EE650 Mini Project Presentation

Cantilever Oscillation Control

Group Members :
Umesh Kumar (201062)
Manas Kumar (200555)
Ritick Gupta (200801)
Kaushlendra (200500)

Instructor:
Dr. Twinkle Tripathy
Electrical Engineering, IIT Kanpur

System Description and deriving state space model

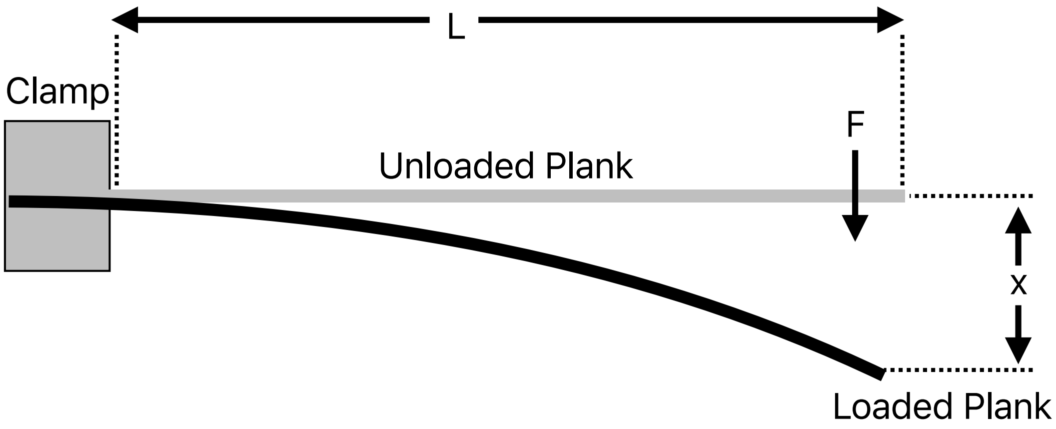
Let F is force applied and x is vertical displacement.

$\boldsymbol{x_1 = x}$ and $\boldsymbol{x_2 = x'}$

Applying Newton's second Law:

$$\boldsymbol{mx'' + kx + cx' = F}$$

Here F is input Force i.e. $\boldsymbol{u}\ (control\ input) = \boldsymbol{F}$



State Space Model (Cont…)

Using the values of x_1 and x_2 we get:

$$\boldsymbol{x_2' = -kx_1/m - cx_2/m + u/m}$$

$$\boldsymbol{x_1' = x_2}$$

The above equations are the state space equations.

As we want to control the vertical displacement i.e. $x1$,

$$\boldsymbol{y = x1 + 0.u = x1} \rightarrow Output\ Equation$$

Using above equation we can find the $A, B, C\ and\ D$ of state space model

State variables, Controllability and Observability matrices

$x_1 \rightarrow$ vertical deflection , $x_2 \rightarrow$ vertical velocity of the edge of the plank

(i) $\dot{x}_1 = x_2$

(ii) $\dot{x}_2 = -\frac{K}{m} x_1 - \frac{c}{m} x_2 + \frac{u}{m}$

$$\Rightarrow \dot{x} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]\, u$$

Controllability Matrix (Q_c)

$$Q_c = \begin{bmatrix} B & AB \end{bmatrix} = \begin{bmatrix} 0 & 1/m \\ 1/m & -c/m^2 \end{bmatrix}$$

