Robotic Link/Joint Control System:
Dynamic Shaping and Linear State Feedback Control Laws

Aly Khater
Lawrence Lai

Introduction

Our system is a single robotic joint/link driven from a gear by an armature-controlled dc servo motor. Our main objective is to have the system be stable, and create an accurate estimation of the output. The solution is to use MATLAB for calculations and simulation. In utilizing MatLab, we will demonstrate our results in a graphical format as well as in various state-space representations.

Objective

For this report the objective will be split into six phases labeled A-F. In phase A, the objective is to show the state space representation of the given transfer function of the system and to plot the input states to a unit step response. The requirements for this phase is that the numerical values of the state space representation comes from Table 1. Another objective of part A is to transform the state space representation into its diagonal canonical form.

For phase B the objective is to test whether or not the system is controllable and if it is to put it into its control canonical form. The requirements for phase B is that the rank of the controllability matrix is the dimension of the system which is three.

The objective of phase C is to verify whether the system satisfies Lyapunov stability analysis and to plot the stability of the system. The way to verify the Lyapunov stability analysis is by locating the eigenvalues of matrix A in the system in any of the state space representation and to check if the eigenvalues are strictly in the left hand plane. If Lyapunov stability analysis fails a second requirement is to check if any of the eigenvalues is in the right hand plane.

Phase D objective is to design a controller for the system and plot its input case. The requirement for this section is to first find its desired dominant eigenvalue using a settling time of 0.5s and then adding two more eigenvalues that is at least 10 times bigger than the dominant one. The controller will have its new eigenvalues using MATLAB pole-placement function.

For phase E the objective is to test for observability and to to put that into its observer canonical form. The requirements for phase E is that the rank of the observability matrix is the dimension of the system which is three.

For the last phase F, the objective is to check for minimal state realization and to design an observer based compensator. The requirement to check for minimal state realization is to find whether the system is controllable and observable. The requirements for the compensator is that its eigenvalues must be 10 bigger than the desired eigenvalues found in phase D. Similarly MATLAB pole-placement function will be used to get the compensator its new eigenvalues.

Background

Our system involves an RL circuit, a motor outputting rotational motion, connected to a gear that is also rotating, which then leads into our load. The load is connected to the robot link/joint to provide its motions. See Figure 1 for each aspect of the system, and the names assigned to each variable.

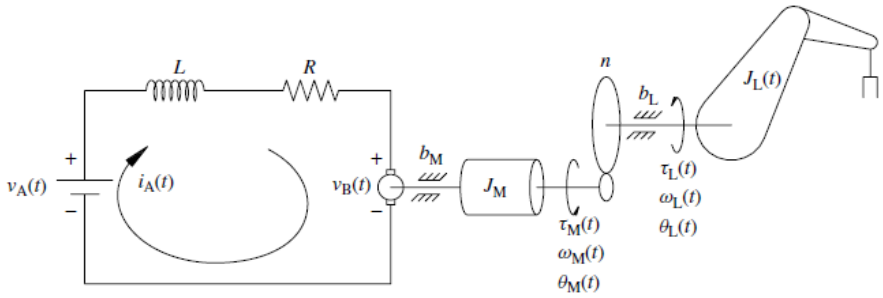


FIGURE 1.16 Diagram for Continuing Exercise 3.

$v_A(t)$	armature voltage	L	armature inductance	R	armature resistance
$i_A(t)$	armature current	$v_B(t)$	back emf voltage	k_B	back emf constant
J_M	motor inertia	b_M	motor viscous damping	$\tau_M(t)$	motor torque
k_T	torque constant	$\omega_M(t)$	motor shaft velocity	$\theta_M(t)$	motor shaft angle
n	gear ratio	$J_L(t)$	load inertia	b_L	load viscous damping
$\tau_L(t)$	load shaft torque	$\omega_L(t)$	load shaft velocity	$\theta_L(t)$	load shaft angle

Figure 1: Robot Link/joint system connected to motor and RL circuit

Variables that are controlled in this system are: $L,\ R,\ k_B,\ J_M,\ b_M,\ k_t,\ n,\ J_L,\ b_L$.

Design and Simulation Results

We split each stage of the design into separate phases, with each phase design and results below. For the purpose of this report, we will mainly showcase Case I of our system. Plots of Case II and Case III can be found in Appendix G.

Phase A: State Space Representation and Diagonal Canonical Form

For our main design, our state space representation coefficient matrices, A, B, C, D are represented as follows:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -\frac{(R \cdot b + k_T \cdot k_b)}{L \cdot J} & -\frac{(L \cdot b + R \cdot J)}{L \cdot J} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{k_T}{L \cdot J \cdot n} \end{bmatrix}$$

$$b = b_M + \frac{bL}{n^2}$$

$$J = J_M + \frac{JL}{n^2}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

To simulate open-loop state variable responses, we used the values shown in Table 1.

