Table of Contents


```
% EE P 547
% Project- David Goniodsky, Hiral Mistry

clear;
close all;
clc;
```

Declaration

```
g=9.80665;  % kgm/s^2
kt=0.3233;  % in Nm/A
kb=0.4953;  % in Vs/rad
R=5.2628; % in ohm
rw=0.0216;    % radius of wheel in meter
mp=0.381;    % mass of pendulum in kg
mw=0.037;    % mass of wheel in kg
L=0.07472;     % distance between wheel center and reference point of
 pendulum in meter % Bit problematic, I think should be around 11 cm.
icmw=7.62997*10^(-5);  % moment of inertia at center of mass of wheel
ip=3.6051156*10^(-3);    % moment of inertia at reference point of
 pendulum


xhatini=[0;0;0;0];
tspan=0:0.01:10;
ref=[0;0;0;0];
```

step 1

Finding the matrices, A and B and finding linear continuous time state space representation of system

```
A(1,1)=0;
A(1,2)=1;
A(1,3)=0;
A(1,4)=0;
A(2,1)=(g*L*mp*(icmw+(mp+mw)*rw^2))/(icmw*(ip+L^2*mp)+(L^2*mp*mw
+ip*(mp+mw))*rw^2);
A(2,2)=-(kb*kt*(icmw+rw*(mw*rw+mp*(L+rw))))/(R*(icmw*(ip
+L^2*mp)+(L^2*mp*mw+ip*(mp+mw))*rw^2));
A(2,3)=0;
A(2,4)=-(kb*kt*(icmw+rw*(mw*rw+mp*(L+rw))))/(R*rw*(icmw*(ip
+L^2*mp)+(L^2*mp*mw+ip*(mp+mw))*rw^2));
A(3,1)=0;
A(3,2)=0;
A(3,3)=0;
A(3,4)=1;
A(4,1)=g*L^2*mp^2*rw^2/(icmw*(ip+L^2*mp)+(L^2*mp*mw+ip*(mp+mw))*rw^2);
A(4,2)= -(kt*kb*rw*(ip+L*mp*(L+rw)))/(R*(icmw*(ip+L^2*mp)+(L^2*mp*mw
+ip*(mp+mw))*rw^2));
A(4,3)=0;
A(4,4)=-(kt*kb*(ip+L*mp*(L+rw)))/(R*(icmw*(ip+L^2*mp)+(L^2*mp*mw
+ip*(mp+mw))*rw^2));

B(1,1)=0;
B(2,1)=-(kt*(icmw+rw*(mw*rw+mp*(L+rw))))/(R*(icmw*(ip
+L^2*mp)+(L^2*mp*mw+ip*(mp+mw))*rw^2));
B(3,1)=0;
B(4,1)=-(kt*rw*(ip+L*mp*(L+rw)))/(R*(icmw*(ip+L^2*mp)+(L^2*mp*mw
+ip*(mp+mw))*rw^2));

C=eye(4);

D=zeros(4,1);

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

```
sys =

  A =
            x1       x2       x3       x4
   x1        0        1        0        0
   x2    64.35   -22.91        0    -1061
   x3        0        0        0        1
   x4     3.15   -3.544        0   -164.1

  B =
            u1
   x1        0
   x2   -46.25
```

```
  x3       0
  x4  -7.155

 C =
       x1  x2  x3  x4
  y1    1   0   0   0
  y2    0   1   0   0
  y3    0   0   1   0
  y4    0   0   0   1

 D =
       u1
  y1    0
  y2    0
  y3    0
  y4    0

Continuous-time state-space model.
```

Step 2

Measured all the physical parameters and put the measured values in the above declaration section

Step 3

Finding the transfer function

```
[num,den]=ss2tf(A,B,C,D)
tf1=tf(num(1,:),den)
tf2=tf(num(2,:),den)
tf3=tf(num(3,:),den)
tf4=tf(num(4,:),den)


num =

         0         0  -46.2488   -0.0000   -0.0000
         0  -46.2488   -0.0000         0         0
         0         0   -7.1546   -0.0000  314.6927
         0   -7.1546   -0.0000  314.6927         0


den =

   1.0e+03 *

    0.0010    0.1870   -0.0643   -7.2161         0


tf1 =
```

```
  -46.25 s^2 - 2.68e-12 s - 1.467e-27
  -----------------------------------
  s^4 + 187 s^3 - 64.35 s^2 - 7216 s

