Control of Robotic Systems (ENPM667)

Final Project Report

Lowell Lobo (120095719)
Vikram Setty (119696897)
Wei-Li Chen (120378508)

A. James Clark School of Engineering
University of Maryland
College Park - 20742

December 18, 2023

Contents

1 Introduction

This project report contains the analysis, observations, and results made and discovered as a part of the final project of the course *Control of Robotic Systems (ENPM667)* at the University of Maryland.

In this project, we analyze a system containing a crane having two attached dangling pendulums moving in a straight line along a frictionless surface. This system can be visualized using the pictorial representation shown in Figure 1. Throughout the following sections, the system's equations of motion are modeled after which the system is linearized, and checked for controllability, and observability. Further, a LQR controller, Luenberger Observer, and even a LQG controller are designed for the system and an analysis of the performance of these different techniques on driving the system's steady-state response is made.

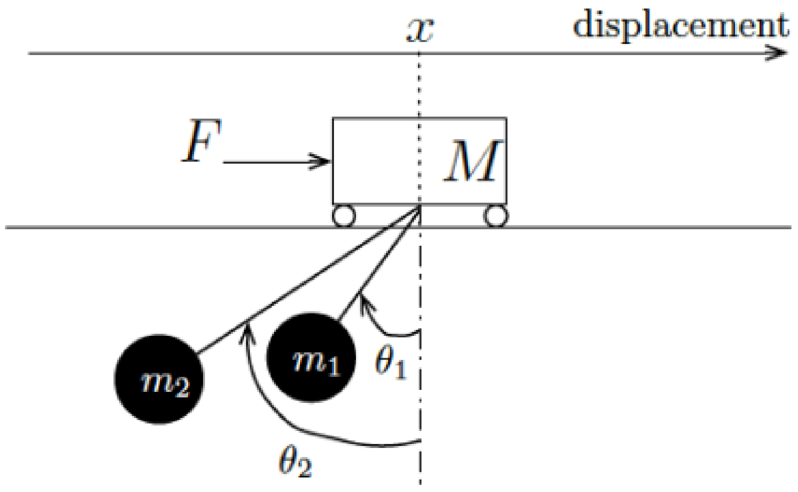


Figure 1: Given System

2 A: System Modeling

To model the equations of motion that govern the movement of the cart-pendulum assembly, it is first necessary to establish a global coordinate system with respect to which all distances and angles are calculated. For the chosen system, we use the same coordinate system partially described in Figure 1. To expand on this frame of reference, we assume the origin of the global coordinate reference frame to be situated at the center of the bottom of the crane (from where both the pendulums hang) at the initial moment before any motion (time t=0). The X-axis of the global coordinate frame extends towards the right side horizontally and the Y-axis goes vertically upwards (in the opposite direction of gravity). This coordinate frame depiction, along with the direction of the acceleration due to the gravity vector is shown more clearly in 2. Further, we also assume the center of mass of the crane (not including the two pendulums) to be present at the origin of the global coordinate frame in the initial state (t=0).

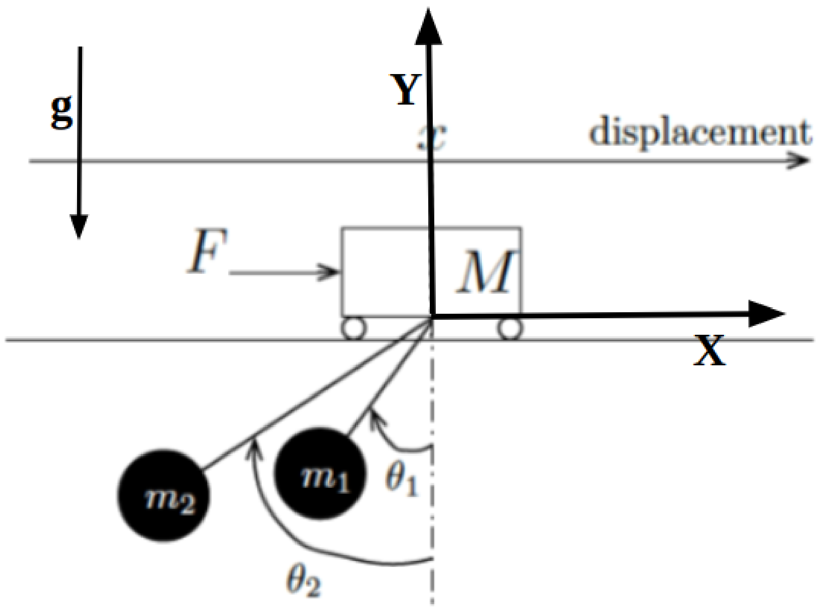


Figure 2: Global Coordinate Frame at the initial state (t=0)

For this depiction of the crane-pendulum system, the three independent variables/parameters that completely define and uniquely identify the movement/state of the system include the horizontal displacement of the crane (x), the vertical angle of the first pendulum (θ_1), and the vertical angle of the second pendulum (θ_2). These three parameters make up the three degrees if freedom of the crane-pendulum system. All these three parameters are functions of time and define the state of the system. Further, it is also possible to represent the general coordinates of each part of the system using these parameters as a function of time.

The position of the crane (of mass M) can be written as shown below.

$$\vec{X_M} = x + 0$$

Similarly, the general position of the two pendulums (of mass m_1 and m_2 and lengths l_1 and l_2) can be written as follows.

$$\vec{X_{m_1}} = (x - l_1 \sin\theta_1) - l_1 \cos\theta_1$$
$$\vec{X_{m_2}} = (x - l_2 \sin\theta_2) - l_2 \cos\theta_2$$

By differentiating these expressions with time, we can obtain the velocities of the crane and the two pendulums.

$$\dot{\vec{X_M}} = \dot{x} + 0$$
$$\dot{\vec{X_{m_1}}} = (\dot{x} - l_1 \cos\theta_1(\dot{\theta_1})) + l_1 \sin\theta_1(\dot{\theta_1})$$
$$\dot{\vec{X_{m_2}}} = (\dot{x} - l_2 \cos\theta_2(\dot{\theta_2})) + l_2 \sin\theta_2(\dot{\theta_2})$$

We can use these expressions to find the equations of motion of the system using the Euler-Lagrange Method. The first step of this method involves formulating the expressions for the potential and kinetic energy of the system. The total kinetic energy of the system ($\mathcal{K}$) expressed as a sum of the individual kinetic energies of each sub-system is listed and simplified as shown below.

$$\mathcal{K} = \frac{1}{2}(\dot{x_M})^T M(\dot{x_M}) + \frac{1}{2}(\dot{x_{m_1}})^T m_1(\dot{x_{m_1}}) + \frac{1}{2}(\dot{x_{m_2}})^T m_2(\dot{x_{m_2}})$$

$$\mathcal{K} = \frac{1}{2}(M + m_1 + m_2)\dot{x}^2 + \frac{1}{2}m_1 l_1 \dot{\theta_1}^2 + \frac{1}{2}m_2 l_2 \dot{\theta_2}^2 - (m_1 l_1 \dot{\theta}_1 \cos\theta_1 + m_2 l_2 \dot{\theta}_2 \cos\theta_2)\dot{x}$$

Using a similar formulation, the expression for the potential energy of the system ($\mathcal{P}$) expressed as a sum of the potential energy of the crane and the two pendulums is shown below.

$$\mathcal{P} = (x_M^T)Mg \begin{bmatrix} 0 \\ 1 \end{bmatrix} + (x_{m_1}^T)m_1 g \begin{bmatrix} 0 \\ 1 \end{bmatrix} + (x_{m_2}^T)m_2 g \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathcal{P} = -mgl_1 \cos\theta_1 - mgl_2 \cos\theta_2$$

Using these expressions for the total kinetic and potential energy of the system, we can now calculate the Lagrangian (L) of the system that comes out to be as shown below.

$$\mathcal{L} = \mathcal{K} - \mathcal{P}$$

$$\mathcal{L} = \frac{1}{2}(M+m_1+m_2)\dot{x}^2 + \frac{1}{2}m_1 l_1 \dot{\theta_1}^2 + \frac{1}{2}m_2 l_2 \dot{\theta_2}^2 - (m_1 l_1 \dot{\theta}_1 \cos\theta_1 + m_2 l_2 \dot{\theta}_2 \cos\theta_2)\dot{x} + mgl_1 \cos\theta_1 + mgl_2 \cos\theta_2$$

According to the Euler-Lagrange Formulation, the following Lyapunov Equation holds to obtain the equation of motion for a general coordinate of the system (with Q being the general force for that coordinate).

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = Q(t)$$

Since we have three degrees of freedom represented by three independent variables (x, θ_1, *and* θ_2), we would get three Lyapunov Equations that fully describe the behavior of the system. The simplification of the Lyapunov equation for the variable x is shown below.

$$\frac{\partial \mathcal{L}}{\partial \dot{x}} = (M + m_1 + m_2)\dot{x} - m_1 l_1 \dot{\theta}_1 \cos\theta_1 - m_2 l_2 \dot{\theta}_2 \cos\theta_2$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) = (M + m_1 + m_2)\ddot{x} - m_1 l_1(\ddot{\theta}_1 \cos\theta_1 - (\dot{\theta_1}^2)\sin\theta_1) - m_2 l_2(\ddot{\theta}_2 \cos\theta_2 - (\dot{\theta_2}^2)\sin\theta_2)$$

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

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

$$(M + m_1 + m_2)\ddot{x} - m_1 l_1(\ddot{\theta}_1 \cos\theta_1 - (\dot{\theta_1}^2)\sin\theta_1) - m_2 l_2(\ddot{\theta}_2 \cos\theta_2 - (\dot{\theta_2}^2)\sin\theta_2) = F(t)$$

We notice that the generalized force for the x dimension is *F(t)* because of the external force indicated in 1. There is however no generalized force for the θ_1 and θ_2 dimensions. The Lyapunov Equation formulation and simplification for θ_1 is shown below.

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = m_1(l_1^2)(\ddot{\theta_1}^2) - m_1 l_1 \dot{x}\cos\theta_1$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \theta_1}\right) = m_1(l_1^2)\ddot{\theta}_1 - m_1 l_1 \ddot{x}\cos\theta_1 + m_1 l_1 \dot{x}\dot{\theta}_1 \sin\theta_1$$

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

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

$$m_1(l_1^2)\ddot{\theta}_1 - m_1 l_1 \cos\theta_1 \ddot{x} + m_1 l_1 g \sin\theta_1 = 0$$

Similarly, the Lyapunov Equation can be applied to θ_2 as well.

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = m_2(l_2^2)(\ddot{\theta_2}^2) - m_2 l_2 \dot{x}\cos\theta_2$$

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial \mathcal{L}}{\partial \theta_2}\right) = m_2(l_2^2)\ddot{\theta}_2 - m_2 l_2 \ddot{x}\cos\theta_2 + m_2 l_2 \dot{x}\dot{\theta}_2 \sin\theta_2$$

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

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

$$m_2(l_2^2)\ddot{\theta}_2 - m_2 l_2 \cos\theta_2 \ddot{x} + m_2 l_2 g \sin\theta_2 = 0$$

The final expressions from the three Lyapunov Equations can be rewritten as shown below to give the expressions for the acceleration for each generalized coordinate (x, θ_1, and θ_2).

$$\ddot{x} = \frac{F(t) + m_1 l_1(\ddot{\theta}_1 \cos\theta_1 - (\dot{\theta_1}^2)\sin\theta_1) + m_2 l_2(\ddot{\theta}_2 \cos\theta_2 - (\dot{\theta_2}^2)\sin\theta_2)}{M + m_1 + m_2}$$

$$\ddot{\theta}_1 = \frac{\ddot{x}\cos\theta_1 - g\sin\theta_1}{l_1}$$

$$\ddot{\theta}_2 = \frac{\ddot{x}\cos\theta_2 - g\sin\theta_2}{l_2}$$

Thus, with these three equations, the crane-pendulum system is completely modeled and motion fully described. Using these expressions, the systems can be written in a non-linear state-space matrix representation displayed below (with $\vec{X}$ representing the state of the system).

$$\vec{X} = \begin{bmatrix} x \\ \dot{x} \\ \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\dot{\vec{X}} = \begin{bmatrix} \dot{x} \\ \frac{F(t)+m_1 l_1(\ddot{\theta}_1 \cos\theta_1 - \dot{\theta_1}^2 \sin\theta_1) + m_2 l_2(\ddot{\theta}_2 \cos\theta_2 - \dot{\theta_2}^2 \sin\theta_2)}{M+m_1+m_2} \\ \dot{\theta}_1 \\ \frac{\ddot{x}\cos\theta_1 - g\sin\theta_1}{l_1} \\ \dot{\theta}_2 \\ \frac{\ddot{x}\cos\theta_2 - g\sin\theta_2}{l_2} \end{bmatrix}$$

3 B: System Linearization

As shown in Section 2, the system obtained is a non-linear function of the chosen state variables. However, a linear system is a lot more versatile in terms of ways it can be observed, controlled, stabilized, and understood. That is why it is crucial to linearize the system. Often, linearization around the equilibrium point is a preferred method as it guarantees lower state velocities and a higher chance of stabilization (by finding an appropriate Lyapunov Function) in most cases.