Table 1: Values and parameters of the robotic system [1]

Parameter	Value	Units	Name
L	0.0006	H	armature inductance
R	1.40	Ω	armature resistance
k_B	0.00867	V/deg/s	motor back emf constant
J_M	0.00844	lb_f-in-s^2	motor shaft polar inertia
b_M	0.00013	lb_f-in/deg/s	motor shaft damping constant
k_T	4.375	lb_f-in/A	torque constant
n	200	unitless	gear ratio
J_L	1	lb_f-in-s^2	load shaft polar inertia
b_L	0.5	lb_f-in/deg/s	load shaft damping constant

Utilizing the values in Table 1 and using MatLab for our calculations, we arrived at the following matrices:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -7508 & -2333 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 4307 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

which returns us with the following characteristic polynomial:

$$x^2 + 2333x + 7508$$

From the characteristic polynomial, we arrive at eigenvalues of -3.2 and -2329.8. We then simulated the system running until it reached steady-state behavior as shown in Figure 2.

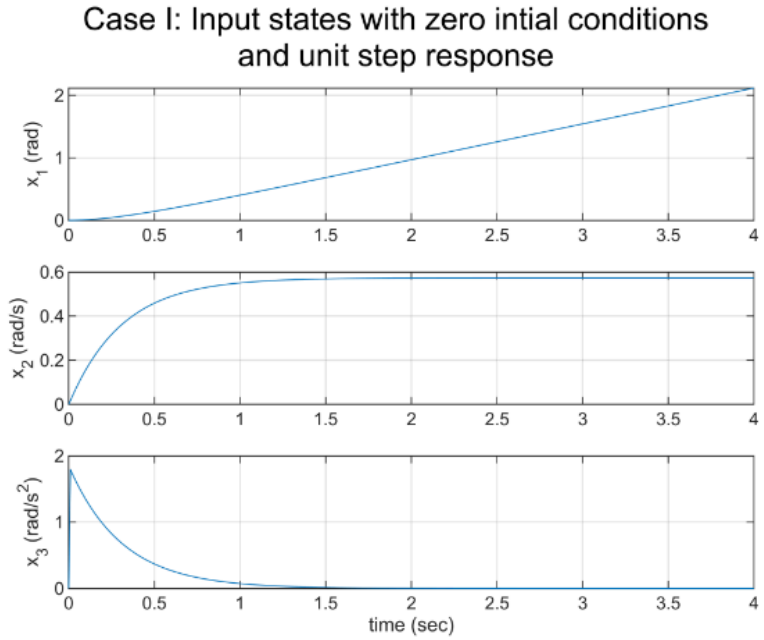


Figure 2: Simulation of the original system.

For the remainder of the report, Matrix D remains at 0, so we will not include matrix D for the rest of the simulation.

The second part of Phase A was to calculate the diagonal canonical form realization as shown:

$$A_{DCF} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -3.2 & 0 \\ 0 & 0 & -2330 \end{bmatrix} B_{DCF} = \begin{bmatrix} .6 \\ 6.3 \\ 4313 \end{bmatrix} C_{DCF} = [1 \quad -.1 \quad 0]$$

For all scripts used in Phase A, refer to Appendix A at the end of this report.

Phase B: Controllability and Control Canonical Form(CCF)

Here, we assessed the controllability of our system and computed the control canonical form of our system.

In using our steady state representation for the system, we used the matlab function *ctrb(sys)*; which computes the controllability matrix from our state-space model in *sys*, which is generated by the following matrix P for a third order system:

$$P = [B \quad AB \quad A^2B]$$

In doing so, our system is **controllable**.

We then find the CCF through a series of transformations on our state space system. To do this, we first found our inverse controllability matrix in CCF, denoted as P_{ccfi}.

$$P_{CCFi} = \begin{bmatrix} \frac{(Rb + k_t k_b)}{LJ} & \frac{(Lb + RJ)}{LJ} & 1 \\ \frac{(Lb + RJ)}{LJ} & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

We then find our transformation matrix $T_{ccf} = PP_{ccfi}$, which is utilized to find the CCF of our state space model. The follow matrices represent our system in CCF:

$$A_{CCF} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -7507 & -2334 \end{bmatrix} B_{CCF} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} C_{CCF} = [4307 \quad 0 \quad 0]$$

For all scripts used in Phase B, refer to Appendix B at the end of this report.

Phase C: Lyapunov Analysis and Stability

For Phase C, we analyzed the system using Lyapunov's analysis to assess the system's stability.

For Lyapunov's, we utilized our characteristic polynomial

$$x^2 + 2333x + 7508$$

to get our poles, but we were unable to successfully use Lyapunov's method to assess our stability. We then moved onto using our eigenvalues to assess if the system is stable. Via eigenvalue analysis, our system turned out to be **marginally stable**. The stability of our system is shown in Figure 3.

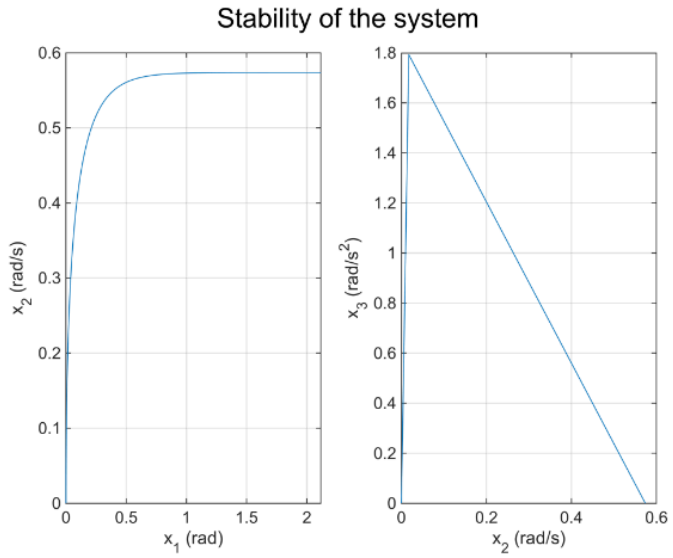


Figure 3: Simulation results of our system, showcasing marginally stability.

