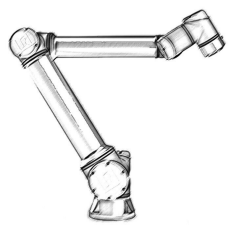
ENPM667 FALL 2023

CONTROL OF ROBOTIC SYSTEMS FINAL REPORT

Caitlin Conn, Piyush Goenka, Yoseph Kebede

Professor Waseem Malik



Engineering Professional Masters, Robotics

The University of Maryland

December 18, 2023

Contents

1 Introduction

This project aims at solving one of the most popular practical examples of controls involving a two mass attached to a moving crane system driven by a forced input. The problem is set with a cart traveling on a one dimensional frictionless track with mass M and acted on by an external force F (as system input). The cart in turn carries to objects of mass m_1 and m_2, that are attached to it by cords l_1 and l_2 respectively illustrated on fig-1 below.

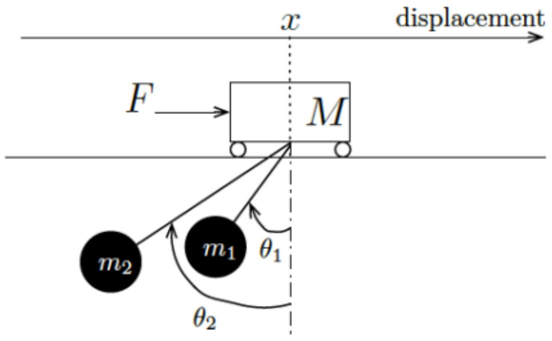


Figure 1: Crane with two pendulum's system

Thus, based on given conditions, and initial conditions, as well as input values considered later on, the project is divided into sections that start by modeling the system, and performing linearization, followed by designing LQR controllers, a Luenberger observer, and finally an LQG controller where each are further refined, and get their results corroborated via simulations. As, a result, through all this analysis, the practical benefit of these controllers are demonstrated when one thinks of stabilizing closed loop systems in general.

2 First Component: Modeling + LQR

2.1 Modeling The Systems

First we begin by modeling the system in which case we obtain the equation of motion, and then put it in state-space representation

2.1.1 Systems of Equations

To minimize the work of assigning coordinate frames, we will use Lagrange-Euler method to compute the Kinetic and Potential Energy of the entire system followed by taking the appropriate derivatives to deduce equations of motions for each degree of freedom. From the figure we can easily identify that the system can be expressed in terms of x, θ_1 and θ_2.

$$L = T - V \tag{1}$$

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) - \frac{\partial L}{\partial q} = F \tag{2}$$

The total kinetic energy of the system can be computed by taking the sum of the kinetic energies of the cart and pendulum masses.

$$\begin{aligned} &T_{tot} = T_M + T_{m1} + T_{m2} \\ &T_{tot} = \tfrac{1}{2}(M * \dot{x}^2) + \tfrac{1}{2}(m_1(\dot{x}^2 - 2l_1 cos(\theta_1)\dot{\theta 1}\dot{x} + l_1^2\dot{\theta}_1^2) + \\ &\tfrac{1}{2}(m_2(\dot{x}^2 - 2l_2^2 cos(\theta_2)\dot{\theta_2}^2\dot{x} + l_2^2\dot{\theta_2}^2) \end{aligned} \tag{3}$$

The total system potential energy is obtained only from oscillating pendulums.

$$\begin{aligned} &V_{tot} = V_{m1} + V_{m2} \\ &V_{tot} = m_1 g l_1(1 - cos(\theta_1)) + m_2 g l_2(1 - cos(\theta_2) \end{aligned} \tag{4}$$

The Lagrange will new be the difference of the system's kinetic and potential energies.

$$L = T_{tot} - V_{tot} \tag{5}$$

$$\begin{aligned} &L = \tfrac{1}{2}(M * \dot{x}^2) + \tfrac{1}{2}(m_1(\dot{x}^2 - 2l_1 cos(\theta_1)\dot{\theta 1}\dot{x} + l_1^2\dot{\theta}_1^2) + \tfrac{1}{2}(m_2(\dot{x}^2 - 2l_2^2 cos(\theta_2)\dot{\theta_2}^2\dot{x} + l_2^2\dot{\theta_2}^2) \\ &-m_1 g l_1(1 - cos(\theta_1)) + m_2 g l_2(1 - cos(\theta_2) \end{aligned} \tag{6}$$

We have three degrees of freedom, which means the variable q from Euler-Lagrange in our case will be x, θ_1, and θ_2. As a result differentiating the Langrange equation interms, we can now compute each terms in 2 to obtain equation of motions of the system. . First, differentiating with respect to x, we get:

$$\frac{\partial L}{\partial x} = 0 \tag{7}$$

as x only defines horizontal motion, ie has no potential energy

Then, differentiating in terms of $\dot{x}$, we get:

$$\frac{\partial L}{\partial \dot{x}} = (M + m_1 + m_2)\dot{x} - m_1 l_1 cos(\theta_1)\dot{\theta_1} - m_2 l_2 cos(\theta_2)\dot{\theta_2} \tag{8}$$

we then take the derivative of resulting term with respect to time

$$\begin{aligned}&\tfrac{d}{dt}(\tfrac{\partial L}{\partial \dot{x}}) = (M + m_1 + m_2)\ddot{x} + m_1 l_1 sin(\theta_1)\dot{\theta_1}^2 \\ &-m_1 l_1 cos(\theta_1)\ddot{\theta_1} + m_2 l_2 sin(\theta_2)\dot{\theta_2}^2 - m_2 l_2 cos(\theta_2)\ddot{\theta_2}\end{aligned} \tag{9}$$

Finally, putting together the two terms we obtain the contribution of the x term to the system's equation of motion as the following.

$$\begin{aligned}&(M + m_1 + m2)\ddot{x} + m_1 l_1 sin(\theta_1)\dot{\theta_1}^2 - m_1 l_1 cos(\theta_1)\ddot{\theta_1} + m_2 l_2 sin(\theta_2)\dot{\theta_2}^2 \\ &-m_2 l_2 cos(\theta_2)\ddot{\theta_2} = F\end{aligned} \tag{10}$$

Following the same steps for θ_1, potential energy contribution we get

$$\tfrac{\partial L}{\partial \theta_1} = m_1 l_1 sin(\theta_1)\dot{\theta_1}\dot{x} - m_1 g l_1 sin(\theta_1) \tag{11}$$

then from kinetic energy portion,

$$\tfrac{\partial L}{\partial \dot{\theta_1}} = -m_1 l_1 cos(\theta_1)\dot{x} + m_1 g l_1^2 \dot{\theta_1} \tag{12}$$

which results as below after derived with time

$$\tfrac{d}{dt}(\tfrac{\partial L}{\partial \dot{\theta_1}}) = m_1 l_1 \ddot{\theta_1} - m_1 l_1 cos(\theta_1)\ddot{x} + m_1 l_1 sin(\theta_1)\dot{\theta_1}\dot{x} \tag{13}$$

Putting the euler-lagrange for θ_1 will be

$$m_1 l_1 cos(\theta_1)\ddot{x} - m_1 l_1 \ddot{\theta_1} - m_1 g l_1 sin(\theta_1) = 0 \tag{14}$$

Finally, θ_2's contribution can similarly be obtained, first by finding potential energy side, which is

$$\tfrac{\partial L}{\partial \theta_2} = m_2 l_2 sin(\theta_2)\dot{\theta_2}\dot{x} - m_2 g l_2 sin(\theta_2) \tag{15}$$

The Kinetic Energy contribution of θ_2 is then,

$$\tfrac{\partial L}{\partial \dot{\theta_2}} = -m_2 l_2 cos(\theta_2)\dot{x} + m_2 g l_2^2 \dot{\theta_2} \tag{16}$$

whose time derivative can be computed to be

$$\tfrac{d}{dt}(\tfrac{\partial L}{\partial \dot{\theta_2}}) = m_2 l_2 \ddot{\theta_2} - m_2 l_2 cos(\theta_2)\ddot{x} + m_2 l_2 sin(\theta_2)\dot{\theta_2}\dot{x} \tag{17}$$

The third and final equation of motion due to θ_2 is then

$$m_2 l_2 cos(\theta_2)\ddot{x} - m_2 l_2 \ddot{\theta_2} - m_2 g l_2 sin(\theta_2) = 0 \tag{18}$$

Putting the equations of motion together that describe the system, we now have

$$\begin{aligned}
&(M + m_1 + m2)\ddot{x} + m_1 l_1 sin(\theta_1)\dot{\theta_1}^2 - m_1 l_1 cos(\theta_1)\ddot{\theta_1} + m_2 l_2 sin(\theta_2)\dot{\theta_2}^2 \\
&-m_2 l_2 cos(\theta_2)\ddot{\theta_2} = F \\
&m_1 l_1 cos(\theta_1)\ddot{x} - m_1 l_1 \ddot{\theta_1} - m_1 g l_1 sin(\theta_1) = 0 \\
&m_2 l_2 cos(\theta_2)\ddot{x} - m_2 l_2 \ddot{\theta_2} - m_2 g l_2 sin(\theta_2) = 0
\end{aligned} \tag{19}$$

Having now obtained the systems of equations that describe motion of objects within this system, let's move on to designing the state spate representation

2.1.2 State Space Representation

Before creating the state space matrix representation, we need to rewrite the system of equations with respect to the highest order of differentials for each linearly independent variable, i.e. $\ddot{x}, \ddot{\theta_1}, \ddot{\theta_2}$. Hence, rearranging equations 10, 14, and 18 by substitution, we obtain the following three forms.

$$\ddot{x} = \frac{F - m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)}{M + m_1 sin(\theta_1)^2 + m_2 sin(\theta_2)^2} \tag{20}$$

$$\ddot{\theta_1} = \frac{Fcos(\theta_1) - m_1 sin(\theta_1)((l_1 cos(\theta_1)\dot{\theta_1}^2 + g) - m_2 sin(\theta_2)cos(\theta_1)(l_2\dot{\theta_2}^2 + gcos(\theta_2) - g\sin(\theta_1)(M + m_2 sin(\theta_2)^2}{l_1(M + m_1 sin(\theta_1)^2 + m_2 sin(\theta_2)^2)} \tag{21}$$

$$\ddot{\theta_2} = \frac{Fcos(\theta_2) - m_1 sin(\theta_1)cos(\theta_2)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2 cos(\theta_2)\dot{\theta_2}^2 + g) - g\sin(\theta_2)(M + m_1 sin(\theta_1)^2}{l_2(M + m_1 sin(\theta_1)^2 + m_2 sin(\theta_2)^2)} \tag{22}$$

We can now write this state space representation as nonlinear system.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta_1} \\ \ddot{\theta_1} \\ \dot{\theta_2} \\ \ddot{\theta_2} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{F - m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)}{M + m_1 sin(\theta_1)^2 + m_2 sin(\theta_2)^2} \\ \dot{\theta_1} \\ \frac{Fcos(\theta_1) - m_1 sin(\theta_1)((l_1 cos(\theta_1)\dot{\theta_1}^2 + g) - m_2 sin(\theta_2)cos(\theta_1)(l_2\dot{\theta_2}^2 + gcos(\theta_2) - g\sin(\theta_1)(M + m_2 sin(\theta_2)^2}{l_1(M + m_1 sin(\theta_1)^2 + m_2 sin(\theta_2)^2)} \\ \dot{\theta_2} \\ \frac{Fcos(\theta_2) - m_1 sin(\theta_1)cos(\theta_2)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2 cos(\theta_2)\dot{\theta_2}^2 + g) - g\sin(\theta_2)(M + m_1 sin(\theta_1)^2}{l_2(M + m_1 sin(\theta_1)^2 + m_2 sin(\theta_2)^2)} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta_1 \\ \dot{\theta_1} \\ \theta_2 \\ \dot{\theta_2} \end{bmatrix} \tag{23}$$

2.1.3 Linearizing The System

We can now linearize the state space at the equilibrium points, which at this stage are:

$$\begin{aligned} x &= 0 \\ \theta_1 &= 0 \\ \theta_2 &= 0 \end{aligned} \tag{24}$$

Following the above points considered, the remaining terms that cause non-linearity in the system either by taking the system variables as inputs or taking their derivatives will in turn be:

$$\begin{aligned} \dot{\theta_1} &= 0 \\ \dot{\theta_2} &= 0 \\ sin(\theta_1) &= 0 \\ sin(\theta_2) &= 0 \\ cos(\theta_1) &= 1 \\ cos(\theta_2) &= 1 \end{aligned} \tag{25}$$

Now we use the Jacobian method to take derivative of each element in the nonlinear A matrix with respect to the state variables, and then insert the equilibrium point values along with the assumed non linear terms to obtain linearized system.

Let's assume $f(x,u)$ to be the nonlinear matrix. It's Jacobian derivative with respect to x will give the A matrix as shown below.