For our system, the point x*=0,* θ_1*=0,* θ_2*=0,* $\dot{x} = 0$, $\dot{\theta_1} = 0$, θ_2*=0* is an equilibrium point. This can be proved by finding each individual state variable velocity at the equilibrium point. The calculation shown below covers this verification.

$$\dot{\vec{X}}_{M_{eq}} = \dot{x} + 0 = 0 + 0$$

$$\dot{\vec{X}}_{m_{1eq}} = (\dot{x} - l_1 \cos\theta_1(\dot{\theta_1})) + l_1 \sin\theta_1(\dot{\theta_1}) = 0 + 0$$

$$\dot{\vec{X}}_{m_{2eq}} = (\dot{x} - l_2 \cos\theta_2(\dot{\theta_2})) + l_2 \sin\theta_2(\dot{\theta_2}) = 0 + 0$$

Thus it is proved that the following state is an equilibrium point.

$$\vec{X_{eq}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

To linearize the system around this equilibrium point, a variety of methods could be used. The most common is a method called Jacobian Linearization wherein the Jacobian of the velocity functions of the non-linear states are constructed from where the linearized state is obtained. However for our current system, the velocity functions are quadratic and Jacobian Linearization though possible, is quite complex. A much simpler and better alternative is to use the small-angle approximation. In the small-angle approximation technique, when an angle approaches zero, we can approximate with marginal error the sinusoidal and tangent of the angle to be the angle value itself and the cosine and cotangent of the angle to be 1. Further, with minimal error, we can approximate the higher order powers (>2) of any quantity near zero to be zero. Thus, with all state variables equal to zero at our equilibrium point, we can assume the following using the small-angle approximation.

$$\sin\theta_1 \approx \theta_1 \;\; as \;\; \theta_1 \to 0$$

$$\sin\theta_2 \approx \theta_2 \;\; as \;\; \theta_2 \to 0$$

$$\cos\theta_1 = \cos\theta_2 \approx 1 \;\; as \;\; \theta_1 \to 0 \;\; and \;\; \theta_2 \to 0$$

$$\dot{\theta_1}^2 = \dot{\theta_2}^2 \approx 0 \;\; as \;\; \dot{\theta_1} \to 0 \;\; and \;\; \dot{\theta_2} \to 0$$

Using these approximations, the expressions for the accelerations for each of the generalized coordinates ($\ddot{x}$, $\ddot{\theta_1}$, and $\ddot{\theta_2}$) can be rewritten as shown below.

$$\ddot{x} = \frac{F(t) + m_1 l_1(\ddot{\theta_1} + m_2 l_2(\ddot{\theta_2}}{M + m_1 + m_2}$$

$$\ddot{\theta_1} = \frac{\ddot{x} - g\theta_1}{l_1}$$

$$\ddot{\theta_2} = \frac{\ddot{x} - g\theta_2}{l_2}$$

Further, by rearranging the previous equations, we can write the expressions for he accelerations for each of the generalized coordinates ($\ddot{x}$, $\ddot{\theta_1}$, and $\ddot{\theta_2}$) in an even more simplified form as depicted below.

$$\ddot{x} = -\frac{m1}{M} g\theta_1 - \frac{m2}{M} g\theta_2 + \frac{F(t)}{M}$$
$$\ddot{\theta_1} = -\frac{M+m_1}{Ml_1} g\theta_1 - \frac{m_2}{Ml_1} g\theta_2 + \frac{F(t)}{Ml_1}$$
$$\ddot{\theta_2} = -\frac{M+m_2}{Ml_2} g\theta_2 - \frac{m_1}{Ml_2} g\theta_1 + \frac{F(t)}{Ml_2}$$

Substituting these expressions into the original system's state update equation, we get a new matrix representation as depicted below.

$$\dot{\vec{X}} = \begin{bmatrix} \dot{x} \\ -\frac{m1}{M} g\theta_1 - \frac{m2}{M} g\theta_2 + \frac{F(t)}{M} \\ \dot{\theta_1} \\ -\frac{M+m_1}{Ml_1} g\theta_1 - \frac{m_2}{Ml_1} g\theta_2 + \frac{F(t)}{Ml_1} \\ \dot{\theta_2} \\ -\frac{m_1}{Ml_2} g\theta_1 - \frac{M+m_2}{Ml_2} g\theta_2 + \frac{F(t)}{Ml_2} \end{bmatrix}$$

Thus, with the system now linearized, we can write the crane-pendulum system as a linear-time-invariant (LTI) state equation as shown below.

$$\dot{\vec{X}} = A\vec{X} + B\vec{U}$$
$$\vec{Y} = C\vec{X} + D\vec{U}$$

In the equation written above, A, B, C, and D are time-invariant matrices. Further, $\vec{U}$ corresponds to the input vector. In the context of this system, we consider a single input that is the force applied on the crane ($F(\vec{t})$). $\vec{Y}$ corresponds to the output, and the vectors C and D depend on the dimensions of and the chosen quantities to represent the output. Considering the above equation, we can write the matrices A and B as shown below.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{m1}{M} g & 0 & -\frac{m2}{M} g & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{M+m_1}{Ml_1} g & 0 & -\frac{m_2}{Ml_1} g & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{m_1}{Ml_2} g & 0 & -\frac{M+m_2}{Ml_2} g & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml_1} \\ 0 \\ \frac{1}{Ml_2} \end{bmatrix}$$

The expressions for the C and D matrices depend on the chosen output and will be described in more detail in further sections. Also, note that with the exception of angles, angular velocities, and angular accelerations being expressed in terms of degrees, all other quantities are written and expressed in their S.I. units.

4 C: System Controllability Check

A system is said to be controllable if there exists a control signal to take the system from its initial state to any desired state in a finite time interval. Controllability depends on the internal dynamics of a system, majorly on how the state velocity changes with the present state and control input signal which is represented by the A and B matrices respectively (or a state transition matrix and B matrix for a linear-time-varying (LTV) case).

There exist multiple ways to test the controllability of a system. A popular method includes using the Rank Test. In this method, we check if the controllability matrix (whose expression is given below) has full rank. If it does, the system is controllable Otherwise, it isn't.

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

If C has full rank, the system is controllable. i.e.

$$rack(C) = n$$

where n is the number of states/the dimension of the state vector ($\vec{X}$).

For our crane-pendulum assembly system, the parametric controllability matrix is

$$\begin{pmatrix} 0 & \frac{1}{M} & 0 & \sigma_2 & 0 & \sigma_1 \\ \frac{1}{M} & 0 & \sigma_2 & 0 & \sigma_1 & 0 \\ 0 & \frac{1}{M\,l_1} & 0 & \sigma_6 & 0 & \sigma_4 \\ \frac{1}{M\,l_1} & 0 & \sigma_6 & 0 & \sigma_4 & 0 \\ 0 & \frac{1}{M\,l_2} & 0 & \sigma_5 & 0 & \sigma_3 \\ \frac{1}{M\,l_2} & 0 & \sigma_5 & 0 & \sigma_3 & 0 \end{pmatrix}$$

where

$$\sigma_1 = \frac{\frac{g^2\,m_1\,(M+m_1)}{M^2\,l_1} + \frac{g^2\,m_1\,m_2}{M^2\,l_2}}{M\,l_1} + \frac{\frac{g^2\,m_2\,(M+m_2)}{M^2\,l_2} + \frac{g^2\,m_1\,m_2}{M^2\,l_1}}{M\,l_2}$$

$$\sigma_2 = -\frac{g\,m_1}{M^2\,l_1} - \frac{g\,m_2}{M^2\,l_2}$$

$$\sigma_3 = \frac{\frac{g^2\,m_1\,(M+m_2)}{M^2\,{l_2}^2} + \frac{g^2\,m_1\,(M+m_1)}{\sigma_7}}{M\,l_1} + \frac{\frac{g^2\,(M+m_2)^2}{M^2\,{l_2}^2} + \frac{g^2\,m_1\,m_2}{\sigma_7}}{M\,l_2}$$

$$\sigma_4 = \frac{\frac{g^2\,m_2\,(M+m_1)}{M^2\,{l_1}^2} + \frac{g^2\,m_2\,(M+m_2)}{\sigma_7}}{M\,l_2} + \frac{\frac{g^2\,(M+m_1)^2}{M^2\,{l_1}^2} + \frac{g^2\,m_1\,m_2}{\sigma_7}}{M\,l_1}$$

$$\sigma_5 = -\frac{g\,(M+m_2)}{M^2\,{l_2}^2} - \frac{g\,m_1}{\sigma_7}$$

$$\sigma_6 = -\frac{g\,(M+m_1)}{M^2\,{l_1}^2} - \frac{g\,m_2}{\sigma_7}$$

$$\sigma_7 = M^2\,l_1\,l_2$$

To check if the system is controllable, we need to ensure this matrix has full rank. We observe that the controllability matrix is a square matrix (a result of having a single input), and thus the full-rank condition is ensured by checking for a non-zero determinant. The determinant of the controllability matrix is represented as a parametric expression shown below.

$$|C| = -\frac{g^6\,{l_1}^2 - 2\,g^6\,l_1\,l_2 + g^6\,{l_2}^2}{M^6\,{l_1}^6\,{l_2}^6}$$

This expression can be simplified to get the one as follows.

$$|C| = -\frac{g^6\,(l_1 - l_2)^2}{M^6\,{l_1}^6\,{l_2}^6}$$

It is evident that the determinant of the controllability matrix would become zero (signifying non-full rank making the system uncontrollable) only in the case when $l_1 = l_2$. In all other cases/combinations of $M, m_1, m_2, l_1,$ *and* l_2, the system is controllable.

Thus, we can conclude that for the system to be controllable, l_1 must not be equal to l_2.

5 D: LQR Controller Design

A common goal of most systems is to drive the system to converge as soon as possible while also using less actuator effort. It is a trade-off that poses control design changes varying from system to system. The LQR (Linear Quadratic Regulator) controller is a technique that is based on state feedback. the LQR controller chooses the optimal gain matrix (K) by minimizing the cost function $(J(K, X(0)))$ based on the chosen and tuned Q and R matrices that trade-off state convergence and actuator effort. The matrix Q penalizes poor performance (a long time for state convergence) represented by the state magnitude $(\vec{X})$, whereas the matrix R penalizes actuator effort, represented by the magnitude of the control signal $(\vec{U})$.

