Final Project Report

ENPM667 FALL 2020

John DiNofrio & Divyam Garg

December 20th, 2020

Table of Contents

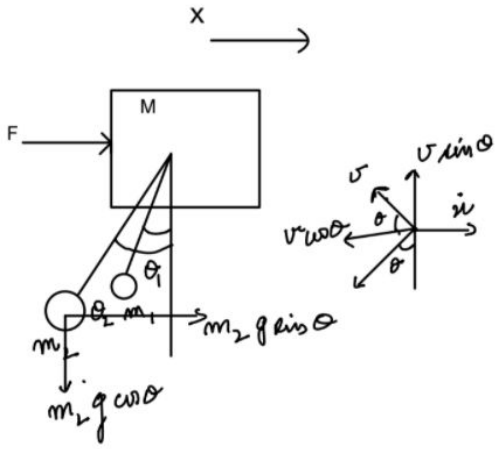
Mobile Robotic Arm Model & Simulation

Introduction

This experiment aims to create a system capable of controlling a double pendulum cart. The equations of motions will be solved by using the Euler-Lagrange equation. With these equations, the state space can be computed and furthermore linearized to help simulate and understand the system better. An LQR will be implemented and the system checked for observability. All of these results will be simulated to show responsiveness of the system.

A. Equations of Motions & State-Space

The first step in computing the equations of motion (EOM) is finding the Kinetic (T) and Potential Energy (V) of the system. The Lagrangian equation can be solved by subtracting the potential energy from the kinetic.



$$L \equiv \mathrm{T} - \mathrm{V} \qquad \text{Eq 1}$$

$$L = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2 \qquad \text{Eq 2}$$

The algebraic process starts by finding the velocity of both pendulum masses.

$$v_1{}^2 = \left(\dot{x} - l_1\dot{\theta}_1 \cos\theta_1\right)^2 + \left(l_1\dot{\theta}_1 \sin\theta_1\right)^2 \qquad \text{Eq 3}$$

$$v_2{}^2 = \left(\dot{x} - l_2\dot{\theta}_2 \cos\theta_2\right)^2 + \left(l_2\dot{\theta}_2 \sin\theta_2\right)^2 \qquad \text{Eq 4}$$

The kinetic energy equation comes out to be

$$T = \frac{1}{2}\left((M + m_1 + m_2)\dot{x}^2 + m_1 {l_1}^2 \dot{\theta_1}^2 + m_2 {l_2}^2 \dot{\theta_2}^2\right) - \dot{x}(m_1 l_1 \dot{\theta_1} \cos\theta_1 + m_2 l_2 \dot{\theta_2} \cos\theta_2)$$

Eq 5

And the potential energy, which uses a reference height of the cart is

$$V = -m_1 g\, l_1 \cos\theta_1 - m_2 g\, l_2 \cos\theta_2$$

Eq 6

Plugging T and V into the Lagrangian equation give

$$L = \frac{1}{2}(M + m_1 + m_2)\dot{x}^2 + \frac{1}{2}(m_1 {l_1}^2 \dot{\theta_1}^2 + m_2 {l_2}^2 \dot{\theta_2}^2) - \dot{x}(m_1 l_1 \dot{\theta_1} \cos\theta_1 + m_2 l_2 \dot{\theta_2} \cos\theta_2) - m_1 g\, l_1 \cos\theta_1 - m_2 g\, l_2 \cos\theta_2$$

Eq 7

Once L is computed, now the important components of the system can be found. To find each element, the following equation is used.

$$\frac{d}{dt}\left(\frac{\partial \mathrm{L}}{\partial \dot{x}}\right) \equiv \frac{\partial \mathrm{L}}{\partial x}$$

Eq 7

The types of motion - x-position, theta1, theta2 - each have individual equations.

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

Eq 8

$$0 = \frac{d}{dt}\left(\frac{\partial \mathrm{L}}{\partial \dot{\theta}_1}\right) - \frac{\partial \mathrm{L}}{\partial \theta_1}$$

Eq 9

$$0 = \frac{d}{dt}\left(\frac{\partial \mathrm{L}}{\partial \dot{\theta}_2}\right) - \frac{\partial \mathrm{L}}{\partial \theta_2}$$

Eq 10

These equations are extremely long and complicated. The first term in each equation requires the partial derivative of L in terms of the first derivative of what is trying to be solved

(the double derivative). Then the time derivative is taken from that. The second term just requires the partial derivative of L in terms of the position or angle.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = (M + m_1 + m_2)\ddot{x} - m_1 l_1 \ddot{\theta}_1 \cos\theta_1 - m_2 l_2 \ddot{\theta_2} \cos\theta_2 + m_1 l_1 \dot{\theta_1}^2 \sin\theta_1 + m_2 l_2 \dot{\theta_2}^2 \sin\theta_2$$

Eq 11

$$\frac{\partial L}{\partial x} = 0$$

Eq 12

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_1}}\right) = m_1 l_1{}^2 \ddot{\theta_1} - m_1 l_1 \dot{\theta_1} \ddot{x} \cos\theta_1 + m_1 l_1 \dot{\theta_1} \dot{x} \sin\theta_1$$

Eq 13

$$\frac{\partial L}{\partial \theta_1} = m_1 l_1 \dot{\theta_1} \dot{x} \sin\theta_1 - m_1 g\, l_1 \sin\theta_1$$

Eq 14

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_2}}\right) = m_2 l_2{}^2 \ddot{\theta_2} - m_2 l_2 \dot{\theta_2} \ddot{x} \cos\theta_2 + m_2 l_2 \dot{\theta_2} \dot{x} \sin\theta_2$$

Eq 15

By plugging equations (11-15) into equations (8-10), the linear and angular acceleration can be found using algebra. This gives the final equations of motion.

$$\ddot{X} = \frac{F - m1 * g * \sin(theta1) * \cos(\theta1) - m2 * g * \sin(\theta2) * \cos(\theta2) - m1 * l1 * (\dot{\theta1}^2) * \sin(\theta1) - m2 * l2 * (\dot{\theta2}^2) * \sin(\theta2)}{M + m1 + m2 - m1 * \cos^2(\theta1) - m2 * cos^2(\theta2)}$$

$$\ddot{X} = Z$$

Eq 16

$$\ddot{theta2} = \frac{Z * \cos(\theta2) - g * \sin(\theta2)}{l1}$$

Eq 17

$$\ddot{theta2} = \frac{Z * \cos(\theta2) - g * \sin(\theta2)}{l2}$$

Eq 18

From these equations of motions, the state space model can be derived. Equation (19) shows the formula for the model and the variables for X. Equation (20) shows the full state space equation.