$\rightarrow$ Rank = 2

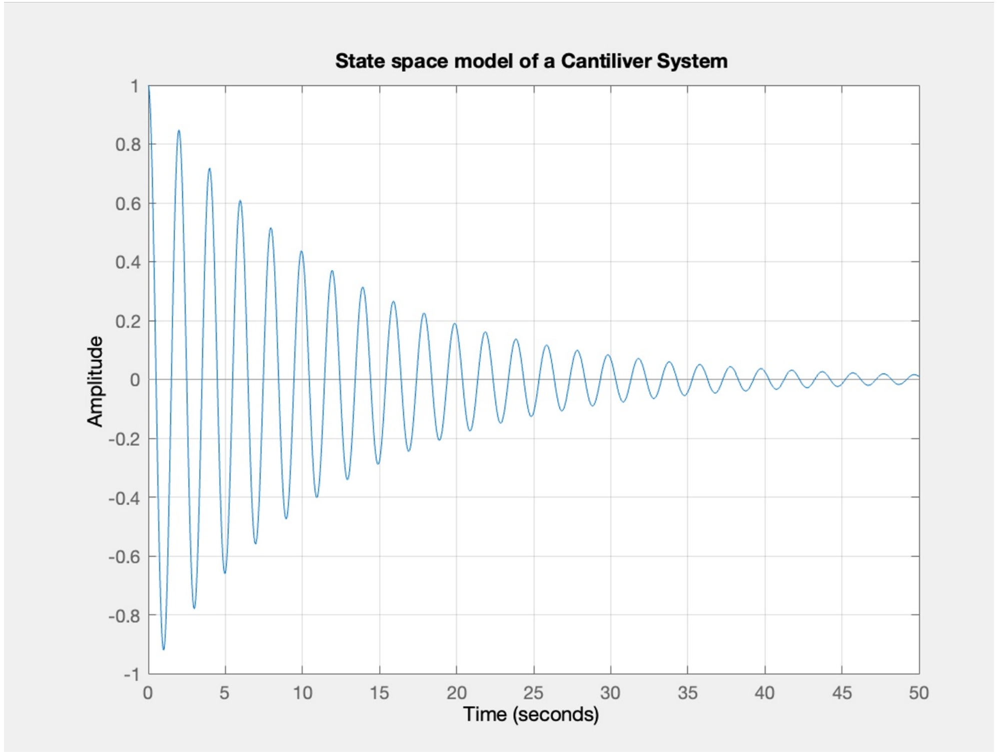
Hence the system is controllable.

Observability Matrix (Q_o)

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

Rank = 2 $\rightarrow$ Hence the system is observable.

System Response without controller:




```
>> stepinfo(sys)

ans =

  struct with fields:

        RiseTime: 0.3363
    SettlingTime: 46.7473
     SettlingMin: 2.5441e-04
     SettlingMax: 0.0032
       Overshoot: 92.0519
      Undershoot: 0
            Peak: 0.0032
        PeakTime: 0.9935
```

Objective of the controller

1. To decrease the peak overshoot of the mass dashpot
2. To decrease the settling time of the mass dashpot

Benefits:

- Reduces the chance of an accident
- Increases the lifetime of swimming board
- Avoiding damage due to stress

Design of the state feedback controller

→ To design a controller for the system we need to check the controllability of the system.

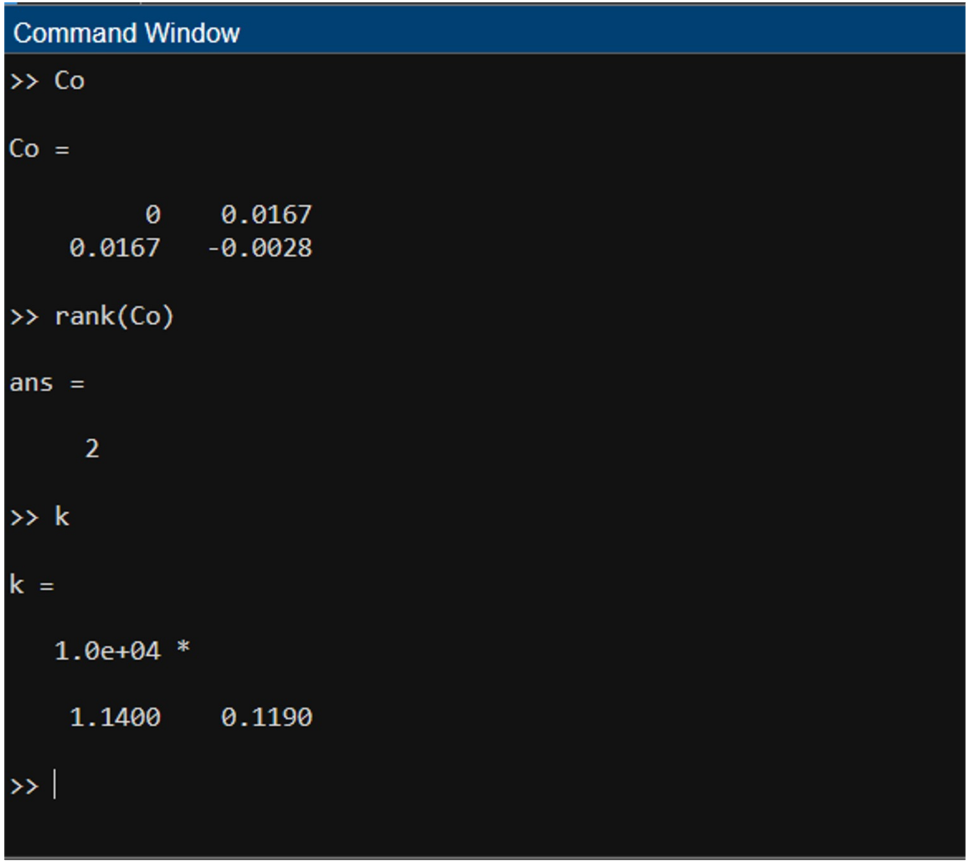
→ The system is controllable if $C = [B\ AB]$ is a full rank matrix