The cost function of the LQR controller $(J(K, X(0)))$ is a convex function and can be expressed as shown below.

$$J(K, X(0)) = \int_0^\infty X(t)^T Q X(t) + U_k(t)^T R U_k(t)\,dt$$

The optimal gain matrix (K) that minimizes the cost function $(J(K, X(0)))$ is given below.

$$K = R^{-1} B_k^T P$$

In the expression for the optimal gain matrix (K), the positive definite matrix P satisfies the Lyapunov Equation shown below.

$$A^T P + PA - PB_k R^{-1} B_k P = -Q$$

5.1 Parameter Tuning and Controllability Check

To test the LQR controller, we must choose a set of values for the parameters of the system (masses and pendulum lengths) along with a reasonable initial condition. The set of chosen parameters and initial condition of the state $(\vec{X})$ are shown below.

- M = 1000 kg
- m_1 = 100 kg
- m_2 = 100 kg
- l_1 = 20 m
- l_2 = 10 m
- $g = 9.8\ m/s^2$

- $\vec{X(0)} = \begin{bmatrix} 0 \\ 0.1 \\ 10 \\ 0.2 \\ 5 \\ 0.3 \end{bmatrix}$

Because we are choosing l_1 and l_2 to be different, **the system is guaranteed to be controllable** as proved in the previous section. Matlab also affirms that the selection is controllable. Note that once again, all units in the initial state matrix are listed in S.I. units except for angle quantities, which are mentioned in degrees.

After iterative fine-tuning, reasonably good choices for the matrices Q and R were chosen as shown below.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

$$R = 0.0001$$

Under these set values for the Q and R matrices, the expressions for the optimal gain matrix (K), and the positive definite matrix P come out as follows.

$$K = \begin{bmatrix} 100 & 1116.5 & 227.5594 & 67.0093 & 95.5984 & -38.9669 \end{bmatrix}$$

$$P = \begin{bmatrix} 11.1651 & 12.3298 & 0.6701 & -24.6441 & -0.3897 & -10.9755 \\ 12.3298 & 137.6343 & 32.1257 & -273.7860 & 6.6248 & -122.9401 \\ 0.6701 & 32.1257 & 2780.7 & -49.7755 & -16.6185 & -68.8101 \\ -24.6441 & -273.7860 & -49.7755 & 5658.4 & 68.5490 & -24.3122 \\ -0.3897 & 6.6248 & -16.6185 & 68.5490 & 1174.3 & -4.9241 \\ -10.9755 & -122.9401 & -68.8101 & -24.3122 & -4.9241 & 1202.6 \end{bmatrix}$$

5.2 Output Response Simulation

Using the optimal gain matrix (K), we can set the input control signal that is the force on the crane as $\vec{F} = K\vec{X}$ to enable state feedback control. Further, for now, we can set the output to be the same as the state vector ($\vec{X}$) to enable complete state feedback with zero disturbances. In such a case, the C and D matrices of the linearized system turn out as shown below.

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

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

The original state response of the system is shown in Figure 3. As expected the outputs are noisy and don't show any convergence. Using LQR control with linearized state feedback (using the A and B matrix representation), the system behaves much better as can be seen in Figure 4. The state variables stabilize to 0 quite quickly in an oscillatory manner. Further, even if we were to use the non-linear state feedback model (original Euler-Lagrange non-linearized equations), with the same LQR controller, we would get an output response that can be visualized in Figure 5. The observed response in this case is similar to linear state feedback, though with a longer convergence time.

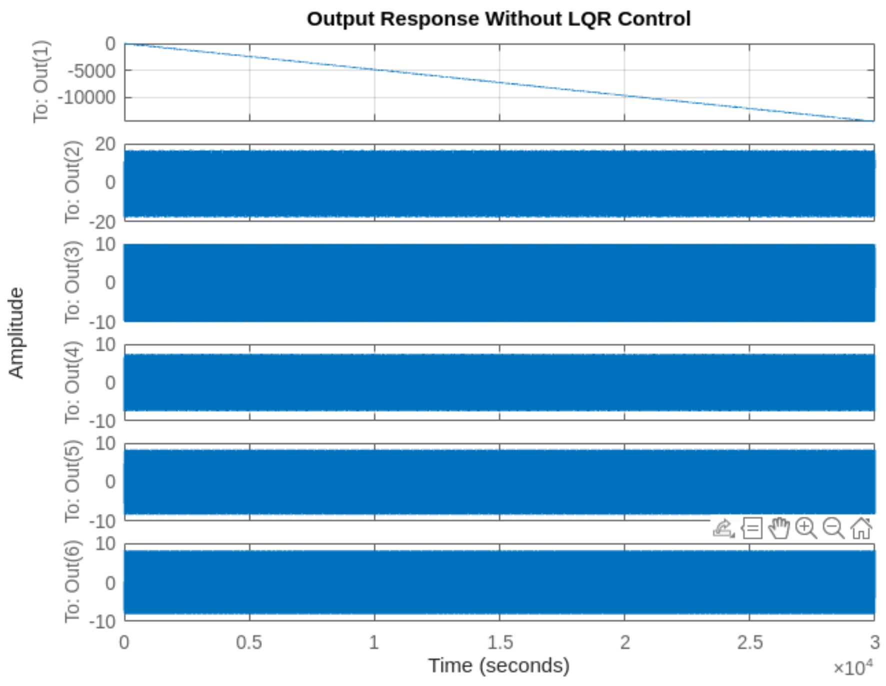


Figure 3: Output Response of the System Without Any Control

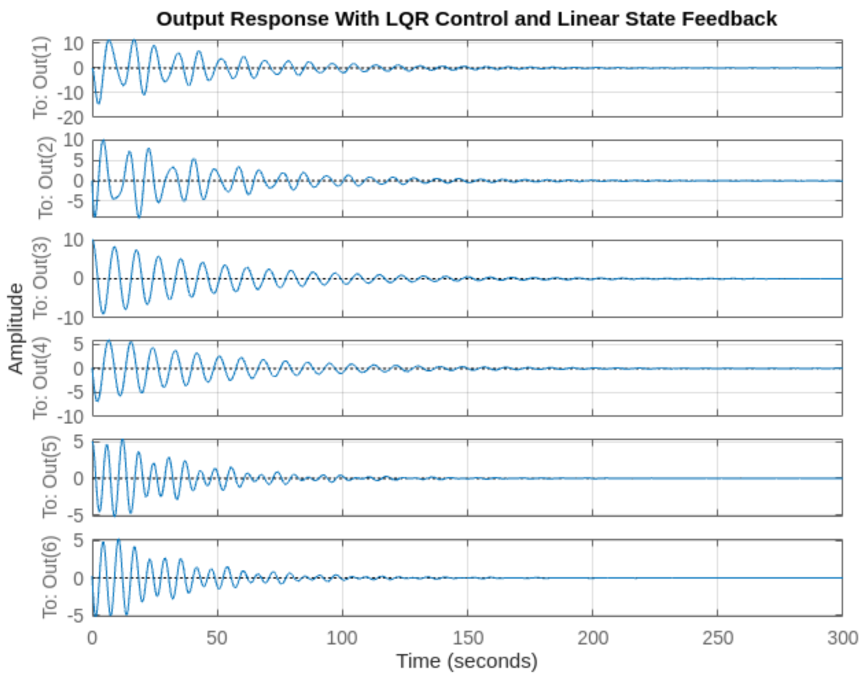


Figure 4: Output Response of the System With LQR Linear State Feedback

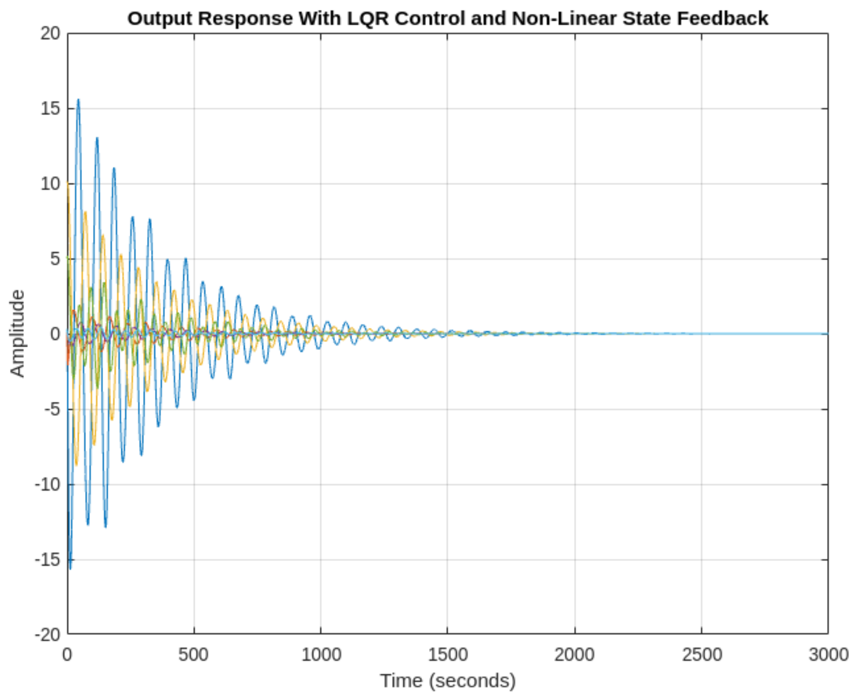


Figure 5: Output Response of the System With LQR Non-Linear State Feedback

5.3 Stability Check Using Lyapunov's Indirect Method

To ensure that the linearized system is stable, we can use Lyapunov's Indirect Method. As a part of this method, we have to check that all the eigenvalues of the closed-loop A matrix (A_c) have a negative real part. The expression for A_c turns out to be as shown below.

$$A_c = A + B_k K$$

In this expression, A_c is the closed loop A matrix, $B = B_k$, and K represents the gain matrix which is nothing but the optimal gain matrix calculated by the LQR controller. The eigenvalues of A_c come out to be the six values shown below.

$$eig(A_c) = \begin{bmatrix} -0.0200 + 0.7120i \\ -0.0200 - 0.7120i \\ -0.0320 + 1.0149i \\ -0.0320 - 1.0149i \\ -0.1007 + 0.0000i \\ -0.9113 + 0.0000i \end{bmatrix}$$

Since all the eigenvalues of A_c have a negative real part, we can say using Lyapunov's Indirect Method that the linearized LQR-controlled system using linearized state feedback is locally stable around the equilibrium point.

6 E: System Observability Check

The observability of the linearized system is assessed for different output vectors, determining the ability to infer internal states from external outputs. Observability is the ability to attain the current state of the system using the output from the system. A system is said to be observable if and only if the Grammian of observability is invertible. But like controllability, there is a much easier method to verify observability. A system is observable if the rank of the observability matrix is n. Where the observability matrix is given as,

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ . \\ . \\ . \\ . \\ CA^{n-1} \end{bmatrix}$$

If the system is observable the following condition holds,

$$rank(O) = n$$

The C matrix for the output vectors can be given as,

- $x(t)$

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

- $\theta_1(t), \theta_2(t)$

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

- $x(t), \theta_2(t)$

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

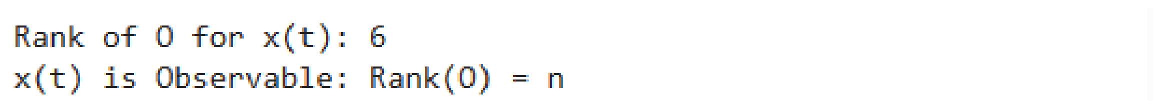
- $x(t), \theta_1(t), \theta_2(t)$

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

Using the above C matrices, the observability of the system is checked.

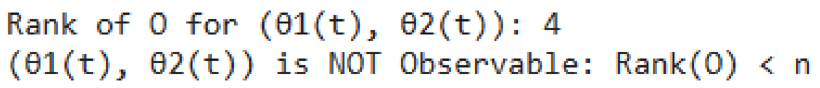
Matlab code is written to check the observability using each output vector both symbolically and using the parameter values given in Section D.