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial f(1)}{\partial \theta_1} & \frac{\partial F}{\partial \dot{\theta_1}} & \frac{\partial f(1)}{\partial \theta_2} & \frac{\partial f(1)}{\partial \dot{\theta_2}} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\partial f(3)}{\partial \theta_1} & \frac{\partial f(3)}{\partial \dot{\theta_1}} & \frac{\partial f(3)}{\partial \theta_2} & \frac{\partial f(3)}{\partial \dot{\theta_2}} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{\partial f(3)}{\partial \theta_1} & \frac{\partial f(3)}{\partial \dot{\theta_1}} & \frac{\partial f(3)}{\partial \theta_2} & \frac{\partial f(3)}{\partial \dot{\theta_2}} \end{bmatrix} \tag{26}$$

Next, we would first derive the nonlinear elements in F accordingly, and then substitute the equilibrium values. Below are the expressions for the partial terms in the A matrix.

For neater expressions, let's take $\Omega = M + m_1 sin(\theta_1)^2 + m_2 sin(\theta_2)^2$

For $\ddot{x}$ term

Partial by θ_1

$$\frac{\partial f(1)}{\partial \theta_1} = \frac{\Omega(-m_1 cos(\theta_1)(l_1 \dot{\theta_1}^2 + gcos(\theta_1)) - m_1 sin(\theta_1)(-gsin(\theta_1)))}{\Omega^2} \frac{-2m_1 sin(\theta_1)cos(\theta_1)(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{\Omega^2} \tag{27}$$

Linearized we get ...

$$\frac{\partial f(1)}{\partial \theta_1} = \frac{-m_1 g}{\Omega} \tag{28}$$

Partial by $\dot{\theta_1}$

$$\frac{\partial f(1)}{\partial \dot{\theta_1}} = \frac{\Omega(-2m_1 sin(\theta_1)l_1\dot{\theta_1}) - 0(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{\Omega^2} \tag{29}$$

Linearized we get ...

$$\frac{\partial f(1)}{\partial \dot{\theta_1}} = 0 \tag{30}$$

Partial by θ_2

$$\frac{\partial f(1)}{\partial \theta_2} = \frac{\Omega(-m_2 cos(\theta_2)(l_2 \dot{\theta_2}^2 + gcos(\theta_2)) - m_2 sin(\theta_2)(-gsin(\theta_2)))}{\Omega^2} \frac{-2m_2 sin(\theta_2)cos(\theta_2)(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{\Omega^2} \tag{31}$$

Linearized we get ...

$$\frac{\partial f(1)}{\partial \theta_2} = \frac{-m_2 g}{\Omega} \tag{32}$$

Partial by $\dot{\theta_2}$

$$\frac{\partial f(1)}{\partial \dot{\theta_2}} = \frac{\Omega(-2m_2 sin(\theta_2)l_2\dot{\theta_2}) - 0(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{\Omega^2} \tag{33}$$

Linearized we get ...

$$\frac{\partial f(1)}{\partial \dot{\theta_2}} = 0 \tag{34}$$

For $\ddot{\theta_1}$ term

Partial by θ_1

$$\frac{\partial f(2)}{\partial \theta_1} = \frac{l_1\Omega(-m_1 cos(\theta_1)(l_1 cos(\theta_1)\dot{\theta_1}^2 + g) - m_1 sin(\theta_1)(-sin(\theta_1)l_1\dot{\theta_1}^2) + m_2 sin(\theta_2)sin(\theta_1)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) - gcos(\theta_1))*}{l_1^2\Omega^2} \frac{(M + m_2 sin(theta_2)^2) - Fsin(\theta_1))}{1} - \frac{2l_1 m_1 sin(\theta_1)cos(\theta_1)(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{l_1^2\Omega^2} \tag{35}$$

Linearized we get ...

$$\frac{\partial f(2)}{\partial \theta_1} = \frac{-g(m_1+M)}{l_1\Omega} \tag{36}$$

Partial by $\dot{\theta_1}$

$$\frac{\partial f(2)}{\partial \dot{\theta_1}} = \frac{l_1\Omega(-2m_1 sin(\theta_1)l_1 cos(\theta_1)\dot{\theta_1}) - 0(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{l_1^2\Omega^2} \tag{37}$$

Linearized we get ...

$$\frac{\partial f(2)}{\partial \dot{\theta_1}} = 0 \tag{38}$$

Partial by θ_2

$$\frac{\partial f(2)}{\partial \theta_2} = \frac{l_1\Omega(-m_2 cos(\theta_2)cos(\theta_1)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) - m_2 sin(\theta_2)cos(\theta_1)(-gsin(\theta_2)) - 2gsin(\theta_1)m_2 sin(\theta_2)cos(\theta_2))}{l_1^2\Omega^2}$$
$$- \frac{2m_2 sin(\theta_2)cos(\theta_2)l_1(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{l_1^2\Omega^2} \tag{39}$$

Linearized we get ...

$$\frac{\partial f(2)}{\partial \theta_1} = \frac{-g(m_2)}{l_1\Omega} \tag{40}$$

Partial by $\dot{\theta_2}$

$$\frac{\partial f(2)}{\partial \dot{\theta_2}} = \frac{l_1\Omega(-2m_2 sin(\theta_2)cos(\theta_1)l_2\dot{\theta_2}) - 0(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{l_1^2\Omega^2} \tag{41}$$

Linearized we get ...

$$\frac{\partial f(2)}{\partial \dot{\theta_2}} = 0 \tag{42}$$

Last, for $\ddot{\theta_2}$ term

Partial by θ_1

$$\frac{\partial f(3)}{\partial \theta_1} = \frac{l_2\Omega(-m_1 cos(\theta_1)cos(\theta_2)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_1 sin(\theta_1)cos(\theta_1)(-gsin(\theta_1)) - 2gsin(\theta_2)m_1 sin(\theta_1)cos(\theta_1))}{l_2^2\Omega^2}$$
$$- \frac{2m_1 sin(\theta_1)cos(\theta_1)l_2(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{l_2^2\Omega^2} \tag{43}$$

Linearized we get ...

$$\frac{\partial f(3)}{\partial \theta_1} = \frac{-g(m_1)}{l_2\Omega} \tag{44}$$

Partial by $\dot{\theta}_1$

$$\frac{\partial f(3)}{\partial \dot{\theta_1}} = \frac{l_2\Omega(-2m_1 sin(\theta_1)cos(\theta_2)l_1\dot{\theta_1}) - 0(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{l_1^2\Omega^2} \tag{45}$$

Linearized we get ...

$$\frac{\partial f(3)}{\partial \dot{\theta_1}} = 0 \tag{46}$$

Partial by θ_2

$$\frac{\partial f(3)}{\partial \theta_2} = \frac{l_2\Omega(-m_2 cos(\theta_2)(l_2 cos(\theta_2)\dot{\theta_2}^2 + g) - m_2 sin(\theta_2)(-sin(\theta_2)l_2\dot{\theta_2}^2) + m_1 sin(\theta_1)sin(\theta_2)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - gcos(\theta_2))*}{l_2^2\Omega^2} \frac{(M + m_1 sin(theta_1)^2) - F sin(\theta_2))}{1} - \frac{2l_2 m_2 sin(\theta_2)cos(\theta_2)(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2)) + F)}{l_2^2\Omega^2} \tag{47}$$

Linearized we get ...

$$\frac{\partial f(3)}{\partial \theta_2} = \frac{-g(m_2 + M)}{l_2\Omega} \tag{48}$$

Partial by $\dot{\theta}_2$

$$\frac{\partial f(3)}{\partial \dot{\theta_2}} = \frac{l_2\Omega(-2m_2 sin(\theta_2)l_2 cos(\theta_2)\dot{\theta_2})}{l_2^2\Omega^2} - \frac{0(-m_1 sin(\theta_1)(l_1\dot{\theta_1}^2 + gcos(\theta_1)) - m_2 sin(\theta_2)(l_2\dot{\theta_2}^2 + gcos(\theta_2))}{l_2^2\Omega^2} \tag{49}$$

Linearized we get ...

$$\frac{\partial f(3)}{\partial \dot{\theta_2}} = 0 \tag{50}$$

Similarly, we can linearize the B matrix by taking Jacobian of F with respect to the input F. This will give us

$$B = \frac{\partial F}{\partial F} = \begin{bmatrix} 0 \\ \frac{1}{\Omega} \\ 0 \\ \frac{cos(\theta)}{l_1\Omega} \\ 0 \\ \frac{cos(\theta_2}{l_2\Omega} \end{bmatrix} \tag{51}$$

Linearizing at the equilibrium points and assumed variables will then yield.

$$B = \frac{\partial f}{\partial F} = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{cos(\theta)}{l_1M} \\ 0 \\ \frac{cos(\theta_2}{l_2M} \end{bmatrix} \tag{52}$$

On the same token, our linearized A matrix will now be

$$A_{lin} = \frac{\partial f}{\partial F} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-m_1g}{M} & 0 & \frac{-m_2g}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-g(m_1+M)}{l_1M} & 0 & \frac{-m_2g}{l_1M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & \frac{-gm_1}{l_2M} & 0 & \frac{-g(m_2+M)}{l_2M} & 0 \end{bmatrix} \tag{53}$$

The state representation in full will then be

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-m_1g}{M} & 0 & \frac{-m_2g}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-g(m_1+M)}{l_1M} & 0 & \frac{-m_2g}{l_1M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & \frac{-gm_1}{l_2M} & 0 & \frac{-g(m_2+M)}{l_2M} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{cos(\theta)}{l_1M} \\ 0 \\ \frac{cos(\theta_2}{l_2M} \end{bmatrix} Fu(\vec{t}) \tag{54}$$

2.1.4 Checking for Controllability

The linearized system now shows to be linear time invariant, and thus can be verified of its controllability via the rank test. The controllability matrix can be computed as

$$C_t = [B; AB; A^2B; A^3B; A^4B; A^5B] \tag{55}$$

reference next section for the result of the controllability matrix. Following result, we then compute the rank of this matrix to determine if the linearized system is controllable, ie full rank meaning controllable and not if other wise

$$rank(C_t) = [B; AB; A^2B; A^3B; A^4B; A^5B] \tag{56}$$

As seen in the controllability matrix in section following section, columns 5 and 6 are scalar multiples of columns 3 and 4 respectively. Therefore, the conditions that retain the full rank of the controllability matrix keeping the system controllable are

$$M, m_1, m_2, l_1, l_2 > 0 \tag{57}$$

and l_1 is not equal to l_2

As a result, with these conditions, the system is controllable.

2.2 LQR Controller

To achieve controllability in a system, the choice of controllers depends on the system model, where both PID and LQR controllers play significant roles. The PID controller employs a classical linear equation approach, while the LQR controller focuses on non-linear models and offers an optimal state-feedback law, minimizing a quadratic objective function. In the case of the LQR Controller, when A and Bk are stabilizable, the objective is to find a value for k that minimizes the cost function J, defined as the integral from 0 to infinity of a combination of state and control input terms. This cost function is represented as:

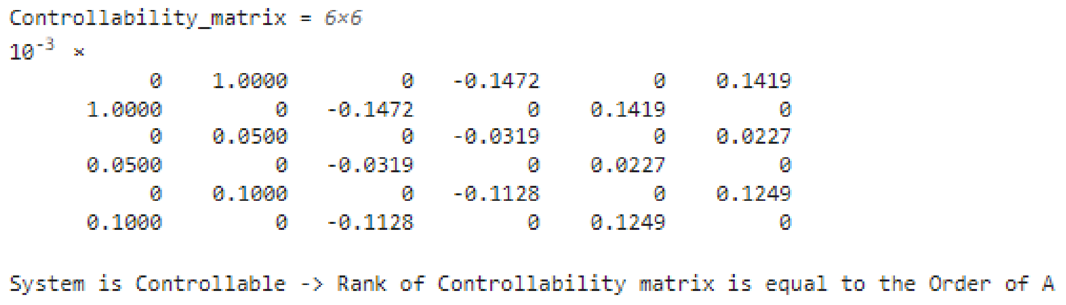
$$J(k, \vec{\mathbf{X}}(0)) = \int_0^\infty [\vec{\mathbf{X}}^T(t)Q\vec{\mathbf{X}}(t) + \vec{\mathbf{U}}_k^T(t)R\vec{\mathbf{U}}_k(t)]\, dt$$

Where Q penalizes poor system performance, and R penalizes actuator effort or energy consumption. The optimal control of the system involves selecting different values for Q and R, and validation through output graphs helps determine the effectiveness of these choices. The Q matrix can be adjusted based on the desired stabilization time for the system and the acceptable error range, while a lower value of R contributes to faster stability by reducing energy consumption.