→ For our system, $\boldsymbol{C} = [\mathbf{0}\ \mathbf{0.0167}\ ;\ \mathbf{0.0167}\ -\mathbf{0.0028}]$, $rank(C) = 2$, hence the system is controllable

→ At present the poles of the system are located at $-\mathbf{0.083} + \mathbf{3.16}\boldsymbol{i}, -\mathbf{0.083} + \mathbf{3.16}\boldsymbol{i}$ which can be found by $\boldsymbol{det}(\boldsymbol{sI} - \boldsymbol{A})$

→ In order to meet the desired objective of the controller we place the poles of the controlled system at $-\mathbf{10} - \mathbf{10}\boldsymbol{i}, -\mathbf{10} + \mathbf{10}\boldsymbol{i}$ using a state feedback controller

- Taking the constants:
  - $k = 600$,
  - $c = 10$,
  - $m = 60$,
- The state feedback controller is given by:
  - $u = -k * x$
  - $k = [k1\ k2]$
- For finding $[k1\ k2]$, we use MATLAB



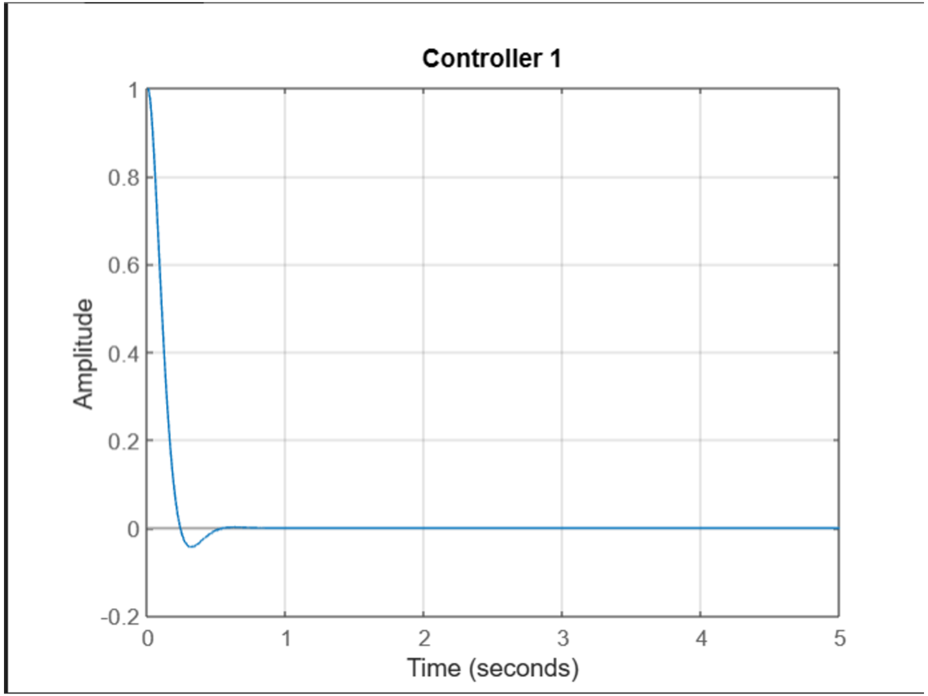
Code for controller

/MATLAB Drive/project.m

```
clc;clear,close all
%% State Space Model
k=600;
c=10;
m=60;