- $x(t)$



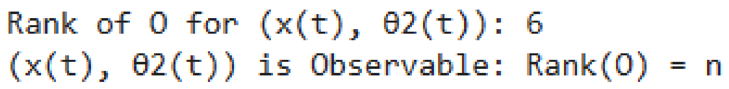
```
Rank of O for x(t): 6
x(t) is Observable: Rank(O) = n
```

- $\theta_1(t), \theta_2(t)$



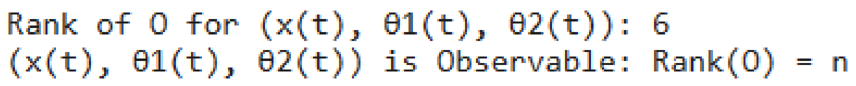
```
Rank of O for (θ1(t), θ2(t)): 4
(θ1(t), θ2(t)) is NOT Observable: Rank(O) < n
```

- $x(t), \theta_2(t)$



```
Rank of O for (x(t), θ2(t)): 6
(x(t), θ2(t)) is Observable: Rank(O) = n
```

- $x(t), \theta_1(t), \theta_2(t)$



```
Rank of O for (x(t), θ1(t), θ2(t)): 6
(x(t), θ1(t), θ2(t)) is Observable: Rank(O) = n
```

where n is the number of states.

Similar results are found with parameter values substituted into the equation.

7 F: Luenberger Observer Design

The Luenberger observer is given by the state space equation,

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

Here, $\hat{X}(t)$ is the estimated state, $Y(t)$ is the system output and L is the observer gain matrix. The Luenberger equation looks identical to a normal state space equation with the addition of a correcting factor or term, $L(Y(t) - C\hat{X}(t))$.

The cost function for the Luenberger observer is that the expectation of $\hat{X}(t)$ is $X(t)$,

In other words, the cost function J is given as,

$$J = \underset{t \longrightarrow \infty}{lim} E\left[X_e^T(t)X_e(t)\right]$$

Here,

$$X_e(t) = X(t) - \hat{X}(t)$$

Using, $X_e(t)$, the state equation can be written as,

$$\dot{X_e}(t) = (A - LC)X_e(t) + B_D U_D(t)$$

where $B_D U_D$ is some disturbance in the system inputs.

This shows that the stability of the observer depends on the poles of $(A - LC)$. An optimal choice of L can make the state estimation equation stable by selecting the poles of $(A - LC)$.

Code is written in Matlab to simulate the system state simulation and the Luenberger observer.

Using the place method of Matlab the L matrix is found by taking poles as,

$$poles = \begin{bmatrix} -1 & -1.1 & -1.2 & -1.3 & -1.4 & -1.5 \end{bmatrix}$$

$$L = \begin{bmatrix} 7.5000 \\ 21.7330 \\ -0.6777 \\ 18.6608 \\ -26.3606 \\ -18.7464 \end{bmatrix}$$

7.1 State Estimation Simulation

Outputs for Linearized Observable State Estimation

- State Estimation for the Output Vector $X(t)$

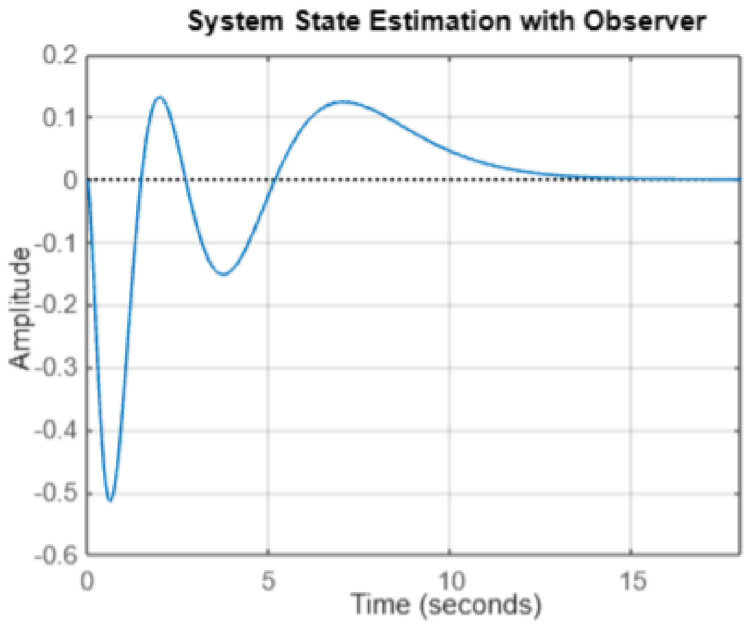


Figure 6: Estimated State of the Output Vector $X(t)$

- State Estimation for the Output Vector $X(t), \theta_2(t)$

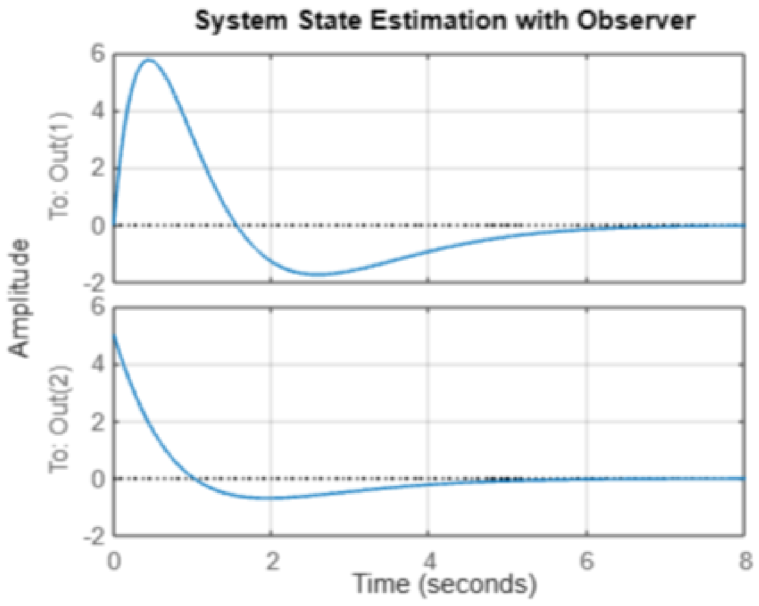


Figure 7: Estimated State of the Output Vector $X(t), \theta_2(t)$

- State Estimation for the Output Vector $X(t), \theta_1(t), \theta_2(t)$

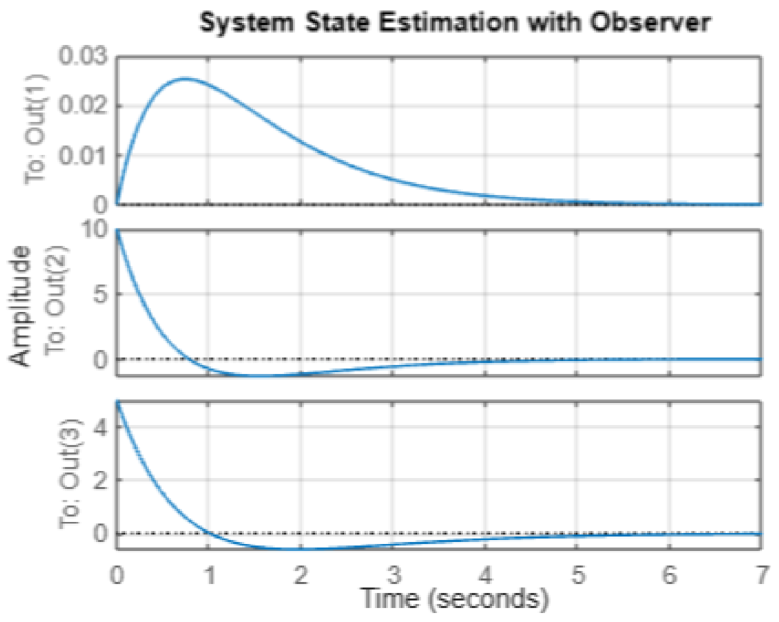


Figure 8: Estimated State of the Output Vector $X(t), \theta_1(t), \theta_2(t)$

7.2 Luenberger Observer Simulation

Outputs for Linearized Observable States with Luenberger Observer: Initial Conditions

- Output Vector $X(t)$

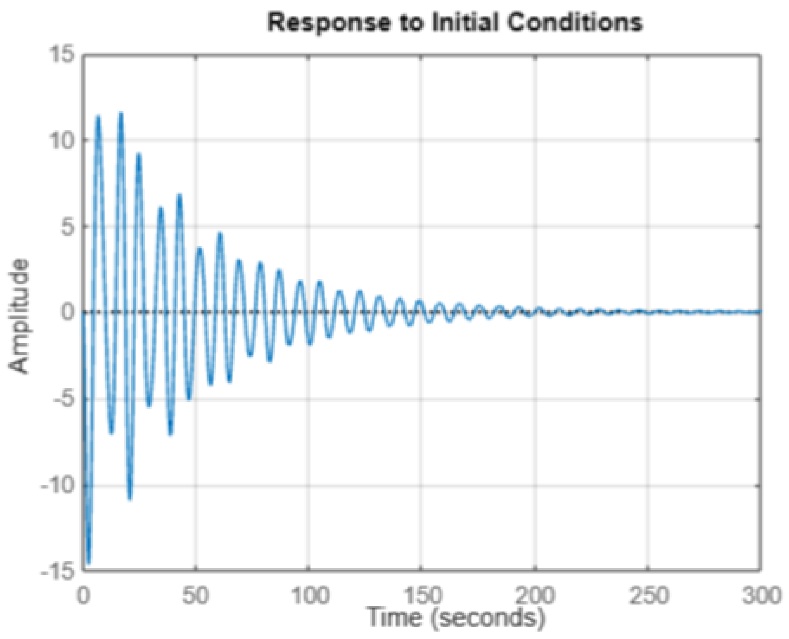


Figure 9: Luenberger Observer Response to the Initial Conditions for the Output Vector $X(t)$

- Output Vector $X(t), \theta_2(t)$

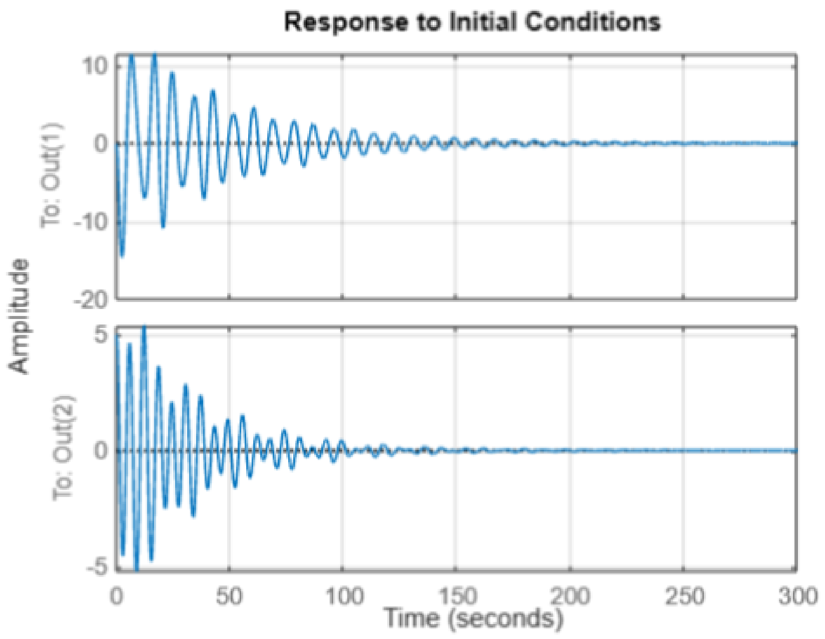


Figure 10: Luenberger Observer Response to the Initial Conditions for the Output Vector $X(t), \theta_2(t)$