For all scripts used in Phase C, refer to Appendix C at the end of this report.

Phase D: Design of the Linear State Feedback Control

We first used the dominant first order system to find out the initial eigenvalue (EV). Our system was given a time constant $\tau = 0.5$ s. Using the inverse of the time constant τ, we selected our desired EV, $\lambda_1 = -2$. As our system is a third order system, we needed two more desired eigenvalues. The second EV, $\lambda_1 = -20$, was selected by multiplying our first desired EV by 10. The third EV, $\lambda_3 = -21$, was selected by subtracting 1 from the second desired EV. Desired EVs are shown below for organization:

$$\lambda_1 = -2;\ \lambda_2 = -20;\ \lambda_3 = -21$$

We then plot our third-order desired EVs alongside our first-order EVs to compare the step response as shown in Figure 4.

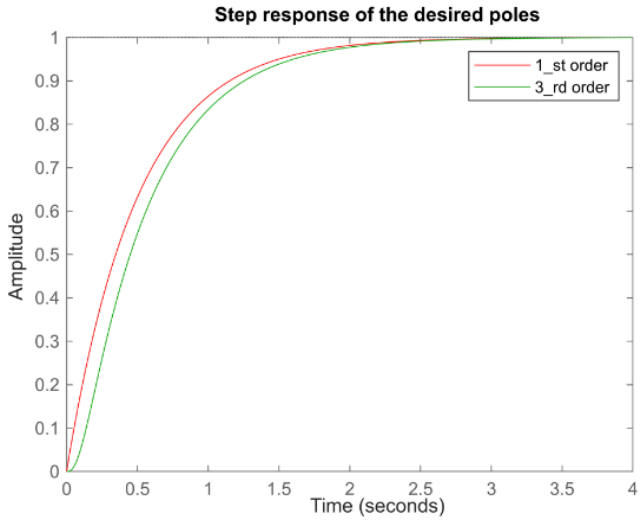


Figure 4: Step response of first-order and third-order system

We now design a state feedback control system for our desired closed-loop EVs. We first needed to solve for K using the MatLab function, *place*.

$$\mathrm{K} = \begin{bmatrix} 0.195 & -1.626 & -0.532 \end{bmatrix}$$

We utilized Ackerman's formula by using the MatLab function, *acker*. With our control law K, we can form our closed loop state space representation of A-BK replacing our A matrix in the open loop version.

$$\text{A-BK} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -840 & -502 & -43 \end{bmatrix}$$

We then plot our newly designed controller versus the original system as shown in Figure 5.

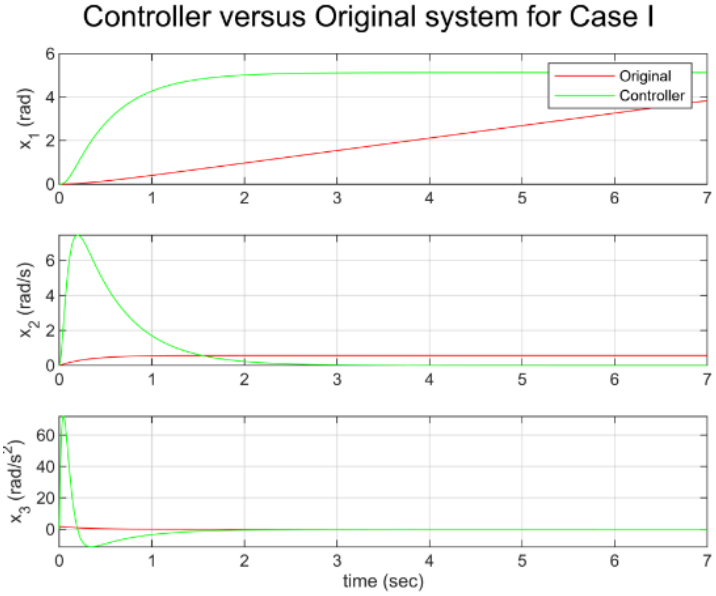


Figure 5: Designed Controller versus the original system.

For all scripts used in Phase D, refer to Appendix D at the end of this report.

Phase E: Observability and Observer Canonical Form(OCF)

For Phase E, we look to find if the system is observable, and to demonstrate our state-space system in OCF.

In using our steady state representation for the system, we used the matlab function *obsv(sys)*; which computes the observability matrix from our state-space model in *sys*, which is generated by the following matrix Q for a third order system:

$$Q = \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix}$$

The function deemed our system **observable.**

We then find the OCF through a series of transformations on our state space system. We find $Q_{OCF} = P_{CCF}$ to get our transformation matrix $T_{OCF} = Q^{-1}Q_{OCF}$. In finding both of these matrices, we can transform our state space matrices into OCF as follows:

$$A_{OCF} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -7507 \\ 0 & 1 & -2334 \end{bmatrix} B_{OCF} = \begin{bmatrix} 4307 \\ 0 \\ 0 \end{bmatrix} C_{OCF} = [0 \quad 0 \quad 1]$$

For all scripts used in Phase E, refer to Appendix E at the end of this report.

