IUPUI

SCHOOL OF ENGINEERING AND TECHNOLOGY

A PURDUE UNIVERSITY SCHOOL

Indianapolis

# ECE 68000: Modern Automatic Control

# Project on Inverted Pendulum


Submitted by,

Sreeram Venkitachalam

University ID: 2000082784

# 1. INTRODUCTION

The purpose of this project is to understand the concepts which were discussed in the lecture. The project helps in understanding the practical implementation of the concepts to real world scenarios. The main concepts that are covered in the project include the following:

a. Finding the State space model for a non-linear system.
b. Linearizing a nonlinear system
c. Analysing the stability of the system using Lyapunov stability theory
d. Linear state feedback law
e. Combined state feedback controller and estimator
f. Linear Quadratic Regulator (LQR)

More details regarding the same are given in the problem statement

## 1.1. Problem Statement

Consider the pole-cart system described in Exercises 3.18 and Figure 3.19 of the textbook. You are asked to perform the following tasks:

a. Derive a nonlinear dynamical model in state-space using the following state variables

$$x_1 = x \qquad x_2 = \dot{x} \qquad x_3 = \theta \qquad x_4 = \dot{\theta}$$

And the output of the variable $y = x_1$

b. Obtain a linearized system model by linearizing the above system about the operating point

$$x = 0 \quad u = 0$$

c. Analyse the stability of the open-loop system using the Lyapunov stability theory.
d. Design a linear state feedback control law using the pole placement method assuming the state variables are measurable. Plot the state and control responses for the closed-loop control system with 3 different sets of control gain matrices.
e. If the state measurements are not available, design a combined state feedback control with state estimator using the Separation Principle. Plot the state and control responses for the closed-loop control system.
f. Design an optimal state feedback controller using LQR/steady-state LQR design method. In this design, you should try to use different weighting matrices and compare your results. Please show your simulation results by plotting appropriate state and control responses.

# 2. Understanding the System

The system consists of a pendulum which is connected to a cart in the inverse direction. Fig. 1 shows the system.

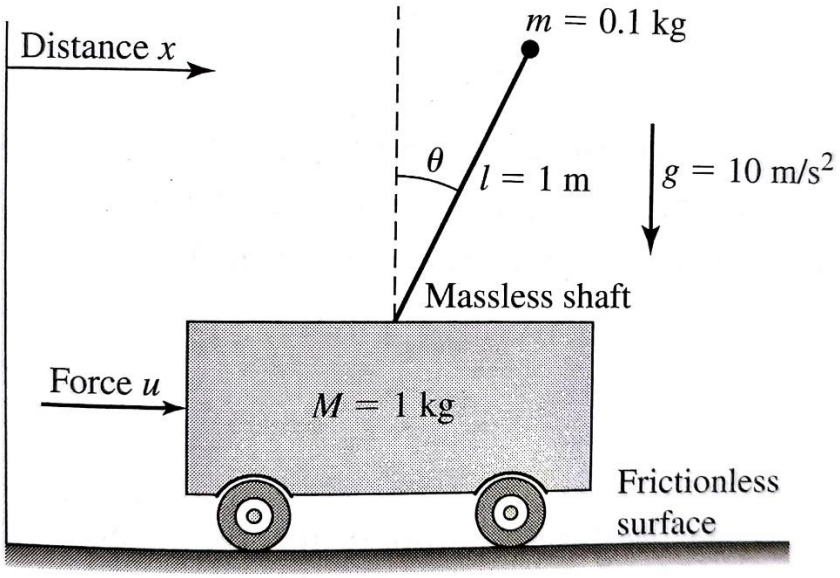


Fig. 1: The inverted pendulum system.

The values of the parameters are also given in the system and $\theta$ represents the angle between the perpendicular and the massless shaft.

## 2.1. Derivation for the non-linear system model

The state space of the system can be defined by using the Lagrange Equation of motion. We can first find the total kinetic energy of the system consisting of the inverted pendulum and the cart. The kinetic energy of the cart is given by:

$$KE_1 = \frac{1}{2} M * \dot{x}^2$$

Where *M* is the Mass of the cart and *x* is the distance covered by the cart pendulum system,

To find the Kinetic energy of the point mass *m*, we find the vertical and horizontal velocity of the pendulum. The horizontal velocity of the system is given by $\frac{d}{dt}(x + lsin\theta)$ and the vertical velocity is given by $\frac{d}{dt}(lcos\theta)$. Thus, the Kinetic Energy of the system is given by:

$$V^2 = \frac{d}{dt}(x + lsin\theta)^2 + \frac{d}{dt}(lcos\theta)^2$$

$$V^2 = \dot{X}^2 + 2l\dot{X}\dot{\theta} * cos\theta + l^2\dot{\theta}^2$$

$$KE_2 = \frac{1}{2} m * V^2 = \frac{1}{2} * m * (\dot{X}^2 + 2l\dot{X}\dot{\theta} * cos\theta + l^2\dot{\theta}^2)$$

The total kinetic energy for the system is

$$KE_{tot} = KE_1 + KE_2$$

$$KE_{tot} = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{X}^2 + 2l\dot{X}\dot{\theta} * cos\theta + l^2\dot{\theta}^2)$$

The Potential Energy of the system is given by

$$U = mgl * cos\theta$$

The Lagrange Equation is given by:

$$L = KE_{tot} - U = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m\left(\dot{X}^2 + 2l\dot{X}\dot{\theta} * cos\theta + l^2\dot{\theta}^2\right) - mgl * cos\theta$$

The first Lagrange Equation is given by

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = u$$

Where $\frac{\partial L}{\partial \dot{x}} = M\dot{x} + m(\dot{x} + l\dot{\theta} * cos\theta)$ and $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = (M+m)\ddot{x} + ml\ddot{\theta} * cos\theta - ml\dot{\theta}^2 * sin\theta$

Since $\frac{\partial L}{\partial x} = 0$, we obtain,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = (M+m)\ddot{x} + ml\ddot{\theta} * cos\theta - ml\dot{\theta}^2 * sin\theta - ml\dot{\theta}^2 * sin\theta = u$$

We now get the equation for the second Lagrange equation,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0$$

Now we can see that $\frac{\partial L}{\partial \dot{\theta}} = m(l\dot{x} * cos\theta + l^2 * \dot{\theta}^2)$ therefore the equation become

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = m\ddot{X} * cos\theta + ml\ddot{\theta} - mgsin\theta = 0$$

$Let\ x_1 = x \quad x_2 = \dot{x} \quad x_3 = \theta \quad x_4 = \dot{\theta}$ , then the equation of the system becomes:

$$\dot{x}_1 = x$$

$$\dot{x}_2 = \frac{-ml\dot{\theta} * cos\theta * sin\theta + gm * cos\theta * sin\theta}{(M+m) - m * cos^2\theta}$$

$$\dot{x}_3 = \theta$$

$$\dot{x}_4 = \frac{-ml\dot{\theta} * cos\theta * sin\theta + gM * sin\theta + gmsin\theta - cos * u}{l(M + m - mcos^2\theta)}$$

The above equations represent the values non-linear dynamical model in state space this answers the first question of the problem statement. The system can be designed for the state space given by

$$\dot{x} = Ax + Bu$$

## 2.2. Model Linearization

This system can be linearized at its operating point **x** =0 **u** = 0. Substituting the values of M, m, l and g. Now the A Matrix for the system can be given by,

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad ; \text{B} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial x_1} \\ \cdot \\ . \\ \frac{\partial f_n}{\partial x_1} \end{bmatrix}$$

Where $f_1 = \dot{x}_1$ ; $f_2 = \dot{x}_2$ ; $f_3 = \dot{x}_3$ ; $f_4 = \dot{x}_4$ ;

Also equating **x** = 0 and u = 0 we get,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 11 & 0 \end{bmatrix}; \qquad B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

Thus, the linearized system model can be determined by the following state space model.