2.3 Checking for Controllability

Before we begin with setting up the controller for our system, we need to first check if the system is controllable or not. For our system with M = 1000Kg, m1 = m2 = 100Kg, l1 = 20m and l2 = 10m we check if our system is controllable by comparing the rank of the Controllability matrix with the order of matrix A. If they are equal, in our case the value for equality would be 6 since A has a order of 6, then we can say that the system is controllable.

```
% Controllability
Controllability_matrix =  ctrb(A, B)
```



```
Controllability_matrix = 6×6
10^-3 ×
         0    1.0000         0   -0.1472         0    0.1419
    1.0000         0   -0.1472         0    0.1419         0
         0    0.0500         0   -0.0319         0    0.0227
    0.0500         0   -0.0319         0    0.0227         0
         0    0.1000         0   -0.1128         0    0.1249
    0.1000         0   -0.1128         0    0.1249         0

System is Controllable -> Rank of Controllability matrix is equal to the Order of A
```

Figure 2: Controllability matrix

2.4 Response to initial conditions

We obtain the initial condition responses for the open-loop linear system, then for the closed-loop linear system with the LQR controller, and at last for the closed-loop non-linear system with the LQR controller. Furthermore, for the closed loop linear system case, we obtain the initial response for non-optimal Q and R and also for the optimal Q and R. We can clearly see that the optimal controller performs much better.

```
% initial conditions, distance is in meters and angles is in radians
% X = 0.1, theta1 = —0.19 and theta2 = 0.31
x_0 = [0.1; 0; —0.19; 0; 0.31; 0];
```

2.4.1 Open Loop Linear System

```
% Compute state space representation using ss function
open_loop_system = ss(A,B,C,D);
```

2.4.2 Closed Loop Linear System

From the above plot we can see that the system state is unstable.

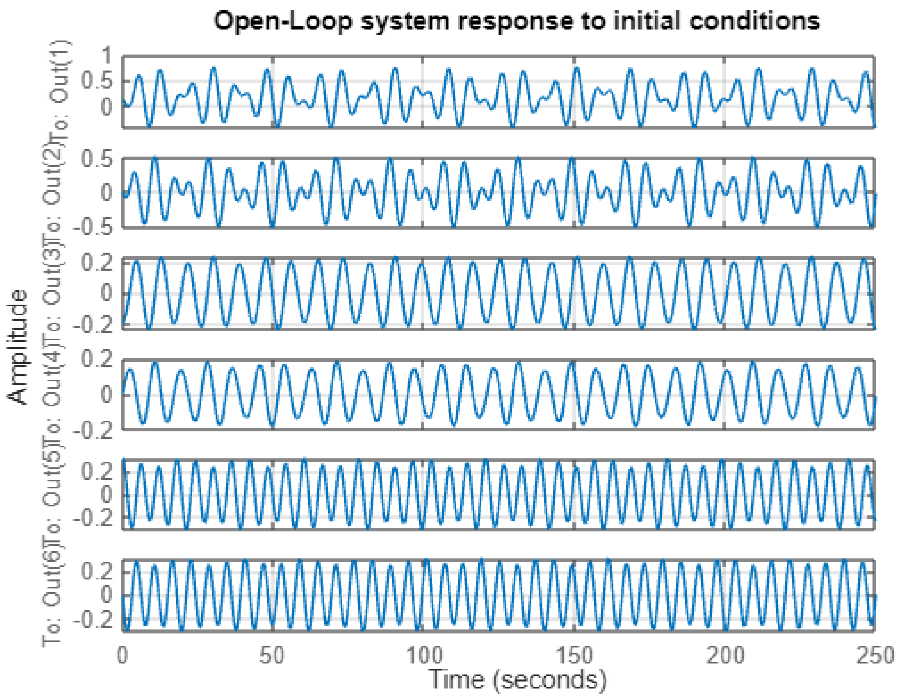


Figure 3: Open loop response to initial condition

```
% LQR controller [1. Arbitrary Q and R]
Q = diag([1, 1, 1, 1, 1, 1]);
R = 0.001;
% LQR Gain Matrix
K = lqr(A, B, Q, R);
closed_loop_system = ss(A—(B*K),B,C,D);
```

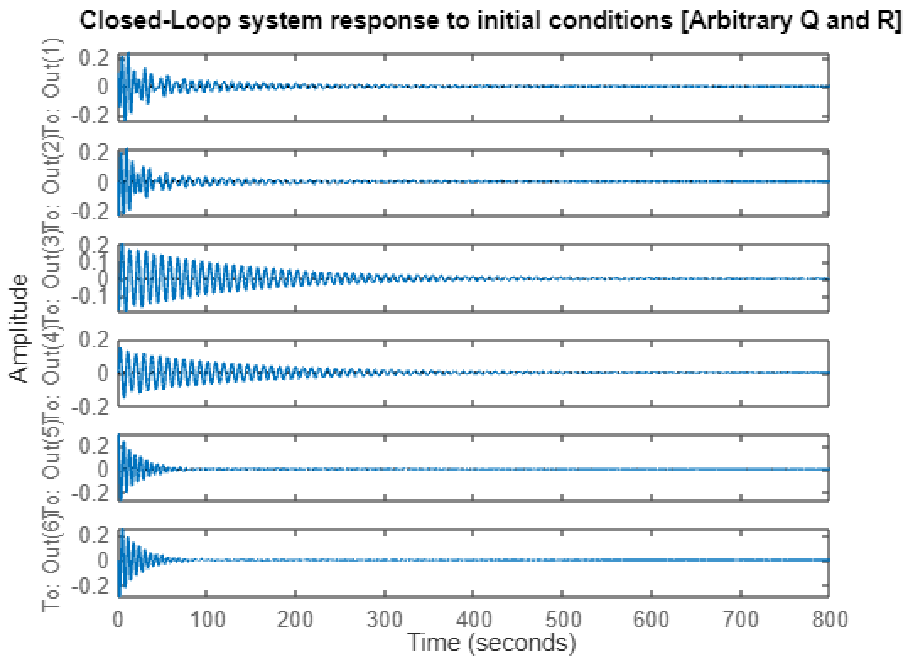


Figure 4: Closed loop response to initial condition with arbitrary Q and R

From the above plot we can see that the system gets stables but there is still high frequency vibrations in the states.

```
% LQR controller design [Arbitrary Q and R]
Q = diag([100, 50, 10, 1, 100, 10]);
R = 0.00001;

% LQR Gain Matrix
K = lqr(A, B, Q, R);

closed_loop_system = ss(A−(B∗K),B,C,D);
x_0 = [0.1; 0; −0.19; 0; 0.31; 0];
```

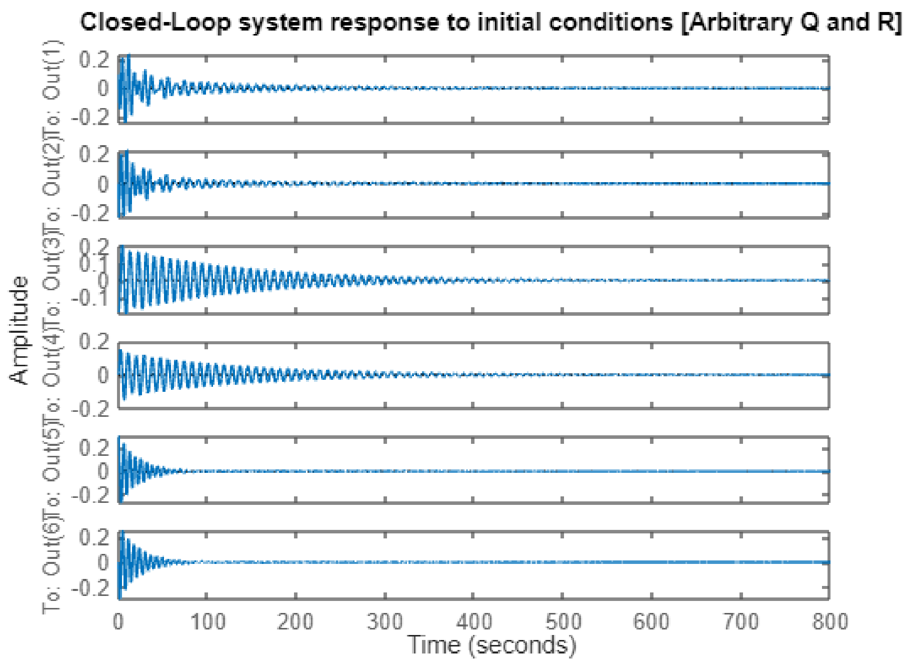


Figure 5: Closed loop response to initial condition with arbitrary Q and R

We can see from the above plot that the system response is better but still the system states do vibrate before getting stable.

```
% Optimal Q and R on system response
Q = diag([316, 315, 320, 800, 320, 800]);
R = 0.001;
```

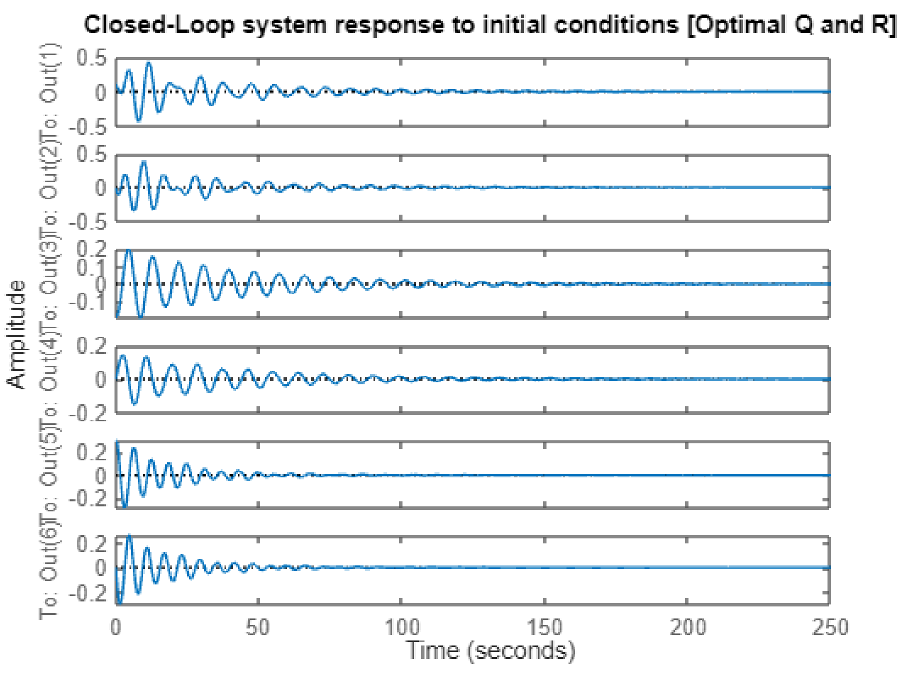


Figure 6: Closed loop response to initial condition with optimal Q and R

2.4.3 Closed Loop Non-Linear System

```
%Checking the initial response of the closed—loop non—linear system
x_0 = [0.1; 0.0; —0.19; 0.0; 0.31; 0];
% ODE solver for non—linear model
ode_nonlinear = @(t, y) cart_model_non_linear(t, y, K, A, B);
[t_nonlinear, solution_nonlinear] = ode45(ode_nonlinear, [0, 200],x_0 );

function dydt = cart_model_non_linear(t, y, K, A, B) % Function to compute non
    linear dynamics of the system
% Control Input
u = K * y;

%Mass of Crane
Mass_Crane= 1000;

% Mass & Cable length of First Load
mass_1= 100;
length_1= 20;
% Mass & Cable length of Second Load
mass_2= 100;
length_2= 10;
```

```
% Gravity component
gravity= 9.81;

% Unpack state variables
d1 = y(2);

% Compute additional terms for the nonlinear dynamics
D_D = Mass_Crane + mass_1 + mass_2 - mass_1 * cos(y(3))^2 - mass_2 * cos(y(5))^2;
term1 = mass_1 * gravity * sin(2 * y(3)) / 2;
term2 = mass_2 * gravity * sin(2 * y(5)) / 2;
term3 = mass_1 * length_1 * y(4)^2 * sin(y(3));
term4 = mass_2 * length_2 * y(6)^2 * sin(y(5));

% Compute state derivatives
d2 = -(1 / D_D) * (term1 + term2 + term3 + term4 + u);
d3 = y(4);
d4 = (1 / length_1) * (d2 * cos(y(3)) - gravity * sin(y(3)));
d5 = y(6);
d6 = (1 / length_2) * (d2 * cos(y(5)) - gravity * sin(y(5)));

% Pack state derivatives into a column vector
dydt = [d1; d2; d3; d4; d5; d6];
end
```