Continuous-time transfer function.


tf2 =

      -46.25 s^3 - 2.68e-12 s^2
  ----------------------------------
  s^4 + 187 s^3 - 64.35 s^2 - 7216 s

Continuous-time transfer function.


tf3 =

   -7.155 s^2 - 8.896e-14 s + 314.7
  ----------------------------------
  s^4 + 187 s^3 - 64.35 s^2 - 7216 s

Continuous-time transfer function.


tf4 =

  -7.155 s^3 - 7.625e-14 s^2 + 314.7 s
  ------------------------------------
   s^4 + 187 s^3 - 64.35 s^2 - 7216 s

Continuous-time transfer function.
```

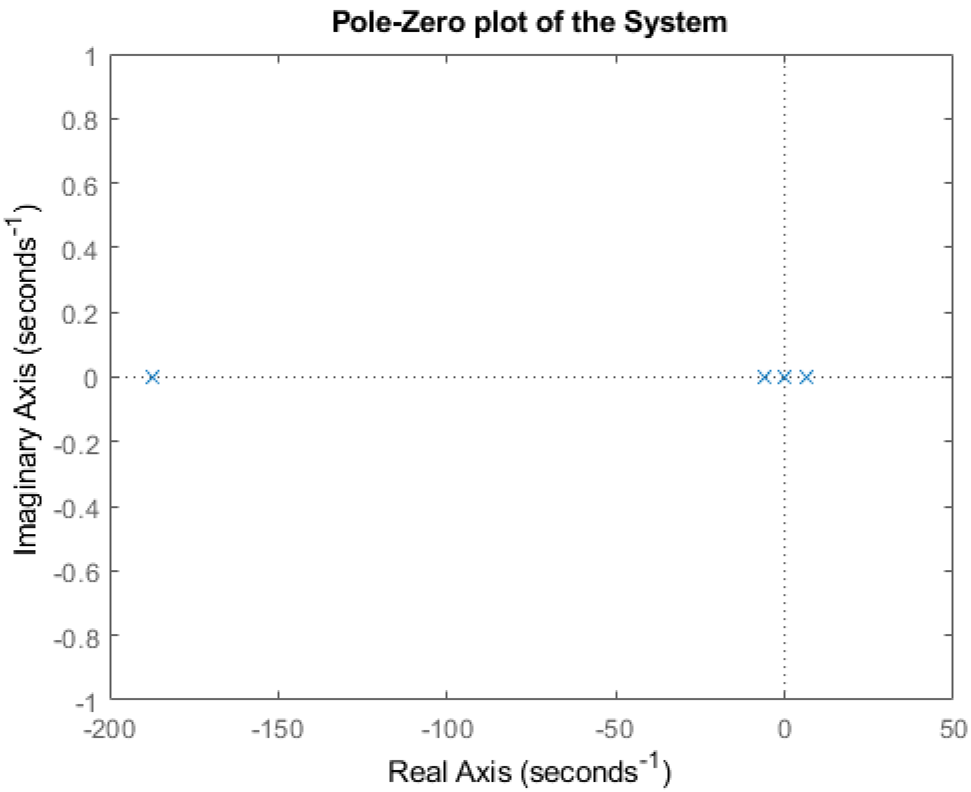
Step 4

Finding eigen values and Characteristic polynomial

```
eigen=eig(A)
CharPoly=poly(A)

% Plotiing the poles of system
figure(1);
pzplot(sys);
title('Pole-Zero plot of the System');
```

```
eigen =

         0
 -187.1037
    6.2795
   -6.1417
```

```
CharPoly =

   1.0e+03 *

    0.0010    0.1870   -0.0643   -7.2161         0
```



Step 5

Check for Asymptotically stable

```
if eigen<0
    fprintf('System is asymptotically stable.\n');
else
    fprintf('System is not asymptotically stable.\n');
end

% system is not asymptotically stable as all one of the poles is lying
 on
% right half of S-plane.
% System is not marginal stable because a pole lies on right half of
% S-plane.

