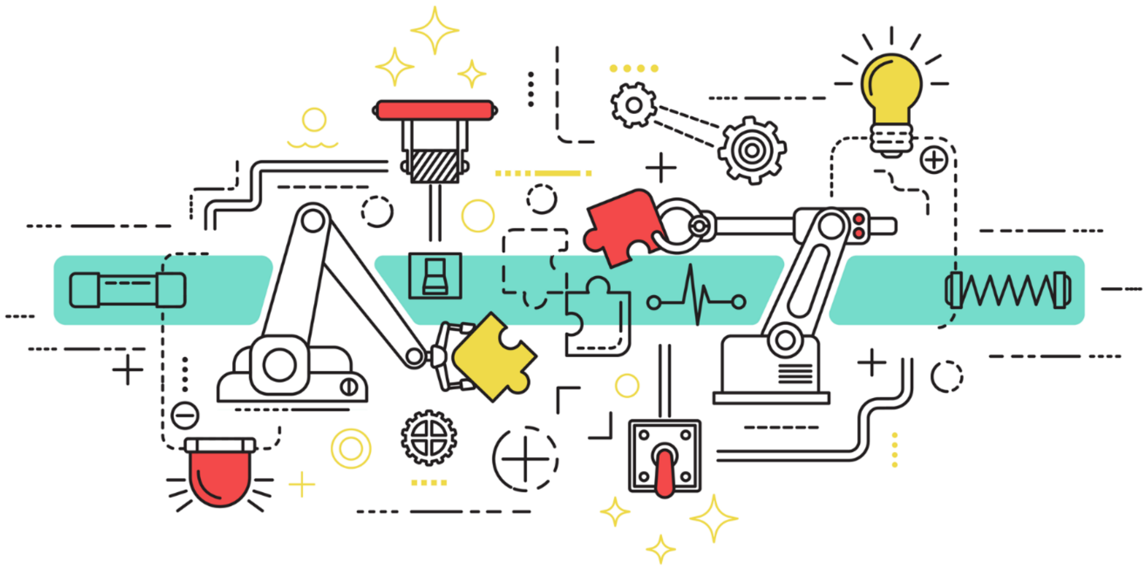
ENPM 667

Control of Robotic Systems



Project - 2

Technical Report

2023

Datta Lohith Gannavarapu
119455395
gdatta@umd.edu

Venkata Sai Sricharan Kasturi
119444788
charan03@umd.edu

Problem: First Component

Consider a crane that moves along an one-dimensional track. It behaves as a frictionless cart with mass M actuated by an external force F that constitutes the input of the system. There are two loads suspended from cables attached to the crane. The loads have mass m1 and m2, and the lengths of the cables are l1 and l2, respectively. The following figure depicts the crane and associated variables used throughout this project.

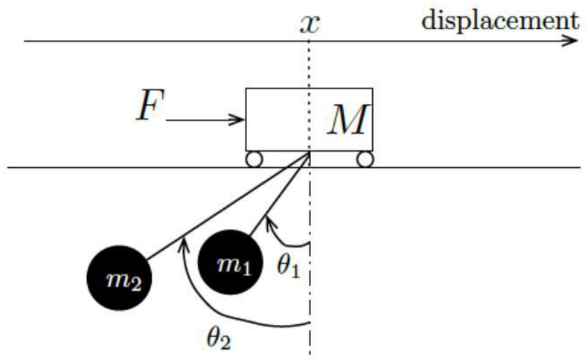


Figure 1: Crane with 2 suspended loads

A) (25 points) Obtain the equations of motion for the system and the corresponding nonlinear state-space representation.

B) (25 points) Obtain the linearized system around the equilibrium point specified by x = 0 and θ1 = θ2 = 0. Write the state-space representation of the linearized system.

C) (25 points) Obtain conditions on M, m1, m2, l1, l2 for which the linearized system is controllable.

D) (25 points) Choose M = 1000Kg, m1 = m2 = 100Kg, l1 = 20m and l2 = 10m. Check that the system is controllable and obtain an LQR controller. Simulate the resulting response to initial conditions when the controller is applied to the linearized system and also to the original nonlinear system. Adjust the parameters of the LQR cost until you obtain a suitable response. Use Lyapunov's indirect method to certify stability (locally or globally) of the closed-loop system.

Solution:

Terminology and variables used in this report:

- M: Mass of the cart
- F: Force applied to the cart
- m1: Mass of Load 1
- m2: Mass of Load 2
- l1: Length of the cable on which load 1 is suspended

- L2: Length of the cable on which load 2 is suspended

Assumptions Considered in this report:

- The payload is treated as if it were a solid particle.
- The rope is assumed to behave like a rigid rod.
- The mass of the rope is neglected when compared to the mass of the payload.
- The trolley moves exclusively in the x-direction.
- The payload's motion occurs on the x-y surface.
- There is no presence of friction within the system.

A) Equations of motion:

Given the assumptions mentioned earlier, we can derive the equations of motion for the system using either the Newton-Euler or Euler-Lagrange method. For this project, we have chosen to employ the Euler-Lagrange technique. This approach relies on determining the Kinetic and Potential Energies of the system to formulate the equations of motion. The Euler-Lagrange equation, which plays a pivotal role in this method, is expressed as follows:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\mathrm{q}}}\right] - \frac{\partial \mathcal{L}}{\partial \mathrm{q}} = \mathrm{f}$$

In the context of the chosen Euler-Lagrange technique for analytical modeling of the cart and pendulum system, the key components are as follows:

- $\mathcal{L}$ represents the Lagrange operator.

- q represents the generalized coordinates of the system.

- T signifies the Kinetic energy of the system.

- V represents the Potential energy of the system.

- f represents the applied force acting on the system in terms of its generalized coordinates.

Lagrangian equation –

The generalized coordinates in our given system are x, θ_1 *and* θ_2. Using the frame of reference shown in Figure 1, position of mass 1 and mass 2 can be computed as follows,

$$X_{m1} = x - l_1 S_1$$

$$Y_{m1} = -l_1 C_1$$

$$X_{m2} = x - l_2 S_2$$

$$\mathrm{Y}_{m2} = -l_2 C_2$$

In this context, x is the displacement of cart in positive-x direction, S *is* $\sin(\theta)$ *and* C *is* $\cos(\theta)$.

Derivation of the above equations gives us the velocities associated with each pendulum loads.

$$\dot{X}_{m1} = \dot{x} - \dot{\theta}_1 l_1 C_1$$

$$\dot{Y}_{m1} = \dot{\theta}_1 l_1 S_1$$

$$\dot{X}_{m2} = \dot{x} - \dot{\theta}_2 l_2 C_2$$

$$\dot{Y}_{m2} = \dot{\theta}_2 l_2 S_2$$

The above equations will yield the components of the pendulum's velocities in both the x-direction and the y-direction. To calculate the magnitude of the velocity vector associated with the pendulum, you can use the following formula:

$$v_1^2 = \dot{X}_{m1}^{\,2} + \dot{Y}_{m1}^{\,2}$$

$$= \left(\dot{x} - \dot{\theta}_1 l_1 C_1\right)^2 + \left(\dot{\theta}_1 l_1 S_1\right)^2$$

$$= \dot{x}^2 + \left(\dot{\theta}_1 l_1\right)^2 - 2\dot{x}\,\dot{\theta}_1 l_1 C_1 \qquad (1)$$

$$v_2^2 = \dot{X}_{m2}^{\,2} + \dot{Y}_{m2}^{\,2}$$

$$= \left(\dot{x} - \dot{\theta}_2 l_2 C_2\right)^2 + \left(\dot{\theta}_2 l_2 S_2\right)^2$$

$$= \dot{x}^2 + \left(\dot{\theta}_2 l_2\right)^2 - 2\dot{x}\,\dot{\theta}_2 l_2 C_2 \qquad (2)$$

Since the displacement of the cart only occurs in the positive-x direction, the velocity of the cart can be represented as $\dot{x}$. By using Equation (1) and Equation (2), the Kinetic Energy (KE) of the entire system can be expressed as the sum of the kinetic energies associated with the cart and both pendulums.

$$K.E. = \frac{1}{2}\dot{x}^2(M) + \frac{1}{2}(m_1 v_1^2) + \frac{1}{2}(m_2 v_2^2)$$