Phase F: Minimal State Realization and Observer Based Compensators

For the final phase of our project, we found the minimal state realization of our system and designed an observer for our system.

To check if the system is minimal, we used the MatLab function, *minreal(sys)*, which eliminates any part of the system that may be uncontrollable or unobservable. As our system was already controllable and observable in previous tests, **our system was already in its minimal state**.

In designing our observer, we needed three desired eigenvalues(EV). To select our desired EVs, we took our designed controller EVs and multiplied each by 10. Our observer desired EVs are as follows:

$$\lambda_1 = -20; \lambda_2 = -200; \lambda_3 = -210$$

We then created our desired characteristic polynomial, and utilized Ackerman's formula to get our gain matrix L and A-LC:

$$L = \begin{bmatrix} -1903 \\ 4.5e6 \\ -1.04e10 \end{bmatrix} A - LC = \begin{bmatrix} 1903 & 1 & 0 \\ -4.5e6 & 0 & 1 \\ -1.04e10 & -7507 & -2334 \end{bmatrix}$$

In getting our L matrix, our closed-loop observer can be shown as A-LC. We then plot our observer versus our controller and original system in Figure 6.

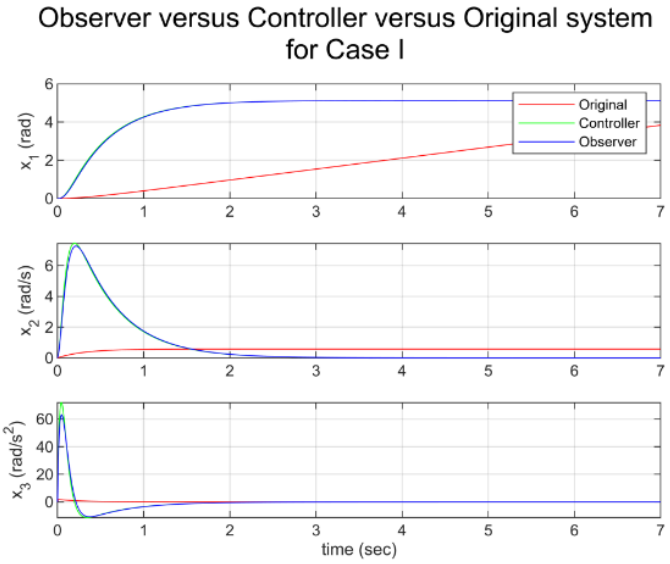


Figure 6: Observer vs. Controller vs. Original System

For all scripts used in Phase F, refer to Appendix F at the end of this report.

Analysis

Phase A: State Space Representation and Diagonal Canonical Form

From the eigenvalues of Phase A, two values are negative, showcasing that they are in the left hand plane. However one of the eigenvalues is a zero which is in the border of the left hand plane. Figure 2 shows that when there is a unit step input applied to the system that the input variable X_1 will go toward infinity which shows that the system is not asymptotically stable. The other variable goes toward a finite number at steady state which will be important in Phase 3. Looking at the diagonal canonical form since the system is decoupled, therefore all the state variables are independent from each other.

Phase B: Controllability and Control Canonical Form

Our system was deemed controllable by the MatLab script by using the controllability matrix. In solving the controllability matrix, we then find the Rank of the matrix, which determines if the vectors of the matrix are linearly independent. If all are linearly independent, then the system is controllable. Our system follows these rules as it is controllable. The control canonical form shows the eigenvalues of the system by looking at the bottom row of its A matrix. A benefit of the control canonical form is that theoretically we can choose which eigenvalue needs to be moved without affecting the other eigenvalue. However, in this report all the eigenvalues will be moved to make it asymptotically stable.

Phase C: Lyapunov Analysis and Stability

Looking at either the diagonal canonical form or control canonical shows that in matrix A one of the eigenvalues is zero. For Lyapunov stability analysis requires that all eigenvalues of the eigenvalues found in Matrix A must strictly be in the left hand plane. Therefore the system does not fit the criteria for Lyapunov stability analysis and it will fail. Using eigenvalue analysis, none of the eigenvalues is in the right hand plane showing marginally stability. Another requirement was to plot the stability of the system shown in figure 3. In order for the system to be asymptotically stable all of the dependent variables must go toward zero as the independent variable goes toward infinity. However if none of the dependent variables go towards zero then the system will be unstable. The finding in figure 3 shows only the second plot going toward zero which verifies the results found using the eigenvalue analysis.

Phase D: Design of the Linear State Feedback Control

Figure 4 shows the normalized first and third order response of a settling time of 0.5s. The first order response is faster than the third order response. However, the shape of the third order is similar to the first order since its dominant eigenvalue is the same as the third order. The reason why a third order response is needed despite being slower than the first order is because the system being designed is in the third order and having three eigenvalues will allow the usage of

Matlab pole-placement technique, which avoids the necessity of manual calculation to find the K gain values.
After using pole-placement to get the new eigenvalue of the system, the system is now asymptotically stable instead of marginally stable. The new A-BK matrix has all negative eigenvalues verifying the stability of the system. Figure 5 is a comparison between the open loop and closed loop system with a unit step response. For all three input states of the controller based system, the steady state value goes toward a finite number instead of infinity. Another difference is that for the original system only X_3 went toward zero for a steady state value and for the controller both X_2 and X_3 went towards zero for its steady state value. What this indicates is that if the stability of the controller is plotted the dependent variables will go towards zero as the independent variables go toward infinity showing an asymptotically stable system.