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 11 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} u \qquad \text{----------} \rightarrow 1$$

$$y = [1\ 0\ 0\ 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

This answer the second question asked in the problem statement.

## 2.3. Analysing the Stability of the system

According to Lyapunov Stability condition the system is said to be unstable if it has any poles on the right-hand side of the S-plane. The state space model of the linearized system is given by Eqn 1. The characteristic equation of the model system can be given by,

$$SI - A = 0$$

$$\begin{bmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & S \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 11 & 0 \end{bmatrix} = 0$$

$$\begin{bmatrix} S & -1 & 0 & 0 \\ 0 & S & 1 & 0 \\ 0 & 0 & S & -1 \\ 0 & 0 & -11 & S \end{bmatrix} = 0$$

Using MATLAB, we can obtain the poles of the state space by calculating the Eigen values of A. The poles of the system are 0.0, 0.0, 3.316, -3.316. Since there is a pole on the right-hand side of the plane (Eigen value is positive), the system is unstable.

## 2.4. Linear State feedback control

The linear state feedback control law, for a system modelled by Eqn 1, is the feedback of a linear combination of all the state variables and has the form,

$$U = -Kx$$

Thus, the closed loop system is then given by,

$$\dot{x} = (A - BK)x$$

The poles of the closed loop system are the roots of the characteristic equation,

$$\det(SI_n - A + BK) = 0$$

The problem statement suggests that we use 3 different poles sets. We have assumed the following pole set `P1 = [-5.1; -5.2; -5.3; -5.4], P2 = [-3.1; -3.2; -3.3; -3.4], P3 = [-1.1; -1.2; -1.3; -1.4]`. The Gain matrix can be calculated by using the place method in MATLAB. The State variables and the Output variable can be found using the ode45 method used in MATLAB. The ode45 will be pointing to the differential equation of the linearized state model. The results for the Linear State feedback can be referred in the result section.

## 2.5. Combined state feedback control with state estimator

The gain matrix of the estimator L can be calculated by equating the characteristic equation of the *A – IC* matrix with the required poles. Generally, the poles of the estimator should be 4 to 6 times more than the poles of the state feedback controller. For this system we are assuming the poles to be at `P = [-20 -21 -22 -23];`

The control law will now be given by $u(t) = -K\tilde{x}(t) + v(t)$.

The combined controller and estimator defines the state variables $x_1, x_2, x_3, x_4$ and estimated errors $e_1, e_2, e_3, e_4$.

The new equations describing the closed loop system are given by,

$$\begin{bmatrix} \dot{x}(t) \\ \dot{\tilde{x}}(t) - \dot{x}(t) \end{bmatrix} = \begin{bmatrix} A - BK & -BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x(t) \\ \tilde{x}(t) - x(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v(t)$$

Once the state feedback gain is obtained along with the estimator gain the state variables and the output variables are obtained by using ode45 command in MATLAB. The results for the Combined state feedback with state estimator can be referred in the result section.

## 2.6. Linear Quadratic Regulator

A system model given by,

$$\dot{x} = Ax + Bu$$

the performance index of the system is given by,

$$J = \int (x^T Q\, x + u^T R u) dt$$

Where $Q = Q^T > 0$ *and* $R = R^T > 0$

For MATLAB there is a command which calculates the solution to the Linear Quadratic Regulator using the Algebraic Riccati Equation. The project uses different Q matrix for the performance index. The cost values used are 1, 10, 100 and the R value is a constant 1. The system then defines the k gain matrix by the *lqr* command in MATLAB. This gain matrix is then used for finding the state variable and the output variable by using ode45 in MATLAB. The results for the linear quadratic regulator can be referred in the result section

# 3. Results

## 3.1. Linear state feedback control

a. The results of the Linear state feedback control with poles at `P1 = [-5.1; -5.2; -5.3; -5.4]` The initial state of the system is $x_0 = [0.1; 0; 0.1; 0]$ and the final state is given at $x_f = [0; 0; 0; 0]$