Solving the above equation with (1) and (2)

$$= \frac{1}{2}\dot{x}^2(m_1 + m_2 + M) + \frac{1}{2}\left(m_1 l_1 \dot{\theta}_1^2\right) + \frac{1}{2}\left(m_2 l_2 \dot{\theta}_2^2\right) - m_1 l_1 c_{s1}\dot{x}\dot{\theta}_1 - m_2 l_2 c_{s2}\dot{x}\dot{\theta}_2$$

To calculate the Potential Energy of the system, we establish the cart's height as the reference point. Consequently, the Potential Energy only involves contributions from the pendulums and can be expressed as:

$$P.E. = -m_1 g l_1 C_1 - m_2 g l_2 C_2$$

Now, we can determine the Lagrangian of the system by subtracting the Potential Energy from the Kinetic Energy:

$$\boldsymbol{\mathcal{L}} = K.E. - P.E.$$

$$\mathcal{L} = \frac{1}{2}\dot{x}^2(m_1 + m_2 + M) + \frac{1}{2}\left(m_1 l_1 \dot{\theta}_1^2\right) + \frac{1}{2}\left(m_2 l_2 \dot{\theta}_2^2\right) - m_1 l_1 c_{s1} \dot{x}\dot{\theta}_1 - m_2 l_2 c_{s2} \dot{x}\dot{\theta}_2 + m_1 g l_1 c_1 + m_2 g l_2 c_2 \qquad (3)$$

From Equation (3), we find the following equations,

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

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

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

Solving the above equations,

$$\frac{\partial \mathcal{L}}{\partial \dot{x}} = \dot{x}(m_1 + m_2 + M) - m_1 l_1 c_1\left(\dot{\theta}_1\right) - m_2 l_2 c_2\left(\dot{\theta}_2\right)$$

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

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) = \ddot{x}(m_1 + m_2 + M) - m_1 l_1\left(\dot{\theta}_1 C_1 - S_1 \dot{\theta}_1^2\right) - m_2 l_2\left(\dot{\theta}_2 C_2 - S_2 \dot{\theta}_2^2\right) = F$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = m_1 l_1 \dot{\theta}_1^2 - m_1 l_1 C_1 \dot{x}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right) = m_1 l_1 \ddot{\theta}_1 - m_1 l_1\left(C_1 \ddot{x} - S_1 \dot{x}\dot{\theta}_1\right)$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = S_1 m_1 l_1 m_1 \dot{x}\dot{\theta}_1 - m_1 g l_1 S_1$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = m_2 l_2 \dot{\theta}_2^2 - m_2 l_2 C_2 \dot{x}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = S_2 m_2 l_2 m_2 \dot{x}\dot{\theta}_2 - m_2 g l_2 S_2$$

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_2}\right) = m_2l_2\ddot{\theta}_2 - m_2l_2\left(C_2\ddot{x} - S_2\dot{x}\dot{\theta}_2\right)$$

Upon substitution and simplification of the above equations, we get,

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_1}\right) - \frac{\partial\mathcal{L}}{\partial\theta_1} = m_1l_1\dot{\theta}_1^2 - m_1l_1\left(C_1\dot{x} - S_1\dot{x}\dot{\theta}_1\right) - S_1m_1l_1m_1\dot{x}\dot{\theta}_1 + m_1gl_1S_1 = 0$$

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_1}\right) - \frac{\partial\mathcal{L}}{\partial\theta_1} = m_1l_1\dot{\theta}_1^2 - m_1l_1C_1\dot{x} + m_1gl_1S_1$$

Now we assume that for very small angle we take the following assumptions:

$$[C_1 = 1, S_1 = \theta_1, \dot{\theta}_1^2 = 0, \dot{\theta}_2^2 = 0]$$

Linearizing the previous equation, we get,

$$m_1l_1\dot{\theta}_1^2 - m_1l_1\dot{x} + m_1gl_1\theta_1 = 0$$

||^{ly},

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_2}\right) - \frac{\partial\mathcal{L}}{\partial\theta_2} = m_2l_2\dot{\theta}_2^2 - m_2l_2\left(C_2\dot{x} - S_2\dot{x}\dot{\theta}_2\right) - S_2m_2l_2m_2\dot{x}\dot{\theta}_2 + m_2gl_2S_2$$

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_2}\right) - \frac{\partial\mathcal{L}}{\partial\theta_2} = m_2l_2\dot{\theta}_2^2 - m_2l_2C_2\dot{x} + m_2gl_2S_2$$

Linearizing the above equation, we get,

$$m_2l_2\dot{\theta}_2^2 - m_2l_2\dot{x} + m_2gl_2\theta_2 = 0$$

$$F = \dot{x}(m_1 + m_2 + M) - m_1l_1\left(\dot{\theta}_1C_1 - S_1\dot{\theta}_1^2\right) - m_2l_2\left(\dot{\theta}_2C_2 - S_2\dot{\theta}_2^2\right)$$

$$m_1l_1\dot{\theta}_1^2 - m_1l_1C_1\dot{x} + m_1gl_1S_1 = 0$$

$$m_2l_2\dot{\theta}_2^2 - m_2l_2C_2\dot{x} + m_2gl_2S_2 = 0$$

$$\ddot{\theta}_1 = \frac{C_1\ddot{x} - gS_1}{l_1}$$

$$\ddot{\theta}_2 = \frac{C_2\ddot{x} - gS_2}{l_2}$$

Substituting values of $\ddot{\theta}_1, \ddot{\theta}_2$ in equation $\ddot{x}$, we get,

$$\ddot{x} = \frac{F + m_1 l_1\left(\ddot{\theta}_1^2 C_1 - S_1\dot{\theta}_1^2\right) + m_2 l_2\left(\ddot{\theta}_2^2 C_2 - S_2\dot{\theta}_2^2\right)}{m_1 + m_2 + M}$$

$$\ddot{x} = \frac{F + m_1 l_1\left(C_1\left(\frac{C_1\ddot{x} - gS_1}{l_1}\right)^2 - S_1\dot{\theta}_1^2\right) + m_2 l_2\left(C_2\left(\frac{C_2\ddot{x} - gS_2}{l_2}\right)^2 - S_2\dot{\theta}_2^2\right)}{m_1 + m_2 + M}$$

$$\ddot{x} = \frac{F + m_1\left(C_1\ddot{x} - gS_1C_1 - l_1S_1\dot{\theta}_1^2\right) + m_2\left(C_2\ddot{x} - gS_2C_2 - l_2S_2\dot{\theta}_2^2\right)}{m_1 + m_2 + M}$$

$$(m_1 + m_2 + M)\ddot{x} = F - m_1\left(gS_1C_1 + l_1S_1\dot{\theta}_1^2\right) - m_2\left(gS_2C_2 + l_2S_2\dot{\theta}_2^2\right)$$

$$\boldsymbol{\ddot{x} = \frac{F - m_1\left(gS_1C_1 + l_1S_1\dot{\theta}_1^2\right) - m_2\left(gS_2C_2 + l_2S_2\dot{\theta}_2^2\right)}{M + m_1(1 - S_1^2) + m_2(1 - S_2^2)}} \qquad \mathbf{(4)}$$

$Next, we substitute the value of \ddot{x} in the previously derived equations of \ddot{\theta}_1, \ddot{\theta} \; we get$

$$\boldsymbol{\ddot{\theta}_1 = \frac{C_1}{l_1}\left[\frac{F - m_1\left(gS_1C_1 + l_1S_1\dot{\theta}_1^2\right) - m_2\left(gS_2C_2 + l_2S_2\dot{\theta}_2^2\right)}{M + m_1(S_1^2) + m_2(S_2^2)}\right] - \frac{gS_1}{l_1}} \qquad \mathbf{(5)}$$

$$\boldsymbol{\ddot{\theta}_2 = \frac{C_2}{l_2}\left[\frac{F - m_1\left(gS_1C_1 + l_1S_1\dot{\theta}_1^2\right) - m_2\left(gS_2C_2 + l_2S_2\dot{\theta}_2^2\right)}{M + m_1(S_1^2) + m_2(S_2^2)}\right] - \frac{gS_2}{l_2}} \qquad \mathbf{(6)}$$

Non-Linear State Space Representation –

The Equations which were derived so far are the non-linear equations of the two-pendulum crane system.

$$\dot{X} = AX + BU \qquad (7)$$

The above equation is the state space representation of the system. Upon expanding $\dot{X}$, from equations (4), (5), (6), we get