- Output Vector $X(t), \theta_1(t), \theta_2(t)$

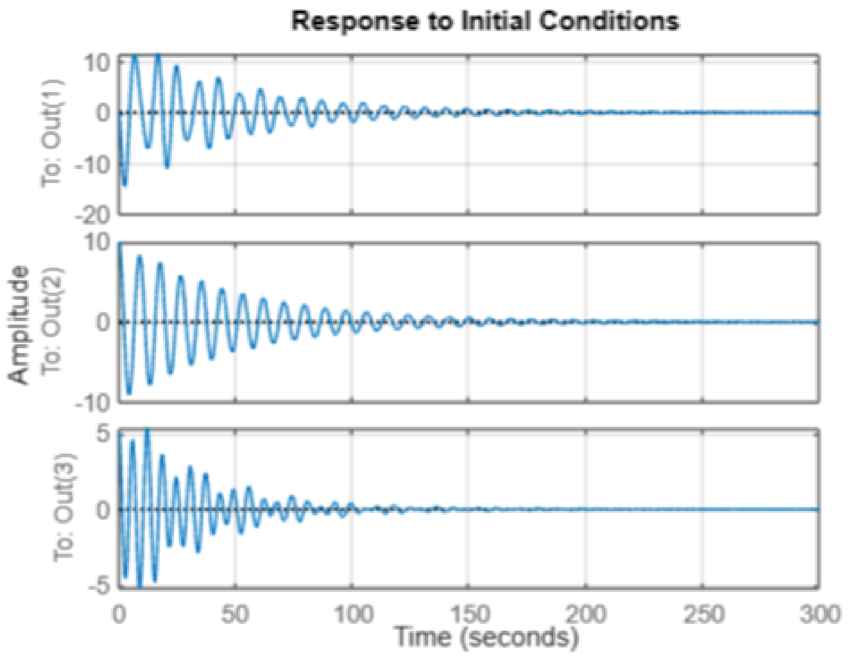


Figure 11: Luenberger Observer Response to the Initial Conditions for the Output Vector $X(t), \theta_1(t), \theta_2(t)$

Outputs for Linearized Observable States with Luenberger Observer: Step Input

- Output Vector $X(t)$

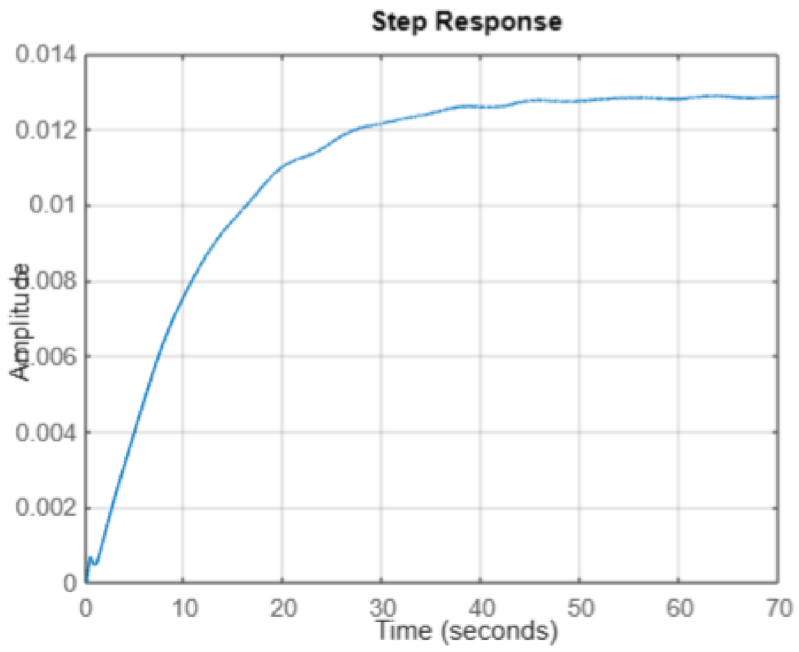


Figure 12: Luenberger Observer Response to a Step Input for the Output Vector $X(t)$

- Output Vector $X(t), \theta_2(t)$

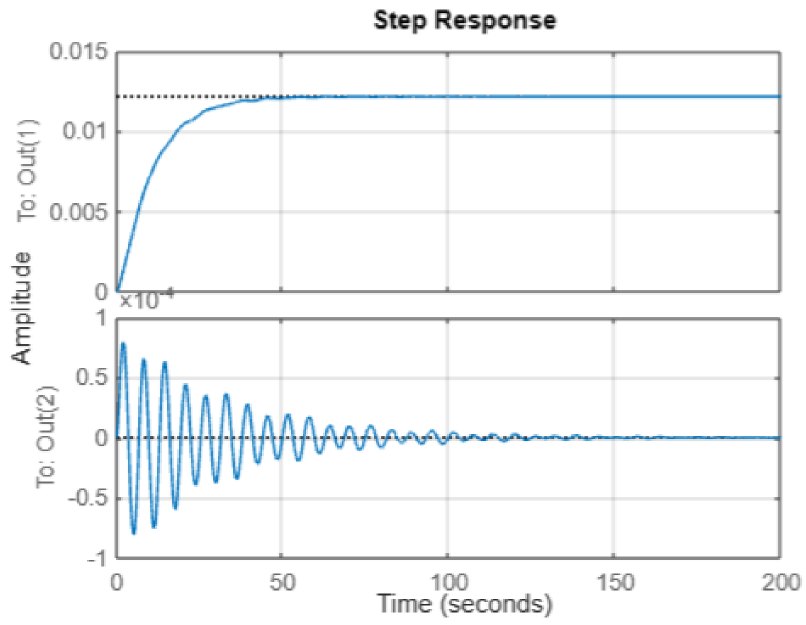


Figure 13: Luenberger Observer Response to a Step Input for the Output Vector $X(t), \theta_2(t)$

- Output Vector $X(t), \theta_1, \theta_2(t)$

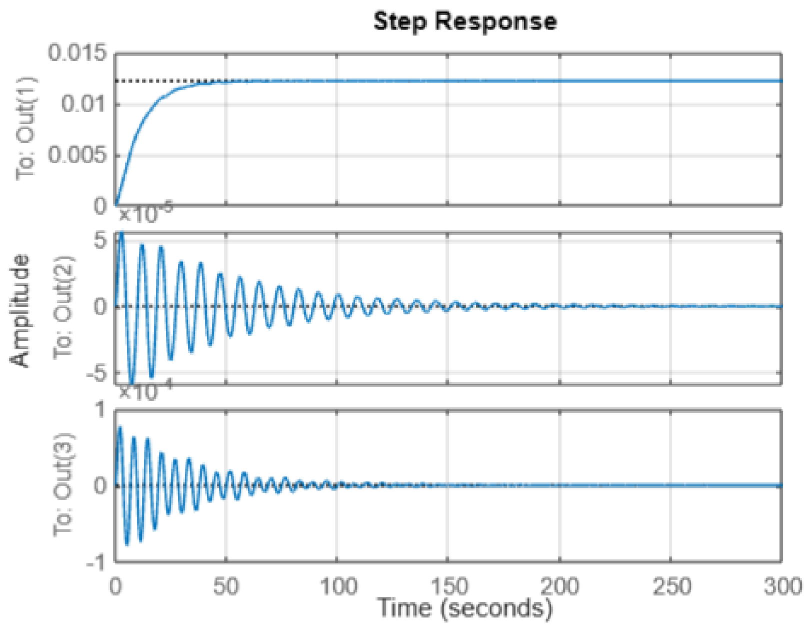


Figure 14: Luenberger Observer Response to a Step Input for the Output Vector $X(t), \theta_1(t), \theta_2(t)$

Outputs for Nonlinear System with Luenberger Observer

- Output Vector $X(t)$

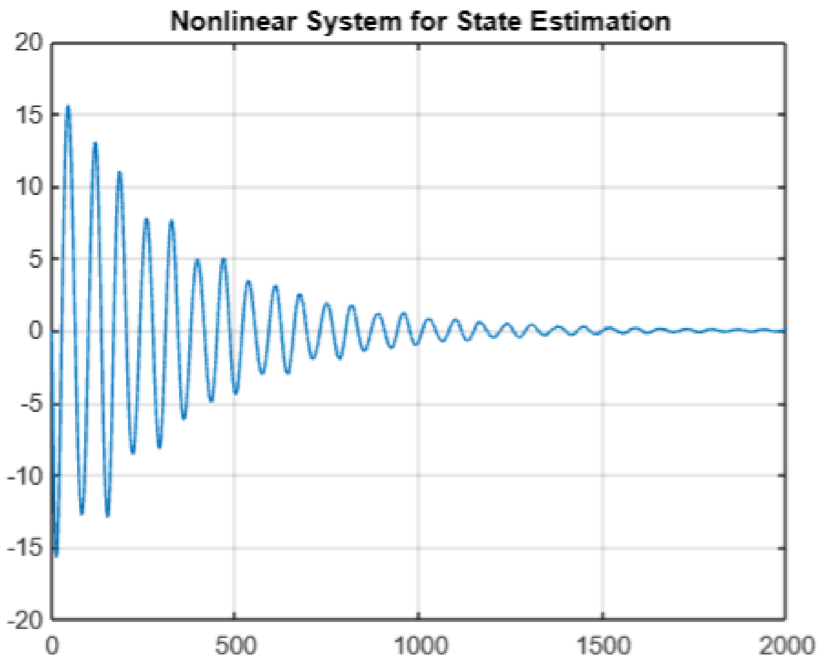


Figure 15: Luenberger Observer Response of the Nonlinear System for the Output Vector $X(t)$

- Output Vector $X(t), \theta_2(t)$

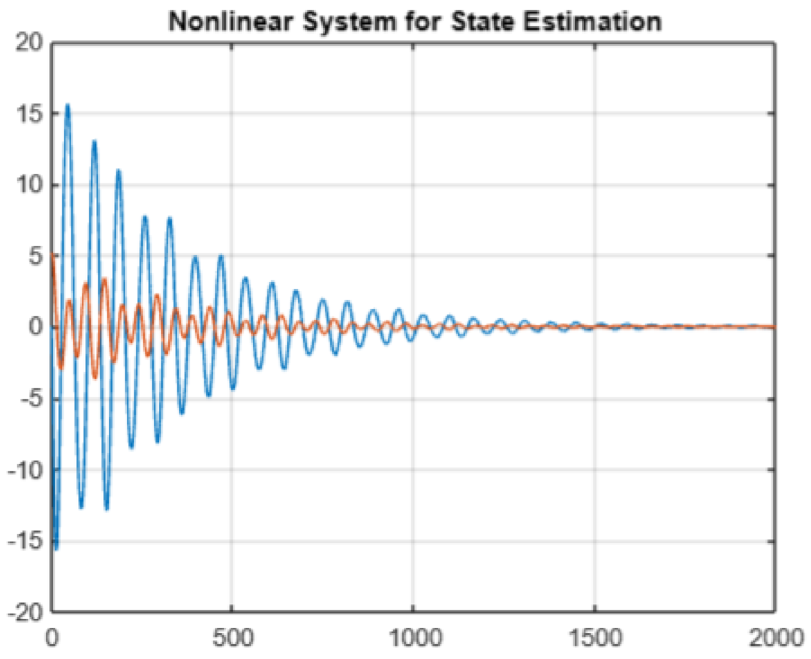


Figure 16: Luenberger Observer Response of the Nonlinear System for the Output Vector $X(t), \theta_2(t)$

- Output Vector $X(t), \theta_1, \theta_2(t)$

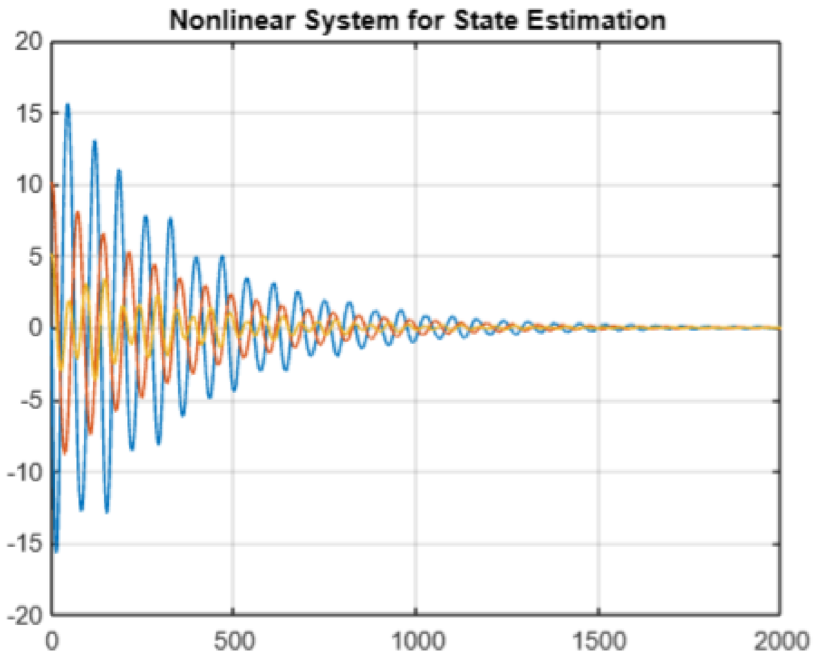


Figure 17: Luenberger Observer Response of the Nonlinear System for the Output vector $X(t), \theta_1(t), \theta_2(t)$

8 G: LQG Controller Design

In real-life applications, input and measurement noises always exist, and the state equations with noise are given as,

$$\dot{X}(t) = AX(t) + B_k U_K(t) + B_D U_D(t)$$
$$Y(t) = CX(t) + V(t)$$

where $U_D(t)$ is input noise and $V(t)$ is measurement noise.

