Control of Robotic Systems (ENPM667)

Report on

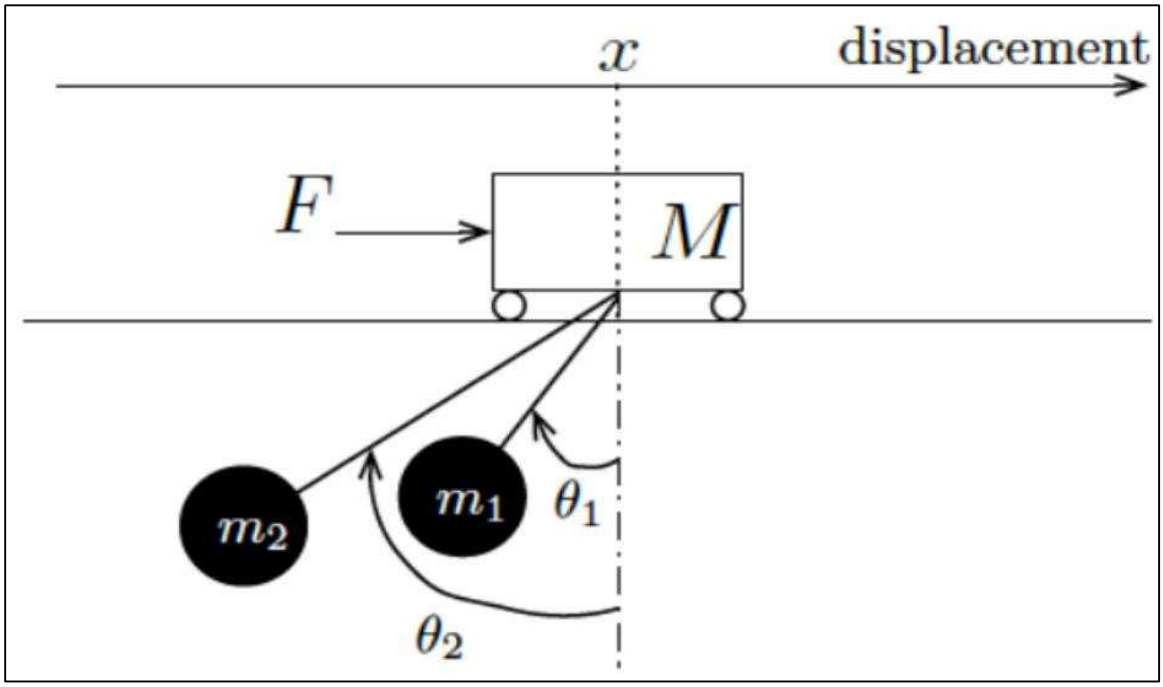
Final Project

Group Member:

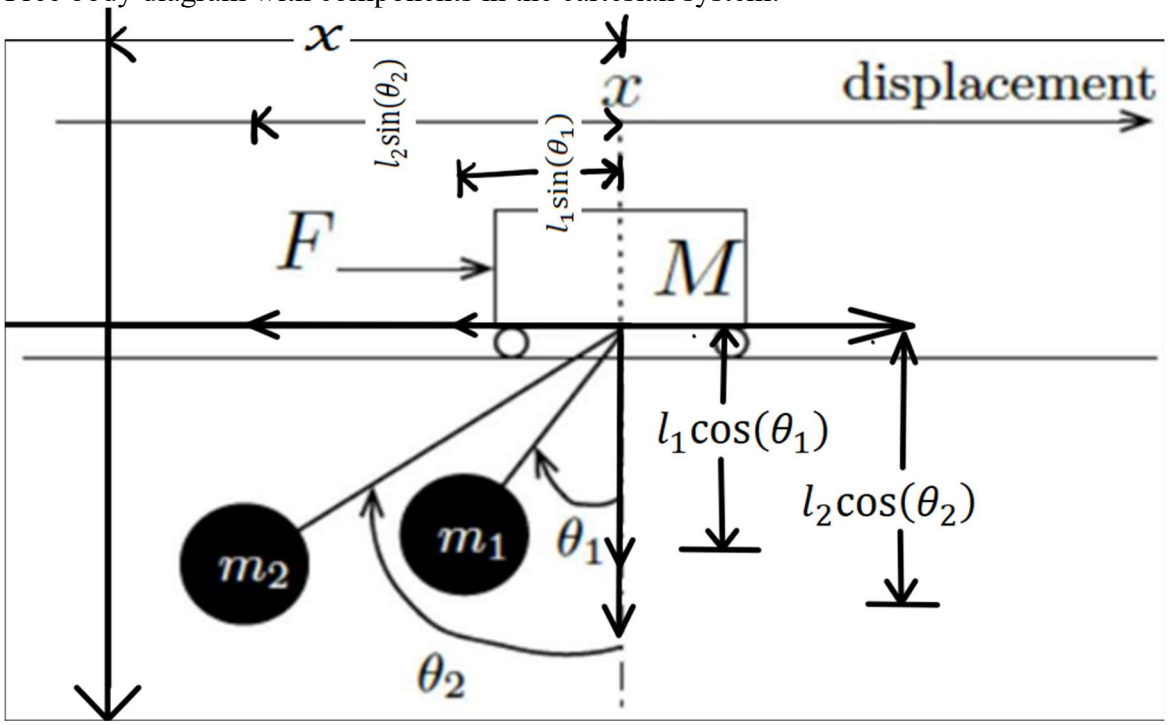
Venkata Tej Kiran Reddy Polamreddy (UID: 119197066)
Krishna Rajesh Hundekari (UID: 119239049)

Table of Contents

A) Equations of motion for the system and the corresponding nonlinear state-space representation.



Free body diagram with components in the cartesian system:



For the above system, the minimum parameters needed to define the system completely are the displacement and velocity of the cart in the X direction (x and $\dot{x}$), angles θ_1 and θ_2 along with the angular velocity $\dot{\theta_1}$ *and* $\dot{\theta_2}$.

To define the Lagrange equation of the system, first we need to compute the Kinetics energy and the potential energy of the system.

To compute the energies, lets first get the position of the mass m_1 and mass m_2 as a function of the respective $\theta's$:

1. For m_1:
   The angle θ_1 in the counterclockwise direction from the negative Y-axis from the figure 1.
   The x component of the mass $m_1 = x - l_1 \sin(\theta_1)$
   The y component of the mass $m_1 = -l_1 \cos(\theta_1)$

   Hence the position vector representation of mass m_1 in terms of θ_1:

$$x_{m1} = \left(x - l_1 \sin(\theta_1)\right)\hat{\imath} + \left(-l_1 \cos(\theta_1)\right)\hat{\jmath}$$

2. For m_2:
   The angle θ_2 in the counterclockwise direction from the negative Y-axis from the figure 1.
   The x component of the mass $m_1 = x - l_2 \sin(\theta_2)$
   The y component of the mass $m_1 = -l_2 \cos(\theta_2)$

   Hence the position vector representation of mass m_2 in terms of θ_2:

$$x_{m2} = \left(x - l_2 \sin(\theta_2)\right)\hat{\imath} + \left(-l_2 \cos(\theta_2)\right)\hat{\jmath}$$

Now to get the remaining state space variables, we have to compute the velocities of the masses by differentiating the position vectors with respect to time:

1. Velocity of mass m_1 :
   By differentiating equation (1) with respect to time and treating θ_1 also a function of time we get:

$$v_{m1} = \left(\dot{x} - l_1 \cos(\theta_1)\dot{\theta_1}\right)\hat{\imath} + l_1 \sin(\theta_1)\dot{\theta_1}\hat{\jmath}$$

2. Velocity of mass m_2 :
   By differentiating equation (2) with respect to time and treating θ_2 also a function of time we get:

$$v_{m2} = \left(\dot{x} - l_2 \cos(\theta_2)\dot{\theta_2}\right)\hat{\imath} + l_2 \sin(\theta_2)\dot{\theta_2}\hat{\jmath}$$

The position vector for the mass M (card) is given by:

$$x = (x)\hat{\imath} + (0)\hat{\jmath}$$

By differentiating above position vector with respect to time, we compute the velocity of the mass M :

$$v_M = (\dot{x})\hat{\imath} + (0)\hat{\jmath}$$

The Kinetic Energy (K.E) of the system is given by:

K.E = (K.E of mass M) + (K.E of mass $m1$) + (K.E of mass $m2$)

(K.E of mass M) = $\frac{1}{2}Mv_M{}^2 = \frac{1}{2}M\dot{x}^2$

(K.E of mass m_1) = $\frac{1}{2}Mv_{m1}{}^2 = \frac{1}{2}m_1\left(\dot{x} - l_1\dot{\theta}_1\cos(\theta_1)\right)^2 + \frac{1}{2}m_1\left(l_1\dot{\theta}_1(\sin(\theta_1))\right)^2$

(K.E of mass m_2) = $\frac{1}{2}Mv_{m2}{}^2 = \frac{1}{2}m_2(\dot{x} - \dot{\theta}_2 l_2\cos(\theta_2)\Big)^2 + \frac{1}{2}m_2\left(l_2\dot{\theta}_2(\sin(\theta_2))\right)^2$

Hence, the total Kinetic Energy is given by:

K.E = $\frac{1}{2}M\dot{x}^2 + \frac{1}{2}m_1\left(\dot{x} - l_1\dot{\theta}_1\cos(\theta_1)\right)^2 + \frac{1}{2}m_1\left(l_1\dot{\theta}_1(\sin(\theta_1))\right)^2 + \frac{1}{2}m_2(\dot{x} - \dot{\theta}_2 l_2\cos(\theta_2)\Big)^2 + \frac{1}{2}m_2\left(l_2\dot{\theta}_2(\sin(\theta_2))\right)^2$

Using the Y component of the position vectors of masses m_1 and m_2, we can compute the potential energy of the respective masses:

(P.E of mass m_1) = $-m_1 g l_1\cos(\theta_1)$

(P.E of mass m_2) = $-m_2 g l_2\cos(\theta_2)$

As the cart (mass M) is constrained to travel only in X direction and the Y component is zero, the P.E of the mass M is zero.

Therefore, the total potential energy of the system is given by:

$$\text{P.E} = (\text{P. E of mass } m_1) + (\text{P. E of mass } m_2)$$

$$\text{P. E} = -m_1 g l_1\cos(\theta_1) - m_2 g l_2\cos(\theta_2) = -g[m_1 l_1\cos(\theta_1) + m_2 g l_2\cos(\theta_2)$$

Using the K.E and P.E of the system, the Lagrange equation of the system is computed as:

$$L = K.E - P.E$$

$$\mathrm{L} = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m_1\dot{x}^2 + \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2\cos^2(\theta_1) - m_1 l_1\dot{\theta}_1\dot{x}\cos(\theta_1) + \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2\sin^2(\theta_1)$$
$$+\frac{1}{2}m_2\dot{x}^2 + \frac{1}{2}m_2 l_2^2\dot{\theta}_2^2\cos^2(\theta_2) - m_2 l_2\dot{\theta}_2\dot{x}\cos(\theta_2) + \frac{1}{2}m_2 l_2^2\dot{\theta}_2^2\sin^2(\theta_1) + g[m_1 l_1\cos(\theta_1) + m_2 l_2\cos(\theta_2)]$$

On further simplifying terms in the Lagrange equation, we get :

$$\mathrm{L} = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}(m_1 + m_2)\dot{x}^2 + \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 l_2^2\dot{\theta}_2^2 -$$
$$\dot{x}\left(m_1 l_1\dot{\theta}_1\cos(\theta_1) + m_2 l_2\dot{\theta}_2\cos(\theta_2)\right) + g[m_1 l_1\cos(\theta_1) + m_2 l_2\cos(\theta_2)]$$

Using Euler- Lagrange's method to compute equation of the motion:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = F$$
$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \left(\frac{\partial L}{\partial \theta_1}\right) = \tau_{\theta_1}$$
$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \left(\frac{\partial L}{\partial \theta_2}\right) = \tau_{\theta_2}$$

1. Compute the Equation of Force(F) along the direction of the state variable x:

   First differentiating L with respect to $\dot{x}$;

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

   Differentiating above equation respect to time:

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

As there are no terms of x in equation L, the derivation of L with respect to x is zero.

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

Hence, the equation for the force along x is given by:

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

2. Compute the Equation of Torque(τ) along the direction of the state variable θ_1:

Using similar approach in the above steps, lets compute the important terms of the Euler-Lagrange's equations:
$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \left(\frac{\partial L}{\partial \theta_1}\right) = \tau_{\theta_1}$

First differentiating L with respect to $\dot{\theta}_1$;

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

Differentiating above equation respect to time:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) = m_1 l_1^2 \ddot{\theta}_1 - \left[m_1 l_1 \ddot{x} \cos(\theta_1) - m_1 \dot{x} l_1 \dot{\theta}_1 \sin(\theta_1)\right]$$

There are some terms of θ_1 in equation L, the derivation of L with respect to θ_1 is:

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

Using derived terms, the equation can be written as :

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

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

As the only activation is through force F, no external torque is applied. Hence, $\tau_{\theta_1} = 0$.

Further simplifying the equation, we get:

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

3. Compute the Equation of Torque(τ) along the direction of the state variable θ_2:
   Using similar approach in the above steps, lets compute the important terms of the Euler-Lagrange's equations:

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

   First differentiating L with respect to $\dot{\theta}_2$;

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

   Differentiating above equation respect to time:

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

   There are some terms of θ_2 in equation L, the derivation of L with respect to θ_2 is:

$$\left(\frac{\partial L}{\partial \theta_2}\right) = m_2 \dot{x} l_2 \dot{\theta}_2 \sin(\theta_2) - m_2 l_2 g \sin(\theta_2)$$

   Using derived terms, the equation can be written as :

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

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

   As the only activation is through force F, no external torque is applied. Hence, $\tau_{\theta_2} = 0$.