$$\dot{X} = AX + Bu \qquad where\ X = \begin{bmatrix} x \\ \dot{x} \\ \theta_1 \\ \dot{\theta_1} \\ \theta_2 \\ \dot{\theta_2} \end{bmatrix}$$

Eq 19

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta_1} \\ \ddot{\theta_1} \\ \dot{\theta_2} \\ \ddot{\theta_2} \end{bmatrix} = \begin{bmatrix} x \\ \dfrac{F + m_1 l_1 \ddot{\theta_1} \cos\theta_1 - m_1 l_1 \dot{\theta_1}^2 \sin\theta_1 + m_2 l_2 \ddot{\theta_2} \cos\theta_2 - m_2 l_2 \dot{\theta_2}^2 \sin\theta_2}{\bar{M}} \\ \dot{\theta_1} \\ \dfrac{\ddot{x}\cos\theta_1 - g\sin\theta_1}{l_1} \\ \dot{\theta_2} \\ \dfrac{\ddot{x}\cos\theta_2 - g\sin\theta_2}{l_2} \end{bmatrix}$$

Eq 20

B. Linearization

Linearization can be found by using the Jacobian matrix for both matrix A and B in the state space equation. The Jacobian matrix will linearize the equations of motion about the given equilibrium point. In this system the equilibrium points are x = 0, theta1,2 = 0. Equation (21) will solve for A and equation (22) will solve for B in the state space model. The MATLAB code is shown below.

$$Jacobian_A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} & \frac{\partial f_1}{\partial x_5} & \frac{\partial f_1}{\partial x_6} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} & \frac{\partial f_2}{\partial x_5} & \frac{\partial f_2}{\partial x_6} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} & \frac{\partial f_3}{\partial x_5} & \frac{\partial f_3}{\partial x_6} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} & \frac{\partial f_4}{\partial x_5} & \frac{\partial f_4}{\partial x_6} \\ \frac{\partial f_5}{\partial x_1} & \frac{\partial f_5}{\partial x_2} & \frac{\partial f_5}{\partial x_3} & \frac{\partial f_5}{\partial x_4} & \frac{\partial f_5}{\partial x_5} & \frac{\partial f_5}{\partial x_6} \\ \frac{\partial f_6}{\partial x_1} & \frac{\partial f_6}{\partial x_2} & \frac{\partial f_6}{\partial x_3} & \frac{\partial f_6}{\partial x_4} & \frac{\partial f_6}{\partial x_5} & \frac{\partial f_6}{\partial x_6} \end{bmatrix}$$

$$Jacobian_A = A$$

Eq 21

$$Jacobian_B = \begin{bmatrix} \frac{\partial f_1}{\partial F} \\ \frac{\partial f_2}{\partial F} \\ \frac{\partial f_3}{\partial F} \\ \frac{\partial f_4}{\partial F} \\ \frac{\partial f_5}{\partial F} \\ \frac{\partial f_6}{\partial F} \end{bmatrix}$$

$$Jacobian_B = B$$

Eq 22

```
f1= xd;
f2= xdd;
f3= o1d;
f4= o1dd;
f5= o2d;
f6= o2dd;

f = [f1 f2 f3 f4 f5 f6];
q = [x xd o1 o1d o2 o2d];


A = cell(6); %cell(6,6);
x= 0;
o1= 0;
o2= 0;
o1d = 0;
o2d = 0;
xd = 0;
```

```
B = cell(1, 6);
for i=1:6
    f_ = f(i);
    B{i} = diff(f_, F);
    for j=1:6
        q_ = q(j);
        diff_ans = diff(f_, q_);

        subs_ans = subs(diff_ans);
        A{i,j} = subs_ans;
    end
end

M = 1000;
m1 = 100;
m2 = 100;
l1 = 20;
l2 = 10;
g = 10;

% Jacobian - A and B
A_subs = zeros(6);
B_subs = zeros(1,6);
for i=1:6
    B_subs(i) = subs(B(i));
    for j=1:6
        A_subs(i,j) = subs(A{i,j});
    end
end
```

After performing the partial derivatives to find the Jacobian, the new linearized state space equation looks like equation (23 &24).

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{g*m1}{M} & 0 & -\frac{g*m2}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{g\left(\frac{m1}{M}+1\right)}{l1} & 0 & -\frac{m2*g}{l1*M} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{m1*g}{l2*M} & 0 & -\frac{g(\frac{m2}{M}+1)}{l2} & 0 \end{bmatrix}$$

Eq 23

$$B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{M * l1} \\ 0 \\ \frac{1}{M * l2} \end{bmatrix}$$

Eq 24

Now at the equilibrium positions given the A and B matrices are,

```
A =
         0    1.0000         0         0         0         0
         0         0   -1.0000         0   -1.0000         0
         0         0         0    1.0000         0         0
         0         0   -0.5500         0   -0.0500         0
         0         0         0         0         0    1.0000
         0         0   -0.1000         0   -1.1000         0

B =
   1.0e-03 *

         0    1.0000         0    0.0500         0    0.1000
```

C. Controllable Conditions

The controllability of a state space system is given by :

$$\boldsymbol{C = [B\ AB\ A^2B\ \dots\ A^{n-1}B]}$$

Eq 25

If the rank(C) = n, 'n' is the number of state variables. In this system there are 6 state variables so the rank should be equal to 6 for it to be controllable, i.e it can reach any state within the system. In the system of inverted pendulum just one input is given as F hence the Controllability matrix would be a square matrix. For the rank to be equal to n determinant of C should not be equal to 0. Hence we can derive the condition for the system to be controllable.

$$\frac{\boldsymbol{g^6 * l1^2 - 2 * g^6 * l1 * l2 + g^6 * l2^2}}{\boldsymbol{M^6 * l1^6 * l2^6}}$$

Eq 26

$$\boldsymbol{\therefore l_1 \neq l_2}$$

Eq 27

After reducing the above equation we infer that l1 should not be equal to l2, which intuitively tells that if the lengths are equal the blobs would collide with each other and would not reach equilibrium position.