The Linear Quadratic Gaussian (LQG) is a controller that combines the LQR with a state estimator. The LQG is applied to the nonlinear system and Matlab code is written to simulate it. A LQR and Kalman Bucy Filter is used in the simulation and randomly generated noise is added to the system. A Kalman Bucy Filter is nothing but a standard Kalman Filter applied to a continuous time domain system.

The state equation that combines LQR and the Kalman Bucy Filter to give the LQG controller is shown below as a matrix representation.

$$\begin{bmatrix} \dot{X}(t) \\ \dot{X}_e(t) \end{bmatrix} = \begin{bmatrix} A + B_K K & -B_K K \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} X(t) \\ X_e(t) \end{bmatrix} + \begin{bmatrix} B_D \\ B_D \end{bmatrix} U_D(t)$$

8.1 LQG Nonlinear Simulation with the Output Vector $X(t)$

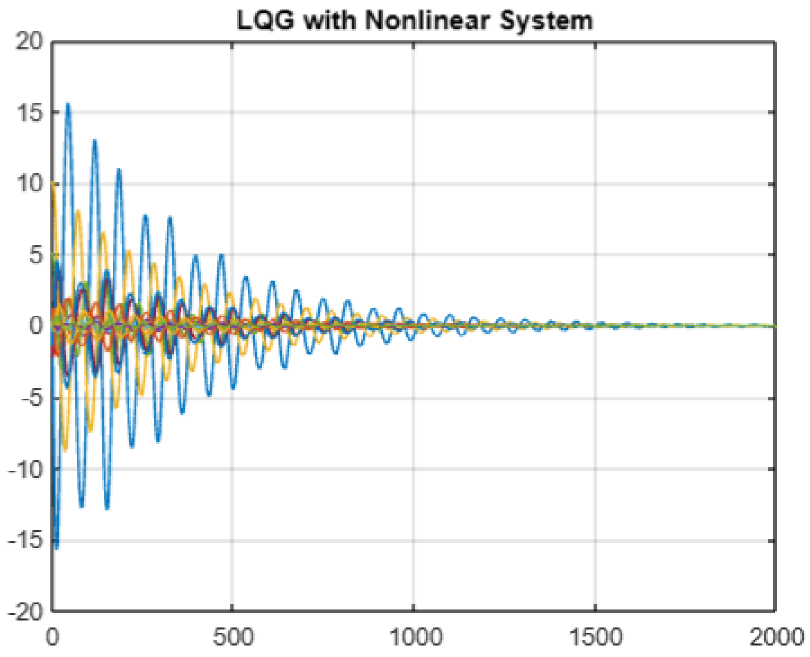


Figure 18: LQG Control on the Nonlinear System for the Output Vector $X(t)$

8.2 LQG Nonlinear Simulation with the Output Vector $X(t), \theta_2(t)$

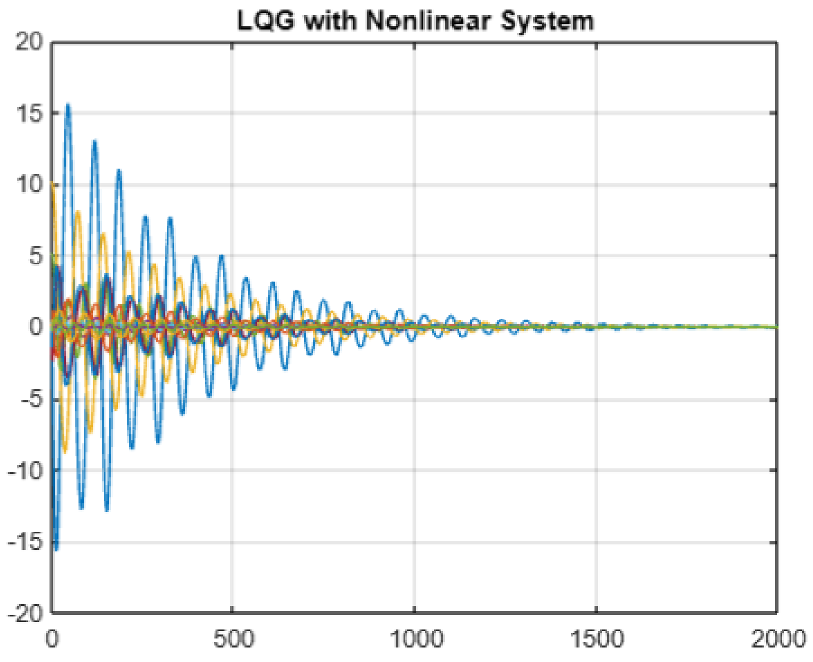


Figure 19: LQG Control on the Nonlinear System for the Output vector $X(t), \theta_2(t)$

8.3 LQG Nonlinear Simulation with the Output Vector $X(t), \theta_1(t), \theta_2(t)$

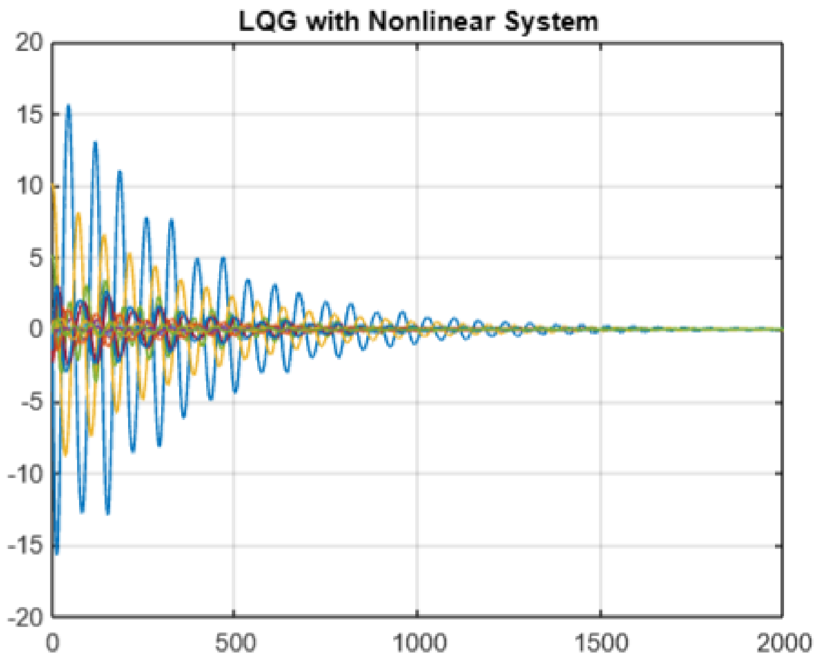


Figure 20: LQG Control on the Nonlinear System for the Output Vector $X(t), \theta_1(t), \theta_2(t)$

For reference tracking operations, the cost function needs to be reconfigured to include the state reference,

$$J = \int (X(t) - X_{ref}^T Q (X(t) - X_{ref} dt \; + \int (U(t) - U_\infty)^T R (U(t) - U_\infty) dt$$

$X_{ref}(t)$ is the reference that needs to be tracked and $U_\infty(t)$ is a condition such that,

$$AX_{ref} + BU_\infty = 0$$

To asymptotically track a constant reference the cost function, J, needs to be minimized.

The normal cost function has X_{ref} as a null vector. To asymptotically track a constant reference on x, the reference vector will be,

$$X_{ref} = \begin{bmatrix} x_{ref} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The LQG is designed to reject any Gaussian disturbance, so if the force disturbance is Gaussian, the LQG controller designed will be easily able to reject the disturbance.

9 Conclusion

In conclusion, the following has been achieved through the project.

- The crane-pendulum has been modeled using the Euler-Lagrange Method to obtain the equations of motion of the system. (Part A)
- Using small-angle approximation, the system has been linearized and expressed through state-space equations in a linear-time-invariant (LTI) system. (Part B)
- The conditions for system controllability based on system parameters have been established by assessing the rank of the controllability matrix. (Part C)
- An LQR controller has been designed for the system and the output response with respect to both linear and non-linear state-feedback has been verified. Further, using Lyapunov's Indirect Method, the stability of the system has been verified. (Part D)
- The different combinations of output representations that make the system observable have been checked by using the rank test on the observability matrix. (Part E)
- The best Luenberger Observer is constructed for each output representation with respect to the initial conditions by using the place method of Matlab. This has been done for both the linear and non-linear system representation. Further, the system's response for all output representations in response to a unit step response has also been simulated. (Part F)
- An LQG controller has been designed on the smallest observable output representation for non-linear state feedback using a Kalman Bucy Filter. (Part G)

The Matlab code used for all simulations along with the videos of them running can be found in the Appendix below.