System is not asymptotically stable.
```

Step 6

Finding poles of transfer function and checking for BIBO stability

```
rts=roots(CharPoly)

% The system is not BIBO stable as one of the pole is on right half of
% S-plane.
```

```
rts =

         0
 -187.1037
    6.2795
   -6.1417
```

Step 7

Check for the system Controllability

```
CtrbMatrix=ctrb(A,B)
CtrbRank=rank(CtrbMatrix)

if CtrbRank==size(A)
   fprintf('System is controllable.\n');
else
    fprintf('System is not controllable.\n');
end

% The system is controllable as the rank of controllability matrix is
 same
% as the size of A
```

```
CtrbMatrix =

   1.0e+08 *

         0   -0.0000    0.0001   -0.0162
   -0.0000    0.0001   -0.0162    3.0304
         0   -0.0000    0.0000   -0.0025
   -0.0000    0.0000   -0.0025    0.4682


CtrbRank =

     4

System is controllable.
```

Step 8

Check for the system Observability

```
ObsvMatrix=obsv(A,C)
ObsvRank=rank(ObsvMatrix)

if ObsvRank==size(A)
   fprintf('System is observable.\n');
else
    fprintf('System is not observable.\n');
end
```

```
ObsvMatrix =

   1.0e+07 *

    0.0000         0         0         0
         0    0.0000         0         0
         0         0    0.0000         0
         0         0         0    0.0000
         0    0.0000         0         0
    0.0000   -0.0000         0   -0.0001
         0         0         0    0.0000
    0.0000   -0.0000         0   -0.0000
    0.0000   -0.0000         0   -0.0001
   -0.0005    0.0004         0    0.0198
    0.0000   -0.0000         0   -0.0000
   -0.0001    0.0001         0    0.0031
   -0.0005    0.0004         0    0.0198
    0.0904   -0.0807         0   -3.7140
   -0.0001    0.0001         0    0.0031
    0.0139   -0.0125         0   -0.5738


ObsvRank =

     4

System is observable.
```

Step 9

Transforming the system to controllable and observable canonical form

```
% controllable canonical form
csys=canon(sys,'companion')

% observable canonical form
[A_obs,B_obs,C_obs,D_obs]=tf2ss(num,den) % observable canonical form
```

```
csys =

  A =
                x1          x2          x3          x4
   x1            0           0           0   3.606e-08
   x2            1           0           0        7216
   x3            0           1           0       64.35
   x4            0           0           1        -187

  B =
       u1
   x1   1
   x2   0
   x3   0
   x4   0

  C =
                 x1           x2           x3           x4
   y1             0       -46.25         8647    -1.62e+06
   y2        -46.25         8647    -1.62e+06     3.03e+08
   y3             0       -7.155         1338   -2.502e+05
   y4        -7.155         1338   -2.502e+05    4.682e+07

  D =
       u1
   y1   0
   y2   0
   y3   0
   y4   0

Continuous-time state-space model.


A_obs =

   1.0e+03 *

   -0.1870    0.0643    7.2161         0
    0.0010         0         0         0
         0    0.0010         0         0
         0         0    0.0010         0