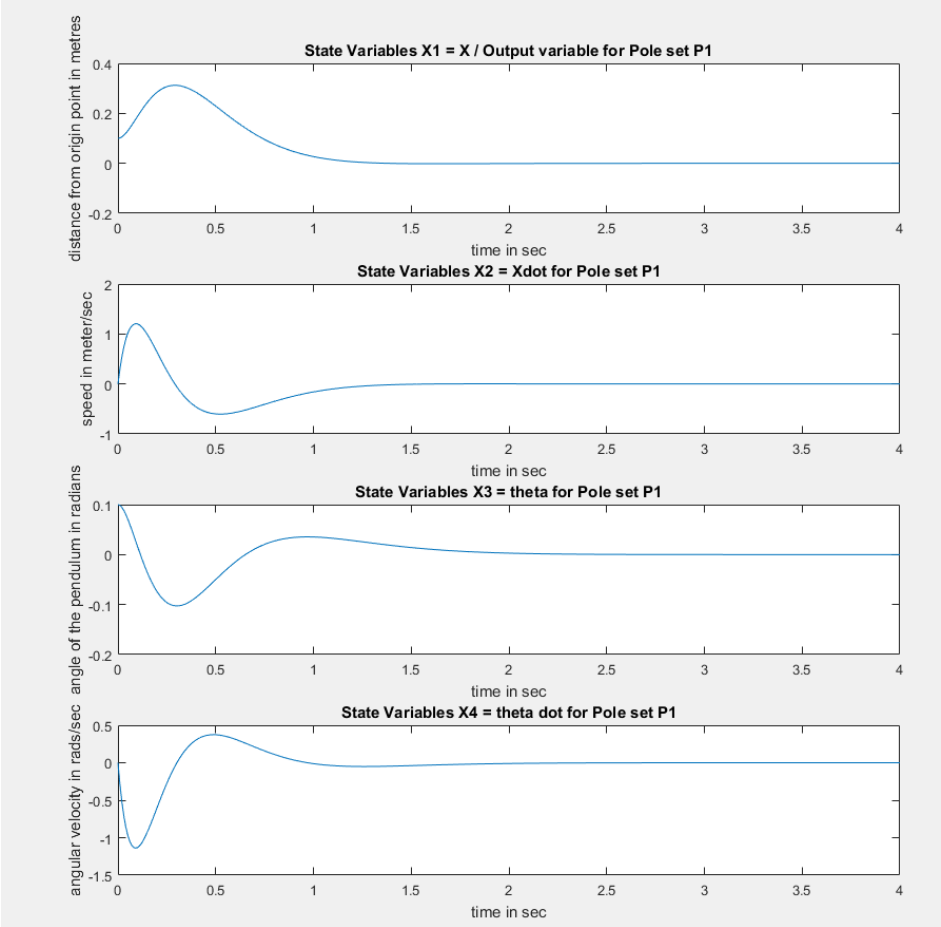


Fig. 2: State response for pole set P1

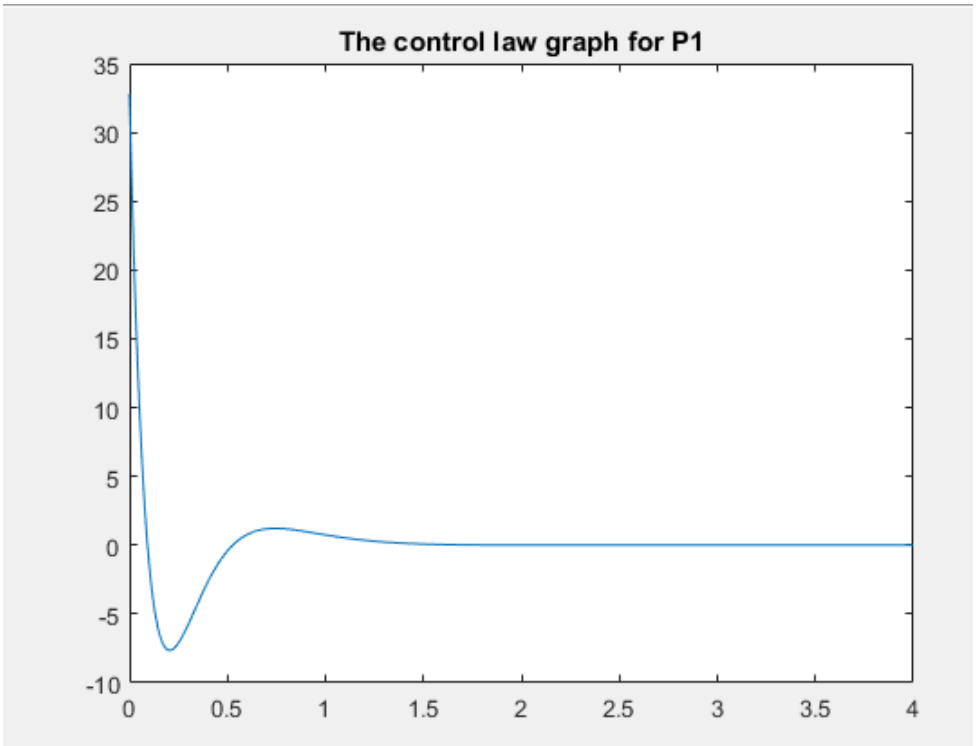


Fig .3: Control Law graph output for Pole set P1

*Observation*: The system is aggressive and settles by around 0.75 seconds. However, the peak value overshoot will be more because of aggressive pole placing.

b. The results of the Linear state feedback control with poles at `P2 = [-3.1; -3.2; -3.3; -3.4]` The initial state of the system is $x_0 = [0.1; 0; 0.1; 0]$ and the final state is given at $x_f = [0; 0; 0; 0]$

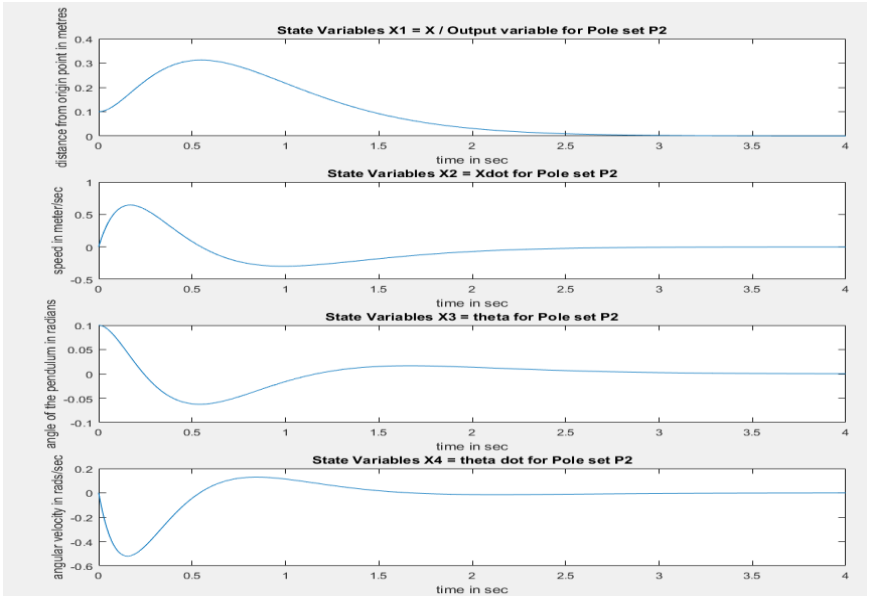


Fig. 4: State response for pole set P2

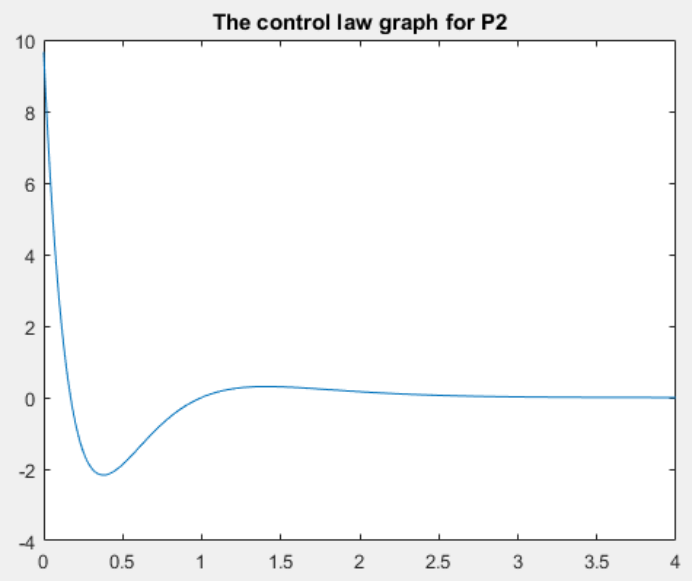


Fig .5: The control law graph for pole set 2

*Observation:* Here the system takes more time to settle down (around 2.5 seconds). However, the peak overshoot is very less as compared to the 1st pole set.

c. The results of the Linear state feedback control with poles at `P3 = [-1.1; -1.2; -1.3; -1.4]` The initial state of the system is $x_0 = [0.1; 0; 0.1; 0]$ and the final state is given at $x_f = [0; 0; 0; 0]$