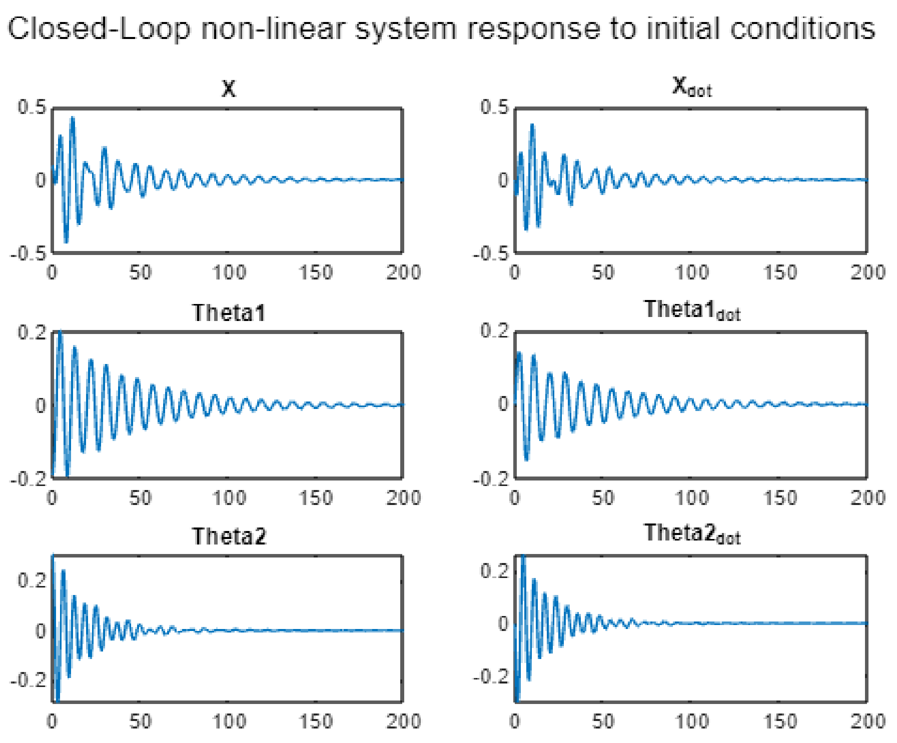


Figure 7: Non linear system response to initial conditions

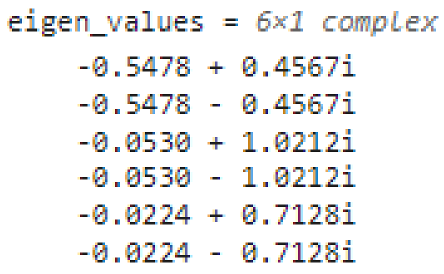
Here, we see that the system oscillates for some time before reaching the equilibrium point. However, it is a good response considering that the system is non-linear to begin with.

2.5 Lyapunov Stability Analysis

In order to check for the stability of our closed-loop system, we use the Lyapunov Indirect method. In this method, we compute the eigen values of the system. If the real part of all the eigen values are negative, then we can conclude that the system is stable.

```
% Lyapunov Indirect Stability Analysis
% Real parts of all eigen values need to be negative inorder for the system
% to be stable
eigen_values = eig(A-B*K)
```

Here, we observe that all the eigen values are negative. Hence, our system is stable.



```
eigen_values = 6×1 complex
    -0.5478 + 0.4567i
    -0.5478 - 0.4567i
    -0.0530 + 1.0212i
    -0.0530 - 1.0212i
    -0.0224 + 0.7128i
    -0.0224 - 0.7128i
```

Figure 8: Eigen value of our system

3 Second Component:

3.0.1 Evaluating Observability

Considering the conditions previously determined that must hold for linearized system to be observable, a set of output vectors are evaluated against the system to check for observability. For each of the four cases, the observability matrix is computed and the observability rank test to performed to test for observability. The state equation can be determined observable if and only if the $np \times n$ observabiltiy matrix satisfies the following test:

$$rank\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = n \quad \text{or equivalently} \quad rank[C^T \quad A^T C^T \quad \cdots \quad A^{T^{n-1}} C^T\,] = n$$

For this specific problem the rank test is:

$$rank\begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \end{bmatrix}$$

Matrix A is defined as:

$$A_F = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & a1 & 0 & a2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & a3 & 0 & a4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & a5 & 0 & a6 & 0 \end{bmatrix} \quad \text{where} \quad \begin{cases} a1 = \ -m1 * g/M \\ a2 = \ -m2 * g/M \\ a3 = -g * (m1 + M)/(l1 * M) \\ a4 = (-m2 * g)/(l1 * M) \\ a5 = (-g * m1)/(l2 * M) \\ a6 = (-g * (m2 + M))/(l2 * M) \end{cases}$$

$$M = 1000kg, \qquad m1 = m2 = 100kg, \qquad l1 = 10m, \qquad l2 = 20m, \qquad g = -9.8m/s^2$$

For each output vector case, a controllability matrix C is constructed. The observability rank test is evaluated by constructing a matrix using the linearized A matrix and each C matrix to determine if the resulting observability matrix is full rank, which is n= 6 in this case. Each output case is explored below by constructing the observability matrices in MATLAB and the rank function to determine the rank. This can also be done by hand by checking for n linear independent columns in the matrix or verifying the determinant of the matrix is not 0 to satisfy observability, otherwise the matrix is not observable.

Output Vector Case	**Observability Matrix**	**Rank**	**Implication**
Case 1: $(x(t))$	$C1 = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$	6	Observable
Case 2: $(\theta_1(t), \theta_2(t))$	$C2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	6	Not Observable
Case 3: $(x(t), \theta_2(t))$	$C3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	4	Observable

Case 4: $(x(t), \theta_1(t), \theta_2(t))$	$C4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	6	Observable

As shown above, the output vectors for cases 1, 3, and 4 are observable.

3.0.2 Luenberger Observer Design and Simulation

For each observable output vector, an optimal Luenberger observer is designed. The blah blah blah

For each case an arbitrary set of eigenvalues are defined such that the system is stable. This implies all the eigenvalues, also referred to as poles throughout literature, are in the negative half of the plane. Different eigenvalues were experimented with to see how the pole placement affected the system initial response and step response. The set of poles analyzed in this section are the following:

The following assumed initial conditions and variables were also considered for designing the Luenberger Observer, which were determined in the first component of this problem analysis:

$$initial_{condition} = [0.1 \quad 0.0 \quad -0.19 \quad 0.0 \quad 0.31 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0]^T$$

$$Q = \begin{bmatrix} 316 & 0 & 0 & 0 & 0 & 0 \\ 0 & 315 & 0 & 0 & 0 & 0 \\ 0 & 0 & 320 & 0 & 0 & 0 \\ 0 & 0 & 0 & 800 & 0 & 0 \\ 0 & 0 & 0 & 0 & 320 & 0 \\ 0 & 0 & 0 & 0 & 0 & 800 \end{bmatrix}$$

$$R = 0.001$$

$$K = [562.1388 \quad 1317.3177 \quad 203.3900 \quad -745.0353 \quad 417.1770 \quad -338.5086]$$

Using MATLAB's place placement function *place* from the Control System Toolbox, the L gain matrix can be determined for each output case. The general syntax for the function is given:

$$L = place(A, B, p)$$

A is the state matrix, B is the input-to-state matrix, and p are the locations of the closed loop poles. The following Ls were derived for output vector cases 1, 3, and 4 respectively:

$$L_1 = [19.8000 \quad 160.3330 \quad -3770.8509 \quad -703.5286 \quad 3088.9699 \quad -753.5992]$$

$$L_2 = \begin{bmatrix} 12.2059 & 7.8472 & 54.1058 & 74.9210 & -101.8767 & -239.0906 \\ -47.9953 & -206.0665 & 0.5560 & 7.5941 & 4.3298 & 17.8970 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} 7.5652 & 0.3540 & -0.0000 & 14.1777 & 0.3286 & -0.9800 \\ 0.3747 & 7.2348 & -0.0000 & 1.3909 & 12.4233 & -0.0490 \\ -0.0000 & -0.0000 & 5.0000 & -0.0000 & -0.0980 & 5.1320 \end{bmatrix}$$

Similarly, the MATLAB Linear-Quadratic Regulator (LQR) design *lqr* function was leveraged to determine the K optimal closed-loop gain matrix. The linearized A and B matrices, as well as the Q and R matrices determined from the first component analysis are passed in as parameters to this function. Finally obtaining the linearized A and B matrices, as well as the K, L, and C matrices, a the closed-loop state space equation can be constructed such that A is 6×6, B_K is 6×1, C dimensions are defined above, K is 1×6, B_D is 6×1, and the L matrices are 1×6, 2×6, and 3×6, respectively for the three output cases. This possible by designing a stable outback feedback system using K derived from the full state feedback analysis and L found by the Luenberger Observer principles. A new closed loop state space system can be written as:

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

The system can also be written as:

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

$$\dot{X}_e(t) = (A - LC)X_e(t) + B_D U_D(t)$$
where $\dot{X}(t) = (A + B_K K)X(t) - B_K K X_e(t) + B_D U_D(t)$
$$X_e(t) = X(t) - \hat{X}(t) \ \ (Estimation\ Error)$$

The system matrices can be passed as parameters to MATLAB's State Space Model function *ss*, which returns a Control Systems Toolbox state-space model object:

$$sys = ss(A, B, C, D)$$

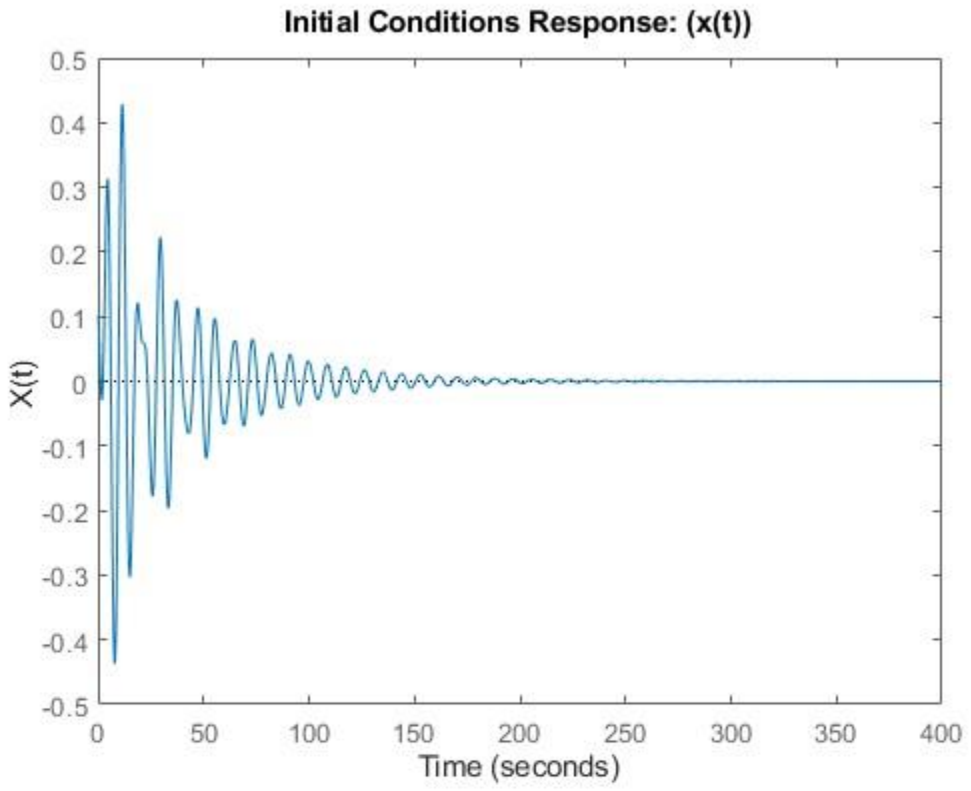
The state-space model object is then passed into the MATLAB's Control System Toolbox's *initial* function to plot the system response to initial conditions and system response to the step input. The general forms for these functions are:

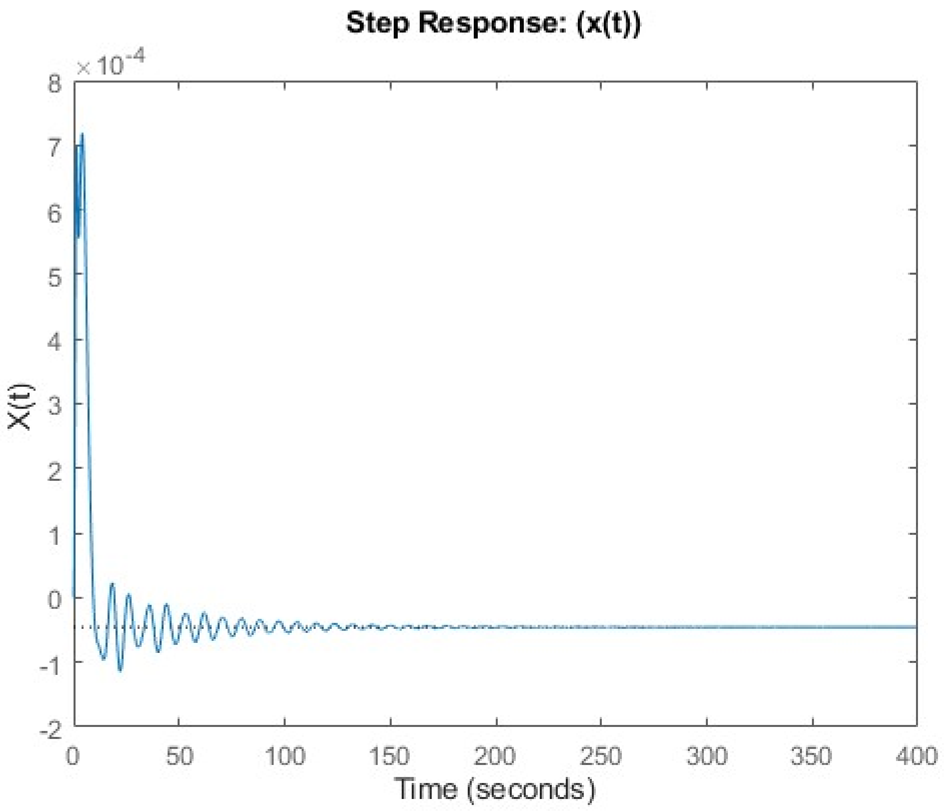
$$[y, tOut] = initial(sys, xinit, tFinal)$$