A = [0 1;-k/m -c/m];
B = [0; 1/m];
C = [1 0];
D = 0;
eigenVal = eig(A);
t=0:0.0001:5;
u=zeros(size(t));
sys = ss(A,B,C,D);
x0=[1 0];
%
figure(1)
lsim(sys,u,t,x0)
title('State space model of a Cantiliver System')
grid on
```

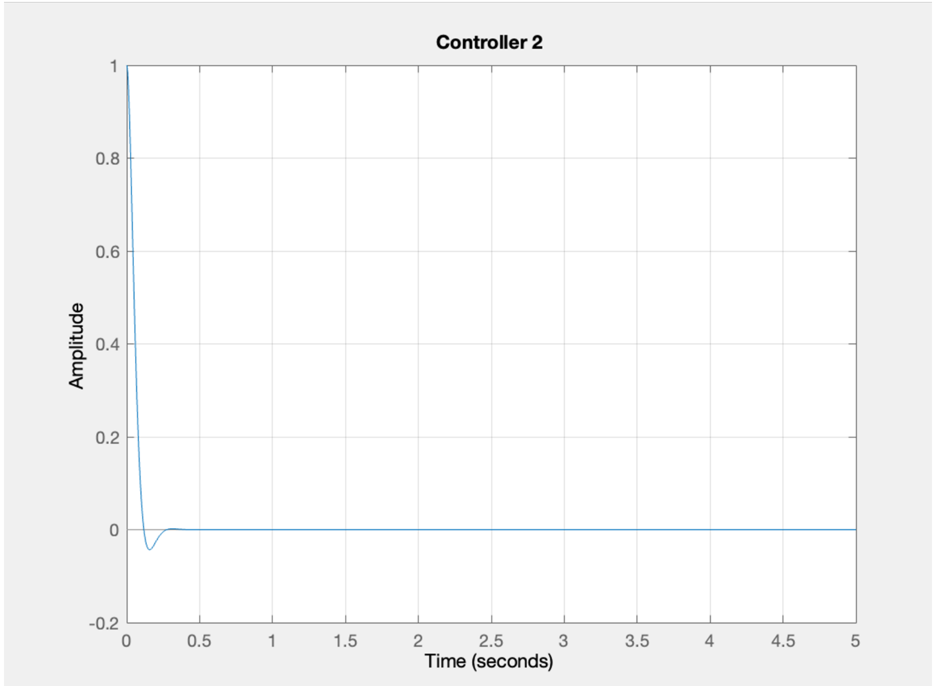
```
%% Controller
Co=ctrb(A,B)
rank(Co)
k=place(A,B,[-10+10i, -10-10i]); %poles for 2 dimensional matrix
sys_cl=ss(A-B*k,B,C,0);
eig(sys_cl);
%
figure(2)
lsim(sys_cl,u,t,x0)
title('Controller 1')
grid on
%
k1=place(A,B,[-20+20i, -20-20i]); %poles for 2 dimensional matrix
sys_cl2=ss(A-B*k1,B,C,0);
eig(sys_cl2);
%
figure(3)
lsim(sys_cl2,u,t,x0)
title('Controller 2')
grid on
```

System Response after using the Controller 1




```
>> stepinfo(sys_cl)

ans =

  struct with fields:

        RiseTime: 0.1519
    SettlingTime: 0.4216
     SettlingMin: 7.5286e-05
     SettlingMax: 8.6934e-05
       Overshoot: 4.3210
      Undershoot: 0
            Peak: 8.6934e-05
        PeakTime: 0.3132
```