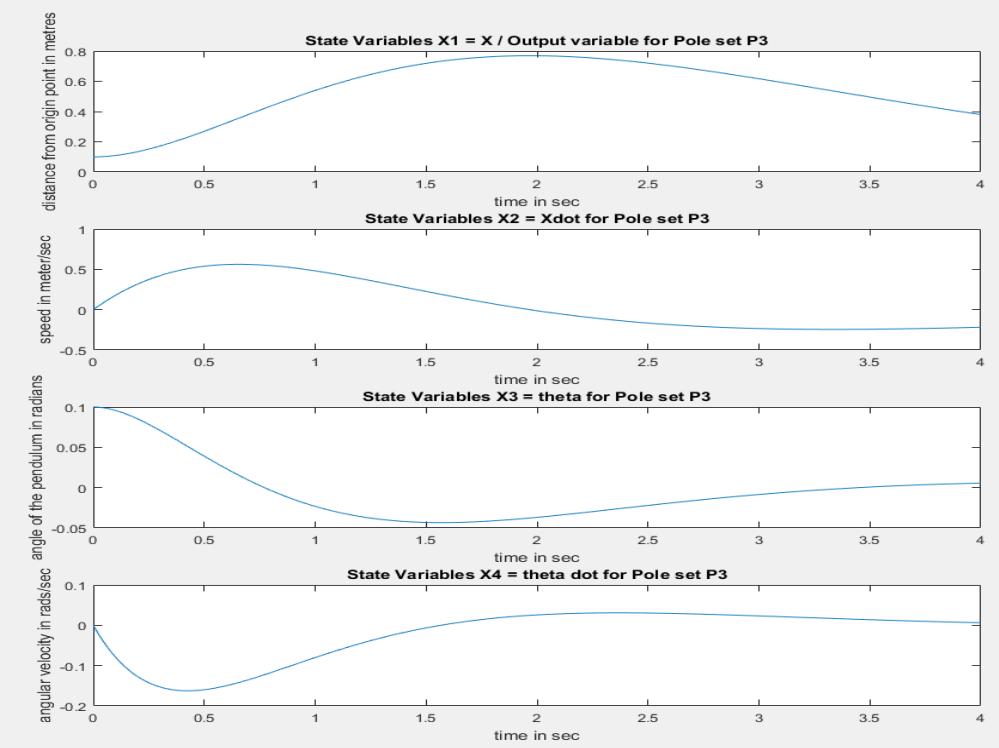


Fig. 6: State response for pole set P3

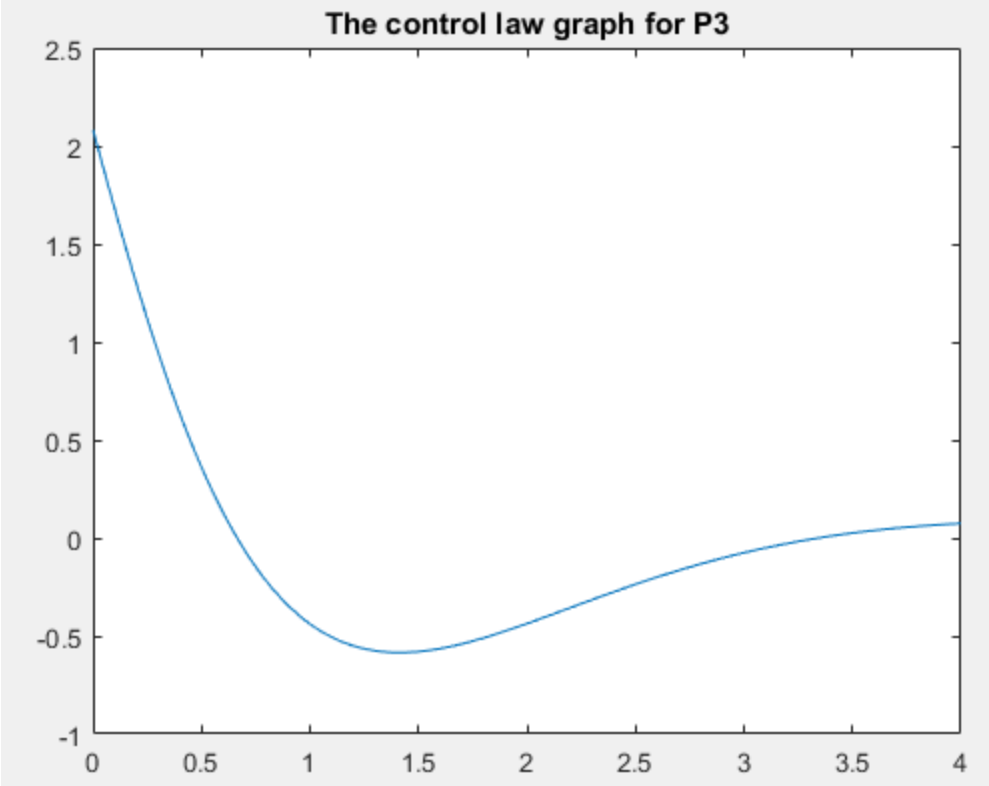


Fig .7: Control law for pole set P3

*Observation:* This set of poles is very slow because of the poles and the system did not settle down even after 4 seconds.

## 3.2. Combined state feedback control and state estimator

The output for the state variables for the combined state feedback controller and estimator is given below. Here the pole for the state feedback model is taken as `P1 = [-5.1; -5.2; -5.3; -5.4]`

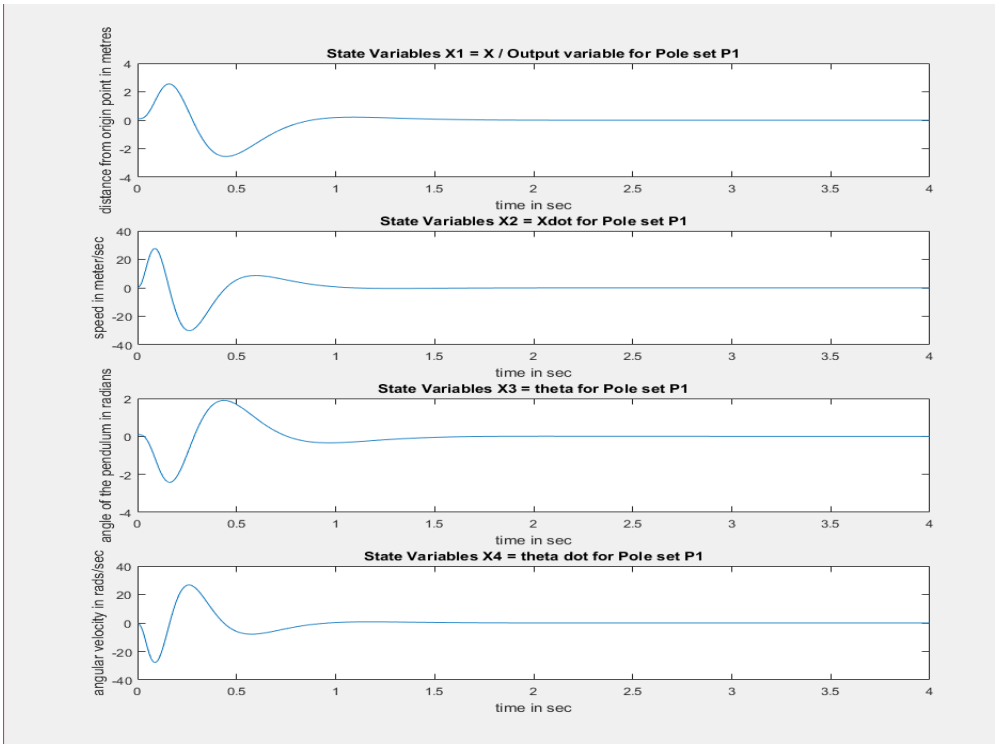


Fig. 8: State response for combined state feedback and estimator

The error of the estimated and the output is given below which later settles to 0.

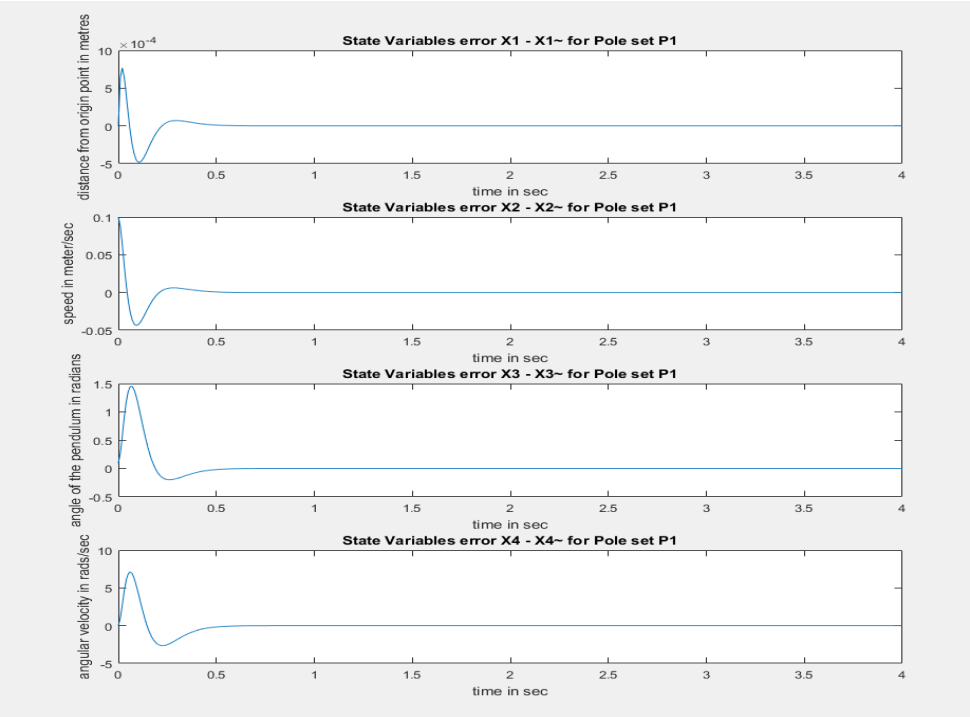


Fig. 9: Error response for the combined state feedback controller and estimator

## 3.3. Linear Quadratic Regulator

a. For LQR there were 3 Q matrices used, for a very less aggressive penalized cost value

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; R = 1$$

*Observation:* The system is less aggressive as the cost to be penalized is very low. We can see that the system is not aggressive enough for the system to be settled.