10 Appendix

A System Controllability & LQR Simulations Code - Part C & D

The link to the video that shows the execution of this code can be found hyperlinked

https://drive.google.com/file/d/14AtCvRM7BaBsk57duZaUSUIkp7hCvEEE/view?usp=sharing.

```
% Initialize symbols
syms M m1 m2 l1 l2 g F t1 t2 x x_ t1_ t2_;

% Setup the functions after small angle approximation
X_dotdot =(F - m1*g*t1 - m2*g*t2)/M;
t1_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t1)/(M*l1);
t2_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t2)/(M*l2);

% Setup state and input vectors
X_states = [x x_ t1 t1_ t2 t2_];
X_eq = [x_ X_dotdot t1_ t1_dotdot t2_ t2_dotdot];
U_input = F;

% Programmatically attain A matrix through Jacobian linearization
A_ = zeros(length(X_eq), length(X_states)) + x;
for i = 1:length(X_eq)
    for j = 1:length(X_states)
        A_(i, j) = diff(X_eq(i), X_states(j));
    end
end
disp(A_)

% Programmatically attain B matrix through Jacobian linearization
B_ = zeros(length(X_states), length(U_input)) + x;
for i = 1:length(U_input)
    for j = 1:length(X_eq)
        B_(j, i) = diff(X_eq(j), U_input(i));
    end
end
disp(B_)

% Test to check for Controllability of Symbolic Matrices
n = size(A_, 1);
ControllabilityMatrix = zeros(n) + x;
ControllabilityMatrix(:,1) = B_;

for i = 1:(n - 1)
    ControllabilityMatrix(:,i+1) = A_^i * B_;
end
disp(ControllabilityMatrix)

% Condition of System Parameters for Controllability
det_ControllabilityMatrix = det(ControllabilityMatrix);
disp('Determinant of the controllability matrix is');
disp(det_ControllabilityMatrix)
disp('size of system is');
disp(n)

% Checking rank of Matrix
rank_ControllabilityMatrix = rank(ControllabilityMatrix);
disp('Rank of C is');
disp(rank_ControllabilityMatrix)

if rank_ControllabilityMatrix == n
    disp('Rank(C)=n, controllable');
else
    disp('Rank(C)<n, not controllable ');
end

% Setup C and D Matrices
C = eye(6);

D = 0;

% Set Initial Condition
X0 = [0 ; 0.1 ; 10 ; 0.2 ; 5 ; 0.3];

% Set System Parameters
M_v = 1000;
m1_v = 100;
m2_v = 100;
l1_v = 20;
l2_v = 10;
g_v = 9.8;

% Substitute System Parameters into Symbolic Matrices
A = double(subs(A_, {M m1 m2 l1 l2 g}, {M_v m1_v m2_v l1_v l2_v g_v
   }))
B = double(subs(B_, {M l1 l2}, {M_v l1_v l2_v}))

% Checking Controllability with System Parameters
val_ControllabilityMatrix = ctrb(A,B)

if (rank(val_ControllabilityMatrix)==6)
    disp(['Having full rank (equal to ', num2str(rank(
       val_ControllabilityMatrix)), '), the pair A,B is not
       controllable'])
else
    disp(['Not having full rank (equal to ', num2str(rank(
       val_ControllabilityMatrix)), '), the pair A,B is not
       controllable'])
end


% Setup Gain Matrices
R = 0.0001;
Q1 = 1;
Q2 = 100;
```

```
Q3 = 1;
Q4 = 100;
Q5 = 10;
Q6 = 10;

Q = diag([Q1, Q2, Q3, Q4, Q5, Q6]);

% Setup Initial State Equation
system_behaviour = ss(A,B,C,D);

figure(1)
initial(system_behaviour,X0)
grid on

% Find LQR Gain Matrix
[K, P, poles] = lqr(A,B,Q,R)
system_stable = 'true' ;

for i=1:6
    if(real(poles(i,1))>0)
        system_stable = 'false';
        break
    end
end

if (strcmp(system_stable,'true'))
    disp("Under the chosen and calculated parameters for the LQR
       controller, the system is stable using Lyapunov's Indirect
       Method")
else
    disp("Under the chosen and calculated parameters for the LQR
       controller, the system is not stable using Lyapunov's
       Indirect Method")
end

% Build New State Space Equation with LQR
new_B = [0 ; 0 ; 0 ; 0 ; 0 ; 0];
lqr_system_behavior = ss(A-(B*K),new_B,C,D);

% Simulate Linear System
figure(2)
initial(lqr_system_behavior,X0)
grid on

% Nonlinear System Simulation
time_period = 0:0.01:10000 ;
[t,X] = ode45(@(t,X) nonlinearfeedback(t, X, K, M_v, m1_v, m2_v,
   l1_v, l2_v, g_v),time_period,X0);
figure(3)
plot(t,X)
grid on
```

```
function y_dot = nonlinearfeedback(t, X, K, M, m1, m2, l1, l2, g)
F = -K*X ;
y_dot = zeros(6,1) ;
y_dot(1,1) = X(2) ;
y_dot(2,1) = (1/(M+m1+m2))*(F+(m1*l1*y_dot(4,1)*cosd(X(3)))+(m2*l2*
    y_dot(6,1)*cosd(X(5)))-(m1*l1*(X(4)^2)*sind(X(3)))-(m2*l2*(X(6)
    ^2)*sind(X(5))));
y_dot(3,1) = X(4) ;
y_dot(4,1) = (y_dot(2,1)*cosd(X(3))-g*(sind(X(3))))/l1 ;
y_dot(5,1) = X(6) ;
y_dot(6,1) = (y_dot(2,1)*cosd(X(5))-g*(sind(X(5))))/l2 ;
end
```

B System Observability Code - Part E

The link to the video that shows the execution of this code can be found hyperlinked

https://drive.google.com/file/d/1hTYo$_l I7PY0wHKkFrVwvjzA6YLWeoIXd/view?usp = sharing$.

```
    % Initialize symbols
syms M m1 m2 l1 l2 g F t1 t2 x x_ t1_ t2_;

% Setup the functions after small angle approximation
X_dotdot =(F - m1*g*t1 - m2*g*t2)/M;
t1_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t1)/(M*l1);
t2_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t2)/(M*l2);

% Setup state and input vectors
X_states = [x x_ t1 t1_ t2 t2_];
X_eq = [x_ X_dotdot t1_ t1_dotdot t2_ t2_dotdot];
U_input = F;

% Programmatically attain A matrix through Jacobian linearization
A_ = zeros(length(X_eq), length(X_states)) + x;
for i = 1:length(X_eq)
    for j = 1:length(X_states)
        A_(i, j) = diff(X_eq(i), X_states(j));
    end
end
disp(A_)

% Programmatically attain B matrix through Jacobian linearization
B_ = zeros(length(X_states), length(U_input)) + x;
for i = 1:length(U_input)
    for j = 1:length(X_eq)
        B_(j, i) = diff(X_eq(j), U_input(i));
    end
end
disp(B_)

n = size(A_, 1);

% Set Initial Condition
X0 = [0 ; 0.1 ; 10 ; 0.2 ; 5 ; 0.3];

% Set System Parameters
M_v = 1000;
m1_v = 100;
m2_v = 100;
l1_v = 20;
l2_v = 10;
g_v = 9.8;

% Substitute System Parameters into Symbolic Matrices
A = double(subs(A_, {M m1 m2 l1 l2 g}, {M_v m1_v m2_v l1_v l2_v g_v
   }))
B = double(subs(B_, {M l1 l2}, {M_v l1_v l2_v}))
```

```
% Define output vectors
C1 = [1 0 0 0 0 0]; % x(t)
C2 = [0 0 1 0 0 0; 0 0 0 0 1 0]; % ( 1 (t),  2 (t))
C3 = [1 0 0 0 0 0; 0 0 0 0 1 0]; % (x(t),  2 (t))
C4 = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0]; % (x(t),  1 (t),  2 (t
   ))

% Check observability and display results
checkObservability(A, C1, n, 'x(t)');
checkObservability(A, C2, n, '( 1 (t),  2 (t))');
checkObservability(A, C3, n, '(x(t),  2 (t))');
checkObservability(A, C4, n, '(x(t),  1 (t),  2 (t))');


% Function to calculate observability matrix

function rank_O = calculateObservability(A, C, n)
    O = obsv(A, C);
    rank_O = rank(O);
end

% Function to check observability and display results
function checkObservability(A, C, n, outputVector)
    rank_O = calculateObservability(A, C, n);
    disp(['Rank of O for ', outputVector, ': ', num2str(rank_O)]);
    if rank_O == n
        disp([outputVector, ' is Observable: Rank(O) = n']);
    else
        disp([outputVector, ' is NOT Observable: Rank(O) < n']);
    end
end
```

C State Estimation and Luenberger Observer Code - Part F

State Estimation

The link to the video that shows the execution of this code can be found hyperlinked

https://drive.google.com/file/d/11kT7TSUolx5rdGCfv0j1oCADoITm0ZA1/view?usp=sharing.

```
% Simulation Code for State Estimation X_hat
    % Initialize symbols
syms M m1 m2 l1 l2 g F t1 t2 x x_ t1_ t2_;

% Setup the functions after small angle approximation
X_dotdot =(F - m1*g*t1 - m2*g*t2)/M;
t1_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t1)/(M*l1);
t2_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t2)/(M*l2);

% Setup state and input vectors
X_states = [x x_ t1 t1_ t2 t2_];
X_eq = [x_ X_dotdot t1_ t1_dotdot t2_ t2_dotdot];
U_input = F;

% Programmatically attain A matrix through Jacobian linearization
A_ = zeros(length(X_eq), length(X_states)) + x;
for i = 1:length(X_eq)
    for j = 1:length(X_states)
        A_(i, j) = diff(X_eq(i), X_states(j));
    end
end
disp(A_)

% Programmatically attain B matrix through Jacobian linearization
B_ = zeros(length(X_states), length(U_input)) + x;
for i = 1:length(U_input)
    for j = 1:length(X_eq)
        B_(j, i) = diff(X_eq(j), U_input(i));
    end
end
disp(B_)

n = size(A_, 1);

% Set Initial Condition
X0 = [0 ; 0.1 ; 10 ; 0.2 ; 5 ; 0.3];

% Set System Parameters
M_v = 1000;
m1_v = 100;
m2_v = 100;
l1_v = 20;
l2_v = 10;
g_v = 9.8;

% Substitute System Parameters into Symbolic Matrices
A = double(subs(A_, {M m1 m2 l1 l2 g}, {M_v m1_v m2_v l1_v l2_v g_v
   }))
```

```
B = double(subs(B_, {M l1 l2}, {M_v l1_v l2_v}))

% Setup C and D Matrices
C = eye(6);

D = 0;


% Setup Gain Matrices
R = 0.0001;
Q1 = 1;
Q2 = 100;
Q3 = 1;
Q4 = 100;
Q5 = 10;
Q6 = 10;

Q = diag([Q1, Q2, Q3, Q4, Q5, Q6]);

% Find LQR Gain Matrix
[K, P, poles] = lqr(A, B, Q, R);
poles = [-1; -1.1; -1.2; -1.3; -1.4; -1.5]; % Arbitrarily change
    pole values

% Define output vector and check observability
C1 = [1 0 0 0 0 0]; % x(t)
C2 = [0 0 1 0 0 0; 0 0 0 0 1 0]; % ( 1 (t), 2 (t))
C3 = [1 0 0 0 0 0; 0 0 0 0 1 0]; % (x(t), 2 (t))
C4 = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0]; % (x(t), 1 (t), 2 (t
    ))

C_obs = {C1 C2 C3 C4};


for i=1:size(C_obs, 2)
    C_using = C_obs{i};

    % Create Observation Matrix
    O1 = obsv(A, C_using);
    if rank(O1) == size(A, 1)
        disp('C is observable.');
        % Design Luenberger observer
        L1 = place(A', C_using', poles)'; %use C4
        B_obs = [B, L1];
        D_obs = zeros(size(C_using, 1), size(B_obs, 2));


        % Simulate the linearized system with the observer
        A_obs = A - L1 * C_using;


        % Assuming initial observer estimates are zero
        X0_obs = [X0; zeros(6, 1)];
        % Create State Space Equation for State Estimation
        system_obs_behavior = ss(A_obs, B_obs, C_using, D_obs);

        % Plot Initial Condition with Linearized System
        figure;
        initial(system_obs_behavior, X0);
        title('System State Estimation with Observer');
        grid on;

        % Plot Step Input with Linearized System
        figure;
        initial(system_obs_behavior, X0);
        title('System State Estimation with Step Input');
        grid on;

        % Simulation parameters
        time_period = 0:0.01:2000;

        % Nonlinear feedback simulation
        [t, Y] = ode45(@(t,X) nonlinearfeedback(t, X, K, M_v, m1_v,
           m2_v, l1_v, l2_v, g_v, C_using, L1), time_period, X0);
        figure;
        plot(t, Y);
        title('Nonlinear System for State Estimation');
        grid on;

    else
        disp('C is not observable.');
    end
end



function y_dot = nonlinearfeedback(t, Y, K, M, m1, m2, l1, l2, g,
   C_using, L1)
    F = -K * Y;
    if size(C_using, 1) == 1
        x = [Y(1)];
    elseif size(C_using, 1) == 2
        x = [Y(1); Y(5)];
    elseif size(C_using, 1) == 3
        x = [Y(1); Y(3); Y(5)];
    end
    correction_term = L1*(x - C_using*Y);
    y_dot = zeros(6, 1);
    y_dot(1) = Y(2);
    y_dot(2) = (1/(M + m1 + m2)) * (F + (m1 * l1 * y_dot(4) * cosd(Y
       (3))) + (m2 * l2 * y_dot(6) * cosd(Y(5))) - (m1 * l1 * (Y(4)
       ^2) * sind(Y(3))) - (m2 * l2 * (Y(6)^2) * sind(Y(5))));
    y_dot(3) = Y(4);
    y_dot(4) = (y_dot(2) * cosd(Y(3)) - g * sind(Y(3))) / l1;
    y_dot(5) = Y(6);
    y_dot(6) = (y_dot(2) * cosd(Y(5)) - g * sind(Y(5))) / l2;
    y_dot = y_dot + correction_term;
end
```