System Response after using the Controller 2




```
>> stepinfo(sys_cl2)

ans =

  struct with fields:

        RiseTime: 0.0760
    SettlingTime: 0.2108
     SettlingMin: 1.8821e-05
     SettlingMax: 2.1734e-05
       Overshoot: 4.3210
      Undershoot: 0
            Peak: 2.1734e-05
        PeakTime: 0.1566
```

Justification for the Performance of the Controller

1. Decrease in the peak overshoot of the oscillations:
   - Peak overshoot without controller = 92.05
   - Peak overshoot with controller = 4.32
   - **Decrease in peak overshoot = 95.3%**
2. Decrease in the settling time of the oscillations:
   - Settling time without controller =46.7473
   - Settling time with controller 2 = 0.2108
   - **Decrease in Settling time = 99.5%**

Design of the Luenberger observer

→ To design an observer we need to check the observability of the system

→ The system is observable if the matrix $\boldsymbol{Q} = [\boldsymbol{C}\,;\ \boldsymbol{CA}]$ is full rank

→ $\boldsymbol{Q} = [\mathbf{1}\ \mathbf{0}\,;\ \mathbf{0}\ \mathbf{1}]$, rank of Q is 2 hence the system is observable

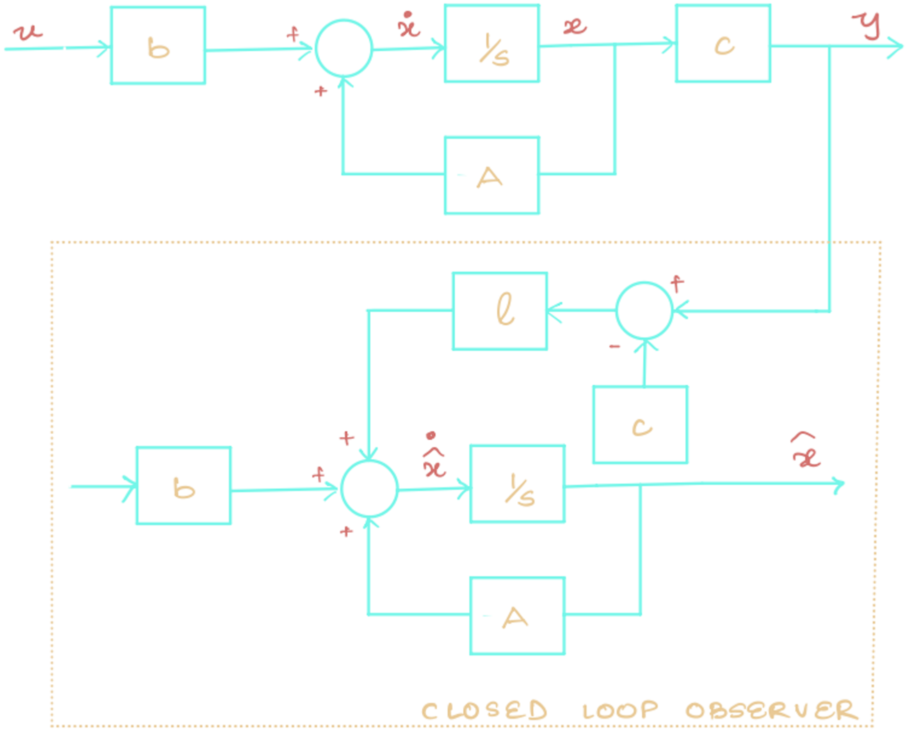
→ In order to design the Luenberger observer we need to find the observer gain L

→ We need to design L such that the eigenvalues of $\boldsymbol{A} - \boldsymbol{L} * \boldsymbol{C}$ lie in OLHP

→ We consider the eigenvalues of $\boldsymbol{A} - \boldsymbol{L} * \boldsymbol{C}$ to be $\mathbf{-10}, \mathbf{-11}$

→ Using MATLAB we find L

Block Diagram



- $\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly$
- $e(t) \; = \; x(t) - \hat{x}(t)$
- $\dot{e} \; = \; (A - LC)e$