Phase E: Observability and Observer Canonical Form

Our system was deemed observable by the MatLab script by using the observability matrix. In solving the observability matrix, we then find the Rank of the matrix, which determines if the vectors of the matrix are linearly independent. If all are linearly independent, then the system is observable. Our system follows these rules as it is observable. The observer canonical form shows the eigenvalues of the system by looking at the rightmost column of its A matrix. Comparing the observer canonical form and diagonal canonical form both forms allows indication of the eigenvalue of the system however the purpose of which they are used is different. When comparing the observer canonical form and control canonical form, the observer's matrices look like transposes of various control canonical form matrices. Observer A matrix is the transpose of control A matrix, observer B matrix is the transpose of control C matrix and observer C matrix is the transpose of control B matrix.

Phase F: Minimal State Realization and Observer Based Compensators

The original system was in its minimal realization because it was found that the system is both controllable and observable which is the definition of minimal realization. Minimal realization means that the system is already the smallest it can be and there is no unnecessary part to the system.
The eigenvalues of the observer have to be bigger than the eigenvalues of the controller because the output of the observer feeds into the controller. Figure 6 plots the input states of the original, controller, observer system with a unit step response. Similarly to phase D, the observer compared to the original looks similar to the controller compared to the original. This is intended as a good observer should match the output of the desired system which would be the controller based system. The requirement was that the observer had a small initial input value to the unit step response, so that the observer and controller are not identical. This is acceptable as an estimate would not look exactly like the desired response in a real world example.

Conclusions

In conclusion, our study extensively analyzed a single robotic joint/link system, utilizing MATLAB for simulation and representation. The project, divided into various phases, focused on understanding and enhancing the system's stability and response characteristics.

Phase A demonstrated lack of its stability due to one of its eigenvalues, as shown in the diagonal canonical form. Phase B confirmed the systems controllability, allowing us to

manipulate our eigenvalues for stability. However, in Phase C, Lyapunov's analysis indicated that the system is marginally stable. With our implementation of the linear state feedback controller in Phase D, we were able to shift the system from marginally stable to asymptotically stable, demonstrating an improvement in stability for the entire system. Phase E's observability analysis allowed us to see that the system states can be understood from its outputs. Finally, Phase F established the system's minimal state realization, showcasing that the system has been fully optimized with no redundant components.

Overall, our analysis of the robotic system using MatLab simulations and the various state-space representations demonstrated problems and solutions into the system's dynamics, controllability, observability, stability, and optimization. This project provides us with a framework for future implementations and improvements in similar control systems.

Appendix

This appendix will contain all the MatLab scripts used to design the robotic system. Every following appendix is appended onto the previous one, creating one continuous script.

Appendix A: State Space Representation and Diagonal Canonical Form

```
%Chapter 2 State Space Representation and Diagonal Canonical Form
L = 0.0006;
R = 1.40;
kB = 0.00867;
JM = 0.00844;
bM = 0.00013;
kT = 4.375;
n = 200;
JL = 1;
bL = 0.5;

J = JM + (JL/(n^2));
b = bM + (bL/(n^2));

A = [0 1 0; 0 0 1; 0 -(R*b + kT*kB)/(L*J) -(L*b + R*J)/(L*J)];
B = [0; 0; kT/(L*J*n)];
C = [1 0 0];
D = 0;

JbkR = ss(A,B,C,D)

t = 0:0.01:4;
U = [ones(size(t))];
x0 = [0; 0; 0];
[Yo,t,Xo] = lsim(JbkR,U,t,x0);

figure;
subplot(311), plot(t,Xo(:,1)); grid on;
ylabel('x_1 (rad)');
```

```
 subplot(312), plot(t,Xo(:,2)); grid on;
 ylabel('x_2 (rad/s)');
 subplot(313), plot(t,Xo(:,3)); grid on;
 xlabel('time (sec)'); ylabel('x_3 (rad/s^2)');
 sgtitle({'Case I: Input states with zero intial conditions';'and unit step
response'})


 U2 = [zeros(size(t))];
 x02 = [0; 1; 0];
 [Yo2,t,Xo2] = lsim(JbkR,U2,t,x02);

 figure;
 subplot(311), plot(t,Xo2(:,1)); grid on;
 ylabel('x_1 (rad)');
 subplot(312), plot(t,Xo2(:,2)); grid on;
 ylabel('x_2 (rad/s)');
 subplot(313), plot(t,Xo2(:,3)); grid on;
 xlabel('time (sec)'); ylabel('x_3 (rad/s^2)');
 sgtitle({'Case II: Input states with intial conditions (x_2 = 1)';'and zero
input response'})


 A2 = [0 1 ; -(R*b + kT*kB)/(L*J) -(L*b + R*J)/(L*J)];
 B2 = [0; kT/(L*J*n)];
 C2 = [1 0];
 D2 = [0];

 JbkR2 = ss(A2,B2,C2,D2)

 x03 = [0; 0];
 [Yo3,t,Xo3] = lsim(JbkR2,U,t,x03);

 figure;
 subplot(211), plot(t,Xo3(:,1)); grid on;
 ylabel('x_1 (rad/s)');
 subplot(212), plot(t,Xo3(:,2)); grid on;
 xlabel('time (sec)'); ylabel('x_2 (rad/s^2)');
 sgtitle({'Case III: Modified Input states with zero intial conditions';'and
unit step response'})


 [Tdcf, eig0] = eig(A);

 Adcf = inv(Tdcf)*A*Tdcf
 Bdcf = inv(Tdcf)*B
 Cdcf = C*Tdcf
```

```
Ddcf = D
```