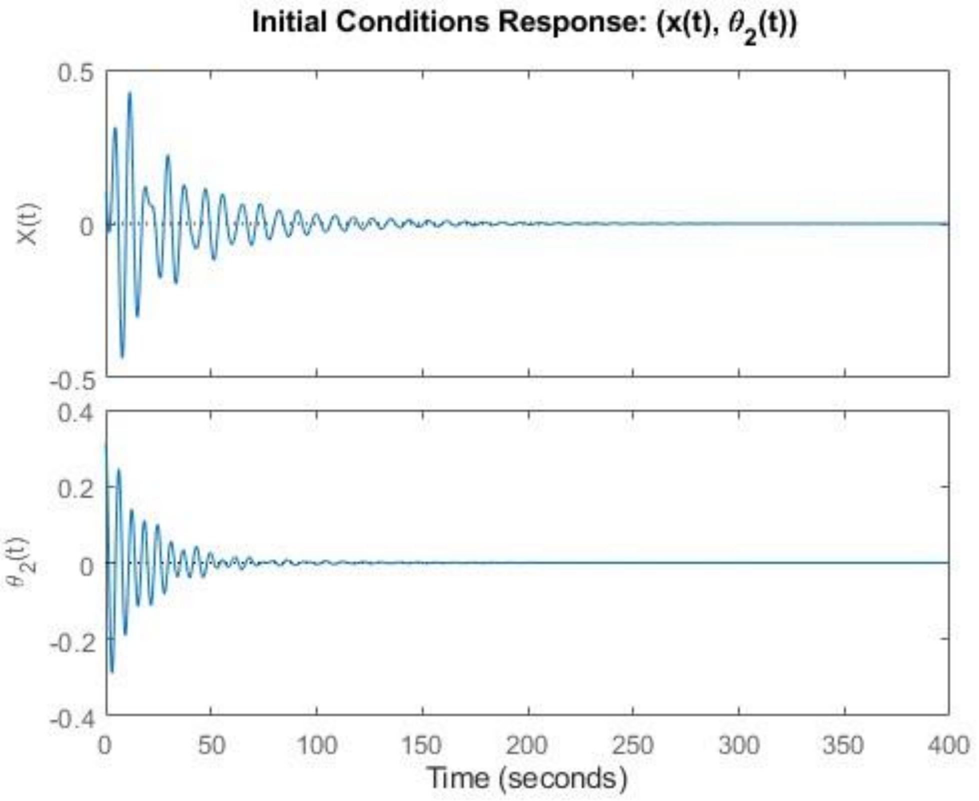
Where *sys* is the system model object, xinit is vector of initial conditions, and tFinal specifies to simulate the response from t = 0 to t = *tFinal*. The output vectors are data and time vectors respectively. The same system model object is passed into MATLAB's Control System Toolbox s*tep* function that simulates the system's step response to a step from time t=0 to time t = *tFinal*.

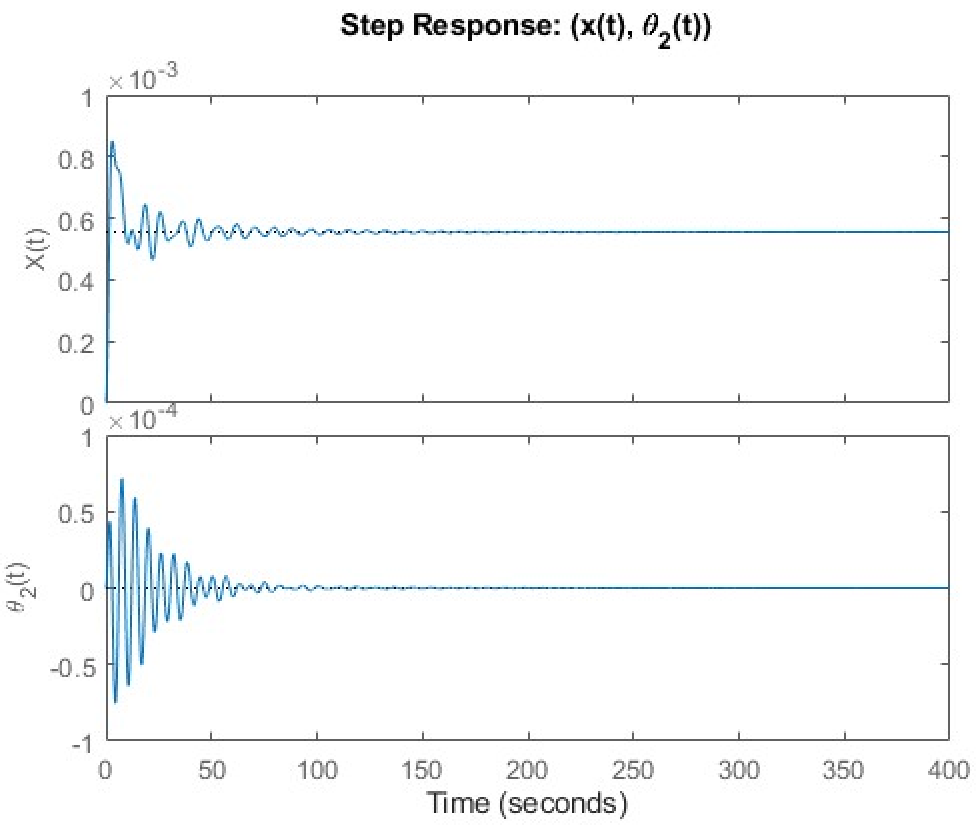
$$[y, tOut] = step(sys, tFinal)$$

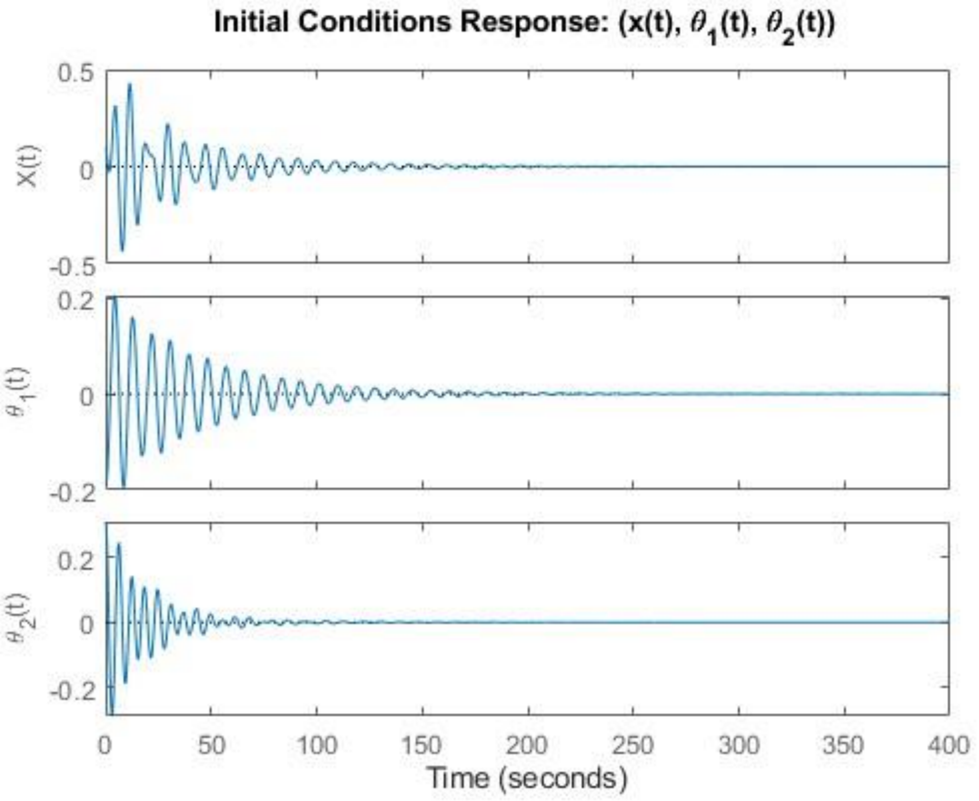
Provided below are generated plots for the linearized system for the initial conditions response and step response for each of the observable output vector cases determined in the section above.

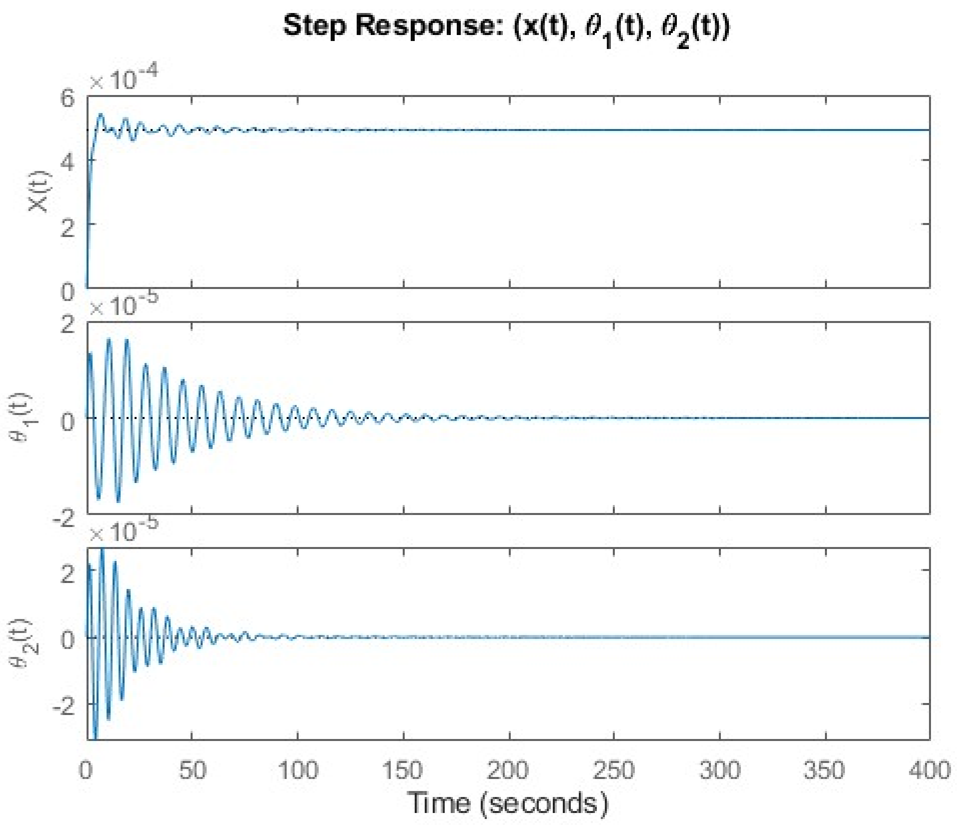



Step Response: (x(t))
$\times 10^{-4}$
8
7
6
5
4
3
2
1
0
-1
-2
X(t)
0
50
100
150
200
250
300
350
400
Time (seconds)


Initial Conditions Response: (x(t), θ_2(t))
0.5
0
-0.5
X(t)
0.4
0.2
0
-0.2
-0.4
θ_2(t)
0
50
100
150
200
250
300
350
400
Time (seconds)


Step Response: (x(t), θ_2(t))
$\times 10^{-3}$
X(t)
$\times 10^{-4}$
θ_2(t)
Time (seconds)

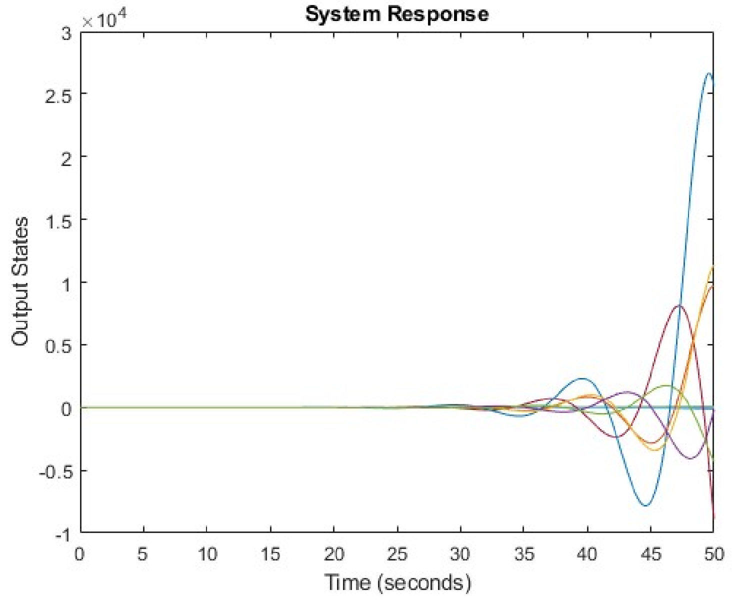

Initial Conditions Response: (x(t), θ_1(t), θ_2(t))
X(t)
θ_1(t)
θ_2(t)
Time (seconds)



