ values.

1.3 Part C: LQG

1.3.1 Design a full order observer with the SRL approach

An observer is designed for PART-A and PART-B using the SRL approach.

1. **PART A**
   The value for ρ has been chosen as 0.5. After choosing this value symmetric

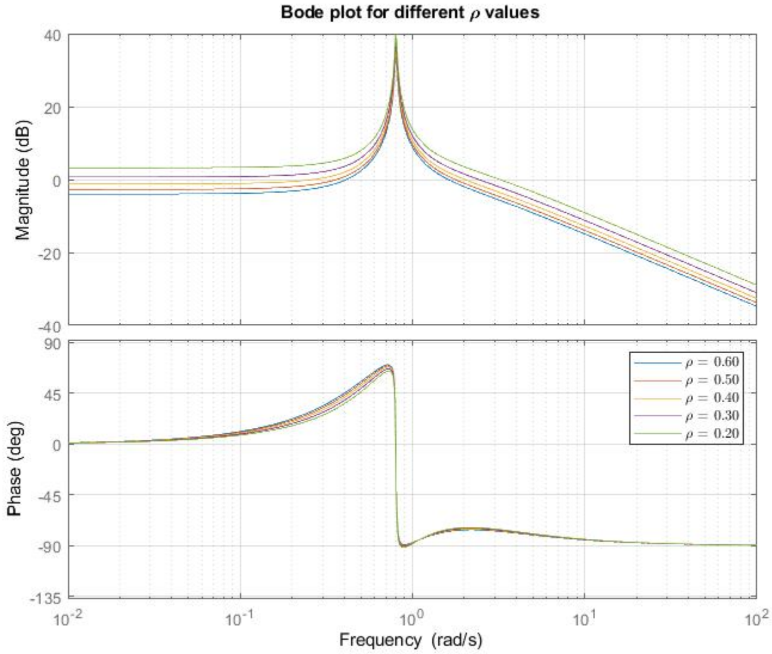

Figure 7: Bode plot

root locus has been plotted and the frequency has been seen as 1.23 rad/s. I tried to select the roots for estimation bandwidth as $BW = 1.23x2.5 = 3.075rad/s$, however the gain was too big. And this can lead to problems. Therefore, I chose the bandwidth as 2.36 rad/s, which has the q value of 145.The corresponding estimator pole locations are:

$$pe = [-1.48 + 2.37i; -3.09; -1.48 - 2.37i] \tag{12}$$

And using the "place" command the corresponding observer gains are found as:

$$L = [1.8525; 4.0746; 2.0170] \tag{13}$$

The figure closed loop poles and observer poles can be seen in 8.

2. **PART B**
The value for ρ has been chosen as 0.5. After choosing this value symmetric root locus has been plotted and the frequency has been seen as 0.8 rad/s. I tried to select the roots for estimation bandwidth as $BW = 0.8x2.5 = 2rad/s$, however 2 rad/s cannot be reached (it is out of bound). Therefore, I had to chose the bandwidth approximately similar. I have obtained my q value as 8.4.The corresponding estimator pole locations are:

$$pe = [-0.378 + 0.709i; -6.09; -0.378 - 0.709i] \tag{14}$$

And using the "place" command the corresponding observer gains are found

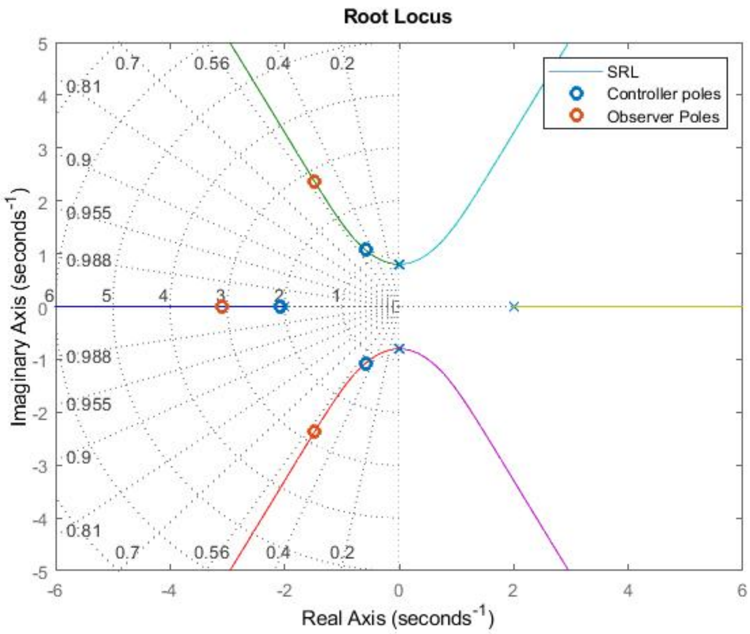

Figure 8: SRL plot for Observer for part-A

as:

$$L = [1.8525; 4.0746; 2.0170] \tag{15}$$

The figure closed loop poles and observer poles can be seen in 9.