   Further simplifying the equation, we get:

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

Therefore, the equations of the motion for the given crane systems are:

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

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

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

These equations can be rearranged to give the equations for linear acceleration $\ddot{x}$ and angular accelerations $\ddot{\theta}_1 \; and \; \ddot{\theta}_2$:

$$\ddot{\theta}_1 = \frac{\cos(\theta_1)\ddot{x} - g\sin(\theta_1)}{l_1}$$
$$\ddot{\theta}_2 = \frac{\ddot{x}\cos(\theta_2) - g\sin(\theta_2)}{l_2}$$

Substituting these in the equation for F we get:

$$[M + m_1 + m_2]\ddot{x} - m_1 l_1 \left[\frac{\cos(\theta_1)\ddot{x} - g\sin(\theta_1)}{l_1}\right]\cos(\theta_1) + m_1 l_1 \dot{\theta_1}^2 \sin(\theta_1)$$
$$- m_2 l_2 \left[\frac{\ddot{x}\cos(\theta_2) - g\sin(\theta_2)}{l_2}\right]\cos(\theta_2) + m_2 l_2 \dot{\theta_2}^2 \sin(\theta_2) = F$$

Upon simplifying we get and rearranging to get $\ddot{x}$:

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

Resubstituting $\ddot{x} \; to \; get \; \ddot{\theta}_1 \; and \; \ddot{\theta}_2$:

$$\ddot{\theta}_1 = \frac{\cos(\theta_1)}{l_1}\left[\frac{F - m_1\left(g\sin(\theta_1)\cos(\theta_1) + l_1\sin(\theta_1)\dot{\theta}_1^2\right) - m_2\left(g\sin(\theta_2)\cos(\theta_2) + l_2\sin(\theta_2)\dot{\theta}_2^2\right)}{\left(M + m_1\left(\sin(\theta_1)\right)^2 + m_2\left(\sin(\theta_2)\right)^2\right)}\right] - \frac{g\sin(\theta_1)}{l_1}$$

$$\ddot{\theta}_2 = \frac{\cos(\theta_2)}{l_2}\left[\frac{F - m_1\left(g\sin(\theta_1)\cos(\theta_1) + l_1\sin(\theta_1)\dot{\theta}_1^2\right) - m_2\left(g\sin(\theta_2)\cos(\theta_2) + l_2\sin(\theta_2)\dot{\theta}_2^2\right)}{\left(M + m_1\left(\sin(\theta_1)\right)^2 + m_2\left(\sin(\theta_2)\right)^2\right)}\right] - \frac{g\sin(\theta_2)}{l_2}$$

the state space variable matrix is given by:

$$[\boldsymbol{x}] = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \\ \boldsymbol{\theta_1} \\ \dot{\boldsymbol{\theta_1}} \\ \boldsymbol{\theta_2} \\ \dot{\boldsymbol{\theta_2}} \end{bmatrix}$$

The derivation matrix of the state space variables with respect to time is given by:

$$[\dot{x}] = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dfrac{F - m_1(g\sin(\theta_1)\cos(\theta_1) + l_1\sin(\theta_1)\dot{\theta}_1^2) - m_2(g\sin(\theta_2)\cos(\theta_2) + l_2\sin(\theta_2)\dot{\theta}_2^2)}{\left(M + m_1(\sin(\theta_1))^2 + m_2(\sin(\theta_2))^2\right)} \\ \dot{\theta}_1 \\ \dfrac{\cos(\theta_1)}{l_1}\left[\dfrac{F - m_1(g\sin(\theta_1)\cos(\theta_1) + l_1\sin(\theta_1)\dot{\theta}_1^2) - m_2(g\sin(\theta_2)\cos(\theta_2) + l_2\sin(\theta_2)\dot{\theta}_2^2)}{\left(M + m_1(\sin(\theta_1))^2 + m_2(\sin(\theta_2))^2\right)}\right] - \dfrac{g\sin(\theta_1)}{l_1} \\ \dot{\theta}_2 \\ \dfrac{\cos(\theta_2)}{l_2}\left[\dfrac{F - m_1(g\sin(\theta_1)\cos(\theta_1) + l_1\sin(\theta_1)\dot{\theta}_1^2) - m_2(g\sin(\theta_2)\cos(\theta_2) + l_2\sin(\theta_2)\dot{\theta}_2^2)}{\left(M + m_1(\sin(\theta_1))^2 + m_2(\sin(\theta_2))^2\right)}\right] - \dfrac{g\sin(\theta_2)}{l_2} \end{bmatrix}$$

B) Linearization around the equilibrium point:

We have linearized our nonlinear system using Jacobian Method of Linearization.

The computed Jacobian is at equilibrium point $x = \theta_1 = \theta_2 = 0$.

$$A = \begin{bmatrix} \frac{\delta f_1}{\delta X_1} & \frac{\delta f_1}{\delta X_2} & \frac{\delta f_1}{\delta X_3} & \frac{\delta f_1}{\delta X_4} & \frac{\delta f_1}{\delta X_5} & \frac{\delta f_1}{\delta X_6} \\ \frac{\delta f_2}{\delta X_1} & \frac{\delta f_2}{\delta X_2} & \frac{\delta f_2}{\delta X_3} & \frac{\delta f_2}{\delta X_4} & \frac{\delta f_2}{\delta X_5} & \frac{\delta f_2}{\delta X_6} \\ \frac{\delta f_3}{\delta X_1} & \frac{\delta f_3}{\delta X_2} & \frac{\delta f_3}{\delta X_3} & \frac{\delta f_3}{\delta X_4} & \frac{\delta f_3}{\delta X_5} & \frac{\delta f_3}{\delta X_6} \\ \frac{\delta f_4}{\delta X_1} & \frac{\delta f_4}{\delta X_2} & \frac{\delta f_4}{\delta X_3} & \frac{\delta f_4}{\delta X_4} & \frac{\delta f_4}{\delta X_5} & \frac{\delta f_4}{\delta X_6} \\ \frac{\delta f_5}{\delta X_1} & \frac{\delta f_5}{\delta X_2} & \frac{\delta f_5}{\delta X_3} & \frac{\delta f_5}{\delta X_4} & \frac{\delta f_5}{\delta X_5} & \frac{\delta f_5}{\delta X_6} \\ \frac{\delta f_6}{\delta X_1} & \frac{\delta f_6}{\delta X_2} & \frac{\delta f_6}{\delta X_3} & \frac{\delta f_6}{\delta X_4} & \frac{\delta f_6}{\delta X_5} & \frac{\delta f_6}{\delta X_6} \end{bmatrix} \quad B = \begin{bmatrix} \frac{\delta f_1}{\delta u} \\ \frac{\delta f_1}{\delta u} \\ \frac{\delta f_1}{\delta u} \\ \frac{\delta f_1}{\delta u} \\ \frac{\delta f_1}{\delta u} \\ \frac{\delta f_1}{\delta u} \end{bmatrix}$$

We wrote a python script to compute the Jacobian using approach mentioned above. The output of the screen from the python terminal prints the final linearized A and B matrices. Please find below the snip of these from the terminal output:

A Linear :

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-g \cdot m_1}{M} & 0 & \frac{-g \cdot m_2}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-g - \frac{g \cdot m_1}{M}}{l_1} & 0 & \frac{-g \cdot m_2}{M \cdot l_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-g \cdot m_1}{M \cdot l_2} & 0 & \frac{-g - \frac{g \cdot m_2}{M}}{l_2} & 0 \end{bmatrix}$$

B Lieanr :

$$\begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{M \cdot l_1} \\ 0 \\ \frac{1}{M \cdot l_2} \end{bmatrix}$$

The state space representation after linearization of the given nonlinear system is:

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

C) Condition for Controllability

The System Model after linearization is given as

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-g \cdot m_1}{M} & 0 & \frac{-g \cdot m_2}{M} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-g \cdot (M + m_1)}{M \cdot l_1} & 0 & \frac{-g \cdot m_2}{M \cdot l_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-g \cdot m_1}{M \cdot l_2} & 0 & \frac{-g \cdot (M + m_2)}{M \cdot l_2} & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{M \cdot l_1} \\ 0 \\ \frac{1}{M \cdot l_2} \end{bmatrix}$$

To make the system controllable, we derived the controllability matrix and tried showing that it is of full rank. For a matrix to be full rank, its determinant should not be equal to zero.

Using Sympy python module, the controllability matrix and its respective determinant condition are computed as below (reference program : “controllability.py”)

$$\text{Determinant of(ctrb)} = \frac{g^6 \cdot \left(-l_1^2 + 2 \cdot l_1 \cdot l_2 - l_2^2\right)^6}{M^6 \cdot l_1^6 \cdot l_2^6} = 0$$

The above equation can be zero only when l1 is equal to l2.

From this we can say that, the given **linearized *system is uncontrollable when the lengths of two pendulums are equal.***

D) Controllability check and LQR Controller Design:

1. System Controllability Check:

For the parameters as given in the question:
M = 1000Kg, m_1 = m_2 = 100Kg, l_1 = 20m and l_2 = 10m
The system model is computed and corresponding system controllability is verified by computing the rank of controllability matrix in MATLAB as follows:

```
Given Parameters of the System\n
M : 1000, m1:100, m2:100, l1:20, l2:10
A =

         0    1.0000         0         0         0         0
         0         0   -0.9800         0   -0.9800         0
         0         0         0    1.0000         0         0
         0         0   -0.5390         0   -0.0490         0
         0         0         0         0         0    1.0000
         0         0   -0.0980         0   -1.0780         0


B =

   1.0e-03 *

         0
    1.0000
         0
    0.0500
         0
    0.1000

--------------------------------------------------------------------------------
Check if system is controllable using rank condition\n
--------------------------------------------------------------------------------
Rank of Controllability matrix is : 6
System is Controllable.\n
```

We can see that the controllability matrix is of full rank. Hence the system is controllable for the given parameters.

2. LQR Controller Design:

The optimal solution of LQR controller is given as

$$K = -R^{-1} Bk P$$

Where P is the symmetric positive definite solution of the following Riccatti equation

$$A^T P + PA - PBkR^{-1}Bk^T P = -Q$$

And the optimal cost is given by

$$J(K, X(0)) = X(0)^T P X(0)$$

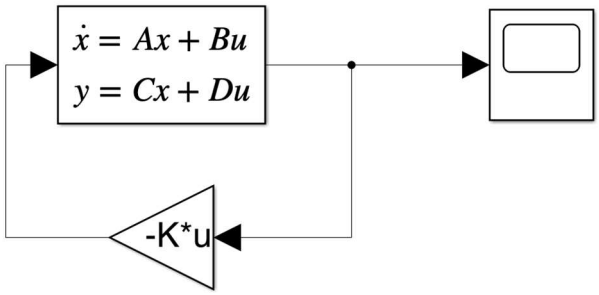
For the above system, values for Q and R are chosen for optimal performance and the respective gain matrix is computed in MATLAB as shown below:

```
------------------------------------------------------------------------
Find gain matrix using LQR method\n
------------------------------------------------------------------------
LQR parameters:

Q =

    10     0     0     0     0     0
     0    10     0     0     0     0
     0     0   100     0     0     0
     0     0     0   100     0     0
     0     0     0     0   100     0
     0     0     0     0     0   100

R =

   1.0000e-03

LQR gains are found as:

K =

  100.0000  503.1648   60.6148 -232.6462   94.9497  -54.2402

Eigen Values of Closed loop system after  applying LQR.\n

ans =

  -0.0245 + 1.0416i
  -0.0245 - 1.0416i
  -0.0120 + 0.7273i
  -0.0120 - 0.7273i
  -0.2066 + 0.2023i
  -0.2066 - 0.2023i
```

From the eigen values it can be seen that the closed loop system is stable.

3. Linear LQR Controller Simulation results:

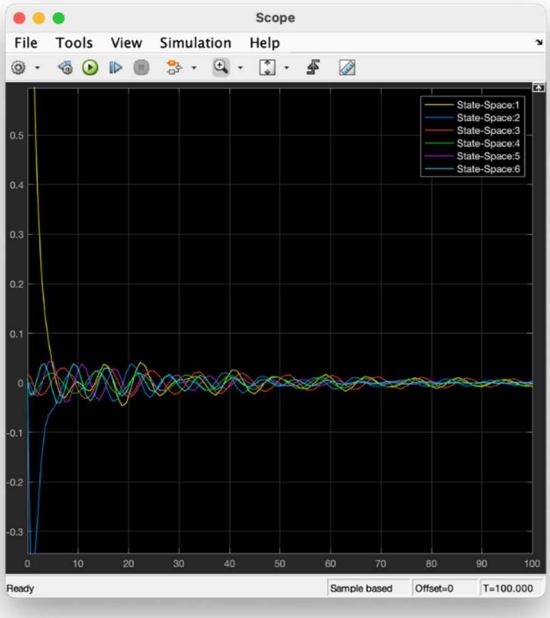
The LQR controller for linear model used for simulations is as follows:



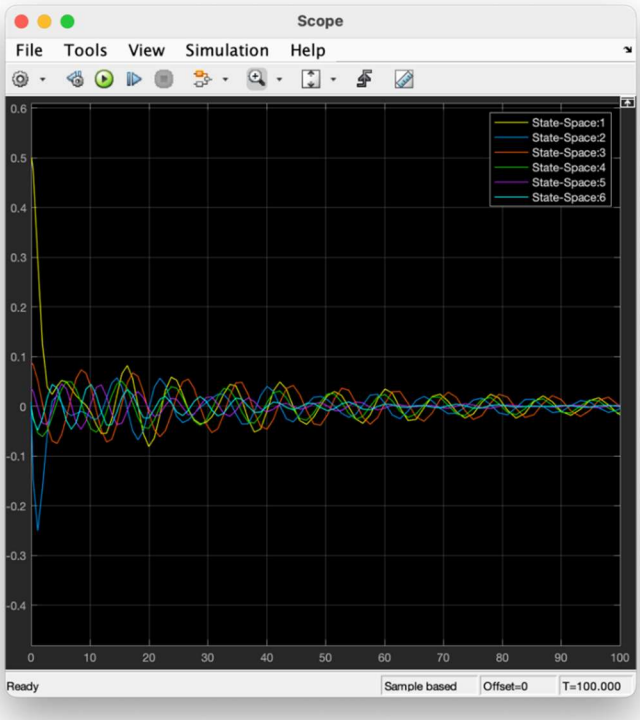
System with LQR Controller

The system response for two initial states are as shown below:
X(0) :

```
Init_S1 =

    1.0000
         0
    0.0175
         0
         0
    0.0087
```



As we can see from the plot, for the given initial state 1 the state is approaching to zero and the system is stabilizing.