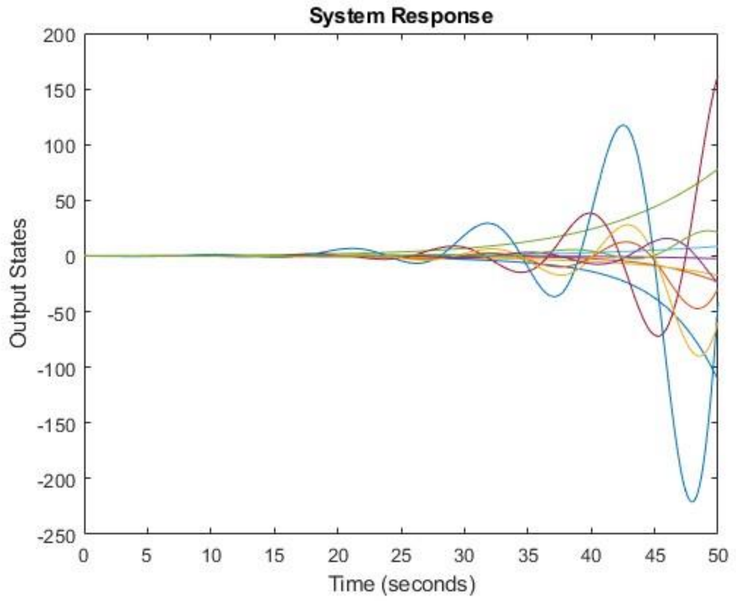
For simulating the nonlinear system in response to the initial conditions and step input, a different approach must be taken. For the case of a nonlinear system, the system matrices A, B, C, and D are not easily extracted as in the linear case. Instead, the general form for the nonlinear case is:

$$\dot{x} = f(x, u, t)$$
$$y = h(x, u, t)$$

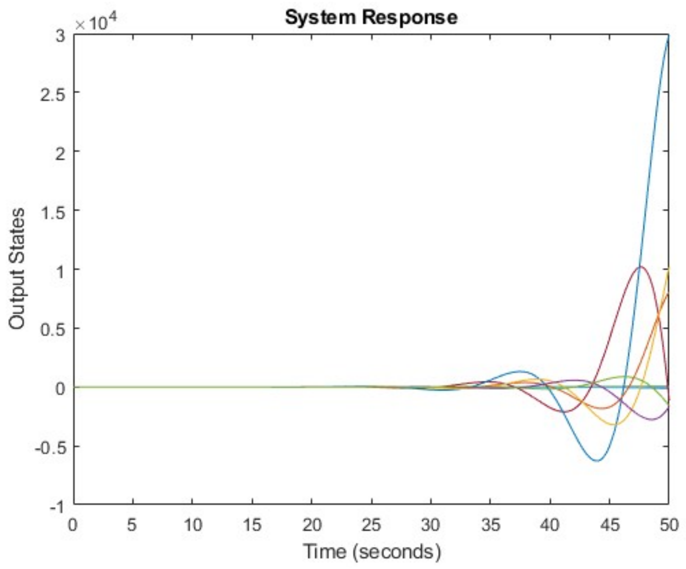
Thus, we can approach modeling the nonlinear simulations by leveraging the MATLAB ode45 function and tracking the state variable. The plots for nonlinear system as shown below for cases 1, 3, and 4 respectively. The nonlinear system response for case 1 is shown below.



The nonlinear system response for case 3 is shown below.



The nonlinear system response for case 4 is shown below.



This has concluded the simulation of both the linear and nonlinear systems. In the following sections we discuss LQG implementation.

3.1 Implementing LQG Control

Now that we have tested Luenberg observer, we have selected the $x(t), \theta_1, \theta_2$ to be our "smallest" output vector. The Linear Quadratic Gaussian is a combination of the Kalman filter (which estimates the state after taking the output Y and input of the system) and the LQR (which now takes the estimated state from the kalman filter and computes a controller K). The LQR outputs an input to the system which is fed back to the kalman filter thereby improving the state estimate.

So, the gain matrix for the kalman filter can be found by first computing the symmetric postive semi definite solution to the Riccati equation (P), and then plugging in the solution to the optimal gain equation (L^*; that is, first obtain P from

$$AP + PA^T + B_D \Sigma_D B_D^T - PC^T \Sigma_V^{-1} CP = 0 \tag{58}$$

and then compute L^* in

$$L = PC^T \Sigma_V^{-1} \tag{59}$$

Hence, once the kalman gain matrix is obtained, by combining it with the LQR system setup previously, one will now yield a state space representation of both actual states and the estimate or the error based on the wants of the designer. state space representation

$$\begin{bmatrix} \dot{x} \\ \dot{x_e} \end{bmatrix} = \begin{bmatrix} A + BK & -BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ x_e \end{bmatrix} + \begin{bmatrix} B_D \\ B_D \end{bmatrix} \vec{u_D}(t) \tag{60}$$

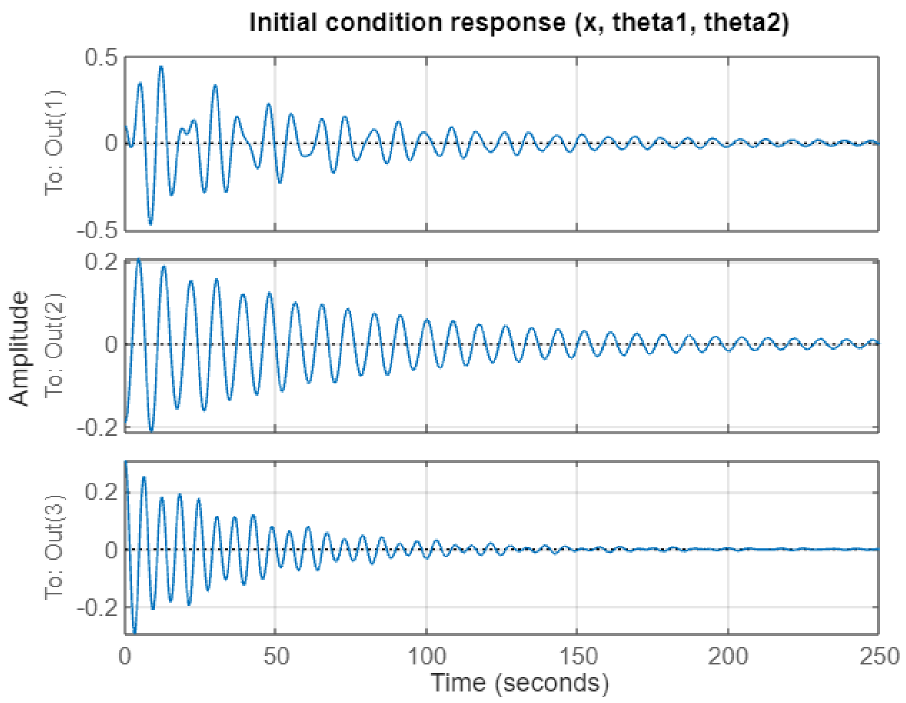


Figure 9: Initial condition response to LGQ controller

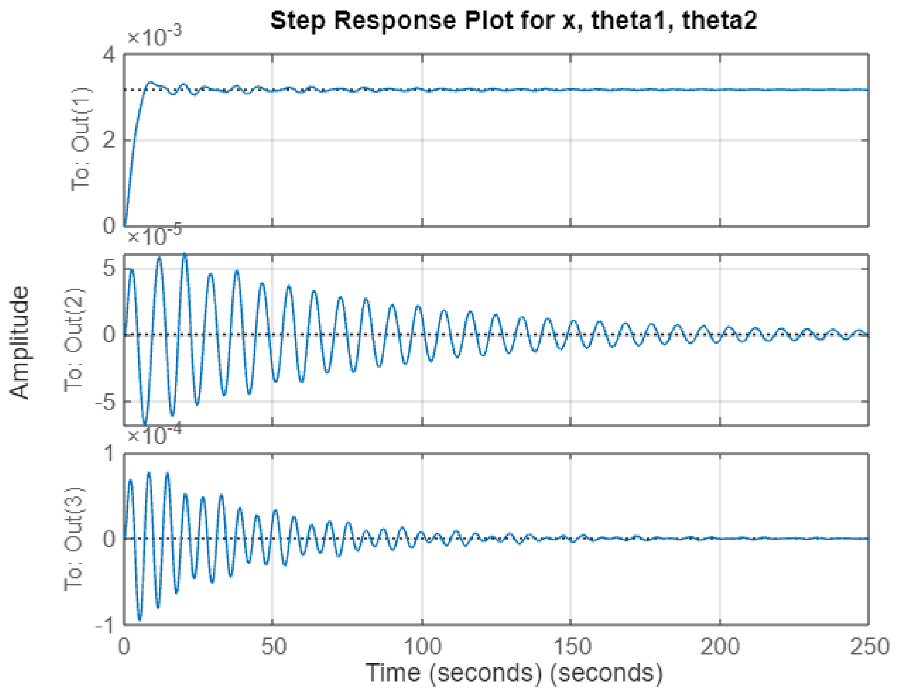


Figure 10: Step response to LQG controller

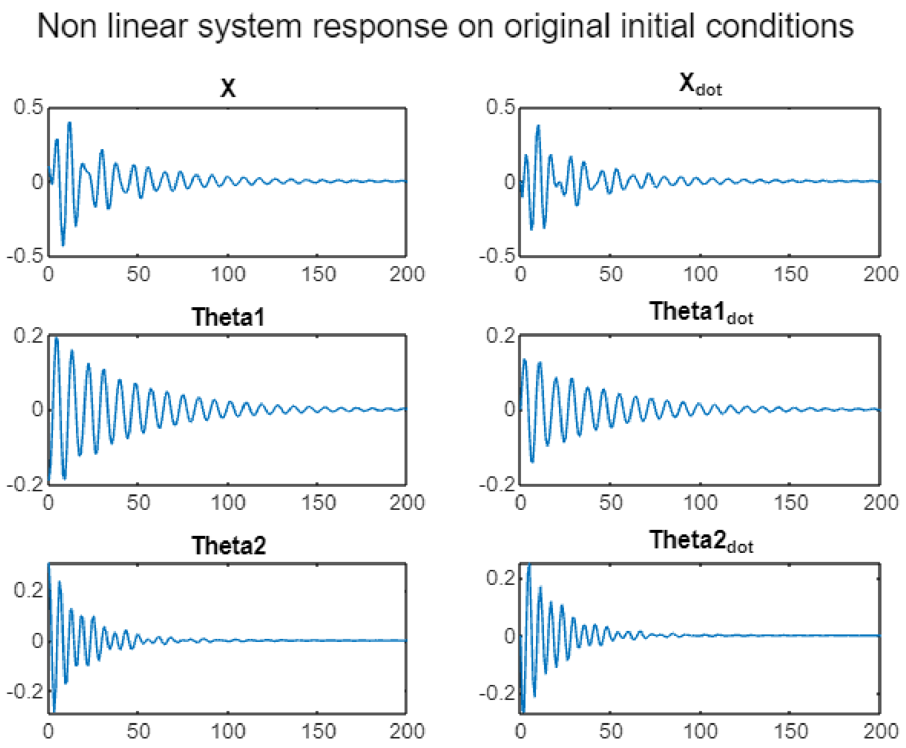


Figure 11: Non linear system response to initial conditions of LQG controller

Given the results from the generated plots we would reconfigure the controller to asymptotically track a constant reference x, say x= 5, by representing the constant reference trajectory which we assumed to be 0 throughout this problem analysis, in the state space equations. This leads to the state error to now equal the state minus constant reference trajectory. To address the question of if the designed system rejects applied disturbances, we observe the plots generated for part G. There is no overshooting and there is clear convergence. Hence, the design rejects constant force disturbances. Additional parameter tweaking can be performed to improve system response performance and disturbance rejection.