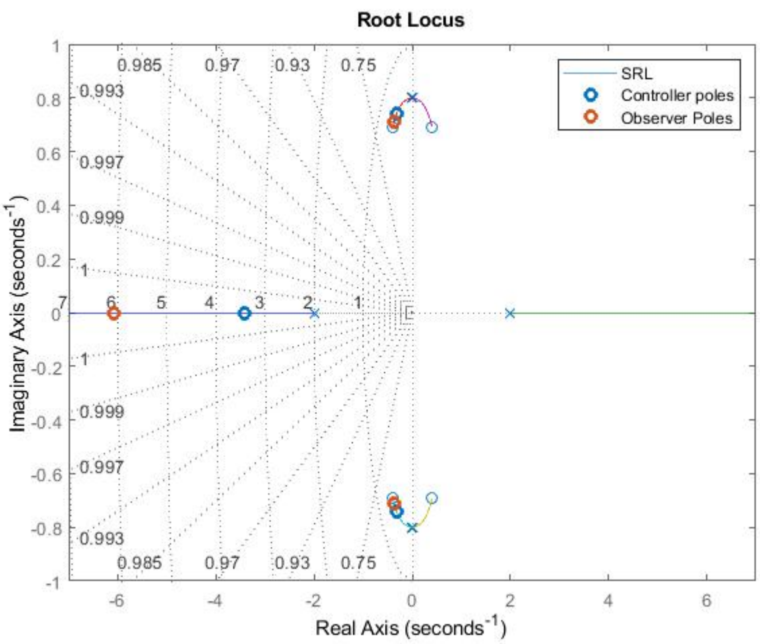

Figure 9: SRL plot for Observer for part-B

1.3.2 Test Performance in Simulink

The simulink model is obtained with the help of [3] and it is shown in figure 10.And, the simulation is made using 2.

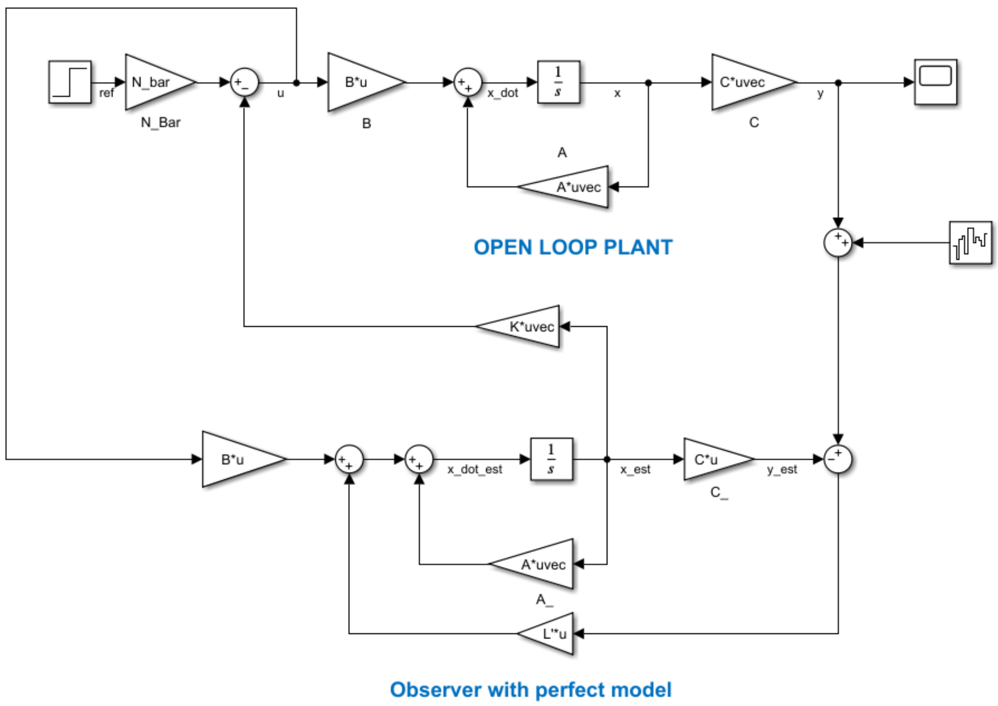

Figure 10: Simulink model

```
%% Simulink Part
N_bar = rscale(sys,K_Lqr);

noise_pow = [0.01, 0.1, 1, 10, 100];
figure;
for i = 1:length(noise_pow)
    set_param('LQR_model/Band-Limited White Noise','Cov',string(noise_pow(i)))
    y = sim('LQR_model')
    plot(y.yout{1}.Values.Time, y.yout{1}.Values.Data, 'linewidth', 1.25)
    hold on
end

legend('0.01', '0.1' , '1' ,'10', '100' )

title('Increased Noise Power and Simulation Results')
```

Algorithm 2: Test Performance in Simulink

1.3.3 Discussions

- **How does the overall system perform as noise is gradually increased?** The noise is gradually increased and the simulation results are shown in 11 for part-A and 12 for part-B. Since the q value is larger in Part A, this observer is more robust to noise compared to Part-b system.

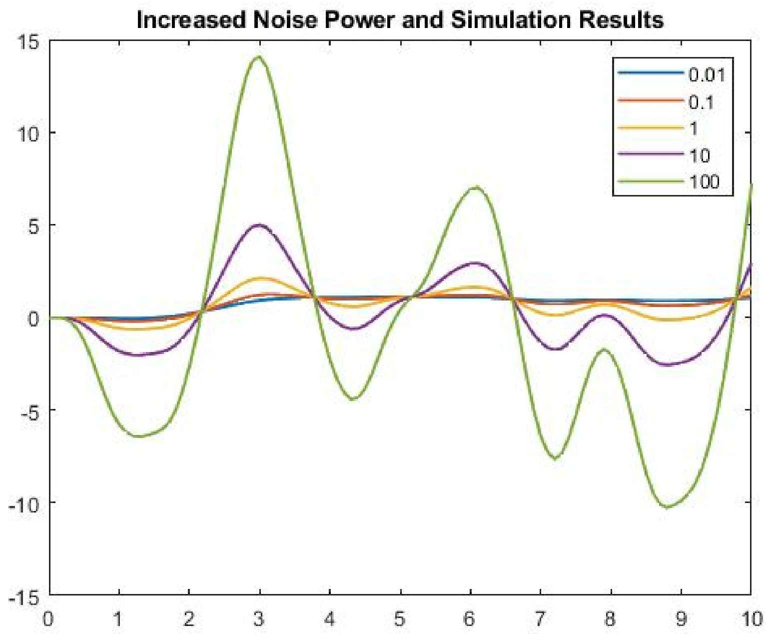

Figure 11: Increased noise and the simulation result for part-A

- **Are you able to satisfy performance specifications?** The step response of 1 and higher values of 1 is not appropriate. Therefore, systems are robust on smaller noise. However, Part-2 is more robust due to the big q value.