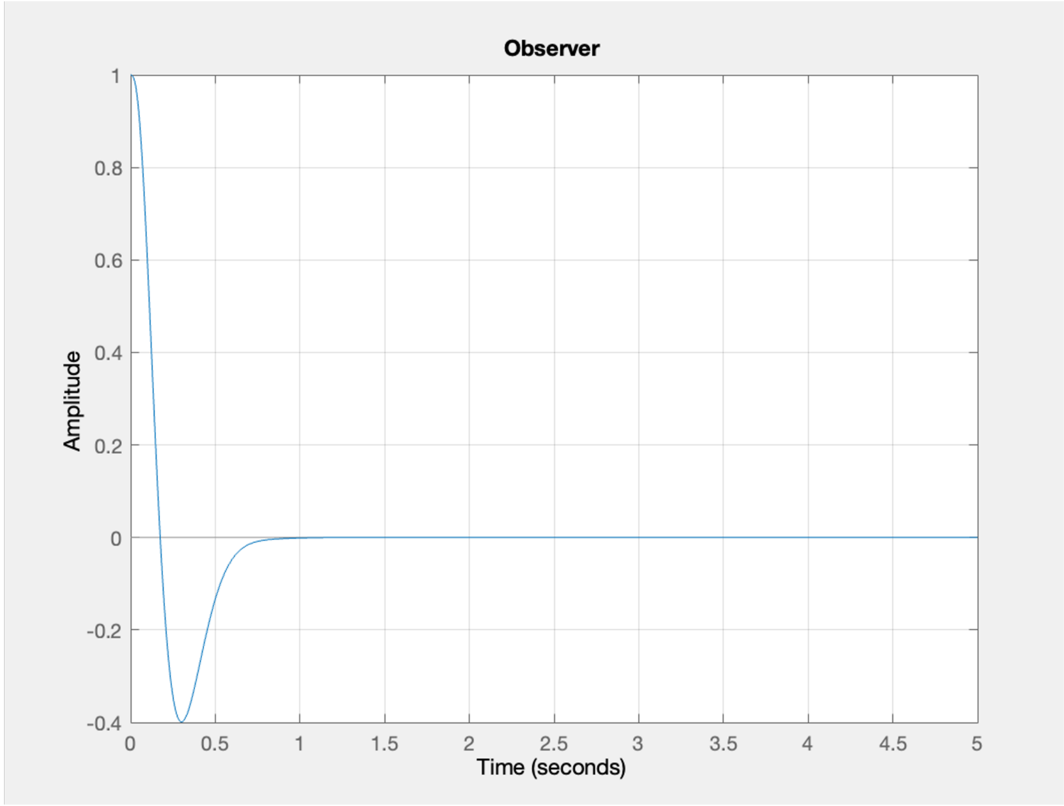
Finding Observer gain

```
>> ob

ob =

     1     0
     0     1

>> rank(ob)

ans =

     2

>> l=place(A',C', [-10,-11])'

l =

   20.8333
   96.5278
```

Code for observer with state estimator

```
%% Luenberger Observer
ob=obsv(A,C)
Ro=rank(ob)
l=place(A',C', [-10,-11])'; %k'
%
at=[A-B*k B*k;zeros(size(A)) A-l*C];
bt=[B; zeros(size(B))];
ct=[C zeros(size(C))];
sys_o=ss(at,bt,ct,0);
%  x1=[0 0];
%
figure(4)
lsim(sys_o,u,t,[x0 x0])
title('Observer')
grid on
```