D. LQR Controller and Simulation

The Linear Quadratic Regulator (LQR) controller is the most optimized method to create a high performance closed-feedback loop. It reduces the work required from the controller while still providing the great performance. The Lyapunov's indirect method assures that the linearized system is stable, and the poles in the left half of the plane verify this.

The poles of the system are negative, so the system is verifiably stable.

```
poles_c =

  -0.0198 + 1.0533i
  -0.0198 - 1.0533i
  -0.0108 + 0.7356i
  -0.0108 - 0.7356i
  -0.0363 + 0.0363i
  -0.0363 - 0.0363i
```

Here we can see that all the poles are negative but they have some imaginary values as well. From this we can infer that state variables might oscillate for some time before reaching the stability. These values further can be altered by tweaking the values of Q and R.

This can also be seen in the simulation results since the system settles down. The simulation is performed with a Simulink LQR controller model shown below.

The system takes the current position and velocity values and force, uses the state space equation to find the new output values, and then produces the position and angles of the system. The LQR controller uses the gain found using the following Q and R values.

$$Q = \begin{bmatrix} 10 \\ 1 \\ 150000 \\ 1 \\ 150000 \\ 1 \end{bmatrix}$$

R = 0.01

The gain value comes out to be [3.1623 87.3782 25.2792 444.5290 45.0557 240.5400]. The LQR does not worry about input cost but instead focuses on stabilizing as fast as possible. The linear LQR code in MATLAB is shown below.

```
Q(1,1) = 10;
Q(2,2) = 1;
Q(3,3) = 150000;
Q(4,4) = 1;
Q(5,5) = 150000;
Q(6,6) = 1;

R = 0.01;

poles = eigs(A_subs);
% getting value of gain by lqr
K = lqr(A_subs,B_subs',Q,R)
% setting initial position
X0 = [5 0 pi/6 0 pi/12 0]
D = [0];
% implementing lqr
Ac = (A_subs - B_subs'*K);
poles_c = eigs(Ac);
C1 = [1 0 0 0 0 0; 0 0 1 0 0 0;0 0 0 0 1 0];
lqr_sys = ss(Ac,B_subs',C1, D);
time= 0:0.1:100
[y,t,x]=initial(lqr_sys,X0,time);
figure,
```

Figure 1 displays stability of crane position with respect to time, similarly figure 2 displays the pendulum angle stability with respect to time.

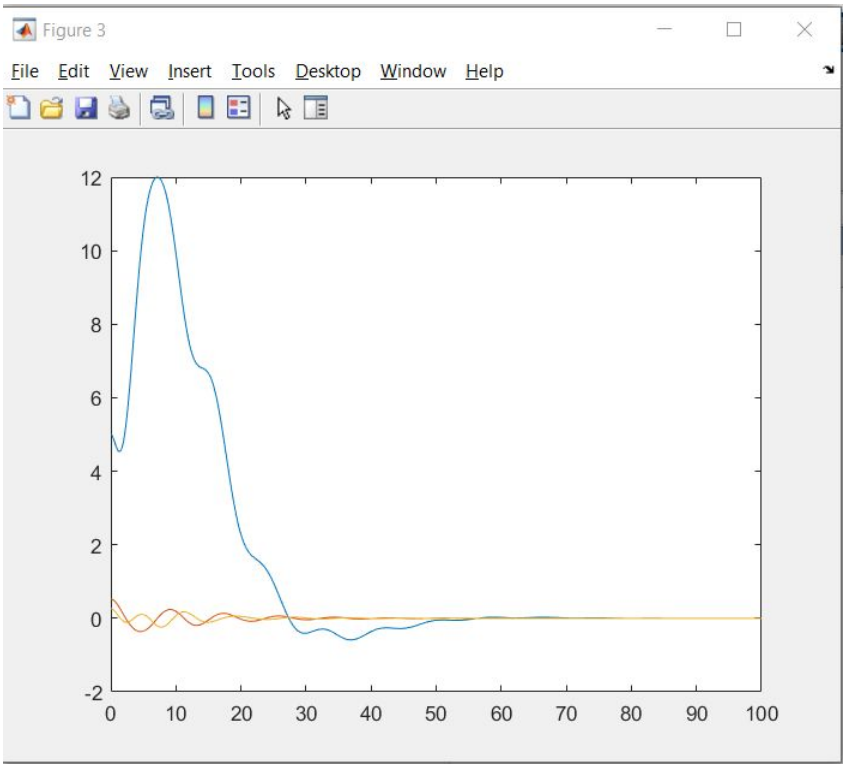


Figure 3
File Edit View Insert Tools Desktop Window Help
12
10
8
6
4
2
0
-2
0 10 20 30 40 50 60 70 80 90 100

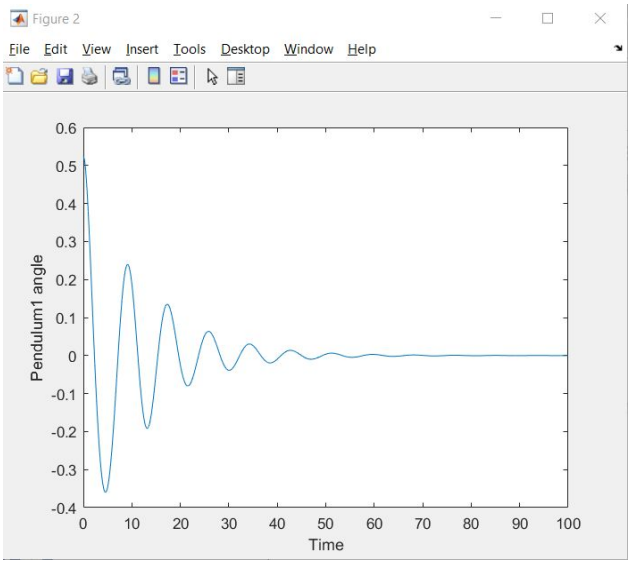


Figure 2
File Edit View Insert Tools Desktop Window Help
Pendulum1 angle
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0 10 20 30 40 50 60 70 80 90 100
Time