- **Do you feel the need to re-design the controller?** Yes, the design can be made for differently by choosing maybe different ρ values in the beginning such as 0.3 or 04.

1.3.4 Design an LQG/LTR controller

The LTR Controller is designed using [4]. LQG control inherently suffers from robustness issues. It is in fact less robust than LQR. While LQR have impressive Gain Margin(=infinity) and Phase margin(=60 degrees), there is no guaranteed stability margin for LQG control. The closed loop system can become unstable if our Kalman filter is not designed properly to take the non-linearities into account[3]. Again for the same, system parameters as before I have design an LTR Controller for part-A and part-B system as seen in figure 13 and 14 respectively. Also, the results are seen in table 3 and 4. And, the calcualtion can be seen in algorithm 3.

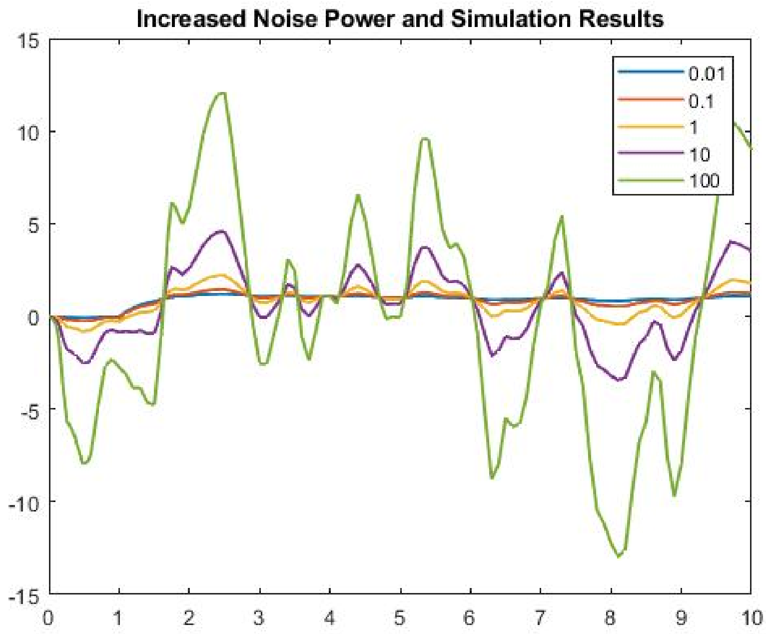

Figure 12: Increased noise and the simulation result for part-B

```
%% Design LTR
w=logspace(-1,3,1000);
rho_ltr = 0.5;
Q_ltr = rho_ltr*C'*C;
r = 1;
tf = K_Lqr*inv(s*eye(size(A))-A)*B;
[K] = lqr(A,B,Q_ltr,r);
sys1 = ss(A,B,K,0);
[maggk1,phasgk1,w]=bode(sys1,w);

q = [1, 10, 100];
rv=1;
% MATLAB lqe
for i = 1:length(q)
    gam=q(i)*B;
    Q1=gam'*gam;
    LTR(i).L=lqe(A,gam,C,Q1,rv);

    % MATLAB bode MATLAB margin
    aa=A-B*K-(LTR(i).L)*C;
    bb=(LTR(i).L);
    cc=K;
    dd=0;
    sysk=ss(aa,bb,cc,dd);
    sysgk=series(sys,sysk);
    [maggk,phsgk,w]=bode(sysgk,w);
    [gm,phm,wcg,wcp] = margin(maggk,phsgk,w)
```



```
end
```

Algorithm 3: Designing LTR

Table 3: LTR Controller Design for Part-A

q	**Observer Gain**	**Gain Margin**	**Phase Margin**	**Wcg**	**Wcp**
1	[-0.2408; 0.2308; 0.4804]	3.5127	64.8465	0.0369	0.9668
10	[47.3596; 24.1566; 4.9149]	6.9243	61.9527	4.4065	1.1746
100	[8602.9; 684; 26.2]	28.7403	69.6066	19.4035	1.2401