4 Appendix

Part D code

```
% Part D — LQR Controller


%Mass of Crane
Mass_Crane= 1000;


% Mass & Cable length of First Load
```

```
mass_1= 100;
length_1= 20;
% Mass & Cable length of Second Load
mass_2= 100;
length_2= 10;
% Gravity component
gravity= 9.81;

%The A and B matrices of our state space representation are as follows

% A matrix
A= [0 1 0 0 0 0;
    0 0 -(mass_1*gravity)/Mass_Crane 0 -(mass_2*gravity)/Mass_Crane 0;
    0 0 0 1 0 0;
    0 0 -((Mass_Crane+mass_1)*gravity)/(Mass_Crane*length_1) 0 -(mass_2*gravity)/(
        Mass_Crane*length_1) 0;
    0 0 0 0 0 1;
    0 0 -(mass_1*gravity)/(Mass_Crane*length_2) 0 -(gravity*(Mass_Crane+mass_2))/(
        Mass_Crane*length_2) 0];

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

% C matrix is Identity Matrix
C = eye(6);

% D matrix is zeros
D = zeros(6, 1);



% Our state space matrices are
A
B
C
D

% Controllability

Controllability_matrix =  ctrb(A, B)
```

```
if (rank(Controllability_matrix)~=6)
    disp("System is Uncontrollable —> Rank of Controllability matrix does not match
        the Order of A")

else
    disp("System is Controllable —> Rank of Controllability matrix is equal to the
        Order of A")
end

% initial conditions
% distance is in meters and angles is in radians
x_0 = [0.1; 0; —0.19; 0; 0.31; 0];

% Compute state space representation using ss function
open_loop_system = ss(A,B,C,D);

rng("default")
%Checking the initial response of the open—loop system
init_plot = initialplot(open_loop_system,x_0)
title('Open—Loop system response to initial conditions')

setoptions(init_plot,'TimeUnits','seconds','Grid','on');
setoptions(init_plot, 'XLim',[0 250]);



% LQR controller design [Arbitrary Q and R]

Q = diag([1, 1, 1, 1, 1, 1]);
R = 0.001;

% LQR Gain Matrix
K = lqr(A, B, Q, R);

closed_loop_system = ss(A—(B∗K),B,C,D);
x_0 = [0.1; 0; —0.19; 0; 0.31; 0];

rng("default")

%Checking the initial response of the closed—loop linear system
```

```
closed_loop_plot = initialplot(closed_loop_system,x_0)
title('Closed-Loop system response to initial conditions [Arbitrary Q and R]')

setoptions(init_plot,'TimeUnits','seconds','Grid','on');
setoptions(init_plot, 'XLim',[0 250]);

% LQR controller design [Arbitrary Q and R]

Q = diag([100, 50, 10, 1, 100, 10]);
R = 0.00001;

% LQR Gain Matrix
K = lqr(A, B, Q, R);

closed_loop_system = ss(A-(B*K),B,C,D);
x_0 = [0.1; 0; -0.19; 0; 0.31; 0];

rng("default")

%Checking the initial response of the closed-loop linear system
closed_loop_plot = initialplot(closed_loop_system,x_0)
title('Closed-Loop system response to initial conditions [Arbitrary Q and R]')

setoptions(init_plot,'TimeUnits','seconds','Grid','on');
setoptions(init_plot, 'XLim',[0 250]);

% Optimal Q and R on system response

Q = diag([316, 315, 320, 800, 320, 800]);
R = 0.001;

% LQR Gain Matrix
K = lqr(A, B, Q, R);

closed_loop_system = ss(A-(B*K),B,C,D);
x_0 = [0.1; 0; -0.19; 0; 0.31; 0];

rng("default")
```

```
%Checking the initial response of the closed-loop linear system
closed_loop_plot = initialplot(closed_loop_system,x_0)
title('Closed-Loop system response to initial conditions [Optimal Q and R]')

setoptions(init_plot,'TimeUnits','seconds','Grid','on');
setoptions(init_plot, 'XLim',[0 250]);

%%Checking the initial response of the closed-loop non-linear system

x_0 = [0.1; 0.0; -0.19; 0.0; 0.31; 0];
% ODE solver for non-linear model
ode_nonlinear = @(t, y) cart_model_non_linear(t, y, K, A, B);
[t_nonlinear, solution_nonlinear] = ode45(ode_nonlinear, [0, 200],x_0 );



figure;

subplot(3, 2, 1);
plot(t_nonlinear, solution_nonlinear(:,1));
title('X');

subplot(3, 2, 2);
plot(t_nonlinear, solution_nonlinear(:,2));
title('X_dot');

subplot(3, 2, 3);
plot(t_nonlinear, solution_nonlinear(:,3));
title('Theta1');

subplot(3, 2, 4);
plot(t_nonlinear, solution_nonlinear(:,4));
title('Theta1_dot');

subplot(3, 2, 5);
plot(t_nonlinear, solution_nonlinear(:,5));
title('Theta2');

subplot(3, 2, 6);
plot(t_nonlinear, solution_nonlinear(:,6));
```

```
title('Theta2_dot');

sgtitle('Closed-Loop non-linear system response to initial conditions');



% Lyapunov Indirect Stability Analysis

% Real parts of all eigen values need to be negative inorder for the system
% to be stable
eigen_values = eig(A-B*K)

function dydt = cart_model_non_linear(t, y, K, A, B) % Function to compute non
    linear dynamics of the system

    % Control Input
    u = K * y;



%Mass of Crane
Mass_Crane= 1000;

% Mass & Cable length of First Load
mass_1= 100;
length_1= 20;
% Mass & Cable length of Second Load
mass_2= 100;
length_2= 10;
% Gravity component
gravity= 9.81;



    % Unpack state variables
    d1 = y(2);

    % Compute additional terms for the nonlinear dynamics
    D_D = Mass_Crane + mass_1 + mass_2 - mass_1 * cos(y(3))^2 - mass_2 * cos(y(5))
        ^2;
    term1 = mass_1 * gravity * sin(2 * y(3)) / 2;
    term2 = mass_2 * gravity * sin(2 * y(5)) / 2;
    term3 = mass_1 * length_1 * y(4)^2 * sin(y(3));
    term4 = mass_2 * length_2 * y(6)^2 * sin(y(5));

    % Compute state derivatives
    d2 = -(1 / D_D) * (term1 + term2 + term3 + term4 + u);
    d3 = y(4);
    d4 = (1 / length_1) * (d2 * cos(y(3)) - gravity * sin(y(3)));
    d5 = y(6);
    d6 = (1 / length_2) * (d2 * cos(y(5)) - gravity * sin(y(5)));

    % Pack state derivatives into a column vector
    dydt = [d1; d2; d3; d4; d5; d6];
end
```

Part EF linear

```
%% Luenberger Observer Implementation References
% https://www.mathworks.com/help/control/ref/dynamicsystem.initial.html
% https://www.mathworks.com/help/ident/ref/dynamicsystem.step.html
% https://www.mathworks.com/help/control/ref/lti.lqr.html

%% Clear all workspaces and command lines
close all; clear; clc

%% User defined inputs can be changed here

%Define Output Case to evaluate
%Can choose values (cases) 1, 3, or 4
output_case = 4

%Define seconds in time to simulate response
sim_time_sec = 400;

% Define arbitrary poles
pole_values = [-2.3, -2.7, -3.1, -3.5, -3.9, -4.3];

% Define initial condition parameters
initial_condition = [0.1; 0.0; -0.19; 0.0; 0.31; 0; 0; 0; 0; 0; 0; 0];

% Use optimal R and Q symmetric postive matrices obtained from Component 1, Part D
```

```
Q = diag([316, 315, 320, 800, 320, 800]);
R = 0.001;

%% Define linearized state space variables and matrices

syms g M m1 m2 l1 l2
syms x x_dot th1 th1d th2 th2d

M = 1000;
m1 = 100;
m2 = 100;
l1 = 20;
l2 = 10;
g = -9.8;
a1 = m1*g/M;
a2 = m2*g/M;
a3 = g*(m1+M)/(l1*M);
a4 = m2*g/(l1*M);
a5 = g*m1/(l2*M);
a6 = g*(m2+M)/(l2*M);
AF = [0 1 0 0 0 0; 0 0 a1 0 a2 0; 0 0 0 1 0 0; 0 0 a3 0 a4 0; 0 0 0 0 0 1; 0 0 a5 0
    a6 0]
BF = transpose([0 1/M 0 1/(l1*M) 0 1/(l2*M)])
D = 0;

%% Define C
switch output_case
    case 1 % Y is function of x(t)
        c1r1 = [1 0 0 0 0 0];
        C1 = c1r1;
        C = C1;
    case 3 % Y is function of x(t) and th2(t)
        c3r1 = [1 0 0 0 0 0];
        c3r3 = [0 0 0 0 1 0];
        C3 = [c3r1; c3r3];
        C = C3;
    case 4 % Y is function of x(t) and th1(t) and th2(t)
        c4r1 = [1 0 0 0 0 0];
        c4r2 = [0 0 1 0 0 0];
        c4r3 = [0 0 0 0 1 0];
        C4 = [c4r1; c4r2; c4r3];
        C = C4;
end


%% Test for Observability

AF_T = transpose(AF);
C_T = transpose(C);

observe_mtx1 = [C; C*AF; C*AF^2; C*AF^3; C*AF^4; C*AF^5];

c1_rank = rank(observe_mtx1);
disp(["Case Rank: ", num2str(c1_rank)])


%% Compute L Gain Matrix
AF_transp = transpose(AF);
L = place(AF_transp,C_T,pole_values);
fprintf('L =\n');
fprintf('   %.4f   %.4f   %.4f   %.4f   %.4f   %.4f\n', L);
formattedString = sprintf('%.5e', L);

L_T = transpose(L);

%% Compute K

K = lqr(AF, BF, Q, R);
fprintf('K =\n');
fprintf('   %.4f   %.4f   %.4f   %.4f   %.4f   %.4f\n', K);

%% Compute new closed loop state space
A_closed = [AF-(BF*K) -BF*K; zeros(size(AF)) AF-(transpose(L)*C)];
B_closed = [BF;BF];
C_closed = [C zeros(size(C))];
D_closed = 0;

%% Pass in state space model matrices to ss function and plots responses
```

```
sys = ss(A_closed, B_closed, C_closed, D_closed);

%% Plot simulation results depending on the output case

if output_case == 4
    % Use step function to generate step input response over time_sec seconds
    figure;
    initial(sys, initial_condition, sim_time_sec);

    % Find all axes objects in the current figure
    axesHandles = findall(gcf, 'type', 'axes');
    % newLabel1 = 'Amplitude';
    newLabel2 = 'X(t)';
    newLabel3 = '\theta_1(t)'; % Assuming you intended different labels for the
        third subplot
    newLabel4 = '\theta_2(t)'; % Assuming you intended different labels for the
        third subplot
    set(get(axesHandles(1), 'YLabel'), 'String', '');
    set(get(axesHandles(2), 'YLabel'), 'String', newLabel4);
    set(get(axesHandles(3), 'YLabel'), 'String', newLabel3);
    set(get(axesHandles(4), 'YLabel'), 'String', newLabel2);
    title(axesHandles(1), "Initial Conditions Response: (x(t), \theta_1(t), \theta_2
        (t))")

    figure, step(sys, sim_time_sec)
    % Find all axes objects in the current figure
    axesHandles = findall(gcf, 'type', 'axes');
    % newLabel1 = 'Amplitude';
    newLabel2 = 'X(t)';
    newLabel3 = '\theta_1(t)'; % Assuming you intended different labels for the
        third subplot
    newLabel4 = '\theta_2(t)'; % Assuming you intended different labels for the
        third subplot
    set(get(axesHandles(1), 'YLabel'), 'String', '');
    set(get(axesHandles(2), 'YLabel'), 'String', newLabel4);
    set(get(axesHandles(3), 'YLabel'), 'String', newLabel3);
    set(get(axesHandles(4), 'YLabel'), 'String', newLabel2);
    title(axesHandles(1), "Step Response: (x(t), \theta_1(t), \theta_2(t))")
```

```
elseif output_case == 3

    % Use step function to generate step input response over time_sec seconds
    figure, initial(sys, initial_condition, sim_time_sec);

    axesHandles = findall(gcf, 'type', 'axes');
    newLabel1 = '';
    newLabel2 = 'X(t)';
    newLabel3 = '\theta_2(t)'; % Assuming you intended different labels for the
        third subplot
    set(get(axesHandles(1), 'YLabel'), 'String', newLabel1);
    set(get(axesHandles(2), 'YLabel'), 'String', newLabel3);
    set(get(axesHandles(3), 'YLabel'), 'String', newLabel2);
    title(axesHandles(1), "Initial Conditions Response: (x(t), \theta_2(t))")


    figure, step(sys, sim_time_sec)

    axesHandles = findall(gcf, 'type', 'axes');
    newLabel1 = '';
    newLabel2 = 'X(t)';
    newLabel3 = '\theta_2(t)'; % Assuming you intended different labels for the
        third subplot
    set(get(axesHandles(1), 'YLabel'), 'String', newLabel1);
    set(get(axesHandles(2), 'YLabel'), 'String', newLabel3);
    set(get(axesHandles(3), 'YLabel'), 'String', newLabel2);
    title(axesHandles(1), "Step Response: (x(t), \theta_2(t))")


elseif output_case == 1

    % Use step function to generate step input response over time_sec seconds
    figure, initial(sys, initial_condition, sim_time_sec);

    axesHandles = findall(gcf, 'type', 'axes');
    newLabel1 = 'X(t)';
    set(get(axesHandles(1), 'YLabel'), 'String', newLabel1);
    title(axesHandles(1), "Initial Conditions Response: (x(t))")
    figure, step(sys, sim_time_sec)

    axesHandles = findall(gcf, 'type', 'axes');
    newLabel1 = 'X(t)';
    set(get(axesHandles(1), 'YLabel'), 'String', newLabel1);
    title(axesHandles(1), "Step Response: (x(t))")

end
```