Appendix B: Controllability and Control Canonical Form

```
%Chapter 3 Controllability and Control Canonical Form
P = ctrb(JbkR);
if (rank(P) == size(A,1))
disp('System is controllable.');
else
disp('System is NOT controllable.');
end


Pccfi = [(R*b + kT*kB)/(L*J) (L*b + R*J)/(L*J) 1; (L*b + R*J)/(L*J) 1 0;1 0 0];
Tccf = P*Pccfi;
Accf = inv(Tccf)*A*Tccf
Bccf = inv(Tccf)*B
Cccf = C*Tccf
Dccf = D
```

Appendix C: Lyapunov Analysis and Stability

```
%Chapter 6 Lyapunov Analysis and Stability
CharPoly = poly(A);
Poles = roots(CharPoly);

if (real(Poles(1)) == 0 | real(Poles(2)) == 0 | real(Poles(3)) == 0)
    if (real(Poles(1)) <= 0 | real(Poles(2)) <= 0 | real(Poles(3)) <= 0)
     disp('System is marginally stable.');
    else
     disp("System is unstable.");
    end
else
   Q = eye(3);
   PQ = lyap(A',Q);
   pm1 = det(PQ(1,1));
   pm2 = det(PQ(1:2,1:2));
   pm3 = det(PQ(1:3,1:3));
   if (pm1 > 0 & pm2 > 0 & pm3 > 0)
    disp('System is asymptotically stable.');
   else
    disp("System is unstable.");
   end
end
```

```
figure;
subplot(121), plot(Xo(:,1), Xo(:,2)); grid on;
xlabel('x_1 (rad)'); ylabel('x_2 (rad/s)');
subplot(122), plot(Xo(:,2), Xo(:,3)); grid on;
xlabel('x_2 (rad/s)'); ylabel('x_3 (rad/s^2)');
sgtitle('Stability of the system')
```

Appendix D: Design of the Linear State Feedback Control

```
% Chapter 7 Design of the the Linear State Feedback Control
Eig1 = -2;
Eig2 = -20;
Eig3 = -21;
den2 = poly([Eig1])

den3 = poly([Eig1; Eig2; Eig3]);
num3 = [den3(4)];
DesEig3 = roots(den3)
Des2 = tf(-Eig1,den2)
Des3 = tf(num3,den3)


figure;
td = [0:.01:7];
step(Des2,t,'r',Des3,t,'g');
title('Step response of the desired poles')
legend('1_st order','3_rd order')


K = place(A,B,DesEig3)

Kack = acker(A,B,DesEig3);
Ac = A-B*K;
Bc = B;
Cc = C;
Dc = D;

JbkRc = ss(Ac,Bc,Cc,Dc)

Ud = [ones(size(td))];
[Yo4,td,Xo4] = lsim(JbkR,Ud,td,x0);
[Yc2,td,Xc2] = lsim(JbkRc,Ud,td,x0);

Ud2 = [zeros(size(td))];
[Yo5,td,Xo5] = lsim(JbkR,Ud2,td,x02);
[Yc3,td,Xc3] = lsim(JbkRc,Ud2,td,x02);
```

```
figure;
subplot(311); plot(td,Xo4(:,1),'r',td,Xc2(:,1),'g'); grid on;
ylabel('x_1 (rad)');
legend('Original','Controller')
subplot(312); plot(td,Xo4(:,2),'r',td,Xc2(:,2),'g'); grid on;
ylabel('x_2 (rad/s)');
subplot(313); plot(td,Xo4(:,3),'r',td,Xc2(:,3),'g'); grid on;
xlabel('time (sec)'); ylabel('x_3 (rad/s^2)');
sgtitle('Controller versus Original system for Case I')


figure;
subplot(311); plot(td,Xo5(:,1),'r',td,Xc3(:,1),'g'); grid on;
ylabel('x_1 (rad)');
legend('Original','Controller')
subplot(312); plot(td,Xo5(:,2),'r',td,Xc3(:,2),'g'); grid on;
ylabel('x_2 (rad/s)');
subplot(313); plot(td,Xo5(:,3),'r',td,Xc3(:,3),'g'); grid on;
xlabel('time (sec)'); ylabel('x_3 (rad/s^2)');
sgtitle('Controller versus Original system for Case II')


den4 = poly([Eig1; Eig2]);
num4 = [den4(3)];
DesEig4 = roots(den4)

Des4 = tf(-Eig1,den4)

K2 = place(A2,B2,DesEig4)

G = 40/((R*b + kT*kB)/(L*J))

Ac2 = A2-B2*K2;
Bc2 = B2*G;
Cc2 = C2;
Dc2 = D2;

JbkRc2 = ss(Ac2,Bc2,Cc2,Dc2)

[Yo6,td,Xo6] = lsim(JbkR2,Ud,td,x03);
[Yc6,td,Xc6] = lsim(JbkRc2,Ud,td,x03);

figure;
subplot(211); plot(td,Xo6(:,1),'r',td,Xc6(:,1),'g'); grid on;
ylabel('x_1 (rad/s)');
legend('Original','Controller')
subplot(212); plot(td,Xo6(:,2),'r',td,Xc6(:,2),'g'); grid on;
xlabel('time (sec)'); ylabel('x_2 (rad/s^2)');
```

```
sgtitle('Controller versus Original system for Case III')
```