Luenberger Observer Simulation

The link to the video that shows the execution of this code can be found hyperlinked

https://drive.google.com/file/d/1yYE2$_kWbelnjDTcBHP2YsP3w3UjQqCwS/view?usp = sharing$.

```
% Simulation Code for State Estimation X_hat
    % Initialize symbols
syms M m1 m2 l1 l2 g F t1 t2 x x_ t1_ t2_;

% Setup the functions after small angle approximation
X_dotdot =(F - m1*g*t1 - m2*g*t2)/M;
t1_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t1)/(M*l1);
t2_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t2)/(M*l2);

% Setup state and input vectors
X_states = [x x_ t1 t1_ t2 t2_];
X_eq = [x_ X_dotdot t1_ t1_dotdot t2_ t2_dotdot];
U_input = F;

% Programmatically attain A matrix through Jacobian linearization
A_ = zeros(length(X_eq), length(X_states)) + x;
for i = 1:length(X_eq)
    for j = 1:length(X_states)
        A_(i, j) = diff(X_eq(i), X_states(j));
    end
end
disp(A_)

% Programmatically attain B matrix through Jacobian linearization
B_ = zeros(length(X_states), length(U_input)) + x;
for i = 1:length(U_input)
    for j = 1:length(X_eq)
        B_(j, i) = diff(X_eq(j), U_input(i));
    end
end
disp(B_)

n = size(A_, 1);

% Set Initial Condition
X0 = [0 ; 0.1 ; 10 ; 0.2 ; 5 ; 0.3];

% Set System Parameters
M_v = 1000;
m1_v = 100;
m2_v = 100;
l1_v = 20;
l2_v = 10;
g_v = 9.8;

% Substitute System Parameters into Symbolic Matrices
```

```
A = double(subs(A_, {M m1 m2 l1 l2 g}, {M_v m1_v m2_v l1_v l2_v g_v
    }))
B = double(subs(B_, {M l1 l2}, {M_v l1_v l2_v}))

% Setup Gain Matrices
R = 0.0001;
Q1 = 1;
Q2 = 100;
Q3 = 1;
Q4 = 100;
Q5 = 10;
Q6 = 10;

Q = diag([Q1, Q2, Q3, Q4, Q5, Q6]);

% C Matrices using Output Vectors
C_x = [1 0 0 0 0 0]
C_xt2 = [C_x; 0 0 0 0 1 0]
C_xt1t2 = [C_x ; 0 0 1 0 0 0 ; 0 0 0 0 1 0]

C_obs = {C_x C_xt2 C_xt1t2};


D = 0;

% Find LQR Gain Matrix
[K, P_lqr, poles_lqr] = lqr(A, B, Q, R);

% Poles for Luenberger Observation Gain Matrix
poles = [-1; -1.1; -1.2; -1.3; -1.4; -1.5];

for i=1:size(C_obs, 2)
    C_using = C_obs{i};

    % Setup Luenberger Gain Matrix
    L = place(A', C_using', poles)'

    % Setup State Estimation Matrices A, B, C
    A_lqg = [A-(B*K) B*K; zeros(n,n) A-(L*C_using)]
    B_lqg = [B; B];
    C_lqg = [ C_using zeros(size(C_using))]

    % Create State Space Equation for LQG
    lqg_system_behaviour = ss(A_lqg, B_lqg, C_lqg, D);
    figure;

    % Simulate LQG with Linearized Initial Condition
    initial(lqg_system_behaviour, [X0; zeros(n, 1)])
    grid on

    % Simulate LQG with Linearized Step Input
    figure;
    step(lqg_system_behaviour)
    grid on
    % Nonlinear System Simulation
    time_period = 0:0.01:2000 ;
    [t,X] = ode45(@(t,X) nonlinearfeedback(t, X, K, L, A, C_using,
       M_v, m1_v, m2_v, l1_v, l2_v, g_v),time_period,[X0; zeros(n,
       1)]);
    figure;
    if size(C_using, 1) == 1
        plot(t,X(:, 1))
    elseif size(C_using, 1) == 2
        plot(t,X(:, 1))
        hold on
        plot(t,X(:, 5))
    elseif size(C_using, 1) == 3
        plot(t,X(:, 1))
        hold on
        plot(t,X(:, 3))
        hold on
        plot(t,X(:, 5))
    end
    title('Nonlinear System for State Estimation');
    grid on
end



function y_dot = nonlinearfeedback(t, X, K, L, A, C, M, m1, m2, l1,
   l2, g)
F = -K*X(1:6) ;
X_hat = (A-(L*C))*X(7:12);
y_dot = zeros(12,1) ;
y_dot(1,1) = X(2) ;
y_dot(2,1) = (1/(M+m1+m2))*(F+(m1*l1*y_dot(4,1)*cosd(X(3)))+(m2*l2*
   y_dot(6,1)*cosd(X(5)))-(m1*l1*(X(4)^2)*sind(X(3)))-(m2*l2*(X(6)
   ^2)*sind(X(5))));
y_dot(3,1) = X(4) ;
y_dot(4,1) = (y_dot(2,1)*cosd(X(3))-g*(sind(X(3))))/l1 ;
y_dot(5,1) = X(6) ;
y_dot(6,1) = (y_dot(2,1)*cosd(X(5))-g*(sind(X(5))))/l2 ;
y_dot(7,1) = y_dot(1,1) + X_hat(1);
y_dot(8,1) = y_dot(2,1) + X_hat(2);
y_dot(9,1) = y_dot(3,1) + X_hat(3);
y_dot(10,1) = y_dot(4,1) + X_hat(4);
y_dot(11,1) = y_dot(5,1) + X_hat(5);
y_dot(12,1) = y_dot(6,1) + X_hat(6);
end
```

D LQG Controller Code - Part G

The link to the video that shows the execution of this code can be found hyperlinked

https://drive.google.com/file/d/1UydqHAcCql-B$_y4ZGzO_yOotl6KWPK9n/view?usp = sharing$.

```
    % Initialize symbols
syms M m1 m2 l1 l2 g F t1 t2 x x_ t1_ t2_;

% Setup the functions after small angle approximation
X_dotdot =(F - m1*g*t1 - m2*g*t2)/M;
t1_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t1)/(M*l1);
t2_dotdot = (F - m1*g*t1 - m2*g*t2 - M*g*t2)/(M*l2);

% Setup state and input vectors
X_states = [x x_ t1 t1_ t2 t2_];
X_eq = [x_ X_dotdot t1_ t1_dotdot t2_ t2_dotdot];
U_input = F;

% Programmatically attain A matrix through Jacobian linearization
A_ = zeros(length(X_eq), length(X_states)) + x;
for i = 1:length(X_eq)
    for j = 1:length(X_states)
        A_(i, j) = diff(X_eq(i), X_states(j));
    end
end
disp(A_)

% Programmatically attain B matrix through Jacobian linearization
B_ = zeros(length(X_states), length(U_input)) + x;
for i = 1:length(U_input)
    for j = 1:length(X_eq)
        B_(j, i) = diff(X_eq(j), U_input(i));
    end
end
disp(B_)

n = size(A_, 1);

% Set Initial Condition
X0 = [0 ; 0.1 ; 10 ; 0.2 ; 5 ; 0.3];

% Set System Parameters
M_v = 1000;
m1_v = 100;
m2_v = 100;
l1_v = 20;
l2_v = 10;
g_v = 9.8;

% Substitute System Parameters into Symbolic Matrices
A = double(subs(A_, {M m1 m2 l1 l2 g}, {M_v m1_v m2_v l1_v l2_v g_v
   }))
B = double(subs(B_, {M l1 l2}, {M_v l1_v l2_v}))
```

```
% Setup Gain Matrices
R = 0.0001;
Q1 = 1;
Q2 = 100;
Q3 = 1;
Q4 = 100;
Q5 = 10;
Q6 = 10;

Q = diag([Q1, Q2, Q3, Q4, Q5, Q6]);

% C Matrices using Output Vectors
C_x = [1 0 0 0 0 0]
C_xt2 = [C_x; 0 0 0 0 1 0]
C_xt1t2 = [C_x ; 0 0 1 0 0 0 ; 0 0 0 0 1 0]

C_obs = {C_x C_xt2 C_xt1t2};


D = 0;

% Find LQR Gain Matrix
[K, P_lqr, poles_lqr] = lqr(A, B, Q, R);

% Input Disturbances
inputVariances = rand(1, size(A, 1))/2;
SigmaD = diag(inputVariances)

for i=1:size(C_obs, 2)
    C_using = C_obs{i};

    % Measurement Disturbances
    measurementVariances = rand(1, size(C_using, 1));
    SigmaV = diag(measurementVariances)

    % Find Kalman Gain Matrix
    [L, P_kalman, poles_kalman] = lqe(A, SigmaD, C_using, SigmaD,
       SigmaV)

    % Setup New A, B, C, D Matrices for LQG
    A_lqg = [A-(B*K) B*K; zeros(n,n) A-(L*C_using)]
    B_lqg = [B; B] % No cloue whihc is correct
    B_lqg = [inputVariances'; inputVariances'] % No cloue whihc is
       correct
    C_lqg = [ C_using C_using]

    % Create LQG State Space Equation
    lqg_system_behaviour = ss(A_lqg, B_lqg, C_lqg, D);
    figure;
    initial(lqg_system_behaviour, [X0; zeros(n, 1)])
    grid on

    % Nonlinear System Simulation
    time_period = 0:0.01:2000 ;
    [t,X] = ode45(@(t,X) nonlinearfeedback(t, X, K, L, A, C_using,
       SigmaD, SigmaV, M_v, m1_v, m2_v, l1_v, l2_v, g_v),time_period
       ,[X0; zeros(n, 1)]);
    figure;
    plot(t,X)

    title('LQG with Nonlinear System');
    grid on
end




function y_dot = nonlinearfeedback(t, X, K, L, A, C, SigmaD, SigmaV,
    M, m1, m2, l1, l2, g)
F = -K*X(1:6) ;
X_hat = (A-(L*C))*X(7:12);
y_dot = zeros(12,1) ;
y_dot(1,1) = X(2) ;
y_dot(2,1) = (1/(M+m1+m2))*(F+(m1*l1*y_dot(4,1)*cosd(X(3)))+(m2*l2*
   y_dot(6,1)*cosd(X(5)))-(m1*l1*(X(4)^2)*sind(X(3)))-(m2*l2*(X(6)
   ^2)*sind(X(5))));
y_dot(3,1) = X(4) ;
y_dot(4,1) = (y_dot(2,1)*cosd(X(3))-g*(sind(X(3))))/l1 ;
y_dot(5,1) = X(6) ;
y_dot(6,1) = (y_dot(2,1)*cosd(X(5))-g*(sind(X(5))))/l2 ;
y_dot(7,1) = y_dot(1,1) + X_hat(1);
y_dot(8,1) = y_dot(2,1) + X_hat(2);
y_dot(9,1) = y_dot(3,1) + X_hat(3);
y_dot(10,1) = y_dot(4,1) + X_hat(4);
y_dot(11,1) = y_dot(5,1) + X_hat(5);
y_dot(12,1) = y_dot(6,1) + X_hat(6);
end
```