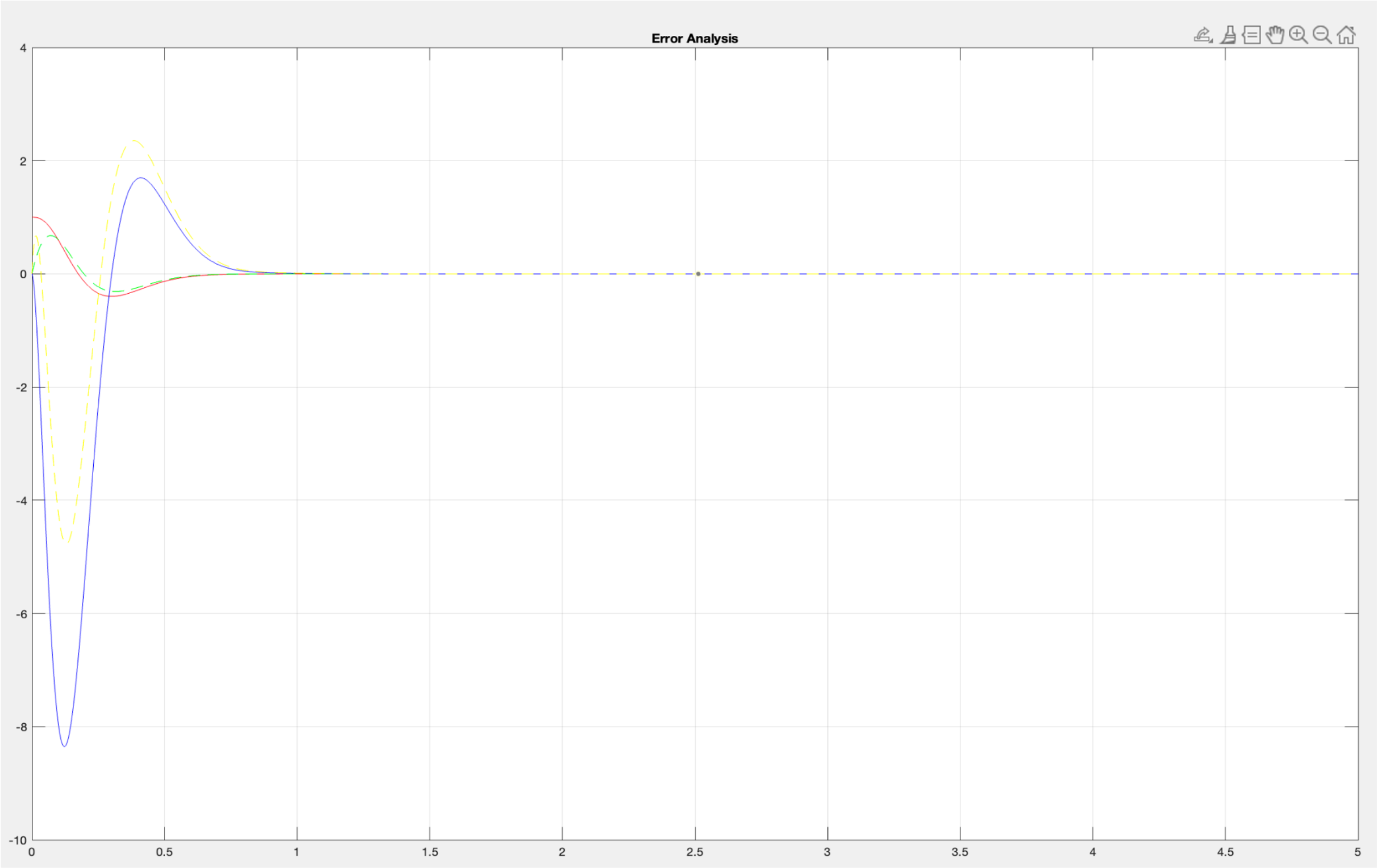
Observer Output



Error Analysis

```
%% Error analysis
n=2;
[y,t,x]=lsim(sys_o,u,t,[x0 x0]);
e=x(:,n+1:end);
x=x(:,1:n);
xe =x-e ; %obs
x1=x(:,1);x2=x(:,2); %plant states
xe1=xe(:,1);xe2=xe(:,2); %observer states
%
figure(5)
plot(t,x1,'-r',t,xe1,'--g',t,x2,'-b',t,xe2,'--y')
title('Error Analysis')
grid on
%
```

Plots



Justification for the Performance of the Observer

→ The estimation error for both the states goes to zero

→ The estimated states follow the system states

→ The observer archives its performance

Thank You