Appendix E: Observability and Observer Canonical Form

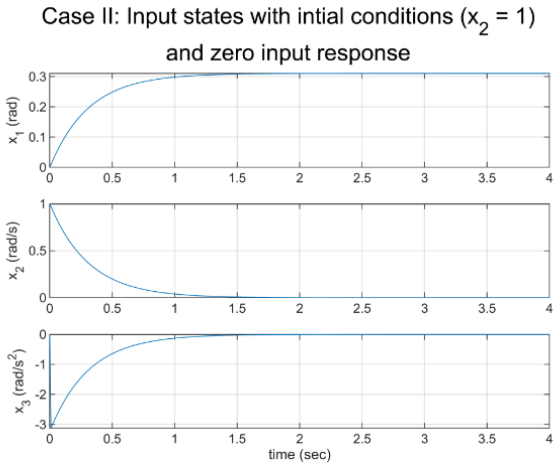
```
%Chapter 4 Observability and Observer Canonical Form
Q = obsv(JbkR);
if (rank(Q) == size(A,1))
disp('System is observable.');
else
disp('System is NOT observable.');
end

Q1 = [C; C*A; C*A^2]


Qocf = inv(Pccfi);
Tocf = inv(Q)*Qocf;
Aocf = inv(Tocf)*A*Tocf
Bocf = inv(Tocf)*B
Cocf = C*Tocf
Docf = D
```

Appendix F: Minimal State Realization and Observer Based Compensators

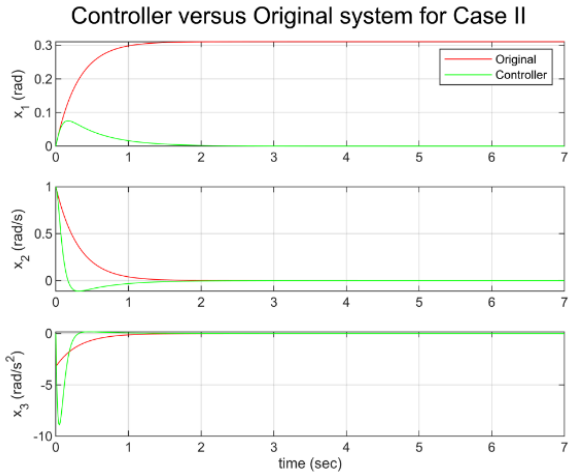
```
%Chapter 5 & 8 Minimal State Realization and Observer Based Compensators
CCF = ss(Accf,Bccf,Cccf,Dccf);
minCCF = minreal(CCF)


Eig1L = -20;
Eig2L = -200;
Eig3L = -210;
den3L = poly([Eig1L; Eig2L; Eig3L]);
num3L = [den3L(4)];
ObsEig3 = roots(den3L);
L2 = place(A',C',ObsEig3)';
Lack = acker(A',C',ObsEig3)';
Ahat = A-L2*C;
Ar = [(A-B*K) B*K;zeros(size(A)) (A-L2*C)];
Br = [B; zeros(size(B))];
Cr = [C zeros(size(C))];
Dr = D;

JbkRr = ss(Ar,Br,Cr,Dr)

x04 = [0; 0; 0; 0; 0.0000016; 0];
[Yr2,td,Xr2] = lsim(JbkRr,Ud,td,x04);
x05 = [0; 1; 0; 0; 0.0000004; 0];
```

```
[Yr3,td,Xr3] = lsim(JbkRr,Ud2,td,x05);

figure;
subplot(311); plot(td,Xo4(:,1),'r',td,Xc2(:,1),'g',td,Xr2(:,1),'b'); grid on;
ylabel('x_1 (rad)');
legend('Original','Controller','Observer')
subplot(312); plot(td,Xo4(:,2),'r',td,Xc2(:,2),'g',td,Xr2(:,2),'b'); grid on;
ylabel('x_2 (rad/s)');
subplot(313); plot(td,Xo4(:,3),'r',td,Xc2(:,3),'g',td,Xr2(:,3),'b'); grid on;
xlabel('time (sec)'); ylabel('x_3 (rad/s^2)');
sgtitle({'Observer versus Controller versus Original system';'for Case I'})


figure;
subplot(311); plot(td,Xo5(:,1),'r',td,Xc3(:,1),'g',td,Xr3(:,1),'b'); grid on;
ylabel('x_1 (rad)');
legend('Original','Controller','Observer')
subplot(312); plot(td,Xo5(:,2),'r',td,Xc3(:,2),'g',td,Xr3(:,2),'b'); grid on;
ylabel('x_2 (rad/s)');
subplot(313); plot(td,Xo5(:,3),'r',td,Xc3(:,3),'g',td,Xr3(:,3),'b'); grid on;
xlabel('time (sec)'); ylabel('x_3 (rad/s^2)');
sgtitle({'Observer versus Controller versus Original system';'for Case II'})


den4L = poly([Eig1L; Eig2L]);
num4L = [den4L(3)];
ObsEig4 = roots(den4L);
L3 = place(A2',C2',ObsEig4)';
Ahat2 = A2-L3*C2;
Ar2 = [(A2-B2*K2) B2*K2;zeros(size(A2)) (A2-L3*C2)];
Br2 = [B2*G; zeros(size(B2))];
Cr2 = [C2 zeros(size(C2))];
Dr2 = D;

JbkRr2 = ss(Ar2,Br2,Cr2,Dr2)

x06 = [0; 0; 0.0000004; 0];
[Yr4,td,Xr4] = lsim(JbkRr2,Ud,td,x06);

figure;
subplot(211); plot(td,Xo6(:,1),'r',td,Xc6(:,1),'g',td,Xr4(:,1),'b'); grid on;
ylabel('x_1 (rad/s)');
legend('Original','Controller','Observer')
subplot(212); plot(td,Xo6(:,2),'r',td,Xc6(:,2),'g',td,Xr4(:,2),'b'); grid on;
xlabel('time (sec)'); ylabel('x_2 (rad/s^2)');
sgtitle({'Observer versus Controller versus Original system';'for Case III'})
```