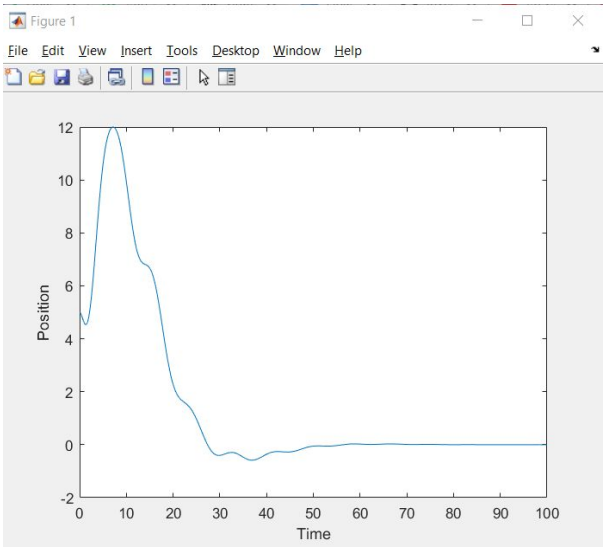
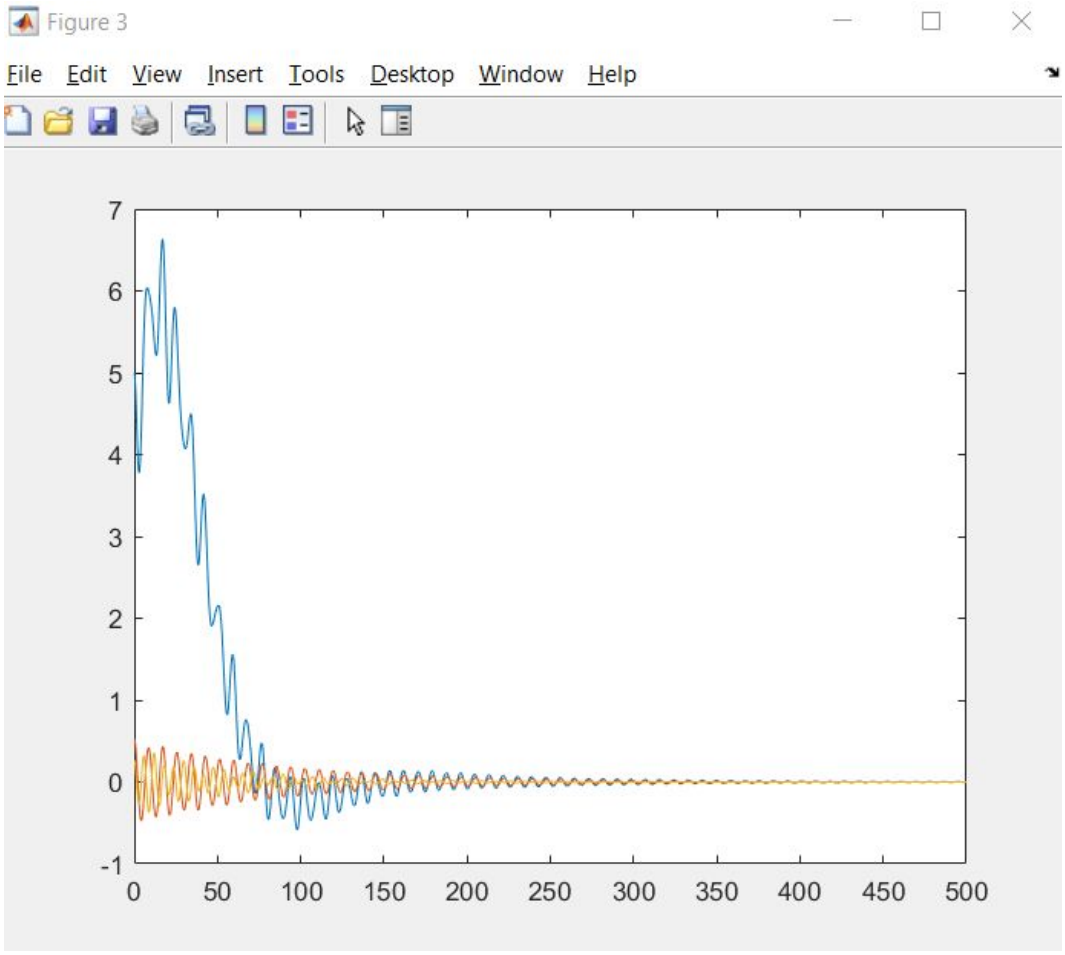
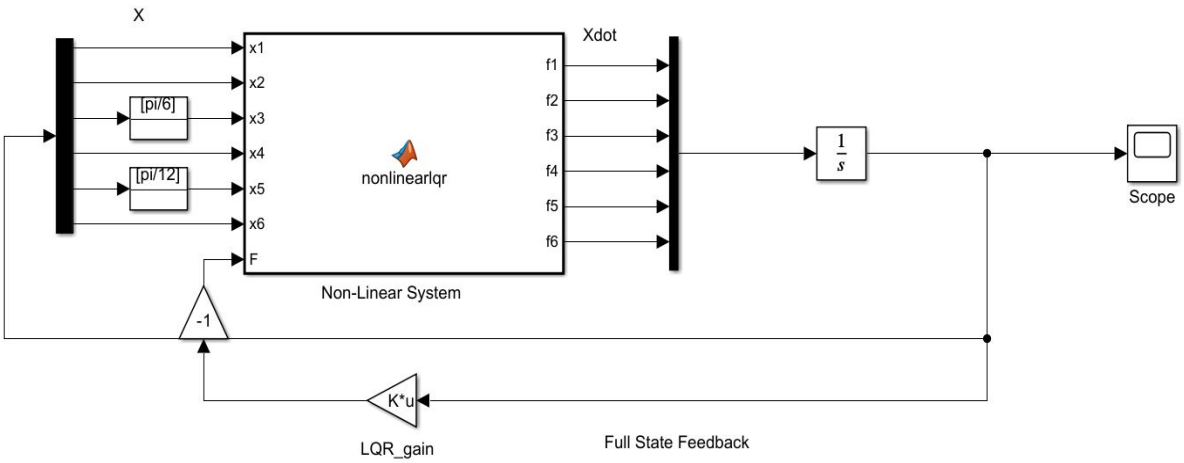