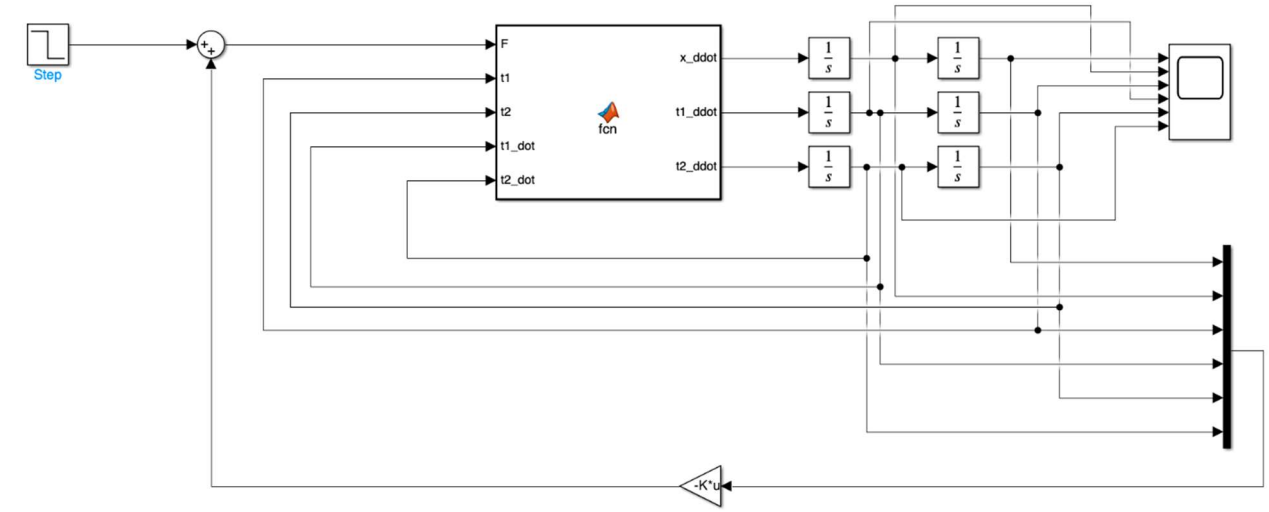

```
Init_S2 =

    0.5000
    0.0100
    0.0873
    0.0002
    0.0349
    0.0002
```

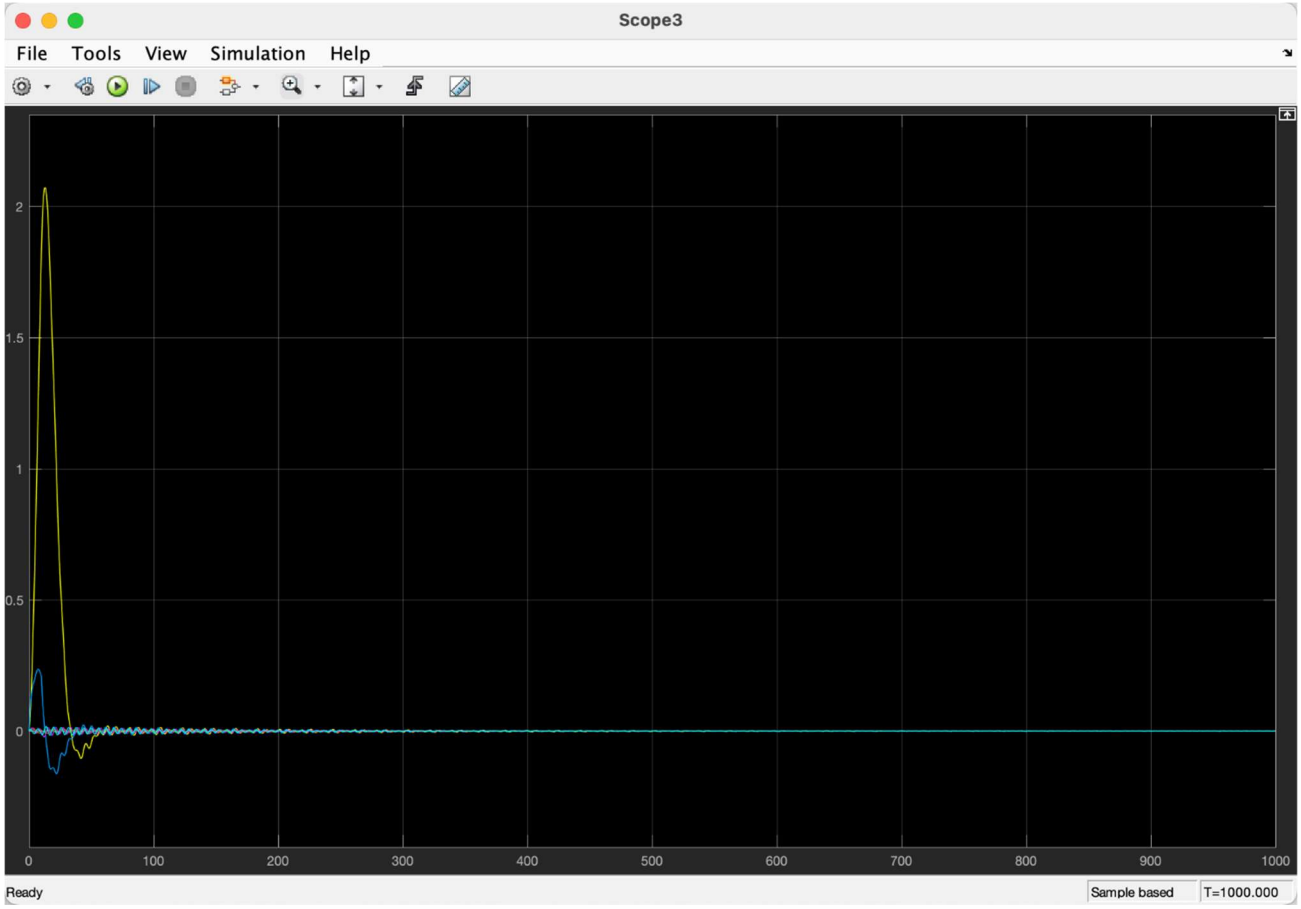
X(0) :

As we can see from the plot, for the given initial state 2 the state is approaching to zero and the system is stabilizing.

4. Non-linear LQR Controller Simulation results:

Non-linear LQR controller model used for the simulation:





5. Stability of Closed loop LQR system from Lyapunov indirect method:

We know that an LTI system in state space form is stable if and only if for any symmetric positive definite matrix Q there exists a symmetric positive definite matrix P such that the following Lyapunov equation holds

$$A^T P + PA = -Q$$

in which $V(X) = X^T P X$ is a Lyapunov function.
For the given problem we have shown the stability through eigen values. The same can be evaluated using the above equation by showing that, for the above closed loop system's Q and A there is exists a P which is positive definite. Using MATLAB this is shown as below(ref: "lqr_controller.m").

Using two ways we have verified the stability of the system:

1. Lyapunov Method by checking the positive definiteness of P:

```
--------------------------------------------------------------------------------
Checking for stabiity of LQR Closed Loop system using Lyapnov indirect method.
--------------------------------------------------------------------------------
For given system

A =

         0    1.0000         0         0         0         0
         0         0   -0.9800         0   -0.9800         0
         0         0         0    1.0000         0         0
         0         0   -0.5390         0   -0.0490         0
         0         0         0         0         0    1.0000
         0         0   -0.0980         0   -1.0780         0


Q =

    10     0     0     0     0     0
     0    10     0     0     0     0
     0     0   100     0     0     0
     0     0     0   100     0     0
     0     0     0     0   100     0
     0     0     0     0     0   100

Lyapnov Solution of Closed loop system after  applying LQR.\n

P =

   1.0e+04 *

    2.4971   -0.0005    0.8670    0.3041    0.0037    0.1011
   -0.0005    1.0794   -0.3041    0.4788   -0.1011    0.1116
    0.8670   -0.3041    0.6140   -0.0050   -0.0863    0.0405
    0.3041    0.4788   -0.0050    0.3248   -0.0405   -0.0367
    0.0037   -0.1011   -0.0863   -0.0405    0.2253   -0.0050
    0.1011    0.1116    0.0405   -0.0367   -0.0050    0.2320

Lyapnov Solution is Positive definite and hence system is stable
```

2. Lyapunov Indirect method using the computed A and B matrices from the Jacobian Linearization:

By computing the eigen values of (A-B*K) matrix, we can see that the real part of all the eigen values is negative. Hence, we can claim that system is locally and globally stable

```
We can compute the closed loop stability using Lyapunov idirect methd as follows:
A Linear :
⎡0  1      0      0      0      0⎤
⎢                                 ⎥
⎢        -g·m₁          -g·m₂     ⎥
⎢0  0   ──────    0   ──────    0⎥
⎢          M              M      ⎥
⎢                                 ⎥
⎢0  0      0      1      0      0⎥
⎢                                 ⎥
⎢           g·m₁                  ⎥
⎢      -g - ────                  ⎥
⎢            M          -g·m₂     ⎥
⎢0  0  ──────────  0   ──────    0⎥
⎢         l₁           M·l₁     ⎥
⎢                                 ⎥
⎢0  0      0      0      0      1⎥
⎢                                 ⎥
⎢                          g·m₂   ⎥
⎢                     -g - ────   ⎥
⎢        -g·m₁              M     ⎥
⎢0  0   ──────    0   ──────────  0⎥
⎣         M·l₂            l₂     ⎦
B Lieanr :
⎡  0  ⎤
⎢     ⎥
⎢  1  ⎥
⎢  ─  ⎥
⎢  M  ⎥
⎢     ⎥
⎢  0  ⎥
⎢     ⎥
⎢  1  ⎥
⎢─────⎥
⎣M·l₁ ⎦
⎡  0  ⎤
⎢     ⎥
⎢  1  ⎥
⎢─────⎥
⎣M·l₂ ⎦

LQR Gains :
[1000.0  1864.2  38.8  -1022.6  330.5  -574.1]

Eigen values from LQR based closed loop Controller A-B*K :
-0.0412403809243072 - 1.00456673689089·i

-0.0154985653233127 - 0.705270024548303·i

-0.82109105375238 - 0.529518969352451·i

-0.82109105375238 + 0.529518969352451·i

-0.0154985653233127 + 0.705270024548303·i

-0.0412403809243072 + 1.00456673689089·i
```

Second Component:

E) Observability

For the give output vectors x(t), (θ_1(t), θ_2(t)), (x(t), θ_2(t)) or (x(t), θ_1(t), θ_2(t)) the matrix C1, C2, C3, C4 should be defined so that for the state

$$X(t) = [\, x(t),\ \dot{x}(t),\ \theta_1(t),\ \dot{\theta_1}(t),\ \theta_2(t),\ \dot{\theta_2}(t)]$$

and system

$$\dot{X}(t) = AX(t) + BU(t)$$
$$y(t) = CX(t) + DU(t)$$

the observability for a given output can be verified by evaluating the below rank condition

$$\text{Rank}([\ \ C^T \ \ (A^T)C^T \ \ (A^T)^2C^T \ \ (A^T)^3C^T \ \ (A^T)^4C^T \ \ (A^T)^5C^T \ \ ]) = 6$$

Using MATLAB the above condition for each output state is verified as below

```
--------------------------------
Case 1 : state = [x(t)]
--------------------------------
Rank of Observability Matrix : 6.000000

State is observable



--------------------------------
Case 2 : state = [θ1(t), θ2(t)]
--------------------------------
Rank of Observability Matrix : 4.000000

State is not observable



--------------------------------
Case 3 : state = [x(t), θ2(t)]
--------------------------------
Rank of Observability Matrix : 6.000000

State is observable



--------------------------------
Case 4 : state = [x(t), θ1(t), θ2(t)]
--------------------------------
Rank of Observability Matrix : 6.000000

State is observable
```

As we can see from above the states

x(t) : ***observable***

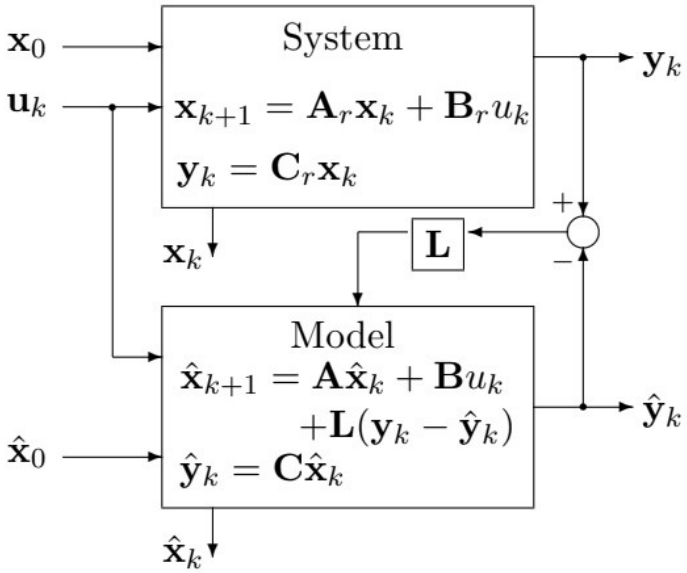
(θ_1(t), θ_2(t)) : ***not observable***

(x(t), θ_2(t)) : ***observable***

(x(t), θ_1(t), θ_2(t)) : ***observable***

F) Luenberger observer Model:

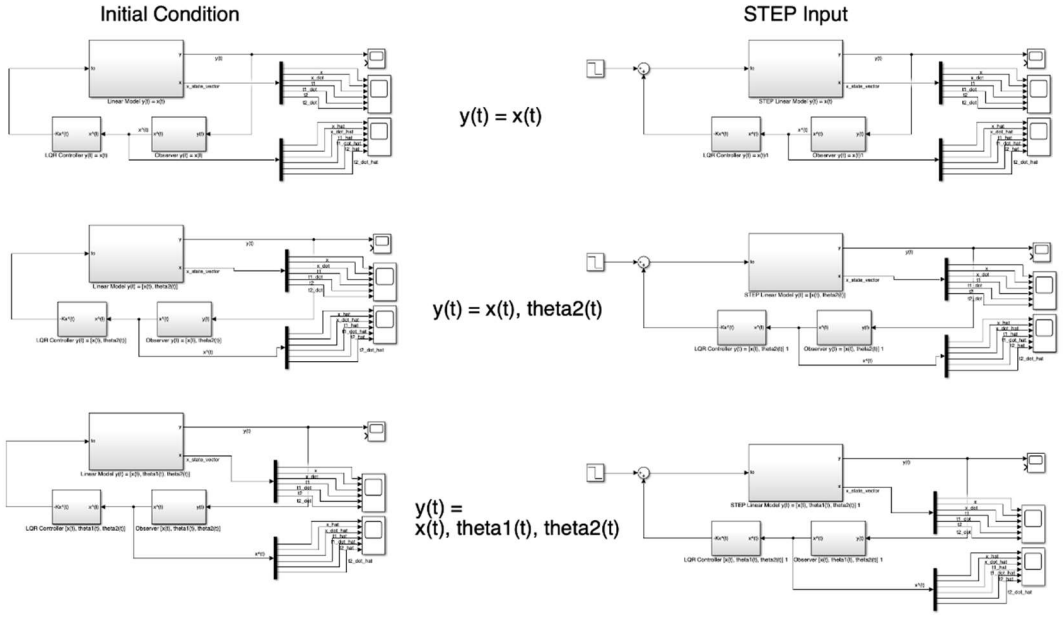
Obtain your "best" Luenberger observer for each one of the output vectors for which the system is observable and simulate its response to initial conditions and unit step input. The simulation should be done for the observer applied to both the linearized system and the original nonlinear system.

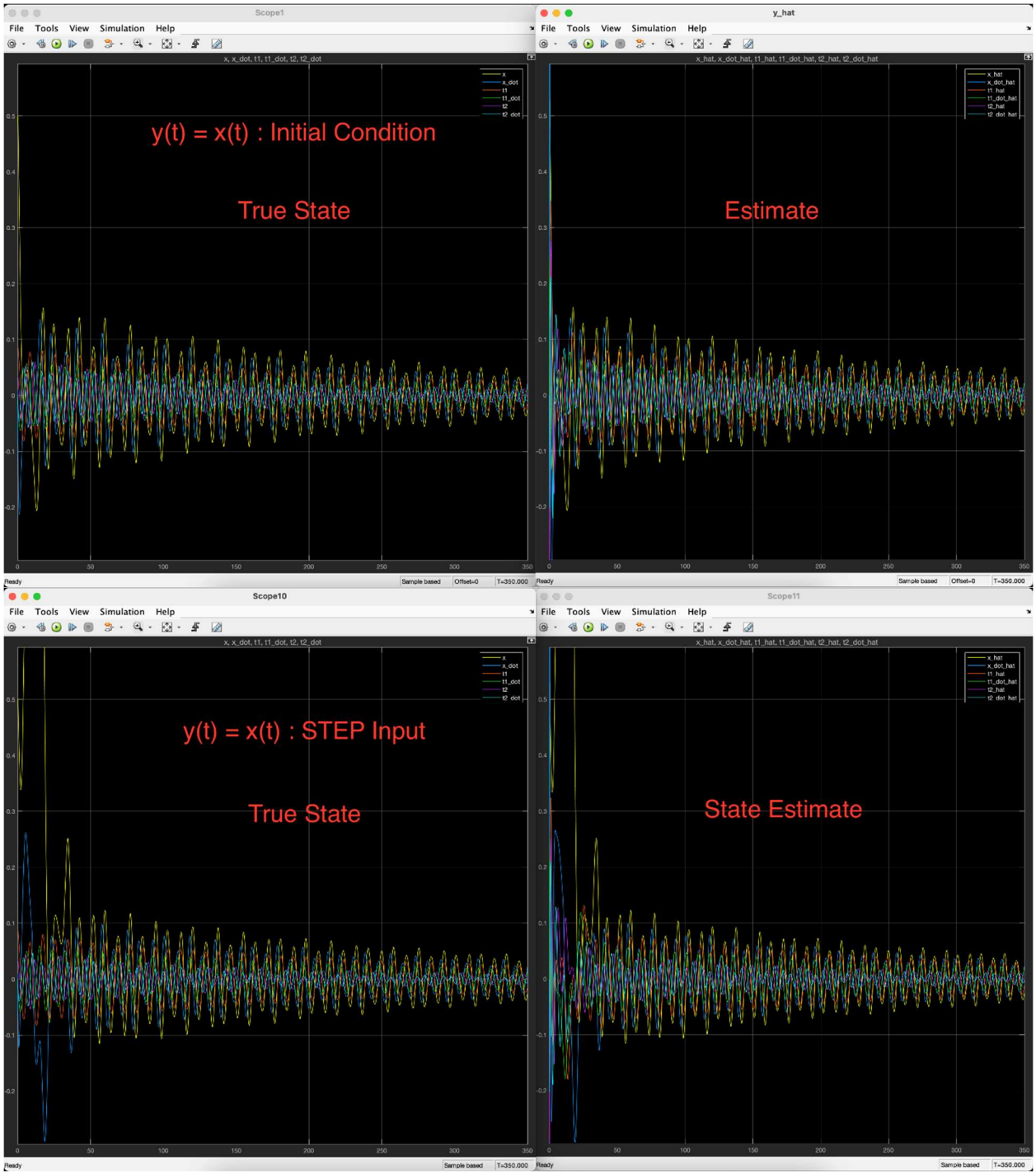


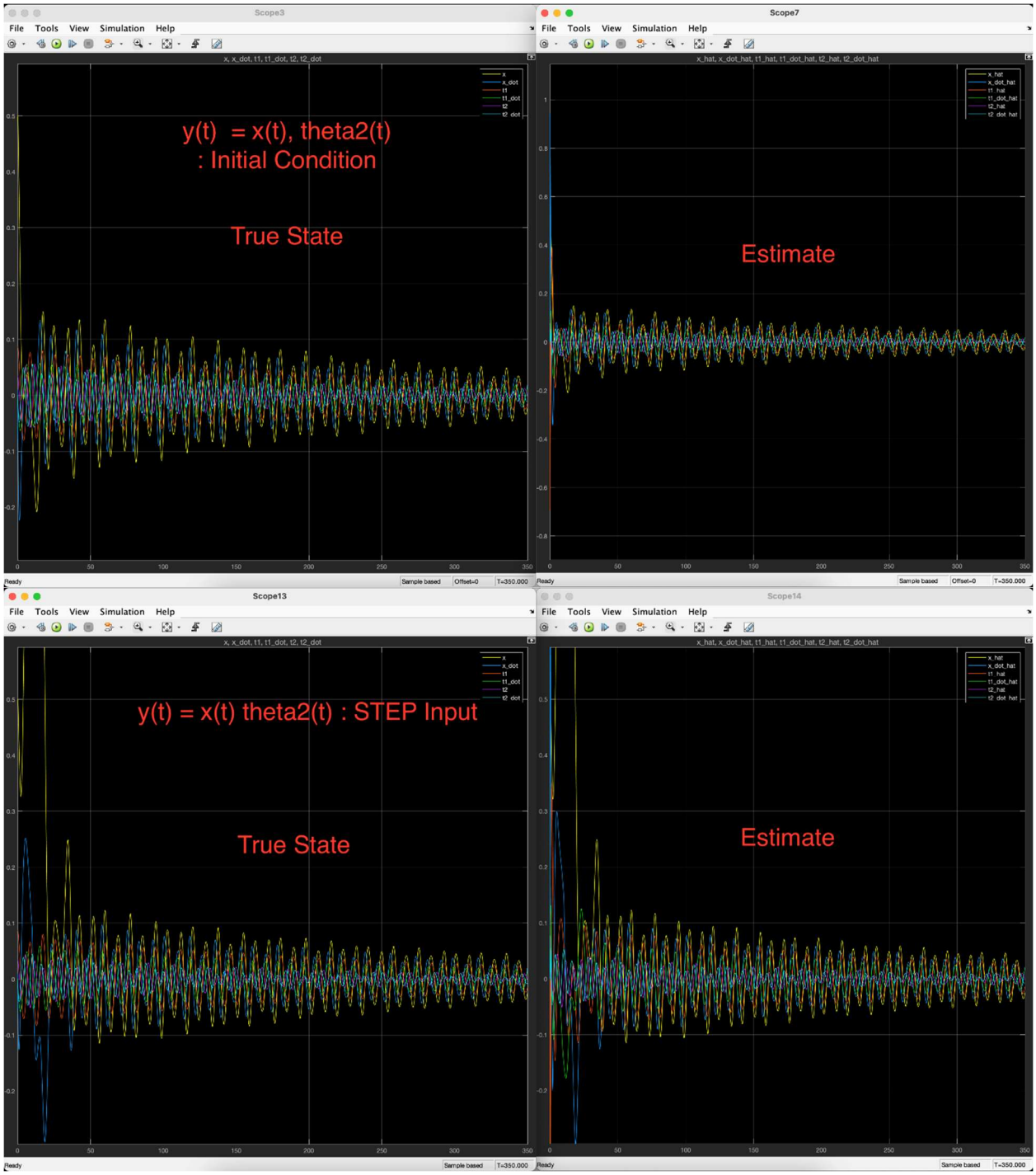
Gains of Luenberger observer are computed for the above observable states as below:

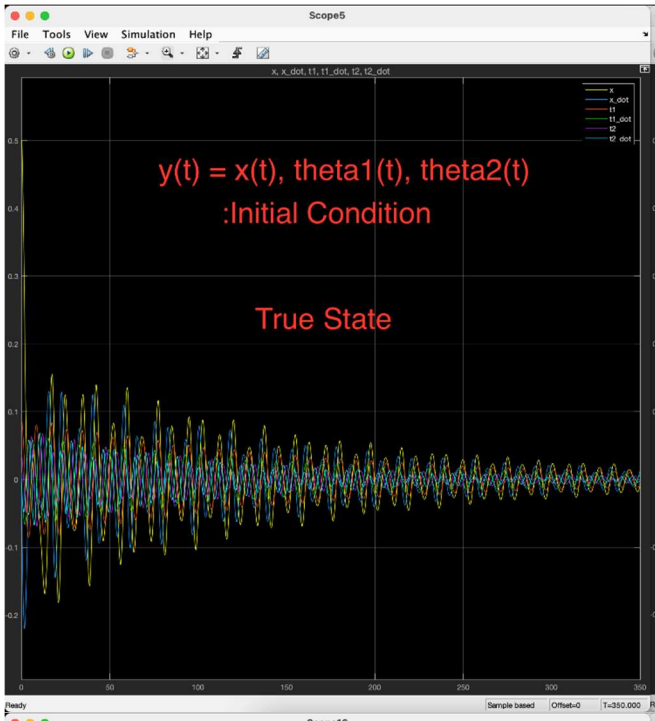
```
Luenberger observer gain for each of these states can be solved using LQR controller for the system [At-Ct*Lt]
LQR parameters:
Q = 100 * eye(6)
R = 0.1

R =

    0.1000

-----------------------------------
Case 1 : Luenberger observer gain for state = [x(t)]
-----------------------------------
LQR gains are found as:

L1 =

   34.0865
   80.9463
  -49.9943
   17.5904
  -33.1328
  -18.2948

-----------------------------------
Case 3 : state = [x(t), θ2(t)]
-----------------------------------
LQR gains are found as:

L3 =

   33.6771    0.0924
   67.0789    2.5821
  -53.1560   -5.3278
    0.8490    0.0272
    0.0924   32.5856
    3.5426   30.9159

-----------------------------------
Case 4 : state = [x(t), θ1(t), θ2(t)]
-----------------------------------
LQR gains are found as:

L4 =

   32.6084   -0.0148   -0.0145
   31.6531   -0.9709   -0.9618
   -0.0148   32.5911   -0.0022
    0.0091   31.0884   -0.0473
   -0.0145   -0.0022   32.5749
    0.0182   -0.0963   30.5633
```