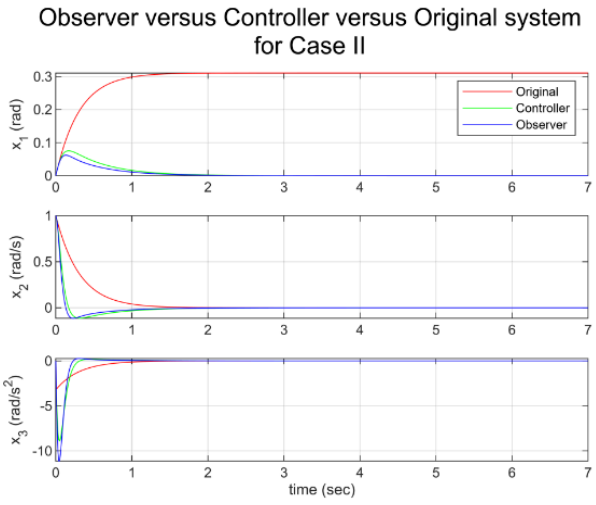
Appendix G: Simulation results for Case II



Chapter 2, Phase A

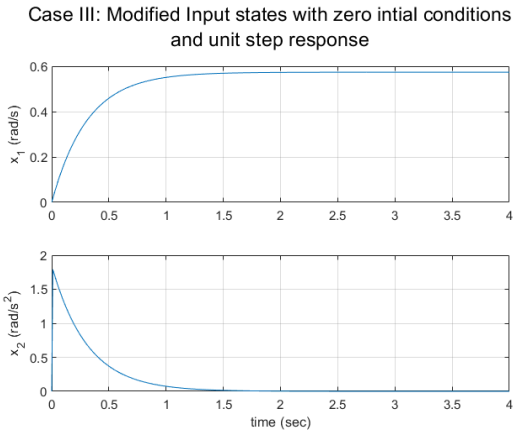


Chapter 7, Phase D

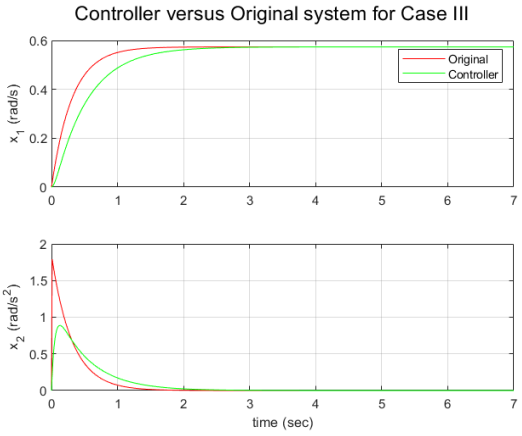


Chapter 8, Phase F

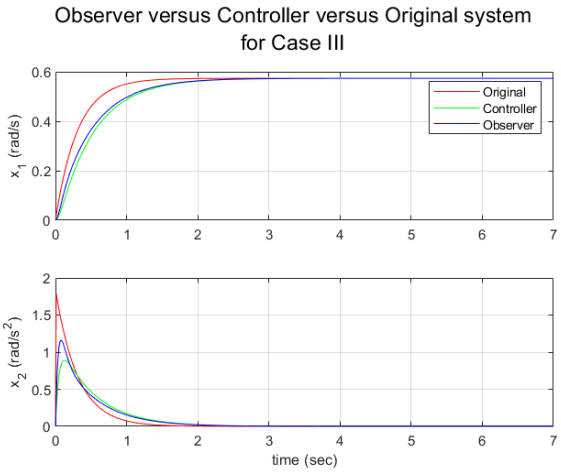
Appendix H: Simulation results for Case III



Chapter 2, Phase A



Chapter 7, Phase D



Chapter 8, Phase F