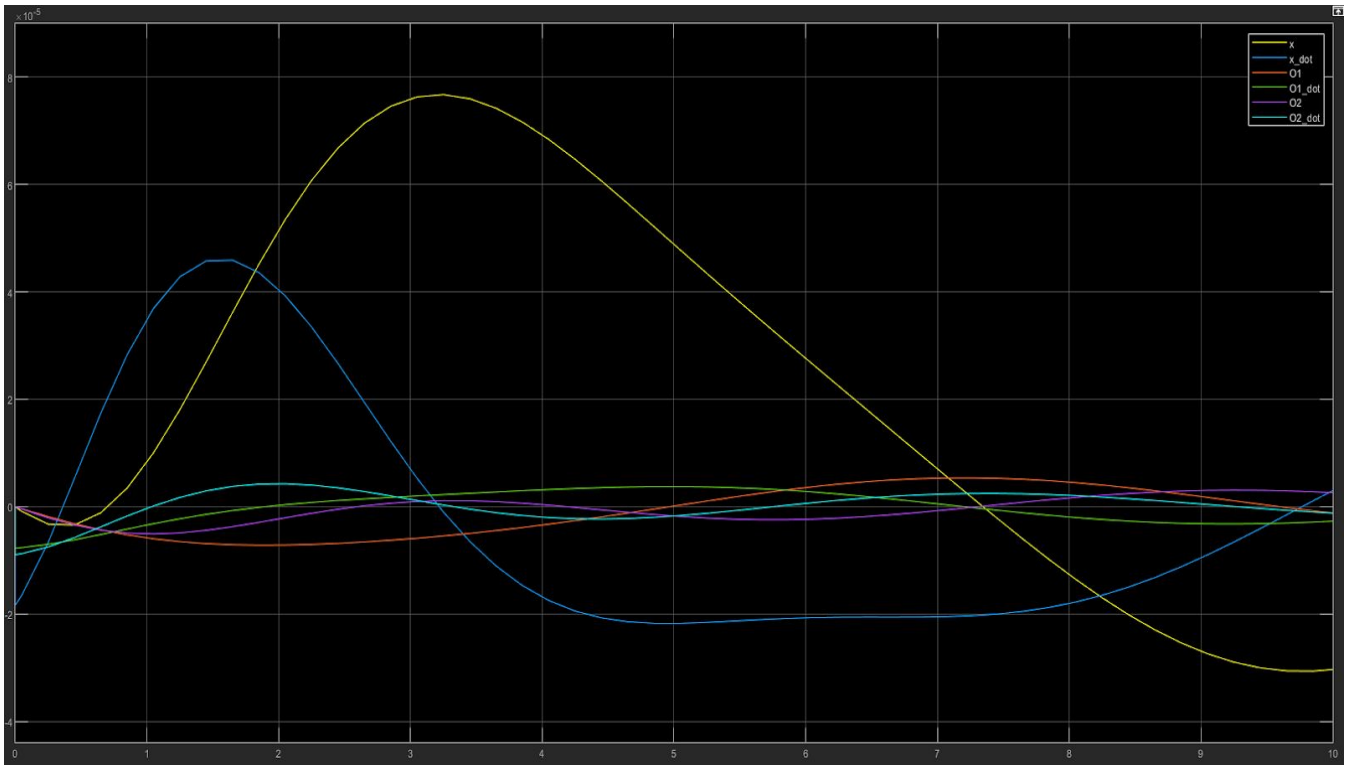
Figure 1
File Edit View Insert Tools Desktop Window Help
Position
12
10
8
6
4
2
0
-2
0 10 20 30 40 50 60 70 80 90 100
Time

Now putting some cost on regulator (R) for actual case where the actuators might be expensive, we get



Here is the following simulink simulation for a Nonlinear system where the system with some initial values of theta1 and theta2 (pi/6 and pi/12) acts as a regulator with some reference force F. In our system we have given the reference force for the input as zero and have shown the model tending to stabilize successfully. For the Nonlinear system LQR might not be the optimal solution.





E. Observability

The C matrices for the new output vectors can be computed. From there, the observability can be determined using the following equation where the determinant cannot be zero and the rank has to be full. Otherwise, the system is not observable.

$$Q = \begin{bmatrix} C\,A \\ C\,A^2 \\ C\,A^3 \\ C\,A^4 \\ C\,A^5 \\ C\,A^6 \end{bmatrix}$$

Eq 28

For each new output, the C matrices come out to be the following with: 1 = x(t), 2 = (theta1(t),theta2(t)), 3 = (x(t),theta2(t)), and 4 = (x(t),theta1(t),theta2(t).

$$C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad C_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad C_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Plugging these values into equation (28), the rank of each observability test comes out to full except for the second one. The rank is only 4, not 6, so (theta1(t),theta2(t)) is not observable. For example observability matrix and rank for C1 is :

```
Ob_po =

    1.0000         0         0         0         0         0
         0    1.0000         0         0         0         0
         0         0   -1.0000         0   -1.0000         0
         0         0         0   -1.0000         0   -1.0000
         0         0    0.6500         0    1.1500         0
         0         0         0    0.6500         0    1.1500

rank is

ans =

     6
```

F. Luenberger Observer

The Luenberger observer allows us to estimate the states of an observable system. $A\hat{x} + Bu$ is the dynamics prediction, $L(y - C\hat{x})$ is the output error correction with L being the gain. Put these together to get.

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \quad \text{Eq 29}$$

The following equation is used to find the error dynamics of the system.

$$\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}} = Ax + Bu - (A\hat{x} + Bu + LC(x - \hat{x})) = (A - LC)\tilde{x} \quad \text{Eq 30}$$

In order to find the desired gain, L, we use the following code in MATLAB, where A and C are the state space matrices and eigs are the pole placements.