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dfrac{F - m_1\left(gS_1C_1 + l_1S_1\dot{\theta}_1^2\right) - m_2\left(gS_2C_2 + l_2S_2\dot{\theta}_2^2\right)}{\left(M + m_1(S_1^2) + m_2(S_2^2)\right)} \\ \dot{\theta}_1 \\ \dfrac{C_1}{l_1}\left[\dfrac{F - m_1\left(gS_1C_1 + l_1S_1\dot{\theta}_1^2\right) - m_2\left(gS_2C_2 + l_2S_2\dot{\theta}_2^2\right)}{\left(M + m_1(S_1^2) + m_2(S_2^2)\right)}\right] - \dfrac{gS_1}{l_1} \\ \dot{\theta}_2 \\ \dfrac{C_2}{l_2}\left[\dfrac{F - m_1\left(gS_1C_1 + l_1S_1\dot{\theta}_1^2\right) - m_2\left(gS_2C_2 + l_2S_2\dot{\theta}_2^2\right)}{\left(M + m_1(S_1^2) + m_2(S_2^2)\right)}\right] - \dfrac{gS_2}{l_2} \end{bmatrix}$$

B) Linearizing Equations of Motion

Linearization around the given equilibrium point $x = 0, \theta_1 = 0, \theta_2 = 0$, we get the Jacobian matrix,

$$A = \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial \dot{x}} & \frac{\partial F_1}{\partial \theta_1} & \frac{\partial F_1}{\partial \dot{\theta}_1} & \frac{\partial F_1}{\partial \theta_2} & \frac{\partial F_1}{\partial \dot{\theta}_2} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial \dot{x}} & \frac{\partial F_2}{\partial \theta_1} & \frac{\partial F_2}{\partial \dot{\theta}_1} & \frac{\partial F_2}{\partial \theta_2} & \frac{\partial F_2}{\partial \dot{\theta}_2} \\ \frac{\partial F_3}{\partial x} & \frac{\partial F_3}{\partial \dot{x}} & \frac{\partial F_3}{\partial \theta_1} & \frac{\partial F_3}{\partial \dot{\theta}_1} & \frac{\partial F_3}{\partial \theta_2} & \frac{\partial F_3}{\partial \dot{\theta}_2} \\ \frac{\partial F_4}{\partial x} & \frac{\partial F_4}{\partial \dot{x}} & \frac{\partial F_4}{\partial \theta_1} & \frac{\partial F_4}{\partial \dot{\theta}_1} & \frac{\partial F_4}{\partial \theta_2} & \frac{\partial F_4}{\partial \dot{\theta}_2} \\ \frac{\partial F_5}{\partial x} & \frac{\partial F_5}{\partial \dot{x}} & \frac{\partial F_5}{\partial \theta_1} & \frac{\partial F_5}{\partial \dot{\theta}_1} & \frac{\partial F_5}{\partial \theta_2} & \frac{\partial F_5}{\partial \dot{\theta}_2} \\ \frac{\partial F_6}{\partial x} & \frac{\partial F_6}{\partial \dot{x}} & \frac{\partial F_6}{\partial \theta_1} & \frac{\partial F_6}{\partial \dot{\theta}_1} & \frac{\partial F_6}{\partial \theta_2} & \frac{\partial F_6}{\partial \dot{\theta}_2} \end{bmatrix} B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml_1} \\ 0 \\ \frac{1}{Ml_2} \end{bmatrix}$$

Equation (7) becomes,

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{gm_1}{M} & 0 & -\frac{gm_2}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{g(M+m_1)}{Ml_1} & 0 & -\frac{gm_2}{Ml_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{gm_1}{Ml_2} & 0 & -\frac{g(M+m_2)}{Ml_2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml_1} \\ 0 \\ \frac{1}{Ml_2} \end{bmatrix}$$

C) Controllability

Code –

```
clc; clear;

% Defining the symbolic variables for the system parameters
syms M mass_1 mass_2 length_1 length_2 g;

% Constructing the 'A' matrix of the state-space representation for the linearized system
A = [0, 1, 0, 0, 0, 0;
     0, 0, -(mass_1*g)/M, 0, -(mass_2*g)/M, 0;
     0, 0, 0, 1, 0, 0;
     0, 0, -((M+mass_1)*g)/(M*length_1), 0, -(mass_2*g)/(M*length_1), 0;
     0, 0, 0, 0, 0, 1;
     0, 0, -(mass_1*g)/(M*length_2), 0, -(g*(M+mass_2))/(M*length_2), 0];
disp('Matrix A:');
disp(A);

% Constructing the 'B' matrix of the state-space representation
B = [0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];
disp('Matrix B:');
disp(B);

% Calculating the controllability matrix of the system
controllability_matrix = [B, A*B, A^2*B, A^3*B, A^4*B, A^5*B];
disp('Controllability Matrix:');
disp(controllability_matrix);

% Calculating the determinant of the controllability matrix
det_controllability_matrix = simplify(det(controllability_matrix));
disp('Determinant of the Controllability Matrix:');
disp(det_controllability_matrix);

% Checking the rank of the controllability matrix for system controllability
rank_controllability_matrix = rank(controllability_matrix);
disp('Rank of the Controllability Matrix:');
disp(rank_controllability_matrix);

% Displaying whether the system is controllable or not based on the rank
if rank_controllability_matrix < 6
    disp('The system is not controllable.');
else
```

```
    disp('The system is controllable.');
end
```

Output –

```
MATLAB Command Window                                                    1 of 1

Matrix A:
[0, 1,                          0, 0,                           0, 0]
[0, 0,                -(g*mass_1)/M, 0,               -(g*mass_2)/M, 0]
[0, 0,                          0, 1,                           0, 0]
[0, 0, -(g*(M + mass_1))/(M*length_1), 0,       -(g*mass_2)/(M*length_1), 0]
[0, 0,                          0, 0,                           0, 1]
[0, 0,       -(g*mass_1)/(M*length_2), 0, -(g*(M + mass_2))/(M*length_2), 0]

Matrix B:
             0
           1/M
             0
1/(M*length_1)
             0
1/(M*length_2)

Controllability Matrix:
[              0,             1/M,
[            1/M,               0,                    - (g*mass_1)/(M^2*length_1) - (g*mass_2)
[              0, 1/(M*length_1),
[1/(M*length_1),               0, - (g*(M + mass_1))/(M^2*length_1^2) - (g*mass_2)/(M^2*ler
[              0, 1/(M*length_2),
[1/(M*length_2),               0, - (g*(M + mass_2))/(M^2*length_2^2) - (g*mass_1)/(M^2*ler

Determinant of the Controllability Matrix:
-(g^6*(length_1 - length_2)^2)/(M^6*length_1^6*length_2^6)

Rank of the Controllability Matrix:
     6

The system is controllable.
>>
```