B_obs =

     1
     0
     0
     0


C_obs =

         0  -46.2488   -0.0000   -0.0000
```

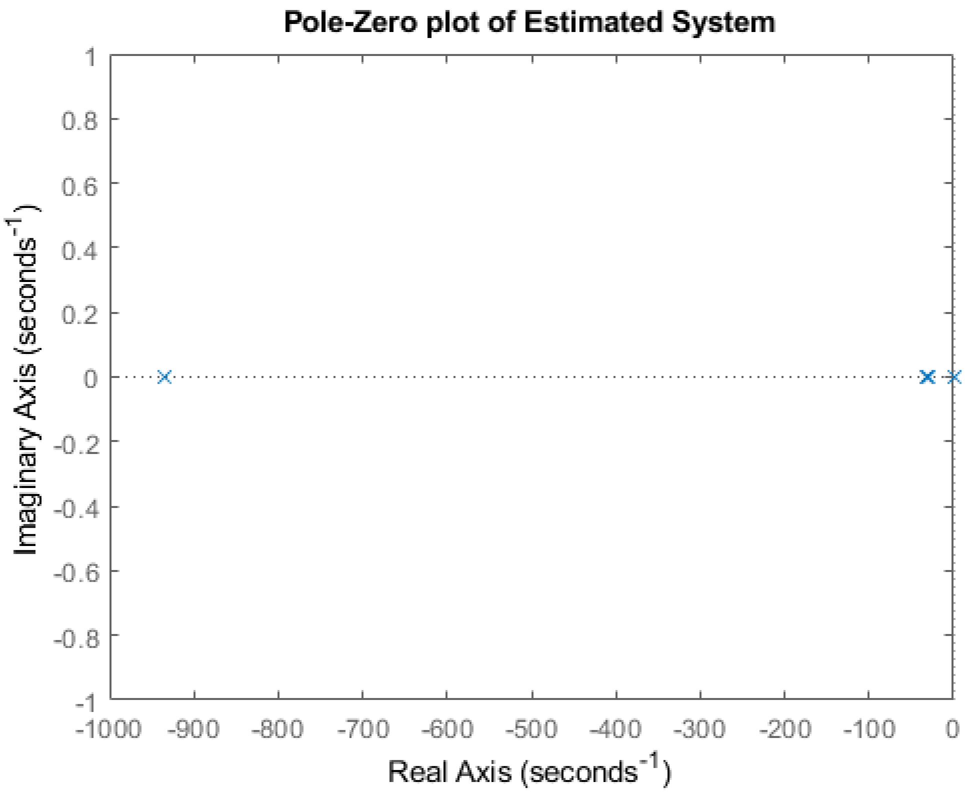
```
  -46.2488   -0.0000         0         0
         0   -7.1546   -0.0000  314.6927
   -7.1546   -0.0000  314.6927         0


D_obs =

     0
     0
     0
     0
```

Step 10

Placing the poles of system in such a way that it stabilize the system and makes 6-8 times faster by placing the poles much more far away from the y-axis in the left-half of S-plane and finding the gain L

```
p1=[0 -187.1037 -6.2795 -6.1417];
poles=5*p1
L=place(A',C',poles);
L=L'
Aobs=A-L*C;
sys_obs=ss(Aobs,B,C,D);

% Plotting the poles of original system and estimated system and
 comparing
% them.
figure(2);
pzplot(sys_obs);
title('Pole-Zero plot of Estimated System');
```

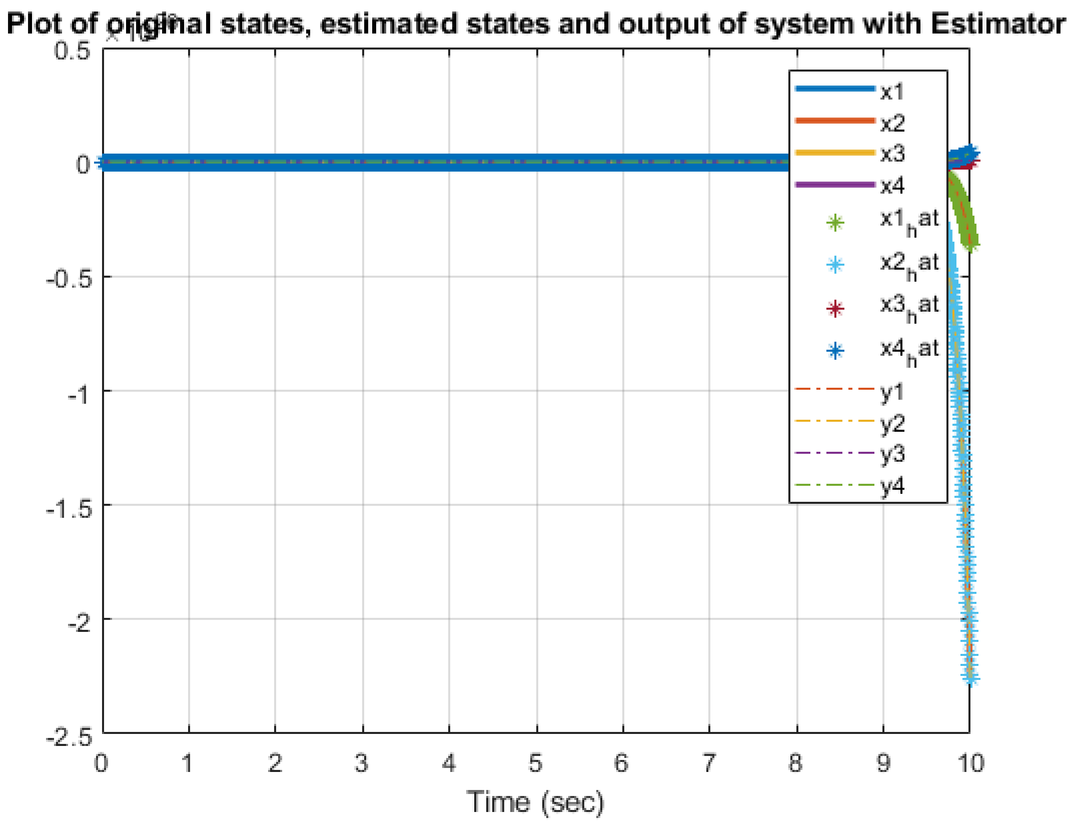
```
poles =