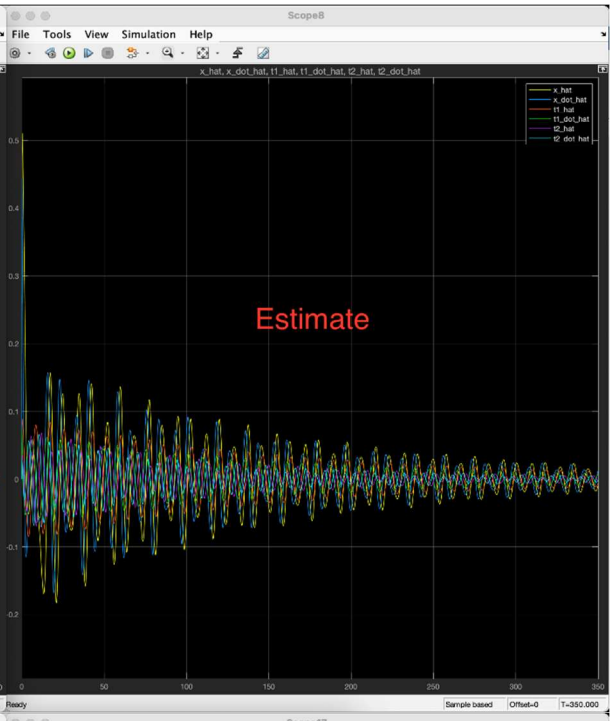
1. Luenberger model for the linearized system:

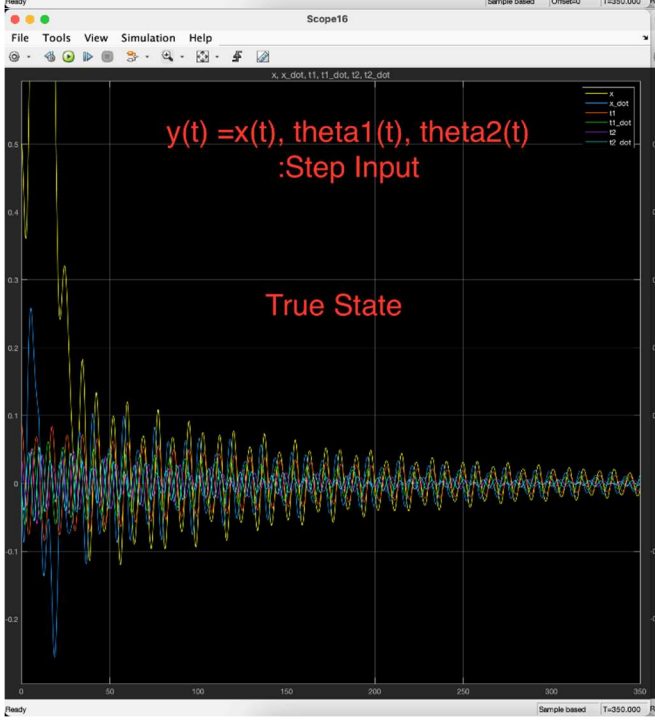


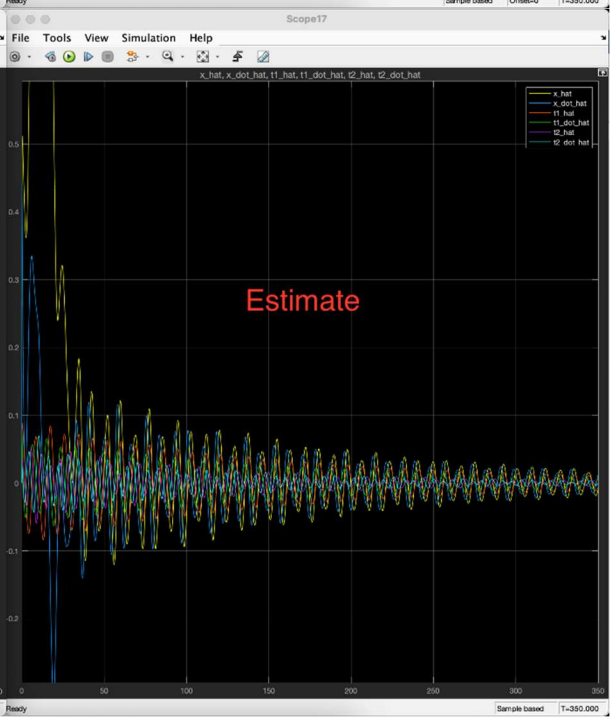

Scope1
File Tools View Simulation Help
x, x_dot, t1, t1_dot, t2, t2_dot
y(t) = x(t) : Initial Condition
True State
y_hat
File Tools View Simulation Help
x_hat, x_dot_hat, t1_hat, t1_dot_hat, t2_hat, t2_dot_hat
Estimate
Scope10
File Tools View Simulation Help
x, x_dot, t1, t1_dot, t2, t2_dot
y(t) = x(t) : STEP Input
True State
Scope11
File Tools View Simulation Help
x_hat, x_dot_hat, t1_hat, t1_dot_hat, t2_hat, t2_dot_hat
State Estimate


Scope3
File Tools View Simulation Help
x, x_dot, t1, t1_dot, t2, t2_dot
y(t) = x(t), theta2(t)
: Initial Condition
True State
Scope7
File Tools View Simulation Help
x_hat, x_dot_hat, t1_hat, t1_dot_hat, t2_hat, t2_dot_hat
Estimate
Scope13
File Tools View Simulation Help
x, x_dot, t1, t1_dot, t2, t2_dot
y(t) = x(t) theta2(t) : STEP Input
True State
Scope14
File Tools View Simulation Help
x_hat, x_dot_hat, t1_hat, t1_dot_hat, t2_hat, t2_dot_hat
Estimate

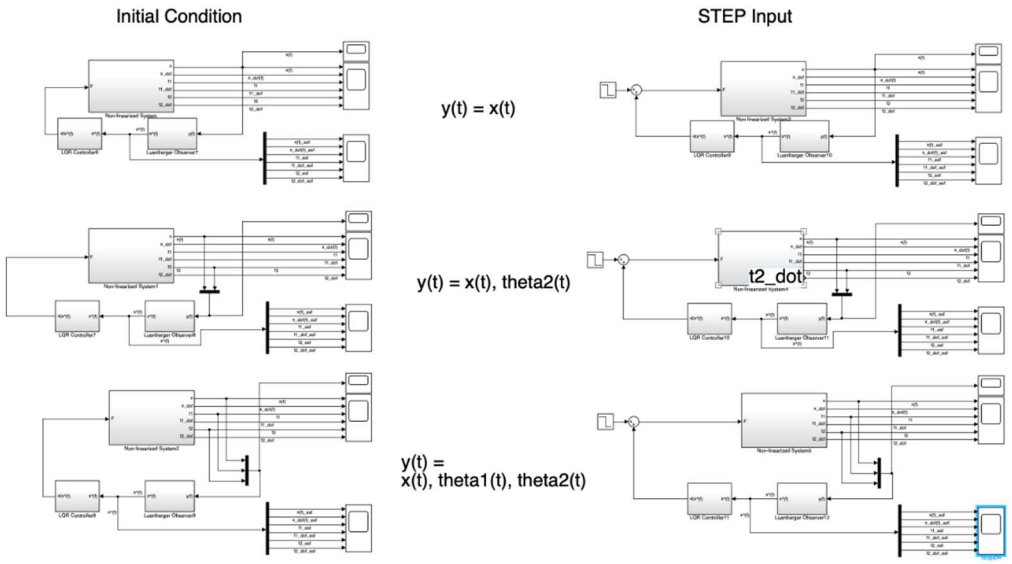

Scope5
File Tools View Simulation Help
x, x_dot, t1, t1_dot, t2, t2_dot
y(t) = x(t), theta1(t), theta2(t)
:Initial Condition
True State
Ready
Sample based
Offset=0
T=350.000

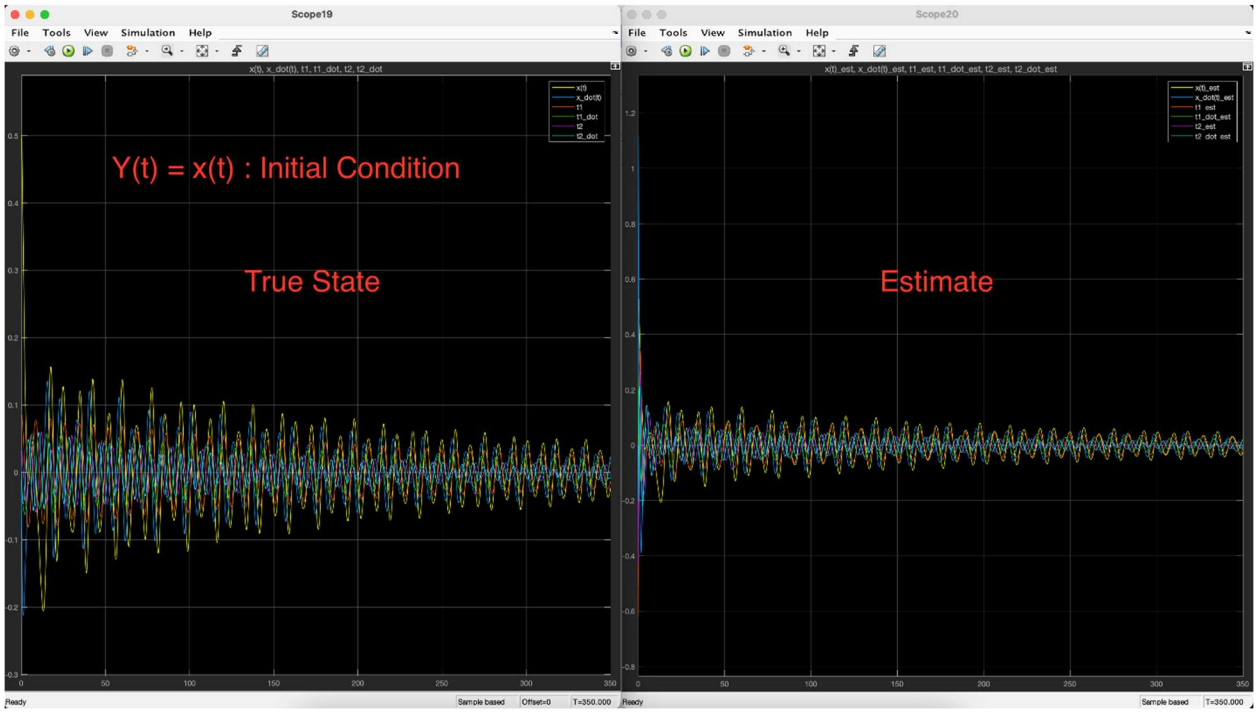

Scope8
File Tools View Simulation Help
x_hat, x_dot_hat, t1_hat, t1_dot_hat, t2_hat, t2_dot_hat
Estimate
Ready
Sample based
Offset=0
T=350.000

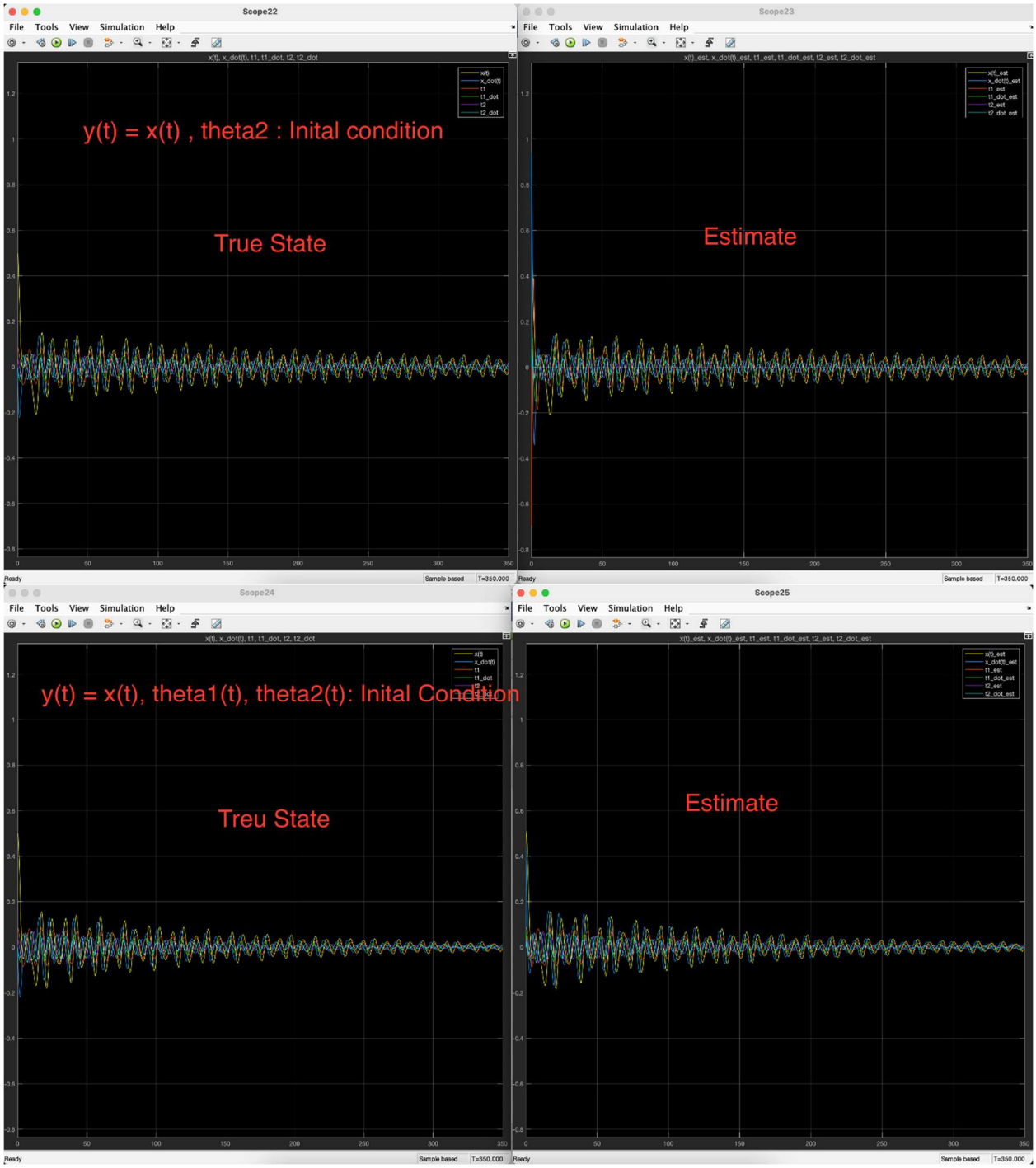

Scope16
File Tools View Simulation Help
x, x_dot, t1, t1_dot, t2, t2_dot
y(t) =x(t), theta1(t), theta2(t)
:Step Input
True State
Ready
Sample based
T=350.000