D) LQR Controller

Code –

```
% Linear System
clc; clear;

% Defining the system parameters
M=1000;  % Defining the mass of the crane in kg
mass_1=100;  % Defining the mass of the first load in kg
mass_2=100;  % Defining the mass of the second load in kg
length_1=20;   % Defining the length of the first cable in meters
length_2=10;   % Defining the length of the second cable in meters
```

```
g=9.81;  % Defining the acceleration due to gravity in m/s^2

% Substituting the physical parameters into the state matrices A and B
A=[0 1 0 0 0 0;
     0 0 -(mass_1*g)/M 0 -(mass_2*g)/M 0;
     0 0 0 1 0 0;
     0 0 -((M+mass_1)*g)/(M*length_1) 0 -(mass_2*g)/(M*length_1) 0;
     0 0 0 0 0 1;
     0 0 -(mass_1*g)/(M*length_2) 0 -(g*(M+mass_2))/(M*length_2) 0];
B=[0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];

% Checking the controllability of the system
controllability_matrix=ctrb(A, B);
if rank(controllability_matrix) == 6
    disp('The system is controllable.');
else
    disp('The system is uncontrollable.');
end

% Setting initial conditions for the system
initial_state=[0;0;10;0;30;0];

% Setting LQR weighting matrices
Q=diag([10 100 1000 1000 100 100]); % Emphasizing the importance of state regulation
R=0.001;            % Representing the cost of control input

% Designing LQR controller
[K,S,eigen_values]=lqr(A, B, Q, R);

% Displaying the LQR results
disp('LQR Gain Matrix K:');
disp(K);
disp('Solution to Riccati Equation S:');
disp(S);
disp('Closed-loop eigen_values:');
disp(eigen_values);

% Defining the new A matrix for the closed-loop system
A_q=A-B*K;

% Creating the state-space models
system_state_space=ss(A,B,eye(6),zeros(6, 1));
controlled_system_state_space=ss(A_q,B,eye(6),zeros(6,1));

% Setting time span for the simulation
t=0:0.01:100;

% Simulating the uncontrolled system response
[Y_uncontrolled,T_uncontrolled,X_uncontrolled]=lsim(system_state_space,zeros(length(t),1),t,initia
l_state);
% Simulating the controlled system response
[Y_controlled,T_controlled,X_controlled]=lsim(controlled_system_state_space,zeros(length(t),1),t,i
nitial_state);

% Preparing graphs for each state variable in a single window
figure;
colors=['b','g', 'r','c','m','y']; % Assigning different colors for each plot

for i=1:6
    % Plotting uncontrolled system response
    subplot(6,2,2*i-1);
    plot(T_uncontrolled,X_uncontrolled(:,i),colors(i));
    title(sprintf('Uncontrolled: State %d',i));
    xlabel('Time(s)');
    ylabel(sprintf('State %d',i));
    grid on;

    % Plotting controlled system response
    subplot(6,2,2*i);
    plot(T_controlled,X_controlled(:,i),colors(i));
    title(sprintf('Controlled: State %d',i));
    xlabel('Time(s)');
    ylabel(sprintf('State %d',i));
    grid on;
end

% Adjusting the figure size for better visibility
set(gcf,'Position',[100,100,1200,800]);
```

Output –

```
MATLAB Command Window                                    1 of 1

Problem_D_1
The system is controllable.
LQR Gain Matrix K:
  100.0000  593.7585  526.4919  606.3359   89.0470 -125.5380

Solution to Riccati Equation S:
   1.0e+04 *

    0.0059    0.0126    0.0061   -0.0307   -0.0013   -0.0109
    0.0126    0.0745    0.0667   -0.1701    0.0035   -0.0660
    0.0061    0.0667    2.5227   -0.0316   -0.1027   -0.1251
   -0.0307   -0.1701   -0.0316    4.9568    0.1174   -0.1709
   -0.0013    0.0035   -0.1027    0.1174    0.5951   -0.0042
   -0.0109   -0.0660   -0.1251   -0.1709   -0.0042    0.6203

Closed-loop eigen_values:
  -0.0281 + 1.0375i
  -0.0281 - 1.0375i
  -0.0340 + 0.7257i
  -0.0340 - 0.7257i
  -0.2436 + 0.1590i
  -0.2436 - 0.1590i

>>
```

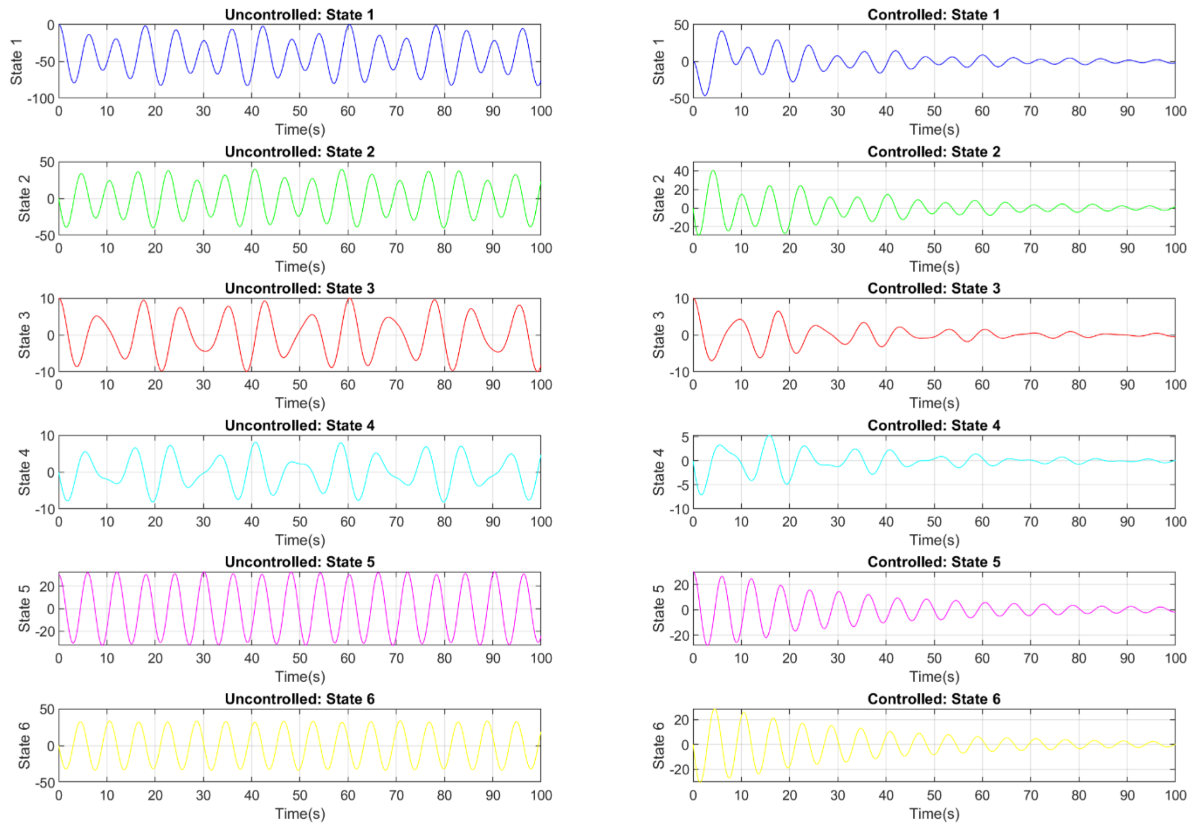


Figure 2: Result of Problem D (Linear System)

Code –

```
% Non - Linear System
clc; clear;

% Initializing state variables for the system
initial_state = [0; 0; 0; 0; 30; 0];  % Defining [x; x_dot; theta1; theta1_dot; theta2;
theta2_dot]

% Setting time span for the simulation
simulation_time_span = 0:0.01:10000;

% Solving the system using the ode45 solver
[t, state_evolution] = ode45(@system_dynamics, simulation_time_span, initial_state);

% Plotting each state variable in separate plots
figure;
subplot(3,2,1);
plot(t, state_evolution(:,1), 'b'); % Plotting x in blue
title('x');
xlabel('Time(s)');
ylabel('x');
grid on;

subplot(3,2,2);
plot(t, state_evolution(:,2), 'g'); % Plotting x_dot in green
title('x\_dot');
```

```
xlabel('Time (s)');
ylabel('x\_dot');
grid on;

subplot(3,2,3);
plot(t, state_evolution(:,3), 'r'); % Plotting theta1 in red
title('theta1');
xlabel('Time(s)');
ylabel('theta1');
grid on;

subplot(3,2,4);
plot(t, state_evolution(:,4), 'c'); % Plotting theta1_dot in cyan
title('theta1\_dot');
xlabel('Time(s)');
ylabel('theta1\_dot');
grid on;

subplot(3,2,5);
plot(t, state_evolution(:,5), 'm'); % Plotting theta2 in magenta
title('theta2');
xlabel('Time(s)');
ylabel('theta2');
grid on;

subplot(3,2,6);
plot(t, state_evolution(:,6), 'y'); % Plotting theta2_dot in yellow
title('theta2\_dot');
xlabel('Time(s)');
ylabel('theta2\_dot');
grid on;

% Defining the system dynamics as a function
function state_derivatives = system_dynamics(t, state)
    % Defining the constants and system parameters
    M = 1000; % Defining the mass of the crane in kg
    mass_1 = 100; % Defining the mass of Load 1 in kg
    mass_2 = 100; % Defining the mass of Load 2 in kg
    length_1 = 20;  % Defining the cable length of Load 1 in meters
    length_2 = 10;  % Defining the cable length of Load 2 in meters
    g = 9.81; % Defining the acceleration due to gravity in m/s^2

    % Constructing system matrices for state-space representation
    A = [0 1 0 0 0 0;
         0 0 -(mass_1*g)/M 0 -(mass_2*g)/M 0;
         0 0 0 1 0 0;
         0 0 -((M+mass_1)*g)/(M*length_1) 0 -(mass_2*g)/(M*length_1) 0;
         0 0 0 0 0 1;
         0 0 -(mass_1*g)/(M*length_2) 0 -(g*(M+mass_2))/(M*length_2) 0];
    B = [0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];

    % Designing LQR weighting matrices
    Q = diag([1000 100 1000 1000 100 100]); % Emphasizing the importance of state regulation
    R = 0.01;

    % Computing the LQR gain matrix
    [K, S, eigen_values] = lqr(A, B, Q, R);

    % Calculating the control force based on LQR gain and current state
    controlled_force = -K * state;

    % Formulating differential equations describing system dynamics
    state_derivatives = zeros(6, 1); % Preallocating for speed
    state_derivatives(1) = state(2); % Computing x_dot
    state_derivatives(2) = (controlled_force - (g/2)*(mass_1*sind(2*state(3)) +
mass_2*sind(2*state(5))) - (mass_1*length_1*state(4)^2*sind(state(3))) -
(mass_2*length_2*state(6)^2*sind(state(5)))) / (M + mass_1*sind(state(3))^2 +
mass_2*sind(state(5))^2); % Computing x_double_dot
    state_derivatives(3) = state(4); % Computing theta1_dot
    state_derivatives(4) = (state_derivatives(2)*cosd(state(3)) - g*sind(state(3))) / length_1; %
Computing theta1_double_dot
    state_derivatives(5) = state(6); % Computing theta2_dot
    state_derivatives(6) = (state_derivatives(2)*cosd(state(5)) - g*sind(state(5))) / length_2; %
Computing theta2_double_dot
end
```