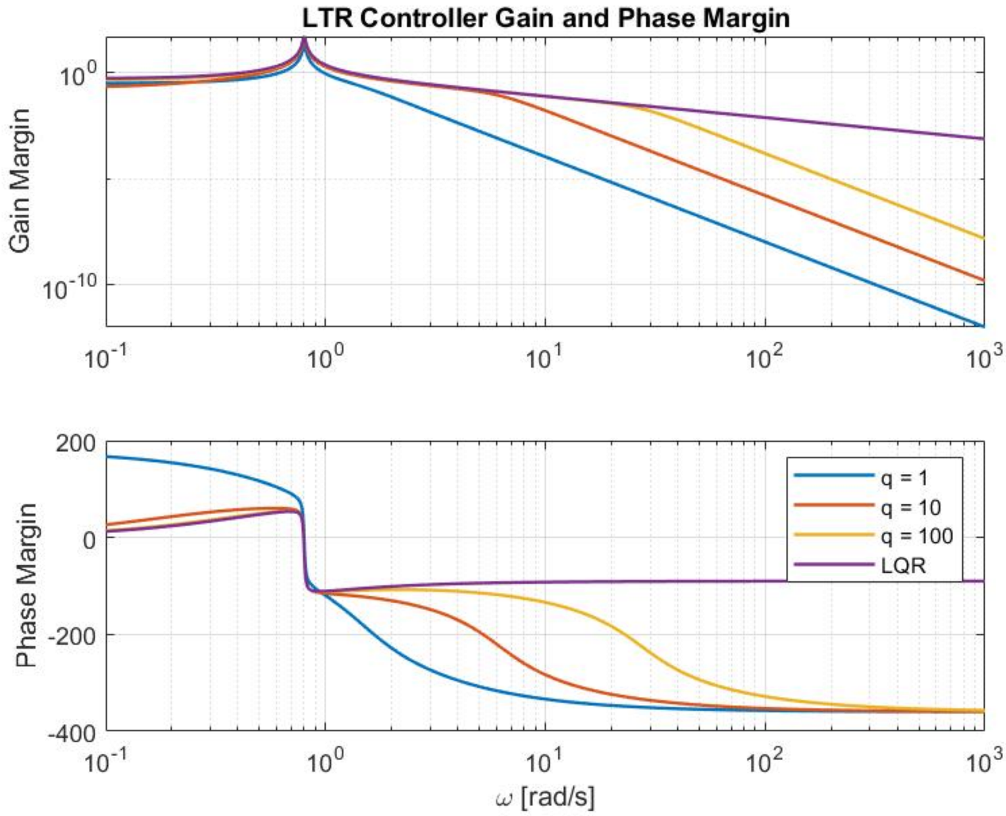

Figure 13: LTR Controller design for part A

Table 4: LTR Controller Design for Part-B

q	Observer Gain	Gain Margin	Phase Margin	Wcg	Wcp
1	[0.4334; 0.2603; 0.0251]	Inf	84.3355	NaN	1.0219
10	[98.9982; 0.4964; 0.0013]	Inf	98.6645	NaN	1.3213
100	[9999; 0.5; 0]	Inf	99.0996	NaN	1.3270

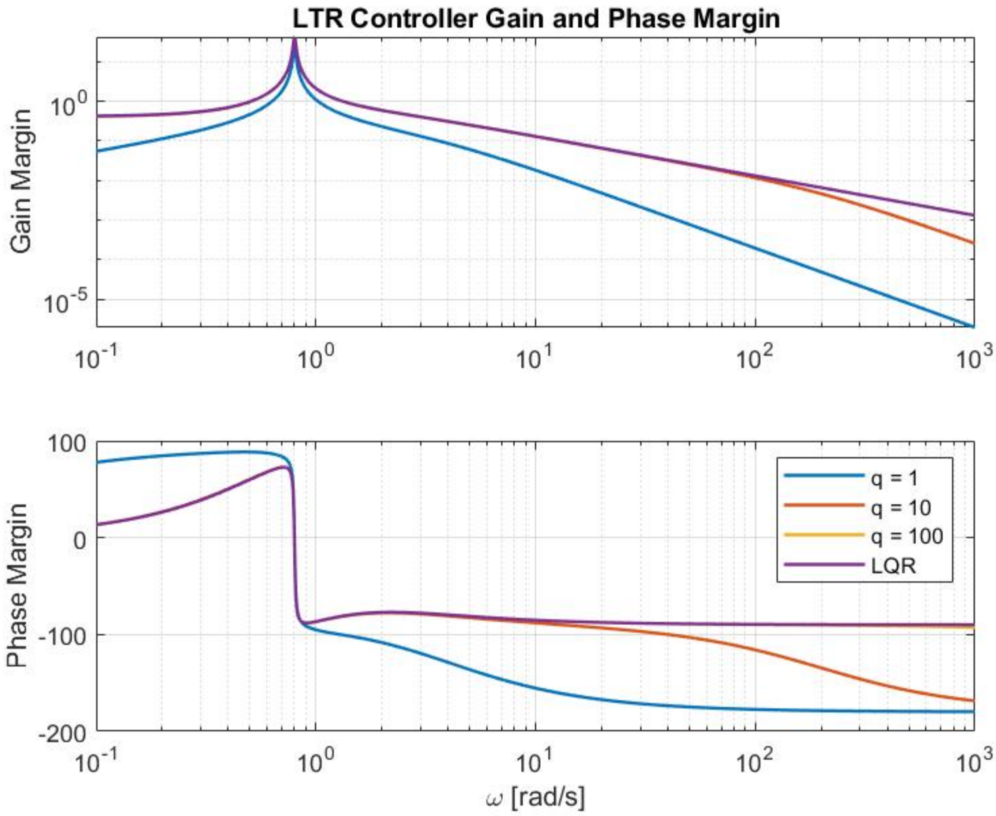

Figure 14: LTR Controller design for part B

1.4 Part D: Robustness

1.4.1 Part-A

The plots are obtained using the "ureal" command with the help of [5].System in part-A is robustly stable for the modeled uncertainty as seen in 15.According to "robstab" command, the system can tolerate up to 433% of the modeled uncertainty.This perturbation causes an instability at the frequency 1.77 rad/seconds. Sensitivity with respect to each uncertain element is:

- 28% for p1. Increasing p1 by 25% decreases the margin by 7%.
- 8% for p2. Increasing p2 by 25% decreases the margin by 2%.
- 11% for p3. Increasing p3 by 25% decreases the margin by 2.75%.
- 0% for p4. Increasing p4 by 25% decreases the margin by 0%.
- 17% for p5. Increasing p5 by 25% decreases the margin by 4.25%.
- 33% for p6. Increasing p6 by 25% decreases the margin by 8.25%.