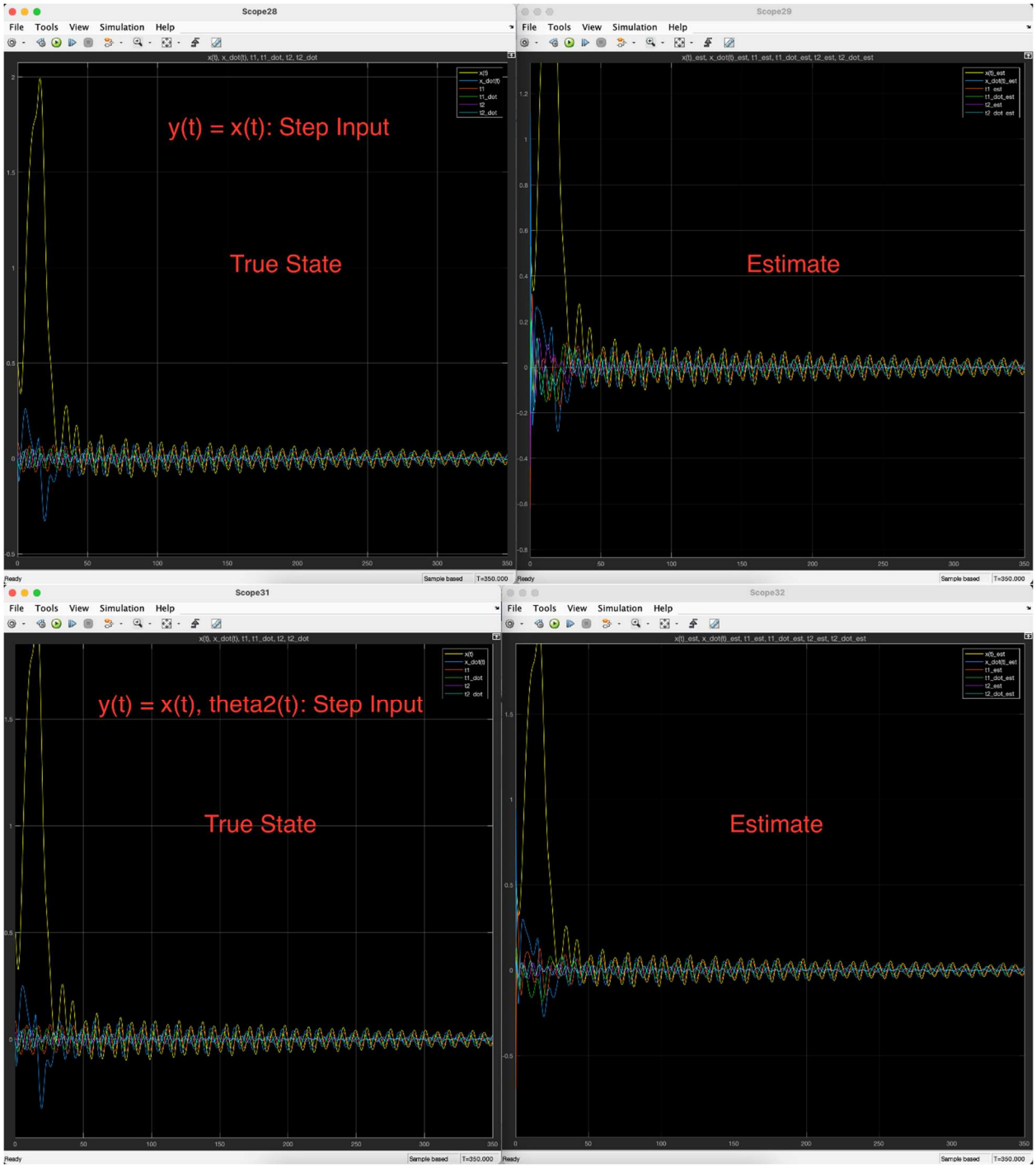

Scope17
File Tools View Simulation Help
x_hat, x_dot_hat, t1_hat, t1_dot_hat, t2_hat, t2_dot_hat
Estimate
Ready
Sample based
T=350.000

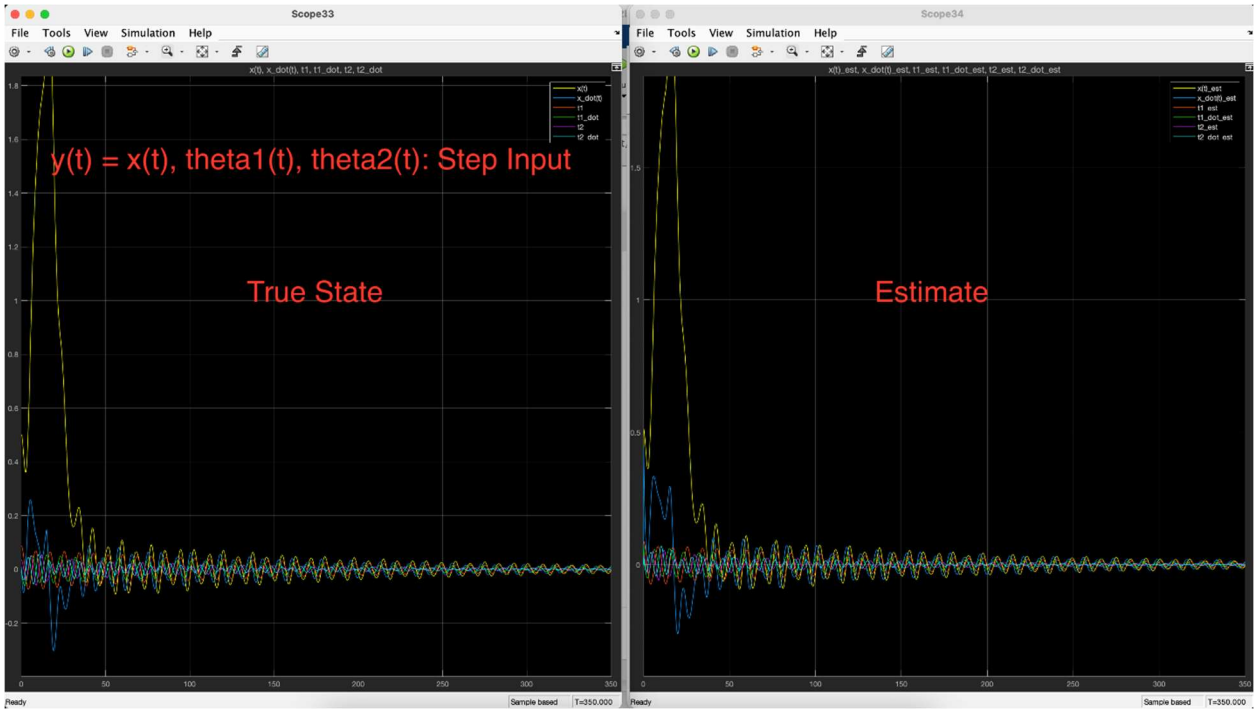
2. Luenberger model for the non-linearized system:






Scope22
File Tools View Simulation Help
x(t), x_dot(t), t1, t1_dot, t2, t2_dot
y(t) = x(t) , theta2 : Inital condition
True State
Scope23
File Tools View Simulation Help
x(t)_est, x_dot(t)_est, t1_est, t1_dot_est, t2_est, t2_dot_est
Estimate
Scope24
File Tools View Simulation Help
x(t), x_dot(t), t1, t1_dot, t2, t2_dot
y(t) = x(t), theta1(t), theta2(t): Inital Condition
Treu State
Scope25
File Tools View Simulation Help
x(t)_est, x_dot(t)_est, t1_est, t1_dot_est, t2_est, t2_dot_est
Estimate


Scope28
File Tools View Simulation Help
x(t), x_dot(t), t1, t1_dot, t2, t2_dot
y(t) = x(t): Step Input
True State
Ready
Sample based
T=350.000
Scope29
File Tools View Simulation Help
x(t)_est, x_dot(t)_est, t1_est, t1_dot_est, t2_est, t2_dot_est
Estimate
Ready
Sample based
T=350.000
Scope31
File Tools View Simulation Help
x(t), x_dot(t), t1, t1_dot, t2, t2_dot
y(t) = x(t), theta2(t): Step Input
True State
Ready
Sample based
T=350.000
Scope32
File Tools View Simulation Help
x(t)_est, x_dot(t)_est, t1_est, t1_dot_est, t2_est, t2_dot_est
Estimate
Ready
Sample based
T=350.000



G) LQG Controller:

For this design we will use y(t) = x(t) as the smallest output vector.

System modifications: In this system, we will consider the disturbances and noise as part of the input to the system and make required changes to the system to accommodates these inputs as shown below

- Augment B matrix to accept input, disturbance and filter the noise terms as follows

  BF = [B Cov_Dist 0*B]

- Augment D matrix to just accepr the measurement noise as part of the input and filter the other parts as follow

  D = [0 0 0 0 0 0 0 Cov_Noise]

- Simulation input Augmentation

  u_aug = [u disturbance noise]

Feedback control: We design feedback control using LQR controller with the following parameters

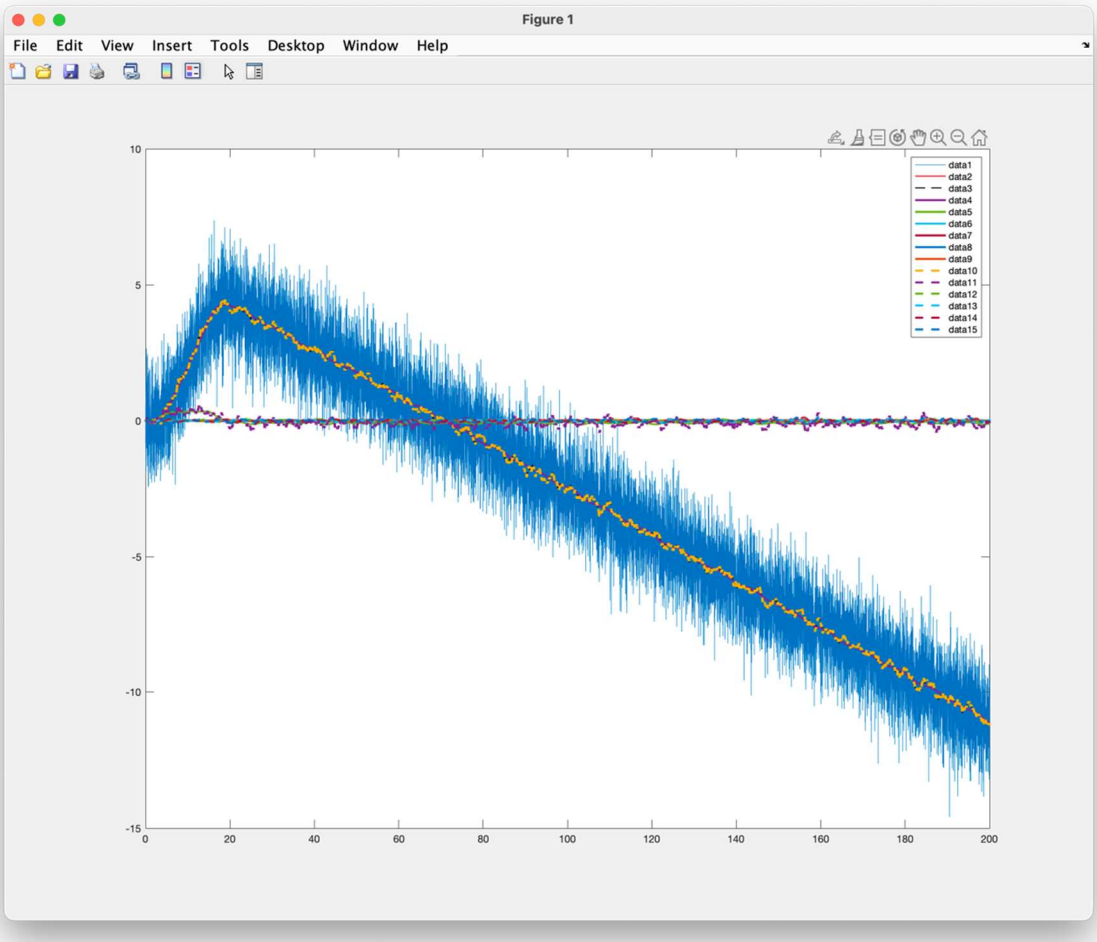
```
LQR parameters:

lqg_Q =

    0.0100         0         0         0         0         0
         0    0.0100         0         0         0         0
         0         0    0.0100         0         0         0
         0         0         0    0.0100         0         0
         0         0         0         0    0.0100         0
         0         0         0         0         0    0.0100


lqg_R =

   100

LQR gains are found as:

lqg_K =

    0.0100    4.8990   -0.0165   -8.1310   -0.0083   -4.0661
```

Kalman Filter: To design LQG controller we model Kalman filter with gain Kf as below