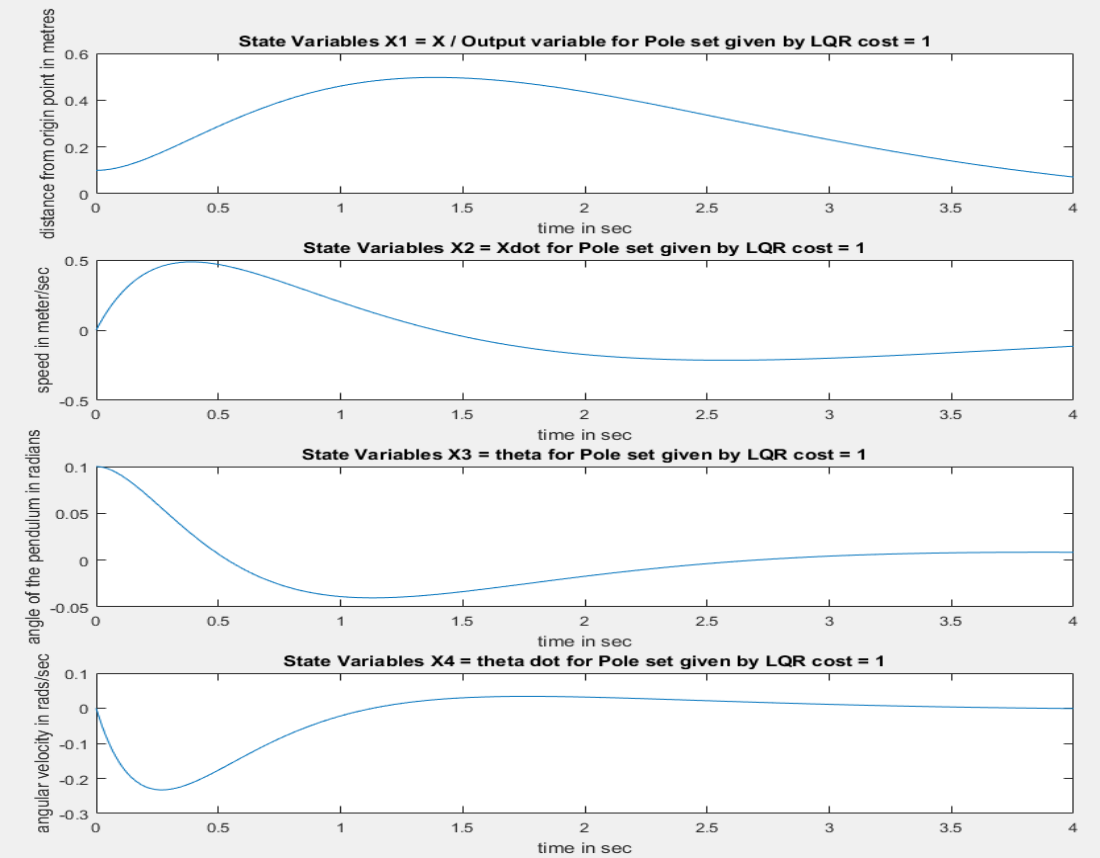


Fig. 10: LQR state response with Q1

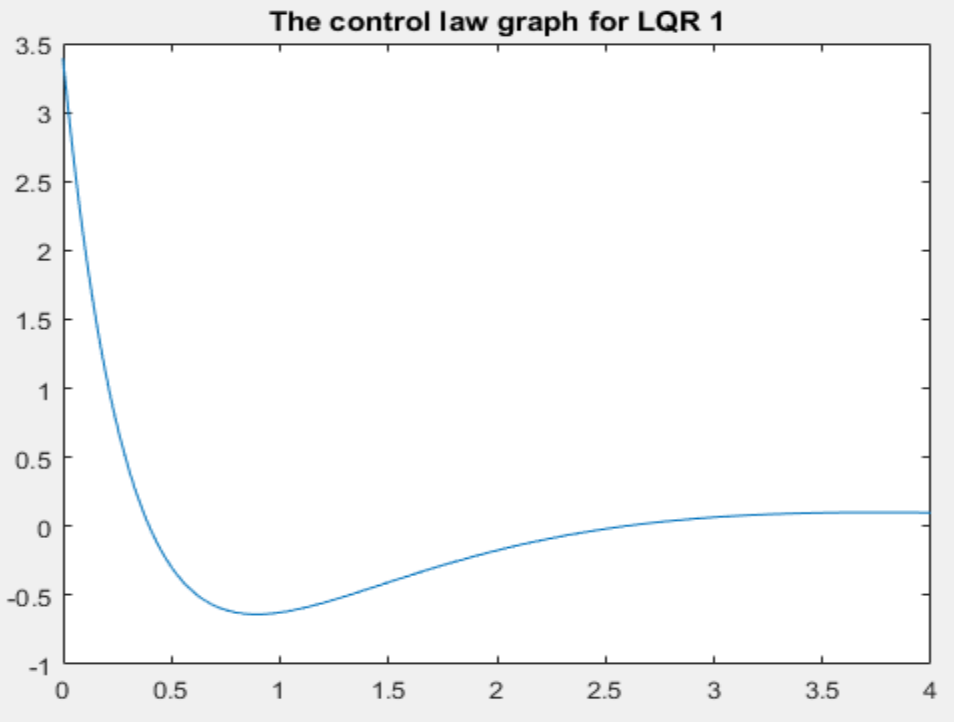


Fig .11: The control law for LQR 1

b. Now increasing the cost

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; R = 1$$

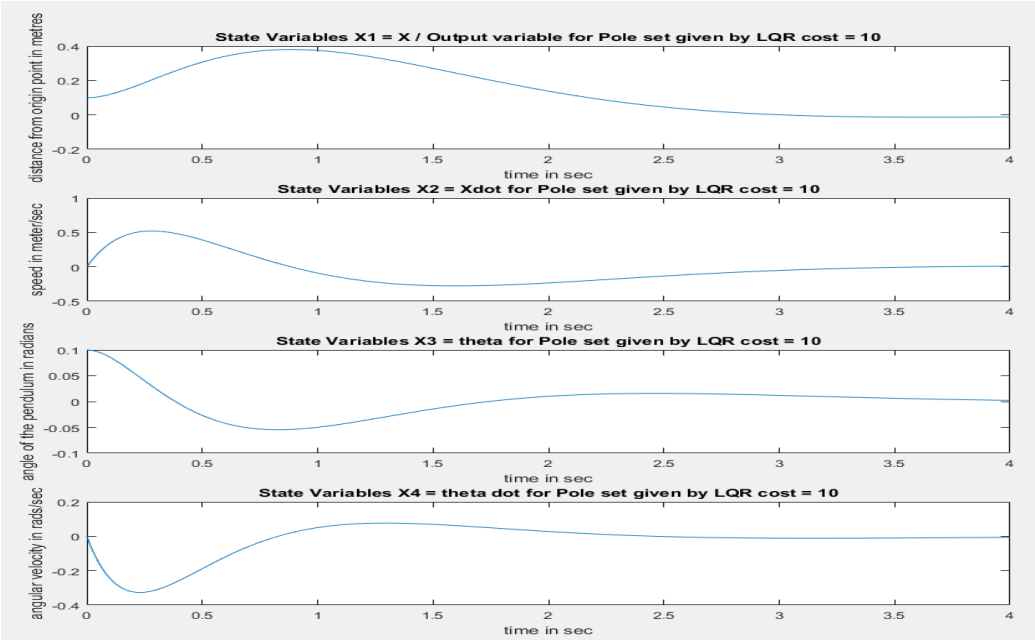


Fig. 12: LQR state response with Q2

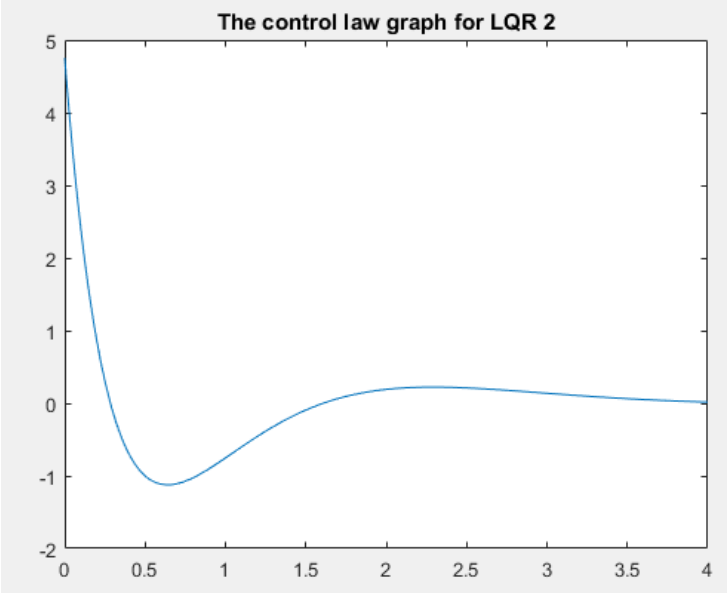


Fig .13: Control Graph response for LQR 2

With this we can see that the settling time has improved. Now further increasing the penalty cost

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; R = 1$$

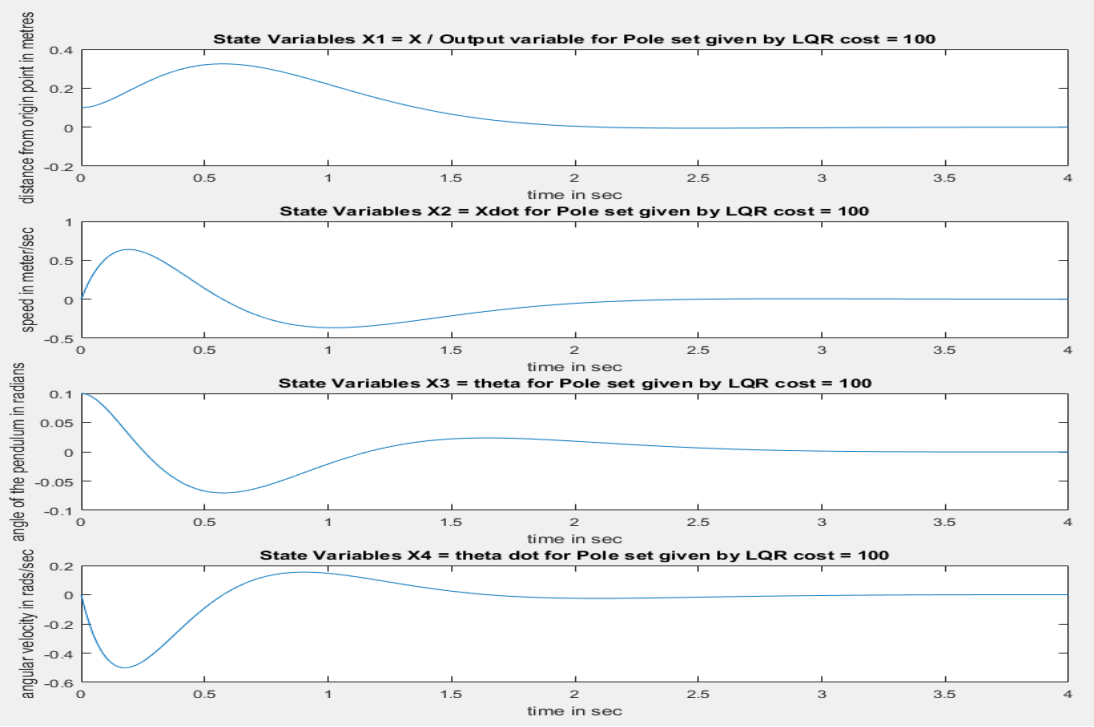


Fig. 14: LQR state response with Q3