Output –

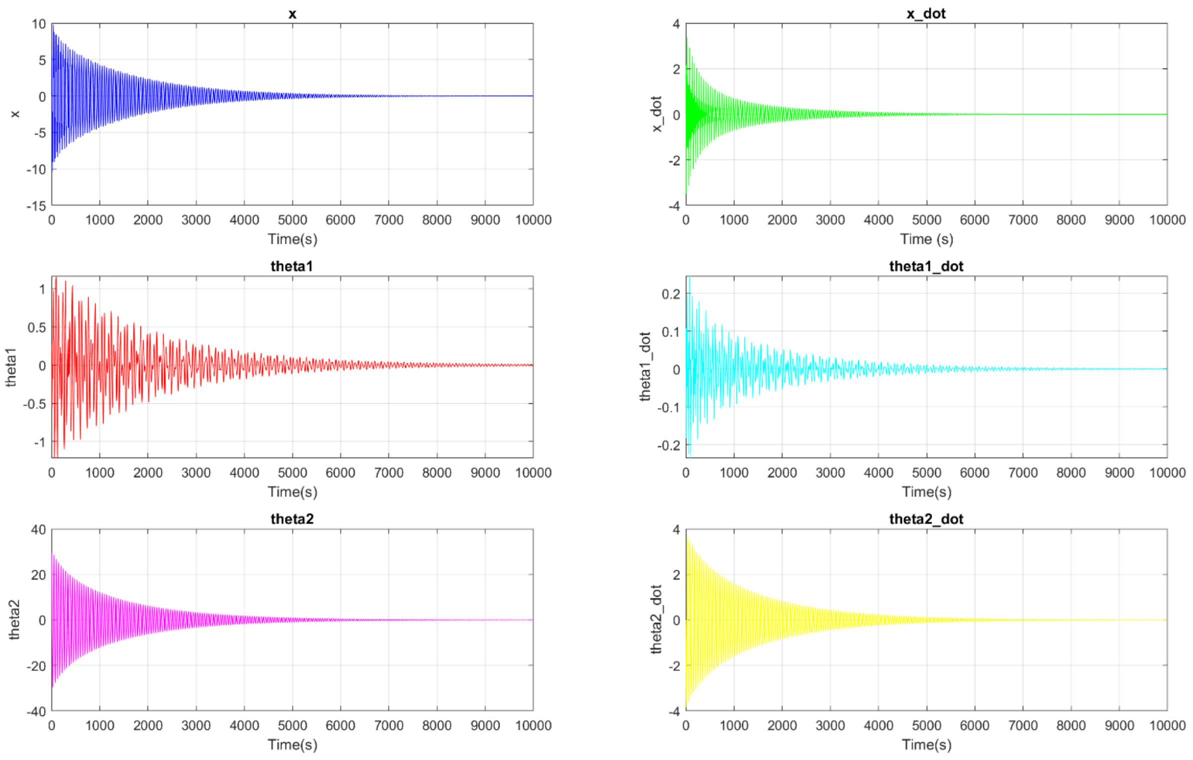


Figure 3: Result of Problem D (Non-Linear System)

Problem: Second Component

Consider the parameters selected in C) above.

E) Suppose that you can select the following output vectors: x(t), (θ 1(t), θ 2(t)), (x(t), θ 2(t)) or (x(t), θ 1(t), θ 2(t)). Determine for which output vectors the linearized system is observable.

F) Obtain your "best" Luenberger observer for each one of the output vectors for which the system is observable and simulate its response to initial conditions and unit step input. The simulation should be done for the observer applied to both the linearized system and the original nonlinear system.

G) Design an output feedback controller for your choice of the "smallest" output vector. Use the LQG method and apply the resulting output feedback controller to the original nonlinear system. Obtain your best design and illustrate its performance in simulation. How would you reconfigure your controller to asymptotically track a constant reference on x ? Will your design reject constant force disturbances applied on the cart ?

Solution:

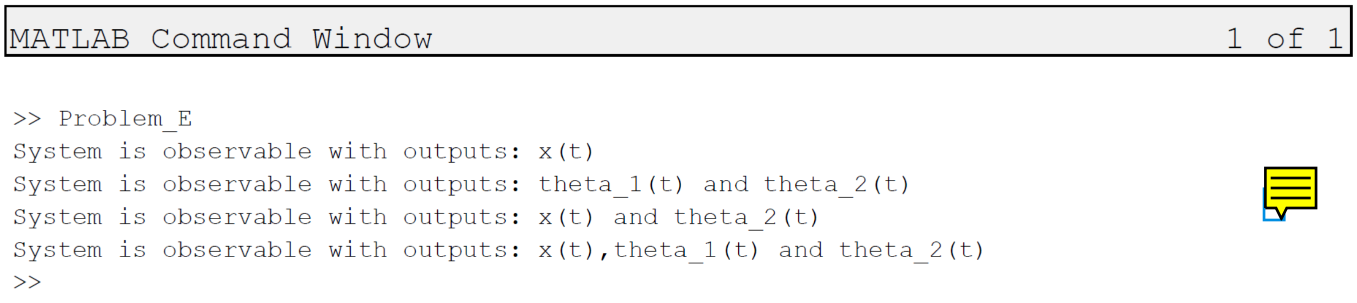
E) Observability

Code –

```
% Clearing the workspace and the command window
clc; clear;

% Defining system parameters
M = 1000;
mass_1 = 100;
mass_2 = 100;
length_1 = 20;
length_2 = 10;
g = 9.81;

% Defining state space matrices A and B with the assigned values
A = [0, 1, 0, 0, 0, 0;
     0, 0, -(mass_1*g)/M, 0, -(mass_2*g)/M, 0;
     0, 0, 0, 1, 0, 0;
     0, 0, -((M+mass_1)*g)/(M*length_1), 0, -(mass_2*g)/(M*length_1), 0;
     0, 0, 0, 0, 0, 1;
     0, 0, -(mass_1*g)/(M*length_2), 0, -(g*(M+mass_2))/(M*length_2), 0];
B = [0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];

% Defining different C matrices for various outputs
C_matrices = {[1, 0, 0, 0, 0, 0];                          % Defining for observing x component
              [0, 0, 1, 0, 0, 0; 1, 0, 0, 0, 0, 0];       % Defining for observing theta1 and
theta2
              [1, 0, 0, 0, 0, 0; 0, 0, 0, 0, 1, 0];       % Defining for observing x and theta2
              [1, 0, 0, 0, 0, 0; 1, 0, 1, 0, 0, 0; 0, 0, 0, 0, 1, 0]}; % Defining for observing x,
theta1, theta2
```

```
% Checking the observability for each C matrix
output_descriptions = {'x(t)', 'theta_1(t) and theta_2(t)', 'x(t) and theta_2(t)', 'x(t),
theta_1(t) and theta_2(t)'};
for i = 1:length(C_matrices)
    Observability_matrix = obsv(double(A), double(C_matrices{i})); % Calculating the observability
matrix
    if rank(Observability_matrix) == 6
        fprintf('System is observable with outputs: %s\n', output_descriptions{i});
    else
        fprintf('System is not observable with outputs: %s\n', output_descriptions{i});
    end
end
```

Output –



```
MATLAB Command Window                                              1 of 1

>> Problem_E
System is observable with outputs: x(t)
System is observable with outputs: theta_1(t) and theta_2(t)
System is observable with outputs: x(t) and theta_2(t)
System is observable with outputs: x(t),theta_1(t) and theta_2(t)
>>
```

F) Observer Design

The Luenberger Observer is written in state-space representation as:

$$\dot{\hat{X}}(t) = A\hat{x} + B_k U_k(t) + L(Y(t) - C\hat{x}(t))$$

Here, $\hat{x}(t)$ is state estimator, L is observer gain matrix, $Y(t) - C\hat{x}(t)$ is correction term and $\hat{x}(0)$ = 0.