         0 -935.5185  -31.3975  -30.7085


L =

   1.0e+03 *

         0    0.0010         0         0
    0.0643    0.9126         0   -1.0605
         0         0    0.0314    0.0010
    0.0032   -0.0035         0   -0.1334
```



Step 11

Developed a Simulink Model and simulated it.

```
sim('FinalProject_EE547_Step11',tspan(end));
figure(3);
plot(t,x,'Linewidth',2);
hold on;
plot(t,x_hat,'*');
hold on;
plot(t,y,'-.');
hold off;
grid on;
xlabel('Time (sec)');
title('Plot of original states, estimated states and output of system
 with Estimator');
legend('x1','x2','x3','x4','x1_hat','x2_hat','x3_hat','x4_hat','y1','y2','y3','y4'
```



Step 12

Designing the controller and finding the proportional gain K

```
% poles1=[-0.1,-0.2,-0.3,-0.4];
% poles1=[-1+i,-1-i,-3,-4];
% poles1=[-1+i,-1-i,-0.5,-2];
% poles1=5*[-1+i,-1-i,-0.5,-2];
% poles1=150*[-1+i,-1-i,-0.5,-2];
% poles1=[-3,-187.1037,-8,-6.14];
% poles1=poles
poles1=[-67.2,-59.2,-12,-6]
K=place(A,B,poles1)
```

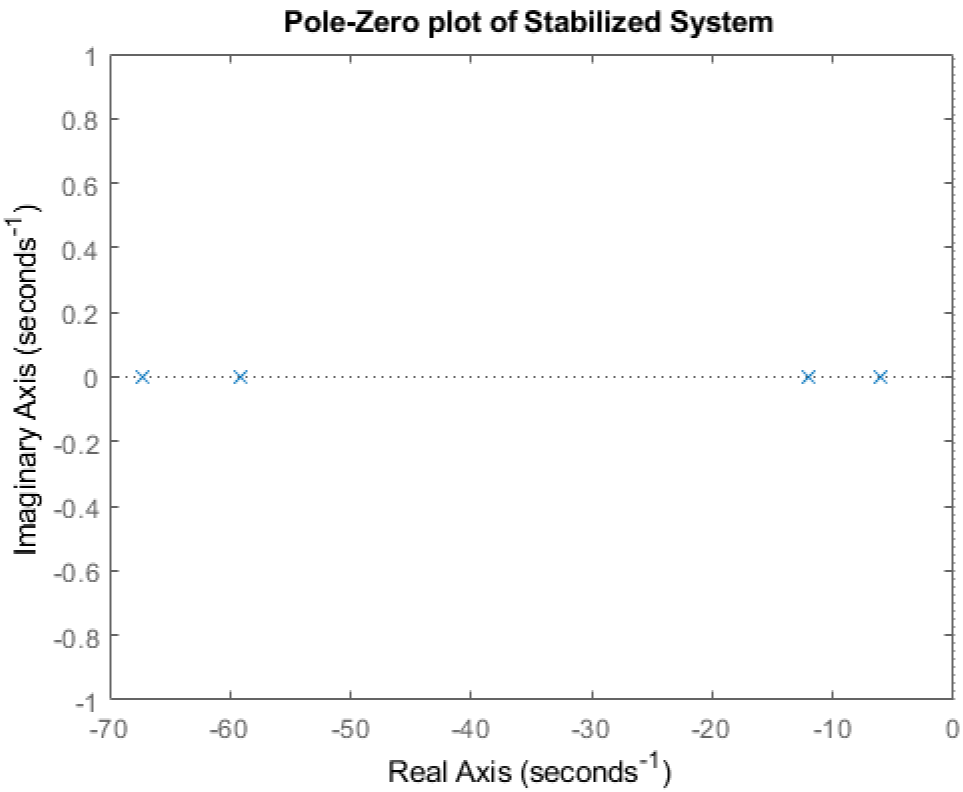
```
poles1 =