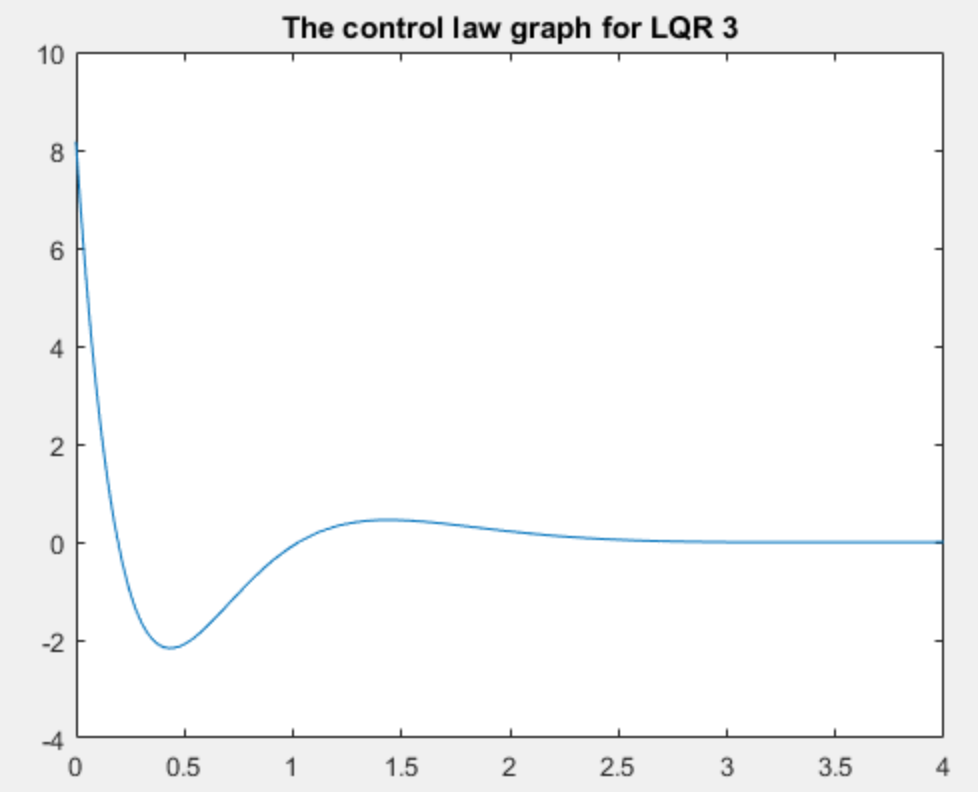


Fig .15: The control Law graph for LQR 3

## 4. Conclusion

This project helped us to analyse and understand how to model a non-linear system, find its stability at the operating point and find the stable controller and estimator gain matrices. I was able to understand the importance of cost matrix in the case of LQR and the placement of the poles for the linear state feedback matrix and how they affect the steeling time and overshoot of the system. The relation between the pole magnitude and the time taken for the system to settle was also understood and studied in the project.