Part EF Non-linear

```
%% Luenberger Observer Implementation References
% https://www.mathworks.com/help/control/ref/dynamicsystem.initial.html
% https://www.mathworks.com/help/ident/ref/dynamicsystem.step.html
% https://www.mathworks.com/help/control/ref/lti.lqr.html
% https://www.mathworks.com/help/matlab/ref/ode45.html

%% Clear all workspaces and command lines
close all; clear; clc

%% User defined inputs can be changed here

%Define Output Case to evaluate
%Can choose values (cases) 1, 3, or 4
output_case = 4

%Define seconds in time to simulate response
sim_time_sec = 50;

% Define arbitrary poles
pole_values = [-2.3, -2.7, -3.1, -3.5, -3.9, -4.3];

% Define initial condition parameters
initial_condition = [0.1; 0.0; -0.19; 0.0; 0.31; 0; 0; 0; 0; 0; 0; 0];

% Use optimal R and Q symmetric postive matrices obtained from Component 1, Part D
Q = diag([316, 315, 320, 800, 320, 800]);
R = 0.001;
```

```
%% Define linearized state space variables and matrices

syms g M m1 m2 l1 l2
syms x x_dot th1 th1d th2 th2d

M = 1000;
m1 = 100;
m2 = 100;
l1 = 20;
l2 = 10;
g = -9.8;
a1 = m1*g/M;
a2 = m2*g/M;
a3 = g*(m1+M)/(l1*M);
a4 = m2*g/(l1*M);
a5 = g*m1/(l2*M);
a6 = g*(m2+M)/(l2*M);
AF = [0 1 0 0 0 0; 0 0 a1 0 a2 0; 0 0 0 1 0 0; 0 0 a3 0 a4 0; 0 0 0 0 0 1; 0 0 a5 0
    a6 0]
BF = transpose([0 1/M 0 1/(l1*M) 0 1/(l2*M)])
D = 0;

%% Define C
switch output_case
    case 1 % Y is function of x(t)
        c1r1 = [1 0 0 0 0 0];
        C1 = c1r1;
        C = C1;
    case 3 % Y is function of x(t) and th2(t)
        c3r1 = [1 0 0 0 0 0];
        c3r3 = [0 0 0 0 1 0];
        C3 = [c3r1; c3r3];
        C = C3;
    case 4 % Y is function of x(t) and th1(t) and th2(t)
        c4r1 = [1 0 0 0 0 0];
        c4r2 = [0 0 1 0 0 0];
        c4r3 = [0 0 0 0 1 0];
        C4 = [c4r1; c4r2; c4r3];
        C = C4;
end



%% Test for Observability

AF_T = transpose(AF);
C_T = transpose(C);

observe_mtx1 = [C; C*AF; C*AF^2; C*AF^3; C*AF^4; C*AF^5];

c1_rank = rank(observe_mtx1);
disp(["Case 1 Rank: ", num2str(c1_rank)])



%% Compute L Gain Matrix
AF_transp = transpose(AF);
L = place(AF_transp,C_T,pole_values); % L = acker(transpose(AF),C,pole_values)
fprintf('L =\n');
fprintf('   %.4f   %.4f   %.4f   %.4f   %.4f   %.4f\n', L);
formattedString = sprintf('%.5e', L);

L_T = transpose(L);

%% Compute K

K = lqr(AF, BF, Q, R);
fprintf('K =\n');
fprintf('   %.4f   %.4f   %.4f   %.4f   %.4f   %.4f\n', K);

%% Compute state space for the nonlinar system using ODE45
sim_time = 0:0.1:sim_time_sec;

[t,x] = ode45(@track_states,sim_time,initial_condition);

plot(t,x)
xlabel("Time (seconds)")
ylabel("Output States")
title("System Response")
```

```
%% ODE45 Function
function dydt = track_states(t,cl_state)

    % Provide covariance values
    process_cov = 0.1*eye(6);
    measure_cov = 1;

    % Redefine params
    R = 0.001;
    Q = diag([316, 315, 320, 800, 320, 800]);

    M = 1000;
    m1 = 100;
    m2 = 100;
    l1 = 20;
    l2 = 10;
    g = -9.8;

    a1 = m1*g/M;
    a2 = m2*g/M;
    a3 = g*(m1+M)/(l1*M);
    a4 = m2*g/(l1*M);
    a5 = g*m1/(l2*M);
    a6 = g*(m2+M)/(l2*M);

    AF = [0 1 0 0 0 0; 0 0 a1 0 a2 0; 0 0 0 1 0 0; 0 0 a3 0 a4 0; 0 0 0 0 0 1; 0 0
        a5 0 a6 0];
    BF = transpose([0 1/M 0 1/(l1*M) 0 1/(l2*M)]);
    D = 0;

    % Compute K Gain
    K = lqr(AF, BF, Q, R);

    % Compute input
    input = -(K*cl_state(1:6));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Modify depending on case 1, 3, or 4
    % Recompute C
    % c1r1 = [1 0 0 0 0 0];
    % C1 = c1r1;
    % C = C1;

    % c3r1 = [1 0 0 0 0 0];
    % c3r3 = [0 0 0 0 1 0];
    % C3 = [c3r1; c3r3];
    % C = C3;

    c4r1 = [1 0 0 0 0 0];
    c4r2 = [0 0 1 0 0 0];
    c4r3 = [0 0 0 0 1 0];
    C4 = [c4r1; c4r2; c4r3];
    C = C4;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Compute L Gain matrix
    LL = lqr(transpose(AF),transpose(C),process_cov,measure_cov);
    LL = transpose(LL);

    xe_dot = (AF—(LL*C)) * cl_state(7:12);

    % Track states below

    % Derive x(t) state variables
    dydt = []';
    dydt(12, 1) = 0;
    % x'
    dydt(1) = cl_state(2);
    % x''
    calc_1 = (input — (g/2)*(m1*sind(2*cl_state(3))+m2*sind(2*cl_state(5))) — (m1*l1
        *(cl_state(4)^2)*sind(cl_state(3)))—(m2*l2*(cl_state(6)^2)*sind(cl_state(5))
        ));
    calc_2 = (M+m1*((sind(cl_state(3)))^2)+m2*((sind(cl_state(5)))^2));
    dydt(2) = calc_1 / calc_2;
    % th1'
    dydt(3) = cl_state(4);
```

```
    % th1''
    dydt(4) = (dydt(2) * cosd(cl_state(3)) — g*(sind(cl_state(3))) ) / l1';
    % th2'
    dydt(5) = cl_state(6);
    % th2''

    % Derive Xe state variables
    dydt(6) = (dydt(2) * cosd(cl_state(5)) — g*(sind(cl_state(5)))) / l2;
    dydt(7) = cl_state(2) — cl_state(10);
    dydt(8) = dydt(2)—xe_dot(2);
    dydt(9) = cl_state(4) — cl_state(11);
    dydt(10) = dydt(4)—xe_dot(4);
    dydt(11) = cl_state(6) — cl_state(12);
    dydt(12) = dydt(6)—xe_dot(6);

end
```

Part G LQG

```
% Part G — LQG Controller

%Mass of Crane
Mass_Crane= 1000;

% Mass & Cable length of First Load
mass_1= 100;
length_1= 20;
% Mass & Cable length of Second Load
mass_2= 100;
length_2= 10;
% Gravity component
gravity= 9.81;

%The A and B matrices

% A matrix
A= [0 1 0 0 0 0;
    0 0 —(mass_1*gravity)/Mass_Crane 0 —(mass_2*gravity)/Mass_Crane 0;
    0 0 0 1 0 0;
    0 0 −((Mass_Crane+mass_1)∗gravity)/(Mass_Crane∗length_1) 0 −(mass_2∗gravity)/(
        Mass_Crane∗length_1) 0;
    0 0 0 0 0 1;
    0 0 −(mass_1∗gravity)/(Mass_Crane∗length_2) 0 −(gravity∗(Mass_Crane+mass_2))/(
        Mass_Crane∗length_2) 0];

B = [0; 1/Mass_Crane; 0; 1/(Mass_Crane∗length_1); 0; 1/(Mass_Crane∗length_2)];

% C matrix
C = eye(6);

% D matrix
D = zeros(6, 1);

Q = diag([100, 50, 10, 1, 100, 10]);
R = 0.001;

% Observable
C = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0];

D = 0;

% Initial Conditions for Leunberger observer
x_0 = [0.1 ;0;−0.19;0;0.31;0;0;0;0;0;0;0];



K =lqr(A,B,Q,R);
pro_n=0.28∗eye(6);
mes_noise=0.9;

A_t = A';
C_t = C';
L_0=lqr(A_t,C_t,pro_n,mes_noise);
L=L_0';

my_system = ss([(A−B∗K) B∗K; zeros(size(A)) (A−L∗C)], [B;zeros(size(B))],[C zeros(
    size(C))], D);

rng("default")
```

```
%Checking the initial response of the closed—loop linear system
init_plot = initialplot(my_system,x_0)
title('Initial condition response (x, theta1, theta2)')

setoptions(init_plot,'TimeUnits','seconds','Grid','on');
setoptions(init_plot, 'XLim',[0 250]);



figure
step(my_system);

grid on

% Limit x—axis values to 250 seconds
xlim([0 250])

% Add labels and title for better visualization
xlabel('Time (seconds)')
ylabel('Amplitude')
title('Step Response Plot for x, theta1, theta2')
% Non linear

tspan=0:1:200;
[t,x] = ode45(@lqg_slvr,tspan,x_0);
figure;

subplot(3, 2, 1);
plot(t, x(:,1));
title('X');

subplot(3, 2, 2);
plot(t, x(:,2));
title('X_dot');

subplot(3, 2, 3);
plot(t, x(:,3));
title('Theta1');
```

```
subplot(3, 2, 4);
plot(t, x(:,4));
title('Theta1_dot');

subplot(3, 2, 5);
plot(t, x(:,5));
title('Theta2');

subplot(3, 2, 6);
plot(t, x(:,6));
title('Theta2_dot');

sgtitle('Non linear system response on original initial conditions');

function diff_eq = lqg_slvr(t,x_ip)
Mass_Crane= 1000;

% Mass & Cable length of First Load
mass_1= 100;
length_1= 20;
% Mass & Cable length of Second Load
mass_2= 100;
length_2= 10;
% Gravity component
gravity= 9.81;

% A matrix
A= [0 1 0 0 0 0;
    0 0 -(mass_1*gravity)/Mass_Crane 0 -(mass_2*gravity)/Mass_Crane 0;
    0 0 0 1 0 0;
    0 0 -((Mass_Crane+mass_1)*gravity)/(Mass_Crane*length_1) 0 -(mass_2*gravity)/(
        Mass_Crane*length_1) 0;
    0 0 0 0 0 1;
    0 0 -(mass_1*gravity)/(Mass_Crane*length_2) 0 -(gravity*(Mass_Crane+mass_2))/(
        Mass_Crane*length_2) 0];

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

Q = diag([316, 315, 320, 800, 320, 800]);
```

```
R = 0.001;



C = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0]; % x , t1 , t2
D = 0;
K = lqr(A,B,Q,R);
u=-K*x_ip(1:6);

s_n=0.29*eye(6);
m_n=0.9;

A_tr = A';
C_tr = C';
L_0=lqr(A_tr,C_tr,s_n,m_n);
L=L_0';


err_terms =(A-L*C)*x_ip(7:12);


diff_eq=zeros(12,1);

diff_eq(1) = x_ip(2);

diff_eq(2)=(u-(gravity/2)*(mass_1*sin(2*x_ip(3))+mass_2*sin(2*x_ip(5)))-(mass_1*
    length_1*(x_ip(4)^2)*sin(x_ip(3)))-(mass_2*length_2*(x_ip(6)^2)*sin(x_ip(5))))/(
    Mass_Crane+mass_1*((sin(x_ip(3)))^2)+mass_2*((sin(x_ip(5)))^2));

diff_eq(3)= x_ip(4);

diff_eq(4)= (diff_eq(2)*cos(x_ip(3))-gravity*(sin(x_ip(3)))))/length_1';

diff_eq(5)= x_ip(6);

diff_eq(6)= (diff_eq(2)*cos(x_ip(5))-gravity*(sin(x_ip(5)))))/length_2;

diff_eq(7)= x_ip(2)-x_ip(10);
diff_eq(8)= diff_eq(2)-err_terms(2);


diff_eq(9)= x_ip(4)-x_ip(11);


diff_eq(10)= diff_eq(4)-err_terms(4);


diff_eq(11)= x_ip(6)-x_ip(12);


diff_eq(12)= diff_eq(6)-err_terms(6);
end
```