Akf. =. A-Kf*C ; BKf = [B Kf]
Ckf = I_{6x6} ; Dkf = 0*[B Kf]

Where the gain Kf is determined using another LQR controller

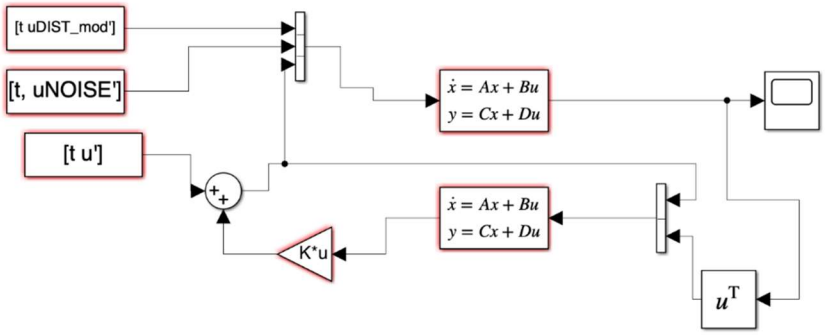
The Kalman filter state estimation for the single output system in simulation is as below:



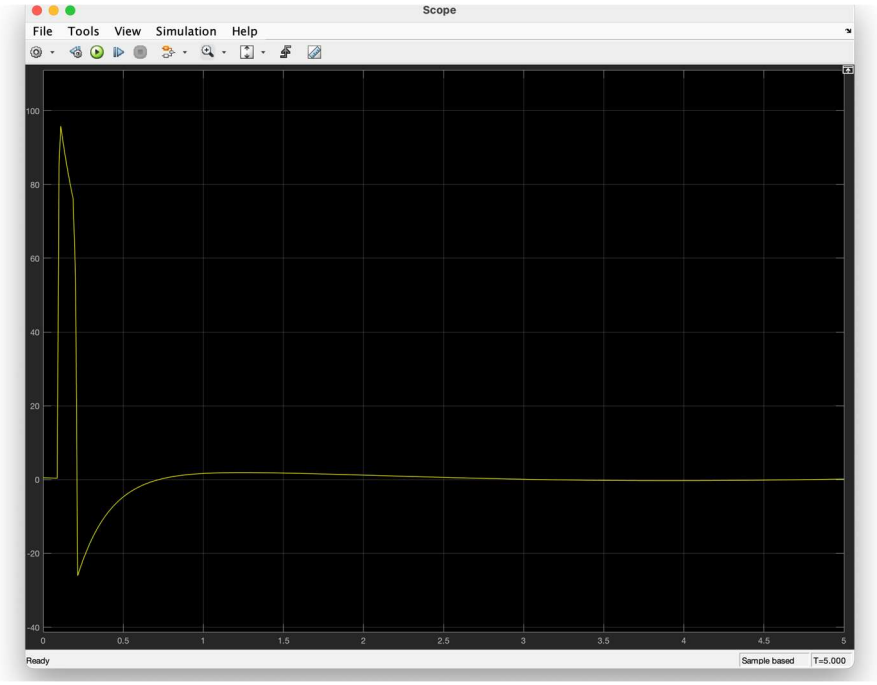
We can see from the above figure that the Kalman estimator is able to accurately estimate the entire state(dotted lines) even from a single noisy measurement of the output.

1. LQG Controller Simulation for linearized system:

Simulation result of Closed loop system with LQR feedback controller and Kalman filter estimator for input with process disturbances and measurement noise.



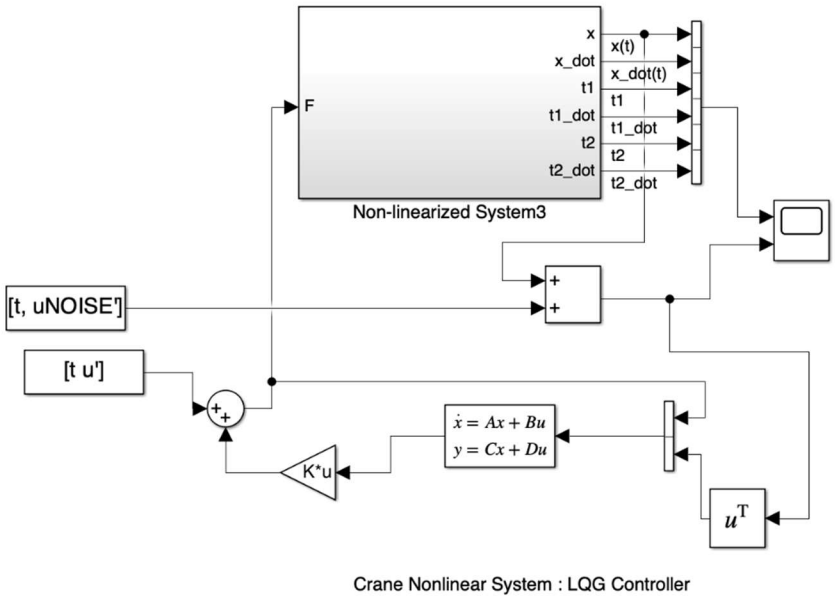
Crane LTI System : LQG Controller



We can see that for a step input with gaussian disturbance and noise the linearized system is able to recover to the stable state.

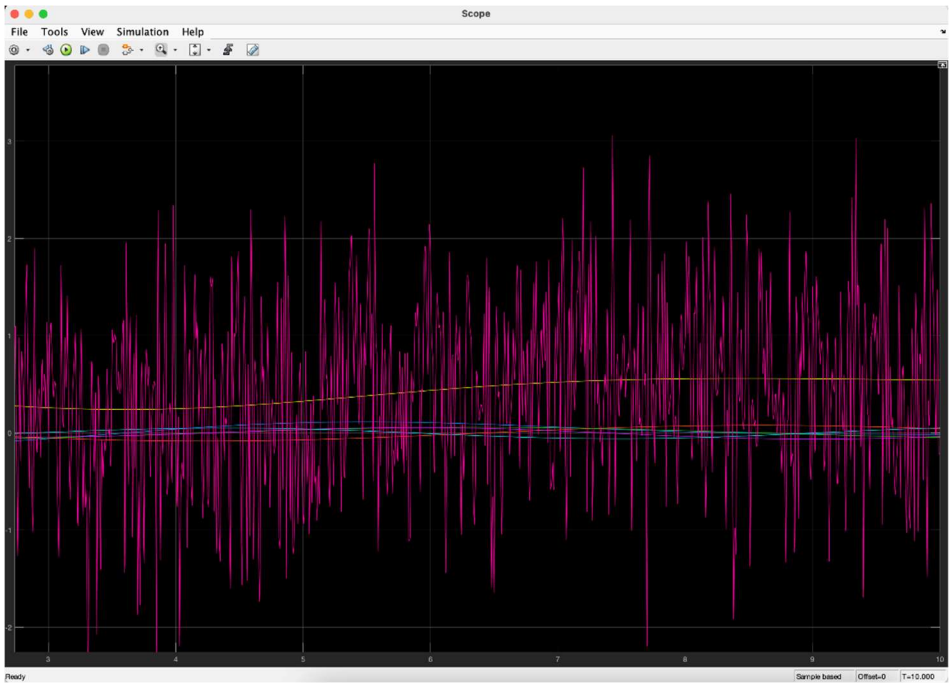
2. LQG Controller Simulation for non-linearized system:

The Simulation model for the nonlinearized system:



Crane Nonlinear System : LQG Controller

We can see that for a step input with gaussian disturbance and noise the nonlinear original system is able to recover to the stable state using the Kalman Filter.



Question:
How would you reconfigure your controller to asymptotically track a constant reference on x ?
Answer:
In order to optically track a constant reference asymptomatically, we will reconfigure our controller (LQG and LQR) to minimize the following cost function:

$$\int_0^\infty [x(t) - x(d)]^{\mathrm{T}} Q\,[x(t) - x(d)] \;+\; [u_k - u\infty]^{\mathrm{T}} R[u_k - u\infty]\,dt$$

Question:
Will your design reject constant force disturbances applied on the cart ?
Answer:
Under the assumption that the force disturbances applied on the cart are of Gaussian Nature, the designed controller will compensate for these disturbances.

Appendix:

Controllability:

```
from sympy import symbols, Matrix, simplify, shape, pprint

M, m1, m2, l1, l2, g, F = symbols(["M", "m1", "m2", "l1", "l2", "g", "F"])

print("This program verifies the controllability of the Linearized Crane system:")

A = Matrix([
      [0, 1, 0, 0, 0, 0],
      [0, 0, -1*g*m1/M, 0, -1*g*m2/M, 0],
      [0, 0, 0, 1, 0, 0],
      [0, 0, -g*(M+m1)/(M*l1), 0, -g*m2/(M*l1), 0],
      [0, 0, 0, 0, 0, 1],
      [0, 0, -1*g*m1/(M*l2), 0, -1*g*(M+m2)/(M*l2), 0]
    ])

B = Matrix([0, 1/M, 0, 1/(M*l1), 0, 1/(M*l2)])

pprint("A : ")
pprint(A)
pprint("B : ")
pprint(B)

pprint("Controllability Matrix of above LTI system is")
pprint("[ B A*B A^2*B A^3*B A^4*B A^5*B ]")

AB = A@B
A2B = A@A@B
A3B = A@A@A@B
A4B = A@A@A@A@B
A5B = A@A@A@A@A@B
ctrb = simplify(Matrix([
  [B[0], B[1], B[2], B[3], B[4], B[5]],
  [AB[0], AB[1], AB[2], AB[3], AB[4], AB[5]],
  [A2B[0], A2B[1], A2B[2], A2B[3], A2B[4], A2B[5]],
  [A3B[0], A3B[1], A3B[2], A3B[3], A3B[4], A3B[5]],
  [A4B[0], A4B[1], A4B[2], A4B[3], A4B[4], A4B[5]],
  [A5B[0], A5B[1], A5B[2], A5B[3], A5B[4], A5B[5]],
]))


pprint("We compute the full rank of the matrix by equating its determinant to zero.")
pprint("Determinant of above controllability matrix is computed as : ")
pprint(simplify(ctrb.det()))
```

Jacobian Linearization:

```
 from sympy import symbols, Matrix, simplify, shape, pprint, sin, cos, diff

M, m1, m2, l1, l2, g, F = symbols(["M", "m1", "m2", "l1", "l2", "g", "F"])

x, x_dot, x_dot_dot    = symbols(["x", "x_dot", "x_dot_dot"])
t1, t1_dot, t1_dot_dot = symbols(["t1", "t1_dot", "t1_dot_dot"])
t2, t2_dot, t2_dot_dot = symbols(["t2", "t2_dot", "t2_dot_dot"])

s1 = sin(t1)
c1 = cos(t1)

s2 = sin(t2)
c2 = cos(t2)


#Systems Non linear equations are given as
f1 = x_dot

f2 = (F - m1*(g*s1*c1 + l1*s1*t1_dot*t1_dot) - m2*(g*s2*c2 + l2*s2*t2_dot*t2_dot))/(M + m1*s1*s1+m2*s2*s2)

f3 = t1_dot

f4 = ((c1*f2) - (g*s1))/l1

f5 = t2_dot

f6 = ((c2*f2) - (g*s2))/l2

#Using Jacobian linearization Compute the system matrices as fllows
A = Matrix([
    [diff(f1, x), diff(f1, x_dot), diff(f1,t1), diff(f1,t1_dot), diff(f1,t2), diff(f1,t2_dot)],
    [diff(f2, x), diff(f2, x_dot), diff(f2,t1), diff(f2,t1_dot), diff(f2,t2), diff(f2,t2_dot)],
    [diff(f3, x), diff(f3, x_dot), diff(f3,t1), diff(f3,t1_dot), diff(f3,t2), diff(f3,t2_dot)],
    [diff(f4, x), diff(f4, x_dot), diff(f4,t1), diff(f4,t1_dot), diff(f4,t2), diff(f4,t2_dot)],
    [diff(f5, x), diff(f5, x_dot), diff(f5,t1), diff(f5,t1_dot), diff(f5,t2), diff(f5,t2_dot)],
    [diff(f6, x), diff(f6, x_dot), diff(f6,t1), diff(f6,t1_dot), diff(f6,t2), diff(f6,t2_dot)],
    ])

B = Matrix([
     [diff(f1, F)],
     [diff(f2, F)],
     [diff(f3, F)],
     [diff(f4, F)],
     [diff(f5, F)],
     [diff(f6, F)],
])

#Lyapunov Indirect Method or Jacobian Linearization around equilibrium point
A_linear = A.subs({t1:0, t2:0, t1_dot:0, t2:0, t2_dot:0})
B_linear = B.subs({t1:0, t2:0, t1_dot:0, t2:0, t2_dot:0})

pprint("A Linear : ")
pprint(A_linear)
pprint("B Lieanr : ")
pprint(B_linear)
```