The estimation error $X_e(t) = X(t) - \hat{X}(t)$ has the following state space representation:

$$\dot{X}_e(t) = \dot{X} - \dot{\hat{X}}(t)$$

$$\dot{X}_e(t) = AX_e(t) - L\big(Y(t) - C\hat{x}(t)\big) + B_d U_d(t)$$

Here, we assume D = 0, *Y(t)* = *Cx(t)*. Therefore, the equation can be written as

$$\dot{X}_e(t) = (A - LC)X_e(t) + B_d U_d(t)$$

Code –

```
% Linear
clc; clear;
```

```
% Defining system parameters
M = 1000;
mass_1 = 100;
mass_2 = 100;
length_1 = 20;
length_2 = 10;
g = 9.81;

% Constructing state space matrices
A = [0 1 0 0 0 0;
     0 0 -(mass_1*g)/M 0 -(mass_2*g)/M 0;
     0 0 0 1 0 0;
     0 0 -((M+mass_1)*g)/(M*length_1) 0 -(mass_2*g)/(M*length_1) 0;
     0 0 0 0 0 1;
     0 0 -(mass_1*g)/(M*length_2) 0 -(g*(M+mass_2))/(M*length_2) 0];

B = [0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];

D = 0;

% Setting LQR parameters
Q = diag([1000 100 1000 1000 100 100]);
R = 0.01;
[K, S, eigen_values] = lqr(A, B, Q, R);

% Defining observer output matrices and corresponding L matrices
C_matrices = {[1, 0, 0, 0, 0, 0];                         % Observing x component
              [0, 0, 1, 0, 0, 0; 1, 0, 0, 0, 0, 0];       % Observing theta1 and theta2
              [1, 0, 0, 0, 0, 0; 1, 0, 1, 0, 0, 0; 0, 0, 0, 0, 1, 0]}; % Observing x and theta
poles = -1:-1:-6;
L_matrices = arrayfun(@(i) place(A', C_matrices{i}', poles)', 1:length(C_matrices),
'UniformOutput', false);

% Setting initial conditions for the observer
initial_conditions = [10, 40, 10, 0, 60, 0, 0, 0, 0, 0, 0, 0];

% Determining the number of systems to analyze
number_of_systems = length(C_matrices);

% Choosing colors for each system
colors = ['b','g','r','c','m','y']; % Adjust as needed for more systems

% Constructing and analyzing each system
figure; % Creating a single figure window

for i = 1:number_of_systems
    A_q = [(A-B*K) B*K; zeros(size(A)) (A-L_matrices{i}*C_matrices{i})];
    B_q = [B; zeros(size(B))];
    C_q = [C_matrices{i} zeros(size(C_matrices{i}))];
    sys = ss(A_q, B_q, C_q, D);

    % Plotting initial response of the system
    subplot(number_of_systems, 2, 2*i-1);
    initial(sys, initial_conditions, colors(i));
    title(sprintf('Initial Response for System %d', i));
    grid on;

    % Plotting step response of the system
    subplot(number_of_systems, 2, 2*i);
    step(sys, colors(i));
    title(sprintf('Step Response for System %d', i));
    grid on;
end

% Adjusting the figure size for better visibility
set(gcf, 'Position', [100, 100, 1200, 800]);
```

Output –

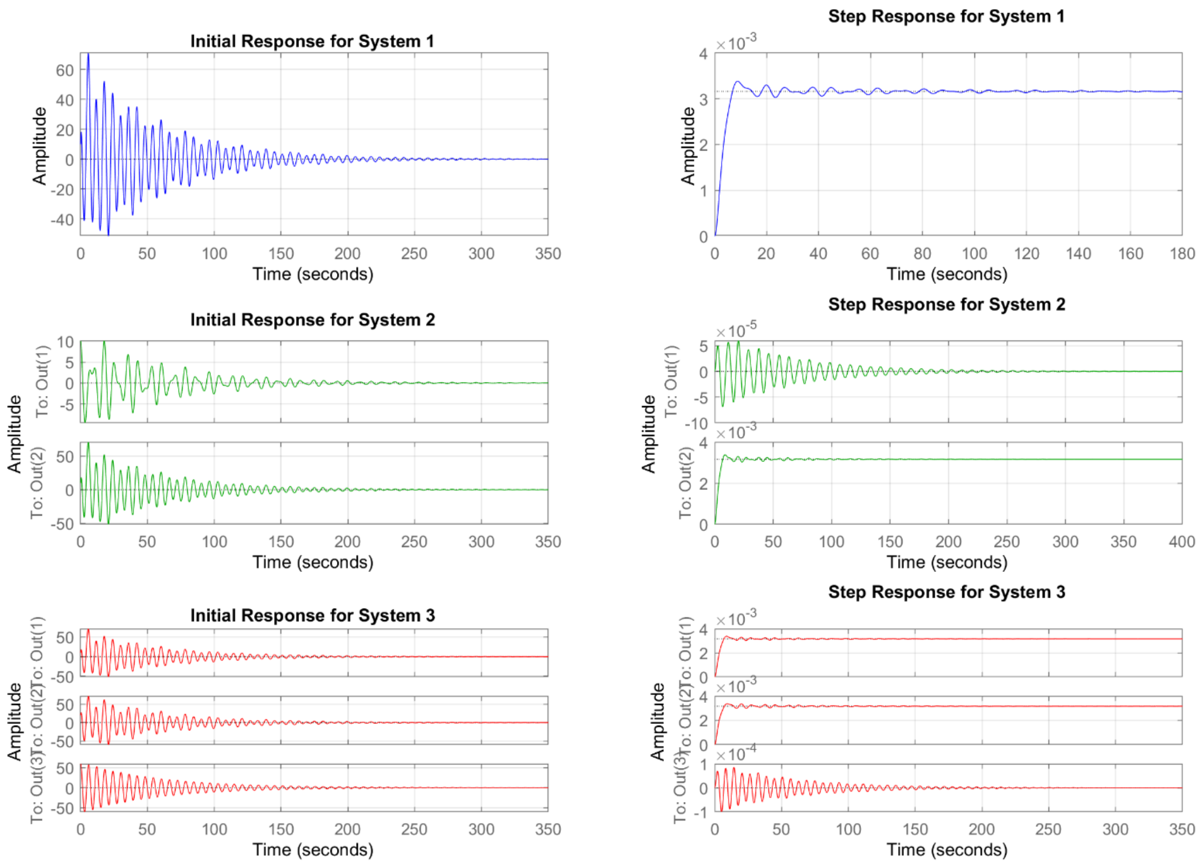


Figure 4: Result of Problem F (Linear System)