  -67.2000  -59.2000  -12.0000   -6.0000


K =

 -278.9672  -42.3023  910.1999  279.4002
```

Step 13

Deriving the state space representation of closed loop system ACL- A matrix of system with controller

```
ACL=A-B*K
sys_CL=ss(ACL,B,C,D);

% Plotting the poles of original system and Stabilized system and
 comparing
% them.
figure(4);
pzplot(sys_CL);
title('Pole-Zero plot of Stabilized System');

% Finding the eigen values and characteristic polynomial of closed
 loop
% system
eig_CL=eig(ACL)
Charpoly_CL=poly(ACL)

% Checking whether the system becomes asymptotically stable or not
if eig_CL < 0
    fprintf('System is asymptotically stable.\n');
else
    fprintf('System is not asymptotically stable.\n');
end
```

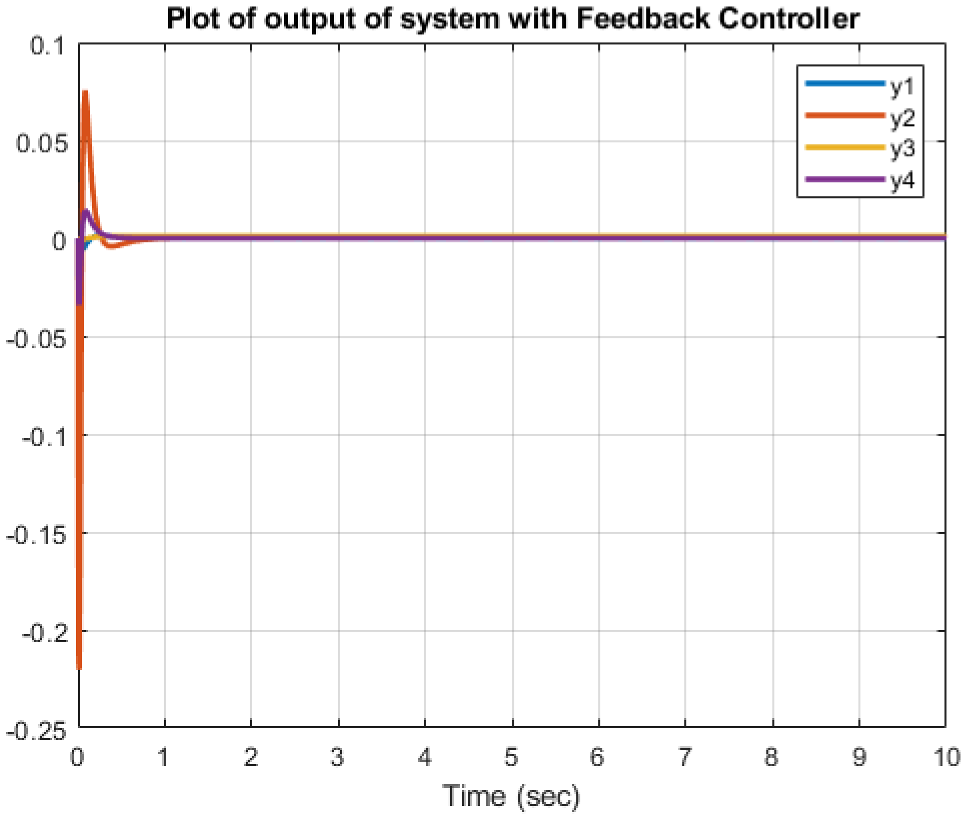
```
ACL =