Lower bound on the actual robust stability margin of the model is 4.3263 wheras upper bound is 4.3353. Perturbations causing instability (wcu) are:

- p1: -1.1420
- p2: -0.3807
- p3: -1.8349
- p4: 1.4028
- p5: 1.4335
- p6: 0.5665

The algorithm can be obtained as seen in 4 for both part A and B by defining their state-space and LQR gains separately..

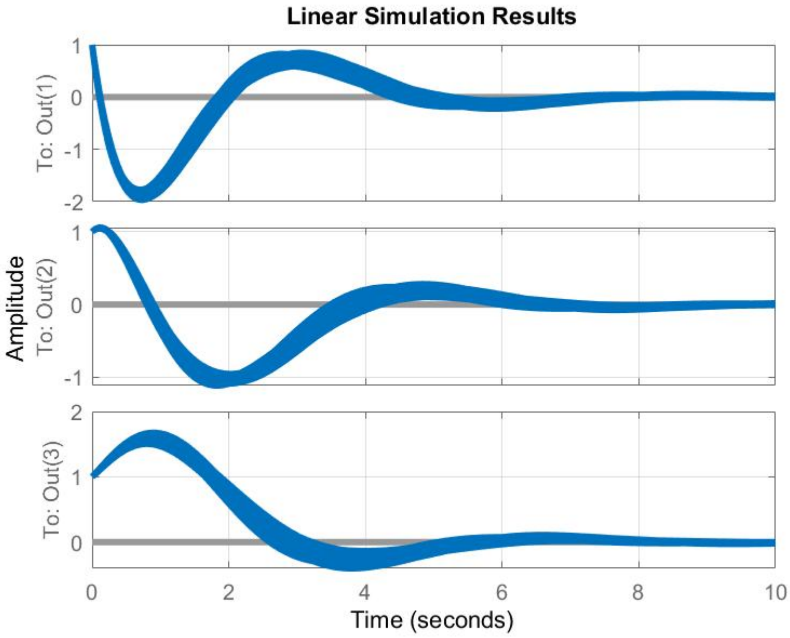

Figure 15: A and B matrices perturbed for part-A

```
%% Perurb separately the A and B matrices

p1 = ureal('p1',-2.0160,'Percentage',10);
p2 = ureal('p2',-0.6720,'Percentage',10);
p3 = ureal('p3',-1.2800,'Percentage',10);
p4 = ureal('p4',1,'Percentage',10);
p5 = ureal('p5',1,'Percentage',10);
p6 = ureal('p6',1,'Percentage',10);
A = [p1 p2 p3; p4 0 0 ;0 p5 0];
B = [p6; 0;0];
C = eye(3); % In order to reach full states
D = [0; 0; 0] % In order to reach full states
sys = ss(A,B,C,D) % Create uncertain state-space model
% % Uncertain closed loop system
Gcl = feedback(sys*K_Lqr, eye(3))

% Check robust stability margins for the uncertain system
opt = robOptions('Display','on','Sensitivity','on')
[stabmarg,wcu] = robstab(Gcl,opt)

% Step response for uncertain systems
T = 0:0.01:10
U = ones(size(T'))*[0 0 0]
x0 = [1 1 1]
lsim(Gcl, U, T, x0)
grid on

set(findall(gcf, 'type', 'line'), 'linewidth',3)
```


Algorithm 4: Perurb separately the A and B matrices

1.4.2 Part-B

Again the system is robustly stable for the modeled uncertainty as seen in figure 16. It can tolerate up to 445% of the modeled uncertainty.There is a destabilizing perturbation amounting to 454% of the modeled uncertainty.This perturbation causes an instability at the frequency 0.945 rad/seconds.Sensitivity with respect to each uncertain element is:

- 18% for p1. Increasing p1 by 25% decreases the margin by 4.5%.
- 9% for p2. Increasing p2 by 25% decreases the margin by 2.25%
- 12% for p3. Increasing p3 by 25% decreases the margin by 3
- 1% for p4. Increasing p4 by 25% decreases the margin by 0.25%.
- 15% for p5. Increasing p5 by 25% decreases the margin by 3.75%.
- 38% for p6. Increasing p6 by 25% decreases the margin by 9.5%.

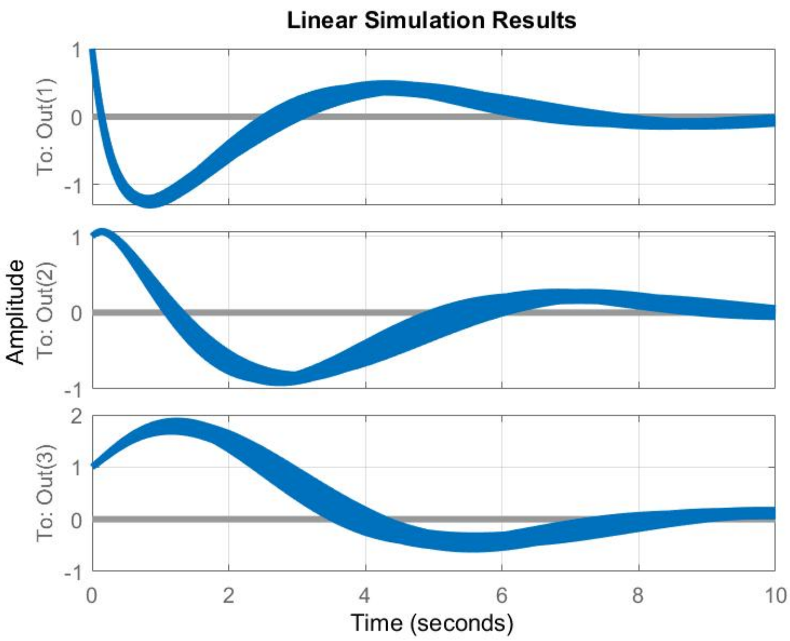

Figure 16: A and B matrices perturbed for part-B

Perturbations causing instability (wcu) are:

- p1: -1.1010
- p2: -0.3670
- p3: -1.8609
- p4: 0.5725

- p5: 1.4539
- p6: 0.5461

Appendix

LQR Design for Part-A and Part-B

```
clc;clear all;close all
%% Define systems (PART A & PART B)
% NOTE: RUN THE SYSTEM IN PART-A OR IN PART-B
s = tf('s');
%% First system --> Part A
Py = 1/((0.5*s + 1)*(s^2+2*(0.01)*(0.8)*s+(0.8)^2));

%% Second system --> Part B
% Py = (s^2+2*0.5*0.8*s+0.8^2)/((0.5*s + 1)*(s^2+2*(0.01)*(0.8)*s
    +(0.8)^2));

%% State-space representation
% Define A,B,C,D Matrices
[A,B,C,D] = tf2ss(cell2mat(Py.numerator),cell2mat(Py.denominator));
% State-space representation
sys = ss(A,B,C,D);
%% Test Controllability of the system
co = ctrb(sys);
controllability = rank(co);

%%
t = 0:0.01:10;
r =ones(size(t));
rho = [0.6, 0.5, 0.4, 0.3, 0.2];
% rho = [1, 0.1, 0.01];
disp('Full state feedback gain matrix(K), solution S of the
    associated algebraic Riccati equation(X) and the closed-loop
    poles(E)')
figure;
for i = 1:length(rho)
    fprintf('For Rho value of %0.2f',rho(i))
    %% Determine Q & R
    Q = C'*C;
    R(i) = rho(i);

    [K,X,E] = lqr(A,B,Q,R(i))

    % New states
    states(i).Ac = [(A-B*K)];
    states(i).Bc = [B];
    states(i).Cc = [C];
    states(i).Dc = [D];
    stepSys.sys_cl(i) = ss(states(i).Ac,states(i).Bc,states(i).Cc,
    states(i).Dc)

    %% Determine the precompensator G
    fprintf('Pre-compansator:')
    sys_ss = ss(A,B,C,D);
    Gbar = rscale(sys_ss,K);
    disp(Gbar);

```

```
    %% Step response for a system with Precompansator --> Means that reference = Gbar*r
    stepSys.sys_cl_precomp(i) = ss(states(i).Ac,states(i).Bc*Gbar,states(i).Cc,states(i).Dc);
    [y,t,x] = lsim(stepSys.sys_cl(i),Gbar*r,t);
    plot(t,y,'LineWidth',1.5);
    legend_name(i) = sprintf("$\\rho$ = %.2f",rho(i));
    hold on;

    %% Lsf
    Lsf(i) = K*inv(s.*eye(size(A))-A)*B;
end
legend(legend_name,'interpreter', 'latex')
ylabel('y');
xlabel('Time [sec]');
title("Step response for different \rho values");
grid;

%% Step response
% % No pre-compensator
% lsim(stepSys.sys_cl(1),stepSys.sys_cl(2),stepSys.sys_cl(3),stepSys.sys_cl(4),r,t)
% % With pre-compansator
% lsim(stepSys.sys_cl_precomp(1),stepSys.sys_cl_precomp(2),stepSys.sys_cl_precomp(3),stepSys.sys_cl_precomp(4),r,t)

%% Symmetric root locus plot
A_srl = [A zeros(size(A));-C'*C -A'];
B_srl = [B;-C'*D];
C_srl = [D'*C B'];
D_srl = D'*D;
sys_srl = ss(A_srl,B_srl,C_srl,D_srl);
figure;
rlocus(sys_srl)
grid
legend('SRL')

%% Loop Gain Lsf
% Compute the magnitude and phase of these responses
w = logspace(-2,2,1000);

figure;
for i = 1:length(rho)
    bode(Lsf(i),w);
    fprintf("GM,PM etc. for the \rho value = %.2f\n",rho(i));
    S = allmargin(Lsf(i))
    hold on;
end
grid minor;
legend(legend_name,'Interpreter','latex');
title('Bode plot for different \rho values');

% mag = squeeze(mag);
% magdb = 20*log10(mag)

%% Design LTR
```

```
w=logspace(-1,3,1000);
rho_ltr = 1;
Q_ltr = rho_ltr*C'*C;
r = 1;
tf = K_Lqr*inv(s*eye(size(A))-A)*B;
[K] = lqr(A,B,Q_ltr,r);
sys1 = ss(A,B,K,0);
[maggk1,phasgk1,w]=bode(sys1,w);

q = [1, 10, 100];
rv=1;
% MATLAB lqe
for i = 1:length(q)
    gam=q(i)*B;
    Q1=gam'*gam;
    LTR(i).L=lqe(A,gam,C,Q1,rv);

    % MATLAB bode MATLAB margin
    aa=A-B*K-(LTR(i).L)*C;
    bb=(LTR(i).L);
    cc=K;
    dd=0;
    sysk=ss(aa,bb,cc,dd);
    sysgk=series(sys1,sysk);
    [maggk,phsgk,w]=bode(sysgk,w);
    [gm,phm,wcg,wcp] = margin(maggk,phsgk,w)

    subplot(2,1,1);
    loglog(w, maggk(:),'linewidth',1.2); % loglog(w,[maggk1(:)
   maggk(:)]);
    grid;
    hold all
    subplot(2,1,2);
    semilogx(w,phsgk(:),'linewidth',1.2); %semilogx(w,[phasgk1(:)
   phsgk(:)]);
    grid;
    hold all;
end

subplot(2,1,1);
title('LTR Controller Gain and Phase Margin')
loglog(w, maggk1(:),'linewidth',1.2);
ylabel("Gain Margin")


subplot(2,1,2);
semilogx(w,phasgk1(:),'linewidth',1.2)
ylabel("Phase Margin")
xlabel('\omega [rad/s]')

legend('q = 1','q = 10','q = 100','LQR')
```