Code –

```
% Non-Linear
% Clearing previous outputs and variables
clc; clear;

% Defining system parameters
M = 1000;
mass_1 = 100;
mass_2 = 100;
length_1 = 20;
length_2 = 10;
```

```
g = 9.81;

% Constructing state space matrices
A = [0 1 0 0 0 0;
     0 0 -(mass_1*g)/M 0 -(mass_2*g)/M 0;
     0 0 0 1 0 0;
     0 0 -((M+mass_1)*g)/(M*length_1) 0 -(mass_2*g)/(M*length_1) 0;
     0 0 0 0 0 1;
     0 0 -(mass_1*g)/(M*length_2) 0 -(g*(M+mass_2))/(M*length_2) 0];
B = [0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];

% Defining output matrices for observable configurations
C_matrices = {[1, 0, 0, 0, 0, 0];                           % Observing x component
              [0, 0, 1, 0, 0, 0; 1, 0, 0, 0, 0, 0];       % Observing theta1 and theta2
              [1, 0, 0, 0, 0, 0; 1, 0, 1, 0, 0, 0; 0, 0, 0, 0, 1, 0]};   % Observing x and theta2
D = 0; % Defining D matrix as zero

% Setting LQR parameters
Q = diag([1000 100 1000 1000 1000 1000]);
R = 0.01;
[K, S, eigen_values] = lqr(A, B, Q, R); % Computing LQR gain matrix

% Setting initial conditions for the observer
initial_conditions = [0, 0, 30, 0, 60, 0, 0, 0, 0, 0, 0, 0];

% Defining desired poles for the observer
poles = [-1; -2; -3; -4; -5; -6];

% Choosing colors for each system
colors = ['b','g','r','c','m','y']; % Adjusting as needed for more systems

% Constructing and analyzing systems for each observer configuration
num_of_systems = length(C_matrices);
figure; % Creating a single figure window for all plots

for i = 1:num_of_systems
    C = C_matrices{i};
    L = place(A', C', poles)'; % Computing observer gain matrix

    % Formulating Luenberger state-space matrices
    A_q = [(A-B*K) B*K; zeros(size(A)) (A-L*C)];
    B_q = [B; zeros(size(B))];
    C_q = [C zeros(size(C))];

    % Creating state-space model
    sys = ss(A_q, B_q, C_q, D);

    % Plotting initial response (Left column)
    subplot(num_of_systems, 2, 2*i-1);
    initial(sys, initial_conditions, colors(i));
    title(sprintf('Initial Response for System %d', i));
    grid on;

    % Plotting step response (Right column)
    subplot(num_of_systems, 2, 2*i);
    step(sys, colors(i));
    title(sprintf('Step Response for System %d', i));
    grid on;
end

% Adjusting the figure size for better visibility
set(gcf, 'Position', [100, 100, 1200, num_of_systems*200]);
```

Output –

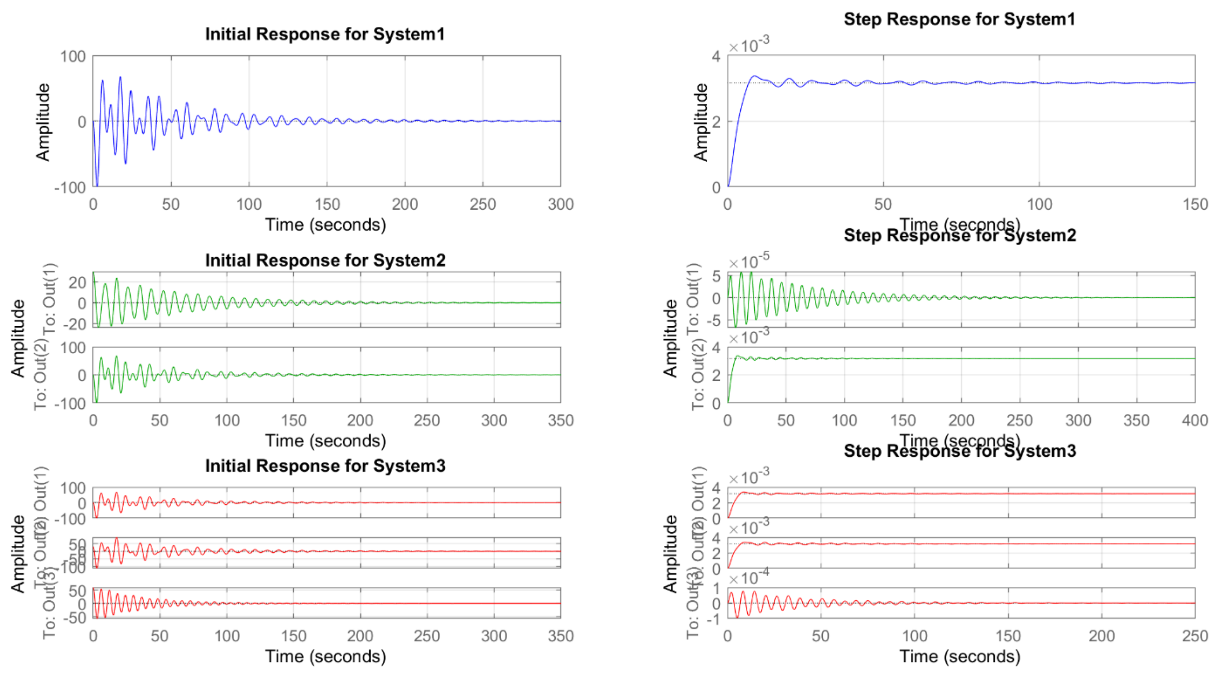


Figure 5: Result of Problem F (Non-Linear System)

G) LQG Design

Code –

```
% Clearing previous outputs and variables
clc; clear;

% Defining system parameters
M = 1000;
mass_1 = 100;
mass_2 = 100;
length_1 = 20;
length_2 = 10;
g = 9.81;

% Constructing state space matrices
A = [0 1 0 0 0 0;
     0 0 -(mass_1*g)/M 0 -(mass_2*g)/M 0;
     0 0 0 1 0 0;
     0 0 -((M+mass_1)*g)/(M*length_1) 0 -(mass_2*g)/(M*length_1) 0;
     0 0 0 0 0 1;
     0 0 -(mass_1*g)/(M*length_2) 0 -(g*(M+mass_2))/(M*length_2) 0];

B = [0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];

% Defining LQR parameters
Q = diag([100 100 100 100 100 100]);
R = 0.001;
```

```
% Defining observable output matrices
C_matrices = {[1, 0, 0, 0, 0, 0];                         % Observing x component
              [0, 0, 1, 0, 0, 0; 1, 0, 0, 0, 0, 0];       % Observing theta1 and theta2
              [1, 0, 0, 0, 0, 0; 1, 0, 1, 0, 0, 0; 0, 0, 0, 0, 1, 0]};
D = 0;

% Setting initial conditions for the Luenberger Observer
initial_conditions = [4; 0; 30; 0; 60; 0; 0; 0; 0; 0; 0; 0];

% Designing LQR and Kalman Filter
[K, S, eigen_values] = lqr(A, B, Q, R);
v_d = 0.3 * eye(6);
v_n = 1;

% Choosing colors for each plot
colors = {'b', 'r', 'g', 'c', 'm', 'y', 'k'};

% Analyzing systems for each observable configuration
num_of_systems = length(C_matrices);

% Creating a single figure window
figure;

% Looping through each configuration
for i = 1:num_of_systems
    C = C_matrices{i};
    K_pop = lqr(A', C', v_d, v_n)';

    % Constructing state-space model
    sys = ss([(A-B*K) B*K; zeros(size(A)) (A-K_pop*C)], [B; zeros(size(B))], [C zeros(size(C))],
D);

    % Plotting initial response
    subplot(num_of_systems, 2, 2*i-1);
    initial(sys, initial_conditions, colors{i});
    title(sprintf('Initial Response for System %d', i));

    % Plotting step response
    subplot(num_of_systems, 2, 2*i);
    step(sys, colors{i});
    title(sprintf('Step Response for System %d', i));
end

grid on;

% Adjusting the figure size for better visibility
set(gcf, 'Position', [100, 100, 1200, 800]);
```