# 5. References

1. Textbook: Systems and controls by Stanislaw h. Zak
2. Class lecture notes
3. https://www.youtube.com/watch?time_continue=314&v=u5Sv7YKAkt4

# 6. Appendix

The Main code ***Project.m*** file is given below,

```
clear all;
clc;
close all;

M = 1;                      % Mass of the cart in Kg (Ref: Exercise 3.18 and
Figure 3.19)
m = 0.1;                    % Mass of the inverted pendulum in Kg (Ref: 3.18
and Figure 3.19)
g = 10 ;                    % Gravitational force is assumed to be 10m/sec^2
as given in Exercise 3.18
l =1; % m

A = [ 0 1 0 0;              % The A matrix which is calculated by hand (Refer
report for more details)
    0 0 -1 0;
    0 0 0 1;
    0 0 11 0;];

B  = [0;1;0;-1];            % The B matrix which is calculated by hand (Refer
report for more details)
C = [1 0 0 0];              % The C matrix which is calculated by hand (Refer
report for more details)
D = [0];                    % The D matrix which is calculated by hand (Refer
report for more details)
tspan = 0:.01:4;

Controllability_Flag = 0;   % The flag is set and the program for feedback
continues only if the flag is set

EiganValue_Given = eig (A);
display = sprintf('The Eigan Values of the given Model are %f, %f, %f, %f',
EiganValue_Given(1), EiganValue_Given(2), EiganValue_Given(3),
EiganValue_Given(4));
disp(display);
Rnk = rank(ctrb(A,B));  % is it controllable

if(Rnk == size(A,1))
    Controllability_Flag = 1;                                          % The
program can proceed further as the system is controllable
end

%-------------------------------------------------------------------------
-------------------%
%-------------------Linear state feedback gain with state variables
available--------------------%
%-------------------------------------------------------------------------
-------------------%

if(Controllability_Flag == 1)
```

```
    P1 = [-5.1; -5.2; -5.3; -5.4];
    P2 = [-3.1; -3.2; -3.3; -3.4];
    P3 = [-1.1; -1.2; -1.3; -1.4];
    K1 = place(A,B,P1);                                % Linear state feedback
gain equation for the 1st set of poles
    K2 = place(A,B,P2);                                % Linear state feedback
gain equation for the 2nd set of poles
    K3 = place(A,B,P3);                                % Linear state feedback
gain equation for the 3rd set of poles

    x0 = [0.1; 0; 0.1; 0];                             % initial state for the
system
    xf = [2;0;0;0];                                    % Final State required
for the system

    %----------------> for the P1 poles set <---------------%

    [t,x] = ode45(@(t,x)diffEq(x,m,M,l,g,-K1*(x - xf)),tspan,x0);
    figure(1);

    display = sprintf('The Linear State feedback gain matrix [%f %f %f %f]',
K1(1), K1(2), K1(3), K1(4));
    disp(display);

    subplot(4,1,1);     plot(t,x(:,1));
    title('State Variables X1 = X / Output variable for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('distance from origin point in metres')  % y-axis label
    subplot(4,1,2);     plot(t,x(:,2));
    title('State Variables X2 = Xdot for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('speed in meter/sec')                       % y-axis label
    subplot(4,1,3);     plot(t,x(:,3));
    title('State Variables X3 = theta for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('angle of the pendulum in radians')      % y-axis label
    subplot(4,1,4);     plot(t,x(:,4));
    title('State Variables X4 = theta dot for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('angular velocity in rads/sec')          % y-axis label
    %------------------------> End <-------------------------%
    %----------------> for the P2 poles set <---------------%
    [t,x] = ode45(@(t,x)diffEq(x,m,M,l,g,-K2*(x - xf)),tspan,x0);
    figure(2);

    display = sprintf('The Linear State feedback gain matrix [%f %f %f %f]',
K1(1), K1(2), K1(3), K1(4));
    disp(display);

    subplot(4,1,1);     plot(t,x(:,1));
    title('State Variables X1 = X / Output variable for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('distance from origin point in metres')  % y-axis label
    subplot(4,1,2);     plot(t,x(:,2));
```

```
    title('State Variables X2 = Xdot for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('speed in meter/sec')                       % y-axis label
    subplot(4,1,3);     plot(t,x(:,3));
    title('State Variables X3 = theta for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('angle of the pendulum in radians')         % y-axis label
    subplot(4,1,4);     plot(t,x(:,4));
    title('State Variables X4 = theta dot for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('angular velocity in rads/sec')             % y-axis label

    %------------------------> End <------------------------%
    %----------------> for the P3 poles set <---------------%

    [t,x] = ode45(@(t,x)diffEq(x,m,M,l,g,-K3*(x - xf)),tspan,x0);
    figure(3);

    display = sprintf('The Linear State feedback gain matrix [%f %f %f %f]',
K1(1), K1(2), K1(3), K1(4));
    disp(display);

    subplot(4,1,1);     plot(t,x(:,1));
    title('State Variables X1 = X / Output variable for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('distance from origin point in metres')     % y-axis label
    subplot(4,1,2);     plot(t,x(:,2));
    title('State Variables X2 = Xdot for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('speed in meter/sec')                       % y-axis label
    subplot(4,1,3);     plot(t,x(:,3));
    title('State Variables X3 = theta for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('angle of the pendulum in radians')         % y-axis label
    subplot(4,1,4);     plot(t,x(:,4));
    title('State Variables X4 = theta dot for Pole set P1');
    xlabel('time in sec')                              % x-axis label
    ylabel('angular velocity in rads/sec')             % y-axis label

    %------------------------> End <------------------------%
end

% %-------------------------------------------------------------------------
---------------------%
% %------------------------------------------ End ------------------------
---------------------%
% %-------------------------------------------------------------------------
---------------------%
% %
% %-------------------------------------------------------------------------
---------------------%
% %------------------------Combined state estimator and controller-----------
---------------------%
% %-------------------------------------------------------------------------
---------------------%
if(Controllability_Flag == 1)
```

```
    P = [-20 -21 -22 -23];              % The estimator poles should be 4 - 6
times more than the poles of the linear state feedback
    L = place(A',C',P)';                % To get the value of L
    A_EstNew = [A-B*K1 -B*K1;zeros(size(A)) A-L*C];
    B_Est = [B;zeros(size(B))];
    C_Est = [C zeros(size(C))];
    D = 0;
    x0 = [0.1;0;0.1; 0;0 ;0.1 ;0.1; 0];

    [t,x] = ode45(@(t,x)diffEq_combined(x,A_EstNew),tspan,x0);


    figure(4);                                  % Represents the state variables
    subplot(4,1,1);    plot(t,x(:,1));
    title('State Variables X1 = X / Output variable for Pole set P1');
    xlabel('time in sec')                               % x-axis label
    ylabel('distance from origin point in metres')  % y-axis label

    subplot(4,1,2);    plot(t,x(:,2));
    title('State Variables X2 = Xdot for Pole set P1');
    xlabel('time in sec')                               % x-axis label
    ylabel('speed in meter/sec')                        % y-axis label

    subplot(4,1,3);    plot(t,x(:,3));
    title('State Variables X3 = theta for Pole set P1');
    xlabel('time in sec')                               % x-axis label
    ylabel('angle of the pendulum in radians')      % y-axis label

    subplot(4,1,4);    plot(t,x(:,4));
    title('State Variables X4 = theta dot for Pole set P1');
    xlabel('time in sec')                               % x-axis label
    ylabel('angular velocity in rads/sec')          % y-axis label


    figure(5);                                  % Represents the estimator error
    subplot(4,1,1);    plot(t,x(:,5));
    title('State Variables error X1 - X1~ for Pole set P1');
    xlabel('time in sec')                               % x-axis label
    ylabel('distance from origin point in metres')  % y-axis label

    subplot(4,1,2);    plot(t,x(:,6));
    title('State Variables error X2 - X2~ for Pole set P1');