LTR Design for part-A

```
%%%%%%%% PART C
clc;clear all;close all
```

```
%% Define systems (PART A & PART B)
% NOTE: RUN THE SYSTEM IN PART-A OR IN PART-B
s = tf('s');
%% First system --> Part A
Py = 1/((0.5*s + 1)*(s^2+2*(0.01)*(0.8)*s+(0.8)^2));

%% State-space representation
% Define A,B,C,D Matrices for PART-A
[A,B,C,D] = tf2ss(cell2mat(Py.numerator),cell2mat(Py.denominator));
% State-space representation
sys = ss(A,B,C,D);

%% Chosen LQR gains for PART-A
% %  For rho = 0.3
% K_Lqr = [1.4711 4.0477 2.5893];

% For rho = 0.5
K_Lqr = [1.2389 3.2651 1.8246];

% Eigenvalues
% Or can be obtained from previous lqr func. result
% Closed loop pole locations
eigg = eig(A-B*K_Lqr);

% Check LQR Gain of the system
K_ack = acker(A,B,eigg);
%% Symmetric root locus plot PART-A
A_srl = [A zeros(size(A));-C'*C -A'];
B_srl = [B;-C'*D];
C_srl = [D'*C B'];
D_srl = D'*D;
sys_srl = ss(A_srl,B_srl,C_srl,D_srl);
figure;
rlocus(sys_srl)
% plot also the eigenvalues
hold all
plot(eigg,'o','Linewidth',2);
legend('SRL','Controller poles')
grid

%% Observer Poles & Find the Observer Gain
% The bandwith is 1.23 rad/s. Therefore, choose pole loc as:
% estimator pole location
% Chose the gain as: G = 145
pe = [-1.48+2.37i; -3.09; -1.48-2.37i];
% Observer Gain
L = place(A',C',pe)

%% tf from process noise to sensor output
% Ge(s)
Ge = C * inv(s*eye(size(A)) - A)*B;
% Ge(-s)
Ge_neg = C * inv(-s*eye(size(A)) - A)*B;
figure;
rlocus(Ge_neg*Ge)
% Estimator SRL eqn: 1+q*Ge*Ge_neg
```

```
%% Symmetric root locus plot with Observer Gains
A_srl = [A zeros(size(A));-C'*C -A'];
B_srl = [B;-C'*D];
C_srl = [D'*C B'];
D_srl = D'*D;
sys_srl = ss(A_srl,B_srl,C_srl,D_srl);
figure;
rlocus(sys_srl)
% plot also the eigenvalues
hold on
plot(eigg,'o','Linewidth',2);
hold on
plot(pe,'o','Linewidth',2);
legend('SRL','Controller poles','Observer Poles')
grid

%%%%%%%%%%%%%%%%%%
%% Simulink Part
N_bar = rscale(sys,K_Lqr);

noise_pow = [0.01, 0.1, 1, 10, 100];
figure;
for i = 1:length(noise_pow)
    set_param('LQR_model/Band-Limited White Noise','Cov',string(
   noise_pow(i)))
    y = sim('LQR_model')
    plot(y.yout{1}.Values.Time, y.yout{1}.Values.Data, 'linewidth',
    1.25)
    hold on
end

legend('0.01', '0.1' , '1' ,'10', '100' )

title('Increased Noise Power and Simulation Results')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Design LTR
w=logspace(-1,3,1000);
rho_ltr = 0.5;
Q_ltr = rho_ltr*C'*C;
r = 1;
tf = K_Lqr*inv(s*eye(size(A))-A)*B;
[K] = lqr(A,B,Q_ltr,r);
sys1 = ss(A,B,K,0);
[maggk1,phasgk1,w]=bode(sys1,w);

q = [1, 10, 100];
rv=1;
% MATLAB lqe
for i = 1:length(q)
    gam=q(i)*B;
    Q1=gam'*gam;
    LTR(i).L=lqe(A,gam,C,Q1,rv);

    % MATLAB bode MATLAB margin
```

```
    aa=A-B*K-(LTR(i).L)*C;
    bb=(LTR(i).L);
    cc=K;
    dd=0;
    sysk=ss(aa,bb,cc,dd);
    sysgk=series(sys,sysk);
    [maggk,phsgk,w]=bode(sysgk,w);
    [gm,phm,wcg,wcp] = margin(maggk,phsgk,w)

    subplot(2,1,1);
    loglog(w, maggk(:),'linewidth',1.2);
    grid;
    hold all
    subplot(2,1,2);
    semilogx(w,phsgk(:),'linewidth',1.2);
    grid;
    hold all;
end

subplot(2,1,1);
title('LTR Controller Gain and Phase Margin')
loglog(w, maggk1(:),'linewidth',1.2);
ylabel("Gain Margin")


subplot(2,1,2);
semilogx(w,phasgk1(:),'linewidth',1.2)
ylabel("Phase Margin")
xlabel('\omega [rad/s]')

legend('q = 1','q = 10','q = 100','LQR')
```