Output –

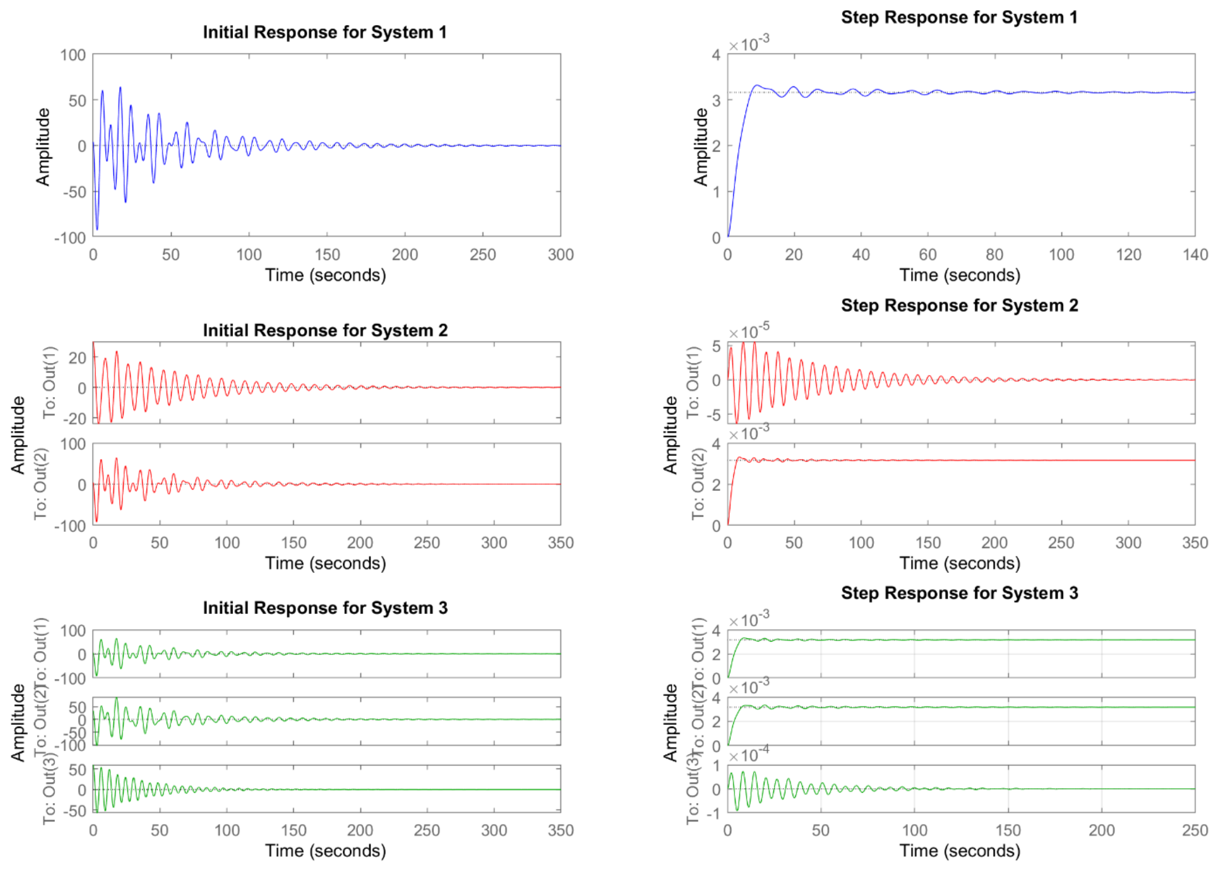


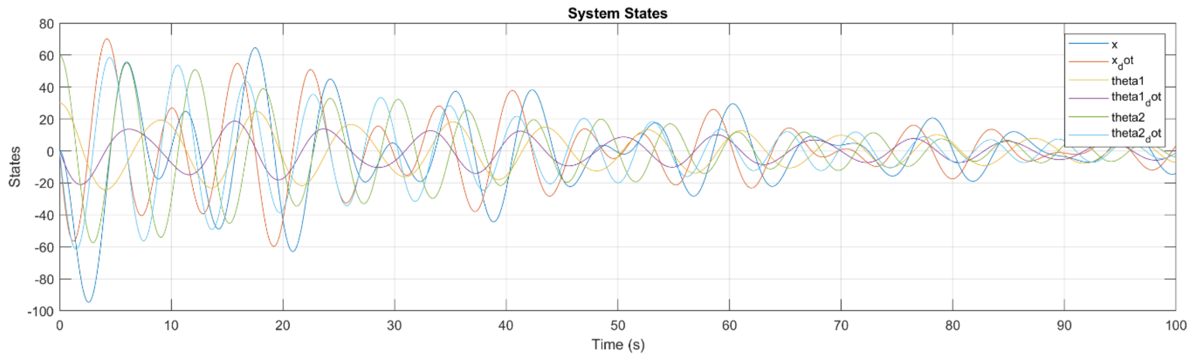
Figure 6: Result of problem G

Code –

```
% Clearing all previous outputs
clc; clear;

% Setting initial conditions and time span for the simulation
initial_conditions = [0; 0; 30; 0; 60; 0; 0; 0; 0; 0; 0; 0];
time_span = 0:0.1:100;

% Solving the system using the ode45 solver
[t, x] = ode45(@twoload_lqg, time_span, initial_conditions);

% Plotting the results in a single window
figure;
subplot(2, 1, 1); % Upper subplot for the system states
plot(t, x(:, 1:6));
title('System States');
xlabel('Time (s)');
ylabel('States');
legend('x', 'x_dot', 'theta1', 'theta1_dot', 'theta2', 'theta2_dot');
grid on;
```

```
subplot(2, 1, 2); % Lower subplot for the estimated states
plot(t, x(:, 7:12));
title('Estimated States');
xlabel('Time(s)');
ylabel('Estimates');
legend('x_hat', 'x_dot_hat', 'theta1_hat', 'theta1_dot_hat', 'theta2_hat', 'theta2_dot_hat');
grid on;

% Defining the twoload_lqg function for nonlinear LQG control
function dy_dt = twoload_lqg(t, y)
    % Defining system parameters
    M = 1000;
    mass_1 = 100;
    mass_2 = 100;
    length_1 = 20;
    length_2 = 10;
    g = 9.81;

    % Constructing state space matrices
    A = [0 1 0 0 0 0;
         0 0 -(mass_1*g)/M 0 -(mass_2*g)/M 0;
         0 0 0 1 0 0;
         0 0 -((M+mass_1)*g)/(M*length_1) 0 -(mass_2*g)/(M*length_1) 0;
         0 0 0 0 0 1;
         0 0 -(mass_1*g)/(M*length_2) 0 -(g*(M+mass_2))/(M*length_2) 0];
    B = [0; 1/M; 0; 1/(M*length_1); 0; 1/(M*length_2)];
    C_matrix = [1 0 0 0 0 0]; % Defining output measurement for x component

    % Designing LQR parameters
    Q = diag([1000 1000 100 10 1000 100]);
    R = 0.01;
    [K, S, eigen_values] = lqr(A, B, Q, R);
    controlled_force = -K * y(1:6); % Calculating control input

    % Designing Kalman filter
    v_d = 0.3 * eye(6);
    v_n = 1;
    K_pop = lqr(A', C_matrix', v_d, v_n)'; % Calculating observer gain

    % Formulating estimator dynamics
    dx_hat = A * y(7:12) + B * controlled_force + K_pop * (C_matrix * y(1:6) - C_matrix *
y(7:12));

    % Formulating system dynamics (controlled system)
    dx = A * y(1:6) + B * controlled_force;

    % Concatenating the system dynamics and observer dynamics
    dy_dt = [dx; dx_hat];
end
```

Output –



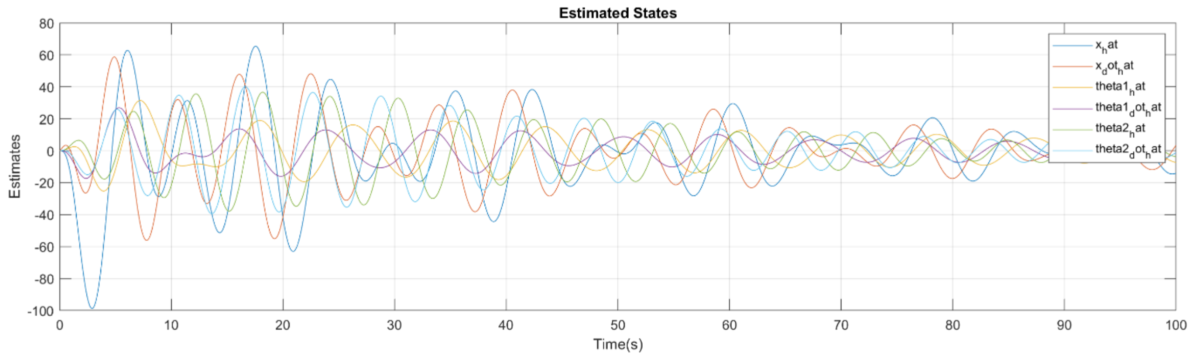


Figure 7: Result of Problem G