   1.0e+04 *

         0    0.0001         0         0
   -1.2838   -0.1979    4.2096    1.1861
         0         0         0    0.0001
   -0.1993   -0.0306    0.6512    0.1835


eig_CL =

  -67.2000
  -59.2000
  -12.0000
   -6.0000


Charpoly_CL =

   1.0e+05 *

    0.0000    0.0014    0.0633    0.8071    2.8643

System is asymptotically stable.
```



Step 14

Making a Simulink model, simulating it and plotting state space variables and output.

```
sim('FinalProject_EE547_Step14',tspan(end));
figure(5);
plot(t,y,'linewidth',2);
grid on;
xlabel('Time (sec)');
legend('y1','y2','y3','y4');
title('Plot of output of system with Feedback Controller');
```

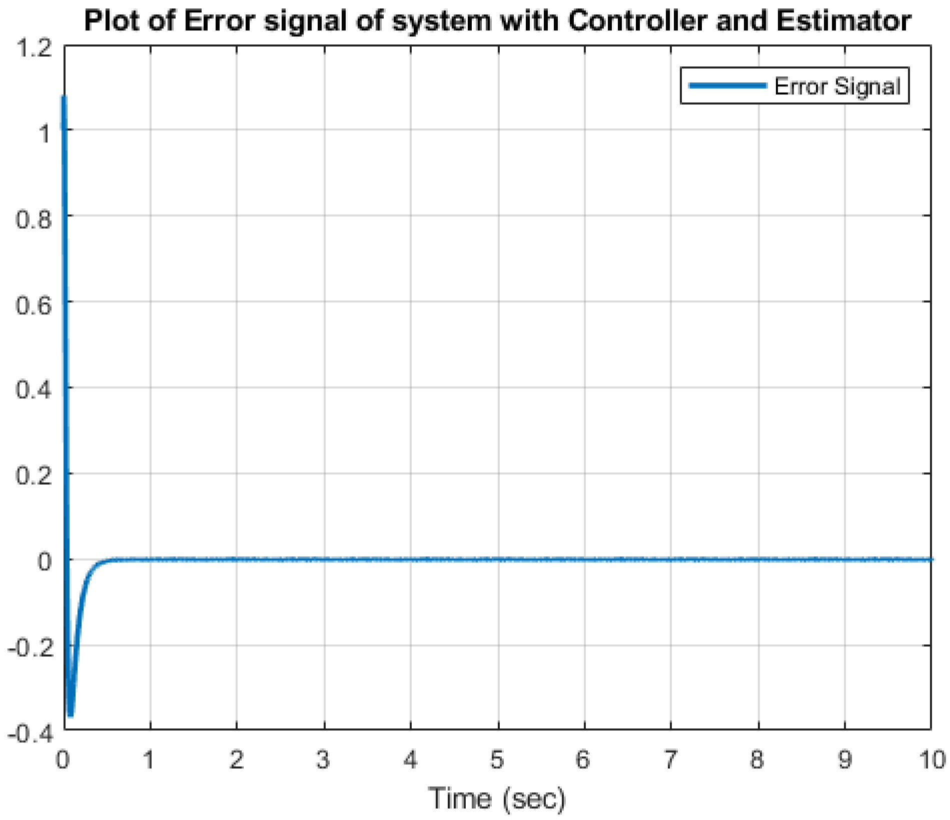


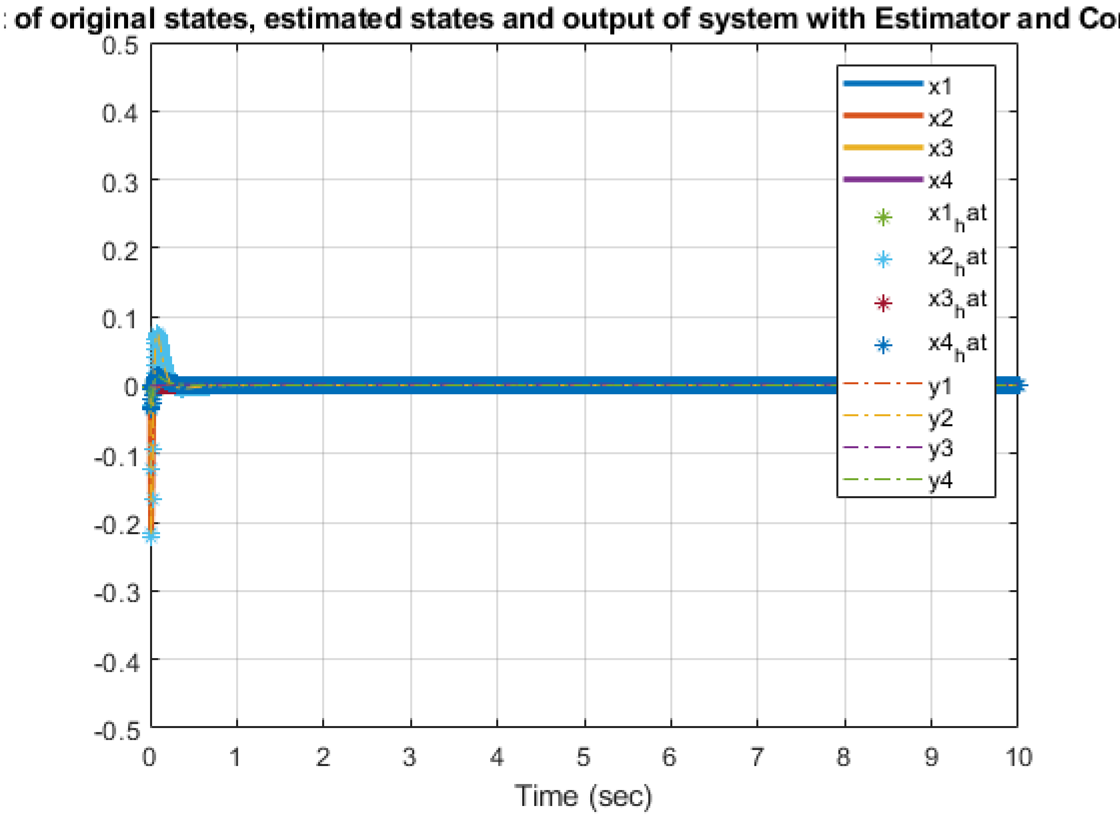
Step 15

Combining the feedback controller model with state estimator model in Simulink. Plotting the error function.

```
sim('FinalProject_EE547_Step15',tspan(end));
figure(6);
plot(t,error,'Linewidth',2);
grid on;
xlabel('Time (sec)');
legend('Error Signal');
title('Plot of Error signal of system with Controller and Estimator');

figure(7);
plot(t,x,'Linewidth',2);
hold on;
plot(t,x_hat,'*');
hold on;
plot(t,y,'-.');
hold off;
grid on;
xlabel('Time (sec)');
ylim([-0.5 0.5]);
title('Plot of original states, estimated states and output of system
 with Estimator and Controller');
legend('x1','x2','x3','x4','x1_hat','x2_hat','x3_hat','x4_hat','y1','y2','y3','y4'
```

```
% It can be seen form the plot that initially the error is very large,
 but
% with time, the error value decreases and eventually it settles down
 to
% zero. That means, the system has now become asymptotically stable.
```





Step 16

```
% Using the LQR method for implementing the model on simulink and
 balancing the Minseg in upright
% position. The complete Simulink model can be found in the attached
 slx
% document.
```

Published with MATLAB® R2018b