    xlabel('time in sec')                               % x-axis label
    ylabel('speed in meter/sec')                        % y-axis label

    subplot(4,1,3);    plot(t,x(:,7));
    title('State Variables error X3 - X3~ for Pole set P1');
    xlabel('time in sec')                               % x-axis label
    ylabel('angle of the pendulum in radians')      % y-axis label

    subplot(4,1,4);    plot(t,x(:,8));
    title('State Variables error X4 - X4~ for Pole set P1');
    xlabel('time in sec')                               % x-axis label
```

```
    ylabel('angular velocity in rads/sec')          % y-axis label

end
% %-------------------------------------------------------------------------
---------------------%
% %------------------------------------------- End -----------------------
---------------------%
% %-------------------------------------------------------------------------
---------------------%
%
% %-------------------------------------------------------------------------
---------------------%
% %------------------------------------------- LQR -----------------------
---------------------%
% %-------------------------------------------------------------------------
---------------------%

%--------------------------------------->LQR Cost = 1<----------------------
--------------%

cost1 = 1;
cost2 = 1;
if(Controllability_Flag == 1)

    Q = [cost1 0 0 0;              % The ARE equation requires Q R in the Q
        0 0 0 0;                % which is taken distance and angle has
        0 0 cost2 0;               % been penalized
        0 0 0 0;];
    R = 1;
    K_lqr = lqr(A,B,Q,R);       % using Matlab inbuilt function to find the
state feedback gain using LQR
    A_lqr = A-B*K_lqr;
    B_lqr = B;
    C_lqr = C;
    D_lqr = D;



    x0 = [0.1; 0; 0.1; 0];      % Initial values at start
    [t,x] = ode45(@(t,x)diffEq(x,m,M,l,g,-K_lqr*(x-[4; 0; 0; 0])),tspan,x0);
    figure(6)

    display = sprintf('The Linear State feedback gain matrix for LQR [%f %f
%f %f]', K_lqr(1), K_lqr(2), K_lqr(3), K_lqr(4));
    disp(display);

    subplot(4,1,1);     plot(t,x(:,1));
    title('State Variables X1 = X / Output variable for Pole set given by LQR
cost = 1');
    xlabel('time in sec')                           % x-axis label
    ylabel('distance from origin point in metres')  % y-axis label

    subplot(4,1,2);     plot(t,x(:,2));
    title('State Variables X2 = Xdot for Pole set given by LQR cost = 1');
    xlabel('time in sec')                           % x-axis label
    ylabel('speed in meter/sec')                    % y-axis label
```

```
    subplot(4,1,3);     plot(t,x(:,3));
    title('State Variables X3 = theta for Pole set given by LQR cost = 1');
    xlabel('time in sec')                           % x-axis label
    ylabel('angle of the pendulum in radians')      % y-axis label

    subplot(4,1,4);     plot(t,x(:,4));
    title('State Variables X4 = theta dot for Pole set given by LQR cost =
1');
    xlabel('time in sec')                           % x-axis label
    ylabel('angular velocity in rads/sec')          % y-axis label

    %-------------------------------------> End <--------------------------
----------%
    %------------------------------------->LQR Cost = 10<------------------
-------------------%
    cost1 = 10;
    cost2 = 10;

    Q = [cost1 0 0 0;              % The ARE equation requires Q R in the Q
        0 0 0 0;                % which is taken distance and angle has
        0 0 cost2 0;               % been penalized
        0 0 0 0;];
    R = 1;
    K_lqr = lqr(A,B,Q,R);       % using Matlab inbuilt function to find the
state feedback gain using LQR
    A_lqr = A-B*K_lqr;
    B_lqr = B;
    C_lqr = C;
    D_lqr = D;


    x0 = [0.1; 0; 0.1; 0];      % Initial values at start
    [t,x] = ode45(@(t,x)diffEq(x,m,M,l,g,-K_lqr*(x-[4; 0; 0; 0])),tspan,x0);
    figure(7)

    display = sprintf('The Linear State feedback gain matrix for LQR [%f %f
%f %f]', K_lqr(1), K_lqr(2), K_lqr(3), K_lqr(4));
    disp(display);

    subplot(4,1,1);     plot(t,x(:,1));
    title('State Variables X1 = X / Output variable for Pole set given by LQR
cost = 10');
    xlabel('time in sec')                           % x-axis label
    ylabel('distance from origin point in metres')  % y-axis label

    subplot(4,1,2);     plot(t,x(:,2));
    title('State Variables X2 = Xdot for Pole set given by LQR cost = 10');
    xlabel('time in sec')                           % x-axis label
    ylabel('speed in meter/sec')                    % y-axis label

    subplot(4,1,3);     plot(t,x(:,3));
    title('State Variables X3 = theta for Pole set given by LQR cost = 10');
    xlabel('time in sec')                           % x-axis label
    ylabel('angle of the pendulum in radians')      % y-axis label
```

```
    subplot(4,1,4);     plot(t,x(:,4));
    title('State Variables X4 = theta dot for Pole set given by LQR cost =
10');
    xlabel('time in sec')                           % x-axis label
    ylabel('angular velocity in rads/sec')          % y-axis label

    %--------------------------------------> End <---------------------------
----------%
    %--------------------------------------->LQR Cost = 100<-----------------
--------------------%

    cost1 = 100;
    cost2 = 100;

    Q = [cost1 0 0 0;              % The ARE equation requires Q R in the Q
        0 0 0 0;                % which is taken distance and angle has
        0 0 cost2 0;               % been penalized
        0 0 0 0;];
    R = 1;
    K_lqr = lqr(A,B,Q,R);       % using Matlab inbuilt function to find the
state feedback gain using LQR
    A_lqr = A-B*K_lqr;
    B_lqr = B;
    C_lqr = C;
    D_lqr = D;


    x0 = [0.1; 0; 0.1; 0];      % Initial values at start
    [t,x] = ode45(@(t,x)diffEq(x,m,M,l,g,-K_lqr*(x-[4; 0; 0; 0])),tspan,x0);
    figure(8)

    display = sprintf('The Linear State feedback gain matrix for LQR [%f %f
%f %f]', K_lqr(1), K_lqr(2), K_lqr(3), K_lqr(4));
    disp(display);

    subplot(4,1,1);     plot(t,x(:,1));
    title('State Variables X1 = X / Output variable for Pole set given by LQR
cost = 100');
    xlabel('time in sec')                           % x-axis label
    ylabel('distance from origin point in metres')  % y-axis label

    subplot(4,1,2);     plot(t,x(:,2));
    title('State Variables X2 = Xdot for Pole set given by LQR cost = 100');
    xlabel('time in sec')                           % x-axis label
    ylabel('speed in meter/sec')                    % y-axis label

    subplot(4,1,3);     plot(t,x(:,3));
    title('State Variables X3 = theta for Pole set given by LQR cost = 100');
    xlabel('time in sec')                           % x-axis label
    ylabel('angle of the pendulum in radians')      % y-axis label

    subplot(4,1,4);     plot(t,x(:,4));
    title('State Variables X4 = theta dot for Pole set given by LQR cost =
100');
```

```
    xlabel('time in sec')                              % x-axis label
    ylabel('angular velocity in rads/sec')             % y-axis label

    %--------------------------------------> End <--------------------------
-----------%
end
% %-------------------------------------------------------------------------
--------------------%
% %------------------------------------------ End ----------------------------
--------------------%
% %-------------------------------------------------------------------------
--------------------%
```

The function ***diffEq.m*** is given below:

```
function dx = diffEq(x,m,M,l,g,u)

%Function definition:
%
% The function is to represent the differnetial equation of the linerized
% system model. The linearization is done by hand refer the report.
% The state variables of the system are given below:
% x1 = x    x2 = xdot      x3 = theta   x4 = theta dot
% The output variable is y = x1

Sx = sin(x(3));
Cx = cos(x(3));

D = (M+m)- m*Cx^2;

dx(1,1) = x(2);                                          % x1 dot
dx(2,1) = (1/D)*(u + m*l*x(4)^2*Sx - m*g*Sx*Cx);         % x2 dot
dx(3,1) = x(4);                                          % x3 dot
dx(4,1) = (1/l*D)*(m*l*x(4)*Sx*Cx + (m+M)*g*Sx - u*Cx); % x4 dot
```

The function ***diffEq_combined.m*** is given below:

```
function dx = diffEq_combined( x,A_EstNew,tspan,x0)
%Function definition:
%
% The function is to represent the differnetial equation combined state
% controller and estimator.
% The state variables of the system are given below:
% x1 = x    x2 = xdot      x3 = theta   x4 = theta dot
% The output variable is y = x1

dx = A_EstNew*x;

end
```