LTR Design for part-B

```
%%%%%%%% PART C -- system B
clc;clear all;close all
% NOTE:
s = tf('s');
%% Second system --> Part B
Py = (s^2+2*0.5*0.8*s+0.8^2)/((0.5*s + 1)*(s^2+2*(0.01)*(0.8)*s
    +(0.8)^2));

%% State-space representation
% Define A,B,C,D Matrices for PART-A
[A,B,C,D] = tf2ss(cell2mat(Py.numerator),cell2mat(Py.denominator));
% State-space representation
sys = ss(A,B,C,D);

%% Chosen LQR gains for PART-A and PART-B
% %  For rho = 0.3
% K_Lqr = [2.7887 2.8440 1.3845];

% For rho = 0.5
K_Lqr = [2.0370 2.1814 0.9370];

% Eigenvalues
```

```
% Or can be obtained from previous lqr func. result
eigg = eig(A-B*K_Lqr);

%% Symmetric root locus plot PART-B
A_srl = [A zeros(size(A));-C'*C -A'];
B_srl = [B;-C'*D];
C_srl = [D'*C B'];
D_srl = D'*D;
sys_srl = ss(A_srl,B_srl,C_srl,D_srl);
figure;
rlocus(sys_srl)
% plot also the eigenvalues
hold all
plot(eigg,'o','Linewidth',2);
legend('SRL','Controller poles')
grid

%% tf from process noise to sensor output
% Ge(s)
Ge = C * inv(s*eye(size(A)) - A)*B;
% Ge(-s)
Ge_neg = C * inv(-s*eye(size(A)) - A)*B;
figure;
rlocus(Ge_neg*Ge)
% Estimator SRL eqn: 1+q*Ge*Ge_neg

%% Observer Poles & Find the Observer Gain
% The bandwith is 1.23 rad/s. Therefore, choose pole loc as:
% estimator pole location
% Chose the gain as: G = 145
pe = [-0.378+0.709i; -6.09; -0.378-0.709i];
% Observer Gain
L = place(A',C',pe)

%% Symmetric root locus plot with Observer Gains
A_srl = [A zeros(size(A));-C'*C -A'];
B_srl = [B;-C'*D];
C_srl = [D'*C B'];
D_srl = D'*D;
sys_srl = ss(A_srl,B_srl,C_srl,D_srl);
figure;
rlocus(sys_srl)
% plot also the eigenvalues
hold on
plot(eigg,'o','Linewidth',2);
hold on
plot(pe,'o','Linewidth',2);
legend('SRL','Controller poles','Observer Poles')
axis([-7 7 -1 1])
grid

%%%%%%%%%%%%%%%%
%% Simulink Part
N_bar = rscale(sys,K_Lqr);

noise_pow = [0.01, 0.1, 1, 10, 100];
```

```
figure;
for i = 1:length(noise_pow)
    set_param('LQR_model/Band-Limited White Noise','Cov',string(
   noise_pow(i)))
    y = sim('LQR_model')
    plot(y.yout{1}.Values.Time, y.yout{1}.Values.Data, 'linewidth',
    1.25)
    hold on
end

legend('0.01', '0.1' , '1' ,'10', '100' )

title('Increased Noise Power and Simulation Results')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Design LTR
w=logspace(-1,3,1000);
rho_ltr = 1;
Q_ltr = rho_ltr*C'*C;
r = 1;
tf = K_Lqr*inv(s*eye(size(A))-A)*B;
[K] = lqr(A,B,Q_ltr,r);
sys1 = ss(A,B,K,0);
[maggk1,phasgk1,w]=bode(sys1,w);

q = [1, 10, 100];
rv=1;
% MATLAB lqe
for i = 1:length(q)
    gam=q(i)*B;
    Q1=gam'*gam;
    LTR(i).L=lqe(A,gam,C,Q1,rv);

    % MATLAB bode MATLAB margin
    aa=A-B*K-(LTR(i).L)*C;
    bb=(LTR(i).L);
    cc=K;
    dd=0;
    sysk=ss(aa,bb,cc,dd);
    sysgk=series(sys,sysk);
    [maggk,phsgk,w]=bode(sysgk,w);
    [gm,phm,wcg,wcp] = margin(maggk,phsgk,w)

    subplot(2,1,1);
    loglog(w, maggk(:),'linewidth',1.2);
    grid;
    hold all
    subplot(2,1,2);
    semilogx(w,phsgk(:),'linewidth',1.2);
    grid;
    hold all;
end

subplot(2,1,1);
title('LTR Controller Gain and Phase Margin')
loglog(w, maggk1(:),'linewidth',1.2);
```

```
ylabel("Gain Margin")


subplot(2,1,2);
semilogx(w,phasgk1(:),'linewidth',1.2)
ylabel("Phase Margin")
xlabel('\omega [rad/s]')

legend('q = 1','q = 10','q = 100','LQR')
```

References

[1] Inverted pendulum: State-space methods for controller design. URL https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum\T1\textsectionion=ControlStateSpace.

[2] Function rscale: Finding the scale factor to eliminate steady-state error. URL https://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_rscale.

[3] Introduction: State-space methods for controller design. URL https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction\T1\textsectionion=ControlStateSpace.

[4] Gene F. Franklin, David J. Powell, and Abbas Emami-Naeimi. *Feedback control of Dynamic Systems*. Pearson, 2020.

[5] Help center. URL https://www.mathworks.com/support/search.html/videos/robust-control-part-4-working-with-parameter-uncertainty-1586760097653.html?fq%5B%5D=category%3Arobust%2Funcertain-models&page=1.