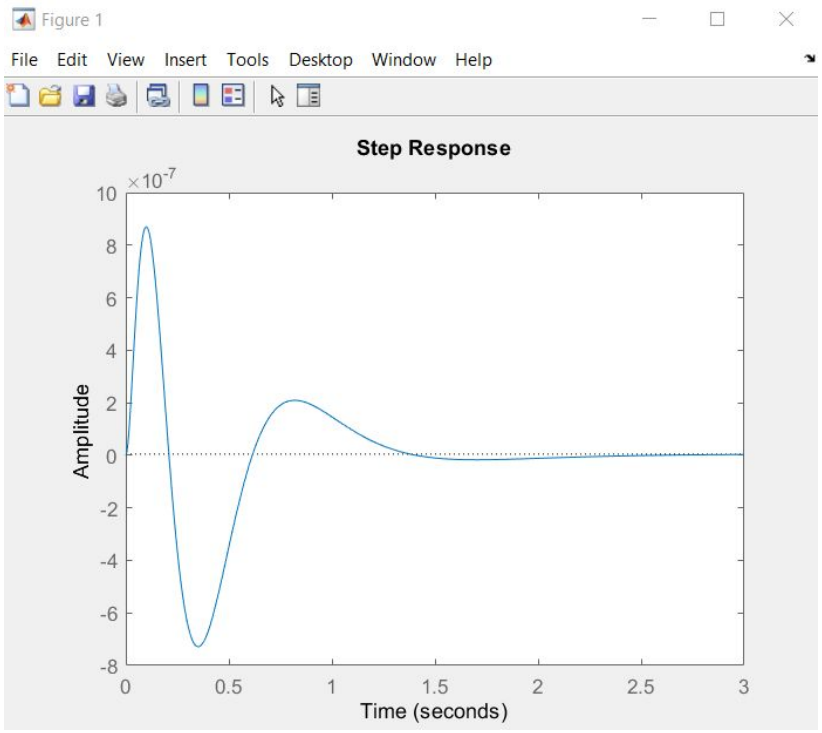
```
L' = place(A', C', eigs)
```

Eq 31

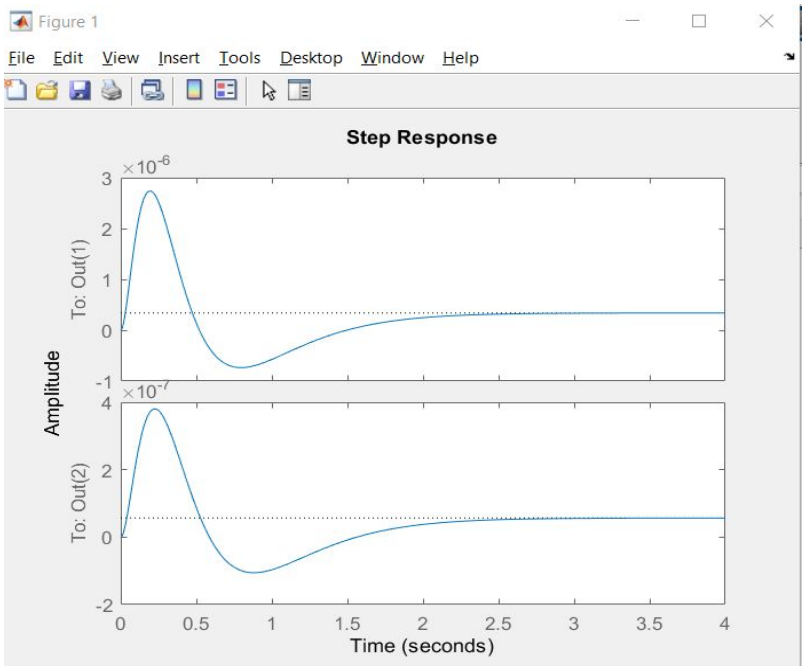
Below are the linear step responses with the modeled Luenberger observer. Each of the test cases is tested out separately, but L2 is not observable and throws error as C2 is not observable from the above section. The MATLAB code is shown below.

```
% place_poles = [2i , -12, -2i, -7, -3, -4];
 place_poles = [-11 , -12, -10, -7, -3, -4];

%implmenting luenberger

L1 = place(A_subs', C1', place_poles)'
luenburger1 = ss(A_subs - L1*C1, B_subs', C1, D)
step(luenburger1)

L2 = place(A_subs', C2', place_poles)'
luenburger2 = ss(A_subs - L2*C2, B_subs', C2, D)
step(luenburger2)

L3 = place(A_subs', C3', place_poles)'
luenburger3 = ss(A_subs - L3*C3, B_subs', C3, D)
step(luenburger3)


L4 = place(A_subs', C4', place_poles)';
luenburger4 = ss(A_subs - L4*C4, B_subs', C4, D);
step(luenburger4);
```

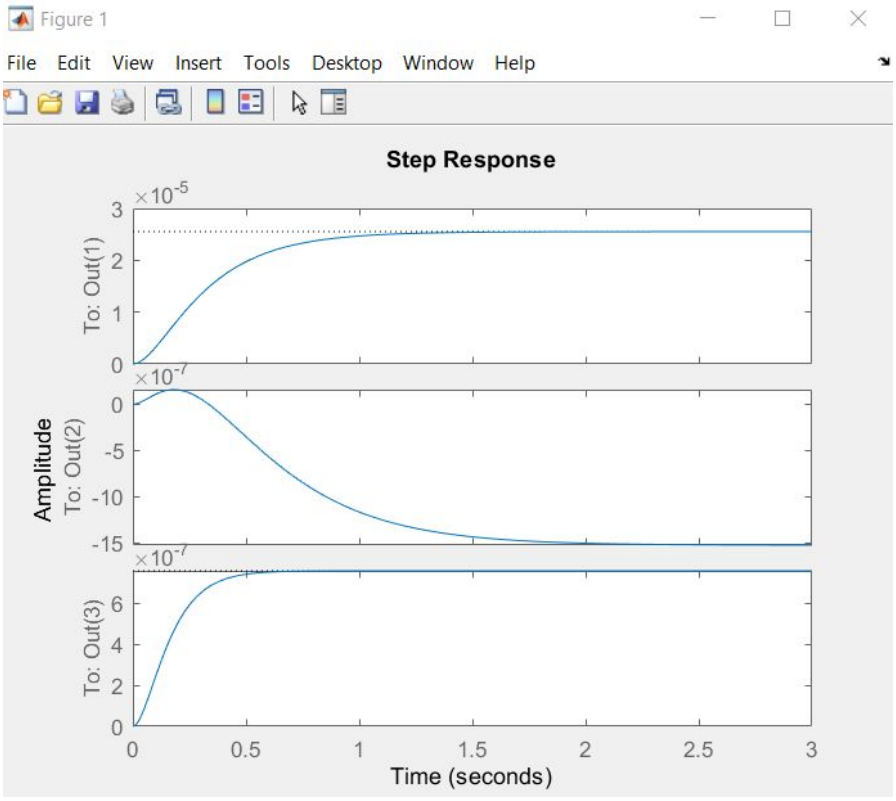
For x(t), C = [1 0 0 0 0 0], and with L1 the step response looks like:



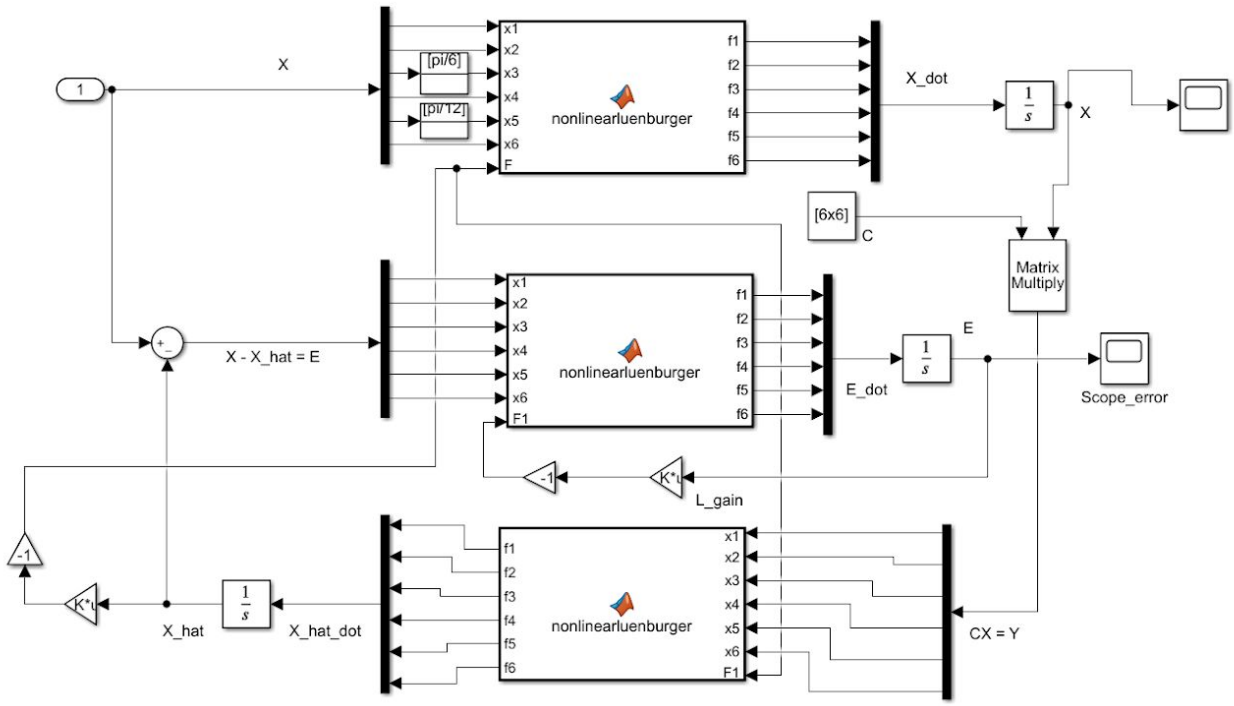
L2 does not have a graph because the poles cannot be placed. When x(t) and theta2(t) are measured, C = [1 0 0 0 0 0; 0 0 0 0 1 0]. A gain with L3 looks like:



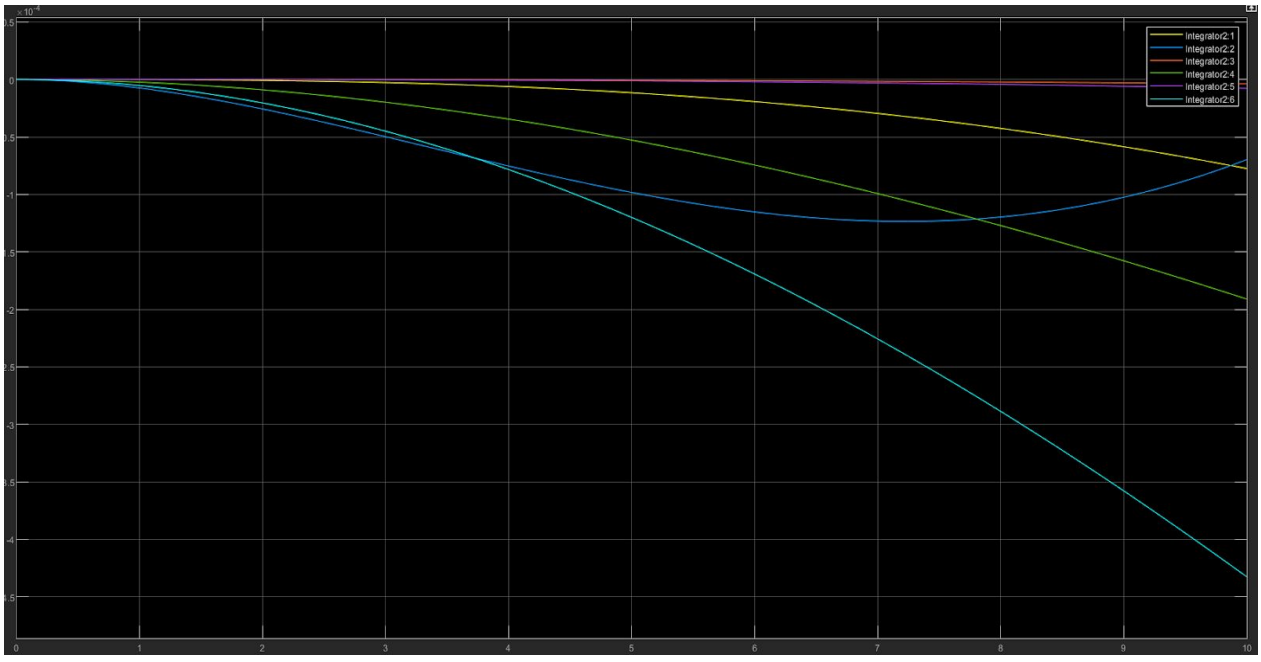
Lastly, when when x(t), theta1(t), and theta2(t) are measured and C = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0], L4 gives the step response:



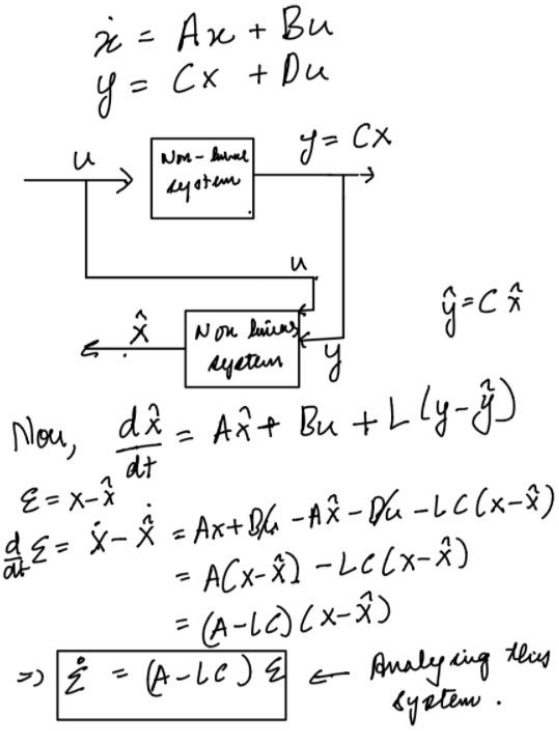
The nonlinear looks like this in Simulink:



The output of the nonlinear system looks like the following:



In this model :



Thus in the above simulink model if kalman filter added, state variables will stabilize

G. LQG Controller and Simulation

This type of controller alternates between a performance tracker and a control effort. It observes and fixes the model using both the LQR controller and a Kalman filter. The LQG controller takes into account disturbances and corrects it. The controller also measures noise. It is basically an optimal estimator and optimal regulator simultaneously.

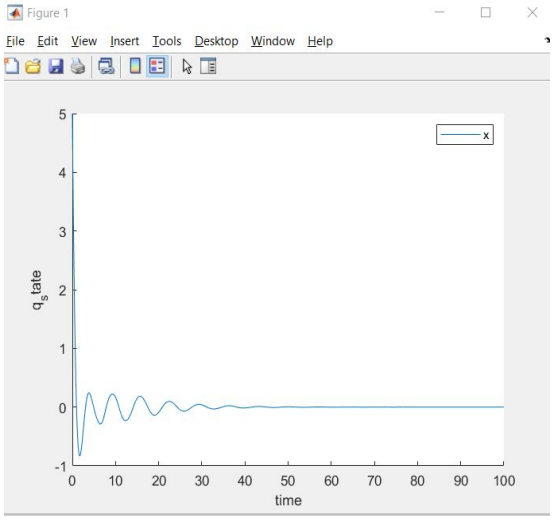
The first step is creating an LQ-optimal gain. Next is making a Kalman filter which acts as a state estimator. By connecting these two, the LQG is created.

```
function S = NonLinear_LQG(t,f,F,L)
M = 1000;
m1 = 100;
m2 = 100;
l1 = 20;
l2 = 10;
g = 10;

x = f(1);
dx = f(2);
t1 = f(3);
dt1 = f(4);
t2 = f(5);
dt2 = f(6);
out = [0; x; 0];
c1 = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 0 0];
control = L*(out - c1*f);
S=zeros(6,1);
S(1) = dx + control(1);
S(2) = (F-((m1*sin(t1)*cos(t1))+(m2*sin(t2)*cos(t2)))*g - (l1*m1*(S(3)^2)*sin(t1)) - (l2*m2*(S(5)^2)*sin(t2)))/(m1+m2+M-(m1*(cos(t1)^2))-(m2*(cos(t2)^2))) + control(2);
S(3) = dt1 + control(3);
S(4) = ((cos(t1)*S(2)-g*sin(t1))/l1) + control(4);
S(5) = dt2 + control(5);
S(6) = (cos(t2)*S(2)-g*sin(t2))/l2 + control(6);
end
```

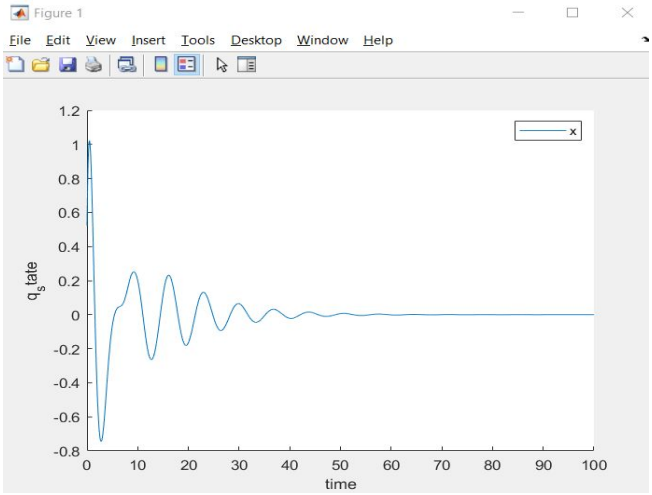
```
% Kalman Estimator
tspan = 0:0.1:100;
B = 0.1*eye(6); %Process Noise
V = 0.01; %Measurement Noise
c = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 0 0];
L = lqe(A,B,c,B,V*eye(3));
Ac1 = A-(L*c);
kalman_sys = ss(Ac1,[B L],c,0);

% Non-linear LQG Response
q_initial = [5 0 pi/6 0 pi/12 0];
[t,q1] = ode45(@(t,q)NonLinear_LQG(t,q,-K*q, L),tspan,q_initial);
figure();
hold on
plot(t,q1(:,3))
ylabel('q_state')
xlabel('time')
legend('x')
hold off
```

The following response of x(t) with c = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 0 0] was

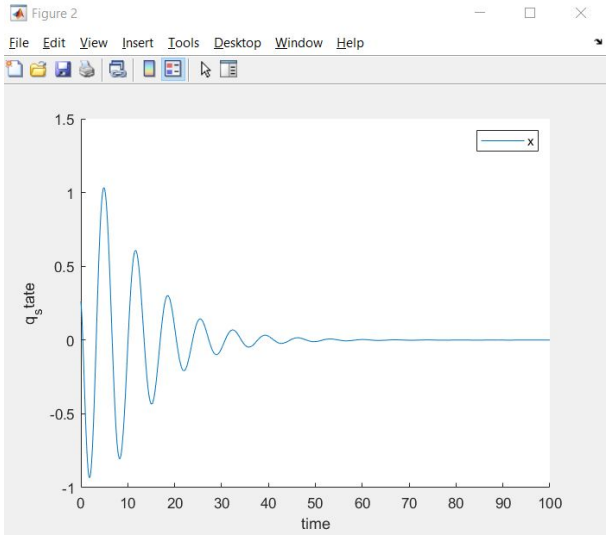
For position :



For theta1 :



For theta2:



Initial conditions taken were : 5, 0, pi/6, 0, pi/12, 0

This Controller can be reconfigured by adding an integral (1/s block) on the input to keep a track. The system may or may not reject force disturbances applied on the cart as LQG unlike LQR is not a robust system.