Lyapunov indirect method

```
from sympy import symbols, Matrix, simplify, shape, pprint, sin, cos, diff

M, m1, m2, l1, l2, g, F = symbols(["M", "m1", "m2", "l1", "l2", "g", "F"])
```

```
x, x_dot, x_dot_dot    = symbols(["x", "x_dot", "x_dot_dot"])
t1, t1_dot, t1_dot_dot = symbols(["t1", "t1_dot", "t1_dot_dot"])
t2, t2_dot, t2_dot_dot = symbols(["t2", "t2_dot", "t2_dot_dot"])

s1 = sin(t1)
c1 = cos(t1)

s2 = sin(t2)
c2 = cos(t2)


#Systems Non linear equations are given as
f1 = x_dot

f2 = (F - m1*(g*s1*c1 + l1*s1*t1_dot*t1_dot) - m2*(g*s2*c2 + l2*s2*t2_dot*t2_dot))/(M +
m1*s1*s1+m2*s2*s2)

f3 = t1_dot

f4 = ((c1*f2) - (g*s1))/l1

f5 = t2_dot

f6 = ((c2*f2) - (g*s2))/l2

#Using Jacobian linearization Compute the system matrices as fllows
A = Matrix([
    [diff(f1, x), diff(f1, x_dot), diff(f1,t1), diff(f1,t1_dot), diff(f1,t2), diff(f1,t2_dot)],
    [diff(f2, x), diff(f2, x_dot), diff(f2,t1), diff(f2,t1_dot), diff(f2,t2), diff(f2,t2_dot)],
    [diff(f3, x), diff(f3, x_dot), diff(f3,t1), diff(f3,t1_dot), diff(f3,t2), diff(f3,t2_dot)],
    [diff(f4, x), diff(f4, x_dot), diff(f4,t1), diff(f4,t1_dot), diff(f4,t2), diff(f4,t2_dot)],
    [diff(f5, x), diff(f5, x_dot), diff(f5,t1), diff(f5,t1_dot), diff(f5,t2), diff(f5,t2_dot)],
    [diff(f6, x), diff(f6, x_dot), diff(f6,t1), diff(f6,t1_dot), diff(f6,t2), diff(f6,t2_dot)],
    ])

B = Matrix([
     [diff(f1, F)],
     [diff(f2, F)],
     [diff(f3, F)],
     [diff(f4, F)],
     [diff(f5, F)],
     [diff(f6, F)],
])


#For Problem C. We can compute the closed loop stability using Lyapunov idirect methd as follows
print("\n\nWe can compute the closed loop stability using Lyapunov idirect methd as follows:")
A_eqlbrm = A.subs({M:1000, m1:100, m2:100, l1:20, l2:10, g:9.8, F:0, t1:0, t2:0, t1_dot:0, t2:0,
t2_dot:0})
B_eqlbrm = B.subs({M:1000, m1:100, m2:100, l1:20, l2:10, g:9.8, F:0, t1:0, t2:0, t1_dot:0, t2:0,
t2_dot:0})
K = Matrix(1,6, [1.0000*1000,    1.8642*1000,    0.0388*1000,   -1.0226*1000,    0.3305*1000,   -
0.5741*1000])
pprint("A Linear : ")
pprint(A_eqlbrm)
pprint("B Lieanr : ")
```

```
pprint(B_eqlbrm)
pprint("\nLQR Gains : ")
pprint(K)

#Eigen values from LQR based closed loop Controller
pprint("\nEigen values from LQR based closed loop Controller A-B*K : ")
for key in (A_eqlbrm-B_eqlbrm*K).eigenvals().keys():
    pprint(key)
    pprint("\n")
```

Instructions to run the Matlab Simulatios:

The required files to run the MATLAB Simulations can be found at the link below:
https://drive.google.com/drive/folders/1TPuixiwANnZ228ZISqYDvnhKHkcSi3zg?usp=sharing

To Run LQR models
linear_system.m
lqr_controller.m

To Run Luenberger Observer model
observability.m
luenberger_observer.m

To Run LQG model
kalman_filter.m
Open relevant model

Linear System Model:

```
syms M m1 m2 l1 l2 g F
syms x theta1 theta2
syms x_dot theta1_dot theta2_dot
syms x_double_dot theta1_double_dot theta2_double_dot

A = [ 0 1 0 0 0 0;
      0 0 -g*m1/M 0 -g*m2/M 0;
      0 0 0 1 0 0;
      0 0 -g*(M+m1)/(M*l1) 0 -g*m2/(M*l1) 0;
      0 0 0 0 0 1;
      0 0 -g*m1/(M*l2) 0 -g*(M+m2)/(M*l2) 0
    ]

B = [0; 1/M; 0; 1/(M*l1); 0; 1/(M*l2)]
```

LQR Controller:

```
disp('Given Parameters of the System\n')
```

```
fprintf('M : %i, m1:%i, m2:%i, l1:%i, l2:%i', 1000, 100, 100, 20, 10)

A = double(subs(A, [M m1 m2, l1, l2, g],[1000 100 100, 20, 10, 9.8]))
B = double(subs(B, [M m1 m2, l1, l2, g],[1000 100 100, 20, 10, 9.8]))

% fprintf(['Eigen values of A :\n'])
% eigs(A)


Sc = ctrb(A, B);

disp('------------------------------------------------------------------------
-----')
disp('Check if system is controllable using rank condition\n')
disp('------------------------------------------------------------------------
-----')
fprintf(['Rank of Controllability matrix ' ...
    'is : %i\n'], rank(ctrb(A, B)))

if rank(ctrb(A, B)) == 6
    disp('System is Controllable.\n')
else
    disp('System is Uncontrollable.\n')
end

disp(newline)
disp('------------------------------------------------------------------------
-----')
disp('Find gain matrix using LQR method\n')
disp('------------------------------------------------------------------------
-----')



disp('LQR parameters:')


 Q = [1 0 0 0 0 0;
      0 1 0 0 0 0;
      0 0 10 0 0 0;
      0 0 0 10 0 0;
      0 0 0 0 10 0;
      0 0 0 0 0 10]
```

```
 R = 0.001

disp('LQR gains are found as:')
K = lqr(A,B,Q,R)

disp('Eigen Values of Closed loop system after  applying LQR.\n')
eigs(A-(B*K))

disp(newline)
disp('-------------------------------------------------------------------------
-----')
disp('Checking for stabiity of LQR Closed Loop system using Lyapnov indirect
method.')
disp('-------------------------------------------------------------------------
-----')
disp('Lyapnov Solution of Closed loop system after  applying LQR.\n')
P = lyap(A-(B*K), Q)


if issymmetric(P) && all(eigs(P) > 0)
    disp('Lyapnov Solution is Positive definite and hence system is stable')
else
    disp('Lyapnov Solution is not Positive definite and hence system is
unstable')
end
C = [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];
D = [0;0;0;0;0;0];
Init_S1 = [1; 0; deg2rad(1); 0; 0; deg2rad(0.5)];
Init_S2 = [0.5; 0.01; deg2rad(5); deg2rad(0.01); deg2rad(2); deg2rad(0.01)];
```

Kalman filter matlab script:

```
%In this system we use the lqr controller for feed back and
% solve for the observer gain using Kalman Bucy estimator

% the system output is considered to
% be the minimum observable state i.e, x(t)

%System and measurement noise covarience are defined as below
Vd = 0.1*eye(6);
Vn = 1;

% x_dot = Ax + Bu + Vd*d +  0*n
```

```
% y     = Cx + Du +  0*d + Vn*n

% Augemented B
BF = [B Vd 0*B]

% Augmented D : [0-for-B [0 0 0 0 0 0]-for-Vd Vn]
D  = [0 0 0 0 0 0 0 Vn]

sysC = ss(A, BF, C1, D)


disp('LQR parameters:')
lqg_Q = 0.01*eye(6)


lqg_R = 100

disp('LQR gains are found as:')
lqg_K = lqr(A,B,lqg_Q,lqg_R)

sysFullOutput = ss(A, BF, eye(6), zeros(6, size(BF, 2)))

%% Build Kalman filter
% [Kf,P,E] = lqe(A, Vd, C, Vd, Vn)

Kf    = (lqr(A', C1', Vd, Vn))'

sysKF = ss(A-Kf*C1, [B Kf], eye(6), 0*[B Kf])


%% Simulate original system
 dt = 0.01;
 t = dt:dt:200

 uDIST  = randn(6, size(t, 2));
 uNOISE = randn(size(t));
 u = 0*t
 u(10:20)   = 100

 uDIST_mod = Vd*Vd*uDIST;

 uAUG = [u; uDIST_mod; uNOISE];

% dt = 0.01;
% t = dt:dt:200
```

```
%
% uDIST  = randn(6, size(t, 2));
% uNOISE = randn(size(t));
% u = 0*t
% u(200:600)   = 100
% u(1500:2000)   = -100
%
% uDIST_mod = Vd*Vd*uDIST;
%
% uAUG = [u; uDIST_mod; uNOISE];

[y, t] = lsim(sysC, uAUG, t);
plot(t, y);

%% Simulate full system
[xtrue, t] = lsim(sysFullOutput, uAUG, t);

hold on

plot(t,xtrue(:,1),'r','LineWidth',1.0)

%% Kalman Filter Estimate
[x,t] = lsim(sysKF, [u; y'], t);
plot(t, x(:,1), 'k--', 'linewidth', 1.0)

plot(t, xtrue, '-', t, x, '--', 'LineWidth', 2)
```

Luenberger Observer:

```
disp('Given Parameters of the System\n')

fprintf('M : %i, m1:%i, m2:%i, l1:%i, l2:%i', 1000, 100, 100, 20, 10)

A = double(subs(A, [M m1 m2, l1, l2, g],[1000 100 100, 20, 10, 9.8]))
B = double(subs(B, [M m1 m2, l1, l2, g],[1000 100 100, 20, 10, 9.8]))

fprintf(['Eigen values of A :\n'])
eigs(A)
```

```
disp('We have seen that the sytem is observable for states [x(t)]; [x(t), θ2(t)];
[x(t), θ1(t), θ2(t)]')
disp(newline)
disp(['Luenberger observer gain for each of these states can be solved ' ...
    'using LQR controller for the system [At-Ct*Lt]'])
disp(newline)
disp('--------------------------------')
disp('Case 1 : Luenberger observer gain for state = [x(t)] ')
disp('--------------------------------')



disp('LQR parameters:')

Q = 100 * eye(6);%rate of performance

R = 0.1;%energy efficiency

disp('LQR gains are found as:')
Lt = lqr(transpose(A),transpose(C1),Q,R);
L1 = transpose(Lt)

disp(newline)
disp('--------------------------------')
disp('Case 3 : state = [x(t), θ2(t)] ')
disp('--------------------------------')



disp('LQR parameters:')


disp('LQR gains are found as:')
Lt = lqr(transpose(A),transpose(C3),Q,R);
L3 = transpose(Lt)



disp(newline)
disp('--------------------------------')
disp('Case 4 : state = [x(t), θ1(t), θ2(t)] ')
disp('--------------------------------')


disp('LQR parameters:')
```

```
disp('LQR gains are found as:')
Lt = lqr(transpose(A),transpose(C4),Q,R);
L4 = transpose(Lt)
```

Observability :

```
disp('Given Parameters of the System\n')

fprintf('M : %i, m1:%i, m2:%i, l1:%i, l2:%i', 1000, 100, 100, 20, 10)

A = double(subs(A, [M m1 m2, l1, l2, g],[1000 100 100, 20, 10, 9.8]));
B = double(subs(B, [M m1 m2, l1, l2, g],[1000 100 100, 20, 10, 9.8]));


disp(newline)
disp('--------------------------------')
disp('Case 1 : state = [x(t)] ')
disp('--------------------------------')

% C1 = [1 0 0 0 0 0;
%       0 0 0 0 0 0;
%       0 0 0 0 0 0;
%       0 0 0 0 0 0;
%       0 0 0 0 0 0;
%       0 0 0 0 0 0];
C1 = [1 0 0 0 0 0];
D1 = 0;

Ct = transpose(C1);
At = transpose(A);

A1C = At*Ct;
A2C = At*At*Ct;
A3C = At*At*At*Ct;
A4C = At*At*At*At*Ct;
A5C = At*At*At*At*At*Ct;

O = [Ct A1C A2C A3C A4C A5C];

R = rank(O);

fprintf('Rank of Observability Matrix : %f', R)
disp(newline)

if R == 6
```

```
    disp("State is observable")
else
    disp("State is not observable")
end

disp(newline)




disp(newline)
disp('---------------------------------')
disp('Case 2 : state = [θ1(t), θ2(t)] ')
disp('---------------------------------')

C2 = [0 0 1 0 0 0; 0 0 0 0 1 0];
D2 = [0;0];

Ct = transpose(C2);
At = transpose(A);

A1C = At*Ct;
A2C = At*At*Ct;
A3C = At*At*At*Ct;
A4C = At*At*At*At*Ct;
A5C = At*At*At*At*At*Ct;

O = [Ct A1C A2C A3C A4C A5C];

R = rank(O);

fprintf('Rank of Observability Matrix : %f', R)
disp(newline)

if R == 6
    disp("State is observable")
else
    disp("State is not observable")
end

disp(newline)



disp(newline)
```

```
disp('--------------------------------')
disp('Case 3 : state = [x(t), θ2(t)] ')
disp('--------------------------------')

C3 = [1 0 0 0 0 0; 0 0 0 0 1 0];
D3 = [0;0];

Ct = transpose(C3);
At = transpose(A);

A1C = At*Ct;
A2C = At*At*Ct;
A3C = At*At*At*Ct;
A4C = At*At*At*At*Ct;
A5C = At*At*At*At*At*Ct;

O = [Ct A1C A2C A3C A4C A5C];

R = rank(O);

fprintf('Rank of Observability Matrix : %f', R)
disp(newline)

if R == 6
    disp("State is observable")
else
    disp("State is not observable")
end

disp(newline)


disp(newline)
disp('--------------------------------')
disp('Case 4 : state = [x(t), θ1(t), θ2(t)] ')
disp('--------------------------------')

C4 = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0];
D4 = [0;0;0];

Ct = transpose(C4);
At = transpose(A);

A1C = At*Ct;
A2C = At*At*Ct;
```

```
A3C = At*At*At*Ct;
A4C = At*At*At*At*Ct;
A5C = At*At*At*At*At*Ct;

O = [Ct A1C A2C A3C A4C A5C];

R = rank(O);

fprintf('Rank of Observability Matrix : %f', R)
disp(newline)

if R == 6
    disp("State is observable")
else
    disp("State is not observable")
end

disp(newline)
```