MARMARA UNIVERSITY

MECHANICAL ENGINEERING DEPARTMENT

ME7009 PROJECT

524618020 ÖZGÜRAZAD ÇELİK

Lecturer: Dr.Öğr.Üyesi UĞUR TÜMERDEM

ISTANBUL, 2022

Contents

System Linearization

Linearization of the System

$$\frac{d\dot{r}}{dt} = \ddot{r} = r\dot{\theta}^2 - \frac{\mu}{r^2} + 2T\sin\phi$$

$$\frac{d\dot{\theta}}{dt} = \ddot{\theta} = \frac{-2\dot{r}\dot{\theta}}{r} + \frac{T}{2r}\cos\phi$$

State vector $x = \begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix}$

Control input vector $u = \begin{bmatrix} \phi \\ T \end{bmatrix}$

Linearized State Equation:

$$\dot{x} = Ax + Bu$$

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \frac{dx}{dt} = \frac{d}{dt}\begin{bmatrix} r \\ \dot{r} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{r} \\ r\dot{\theta}^2 - \frac{\mu}{r^2} + 2T\sin\phi \\ \dot{\theta} \\ \frac{-2\dot{r}\dot{\theta}}{r} + \frac{T}{2r}\cos\phi \end{bmatrix}$$

$$= Ax + Bu = f(x,u)$$

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial r} & \frac{\partial f}{\partial \dot{r}} & \frac{\partial f}{\partial \theta} & \frac{\partial f}{\partial \dot{\theta}} \end{bmatrix}$$

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial r} & \frac{\partial \dot{x}_1}{\partial \dot{r}} & \frac{\partial \dot{x}_1}{\partial \theta} & \frac{\partial \dot{x}_1}{\partial \dot{\theta}} \\ \frac{\partial \dot{x}_2}{\partial r} & \frac{\partial \dot{x}_2}{\partial \dot{r}} & \frac{\partial \dot{x}_2}{\partial \theta} & \frac{\partial \dot{x}_2}{\partial \dot{\theta}} \\ \frac{\partial \dot{x}_3}{\partial r} & \frac{\partial \dot{x}_3}{\partial \dot{r}} & \frac{\partial \dot{x}_3}{\partial \theta} & \frac{\partial \dot{x}_3}{\partial \dot{\theta}} \\ \frac{\partial \dot{x}_4}{\partial r} & \frac{\partial \dot{x}_4}{\partial \dot{r}} & \frac{\partial \dot{x}_4}{\partial \theta} & \frac{\partial \dot{x}_4}{\partial \dot{\theta}} \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dot{\theta}^2 + \frac{2\mu}{r^3} & 0 & 0 & 2r\dot{\theta} \\ 0 & 0 & 0 & 1 \\ \frac{2\dot{r}\theta}{r^2} - \frac{T\cos\phi}{2r^2} & \frac{-2\dot{\theta}}{r} & 0 & -\frac{2\dot{r}}{r} \end{bmatrix}$$

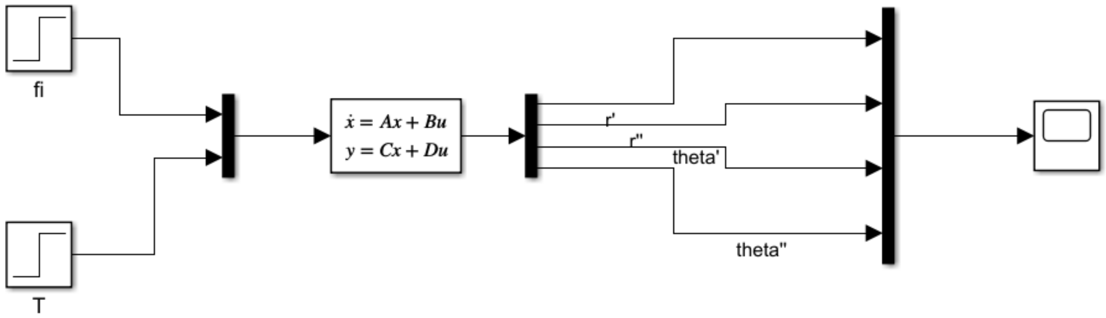
$$B = \frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f}{\partial \phi} & \frac{\partial f}{\partial T} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2T\cos\phi & 2\sin\phi \\ 0 & 0 \\ -\frac{T}{2r}\sin\phi & \frac{\cos\phi}{2r} \end{bmatrix}$$

At the nominal trajectory:

$r = r_0, \ \dot{r} = 0, \ \dot{\theta} = \omega, \ T = 0, \ \mu = r_0^3\omega^2, \ \phi = 90°$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3\omega^2 & 0 & 0 & 2\omega r_0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2\omega}{r_0} & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

System Initialization



We have to control at the first controllability and observability of the system in order to apply control.

```
w=0.05
r0=5000
A=[0 1 0 0;
    3*w^2 0 0 2*w*r0;
    0 0 0 1;
    0 (-2*w)/r0 0 0]
B=[0 0;
    0 2;
    0 0;
    0 0]
C=eye(4,4);
C1=[0 0 0 0;
    0 0 0 0;
    0 0 1 0;
    0 0 0 0];
D=zeros(4,2);




sys = ss(A,B,C,D)
[b,a] = ss2tf(A,B,C,D,2)
tf(sys)


% Co = ctrb(A,B)
Co = ctrb(sys)
```

```
unco = length(A) - rank(Co)


Ob= obsv(A,C)
% Number of unobservable states
unob = length(A)-rank(Ob)
```

```
w = 0.0500
r0 = 5000
A = 4×4
         0    1.0000         0         0
    0.0075         0         0  500.0000
         0         0         0    1.0000
         0   -0.0000         0         0
B = 4×2
     0     0
     0     2
     0     0
     0     0
sys =

  A =
            x1       x2       x3       x4
   x1        0        1        0        0
   x2   0.0075        0        0      500
   x3        0        0        0        1
   x4        0   -2e-05        0        0

  B =
       u1  u2
   x1   0   0
   x2   0   2
   x3   0   0
   x4   0   0

  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  u2
   y1   0   0
   y2   0   0
   y3   0   0
   y4   0   0

Continuous-time state-space model.
b = 4×5
         0         0    2.0000         0         0
         0    2.0000         0         0         0
         0         0         0   -0.0000         0
         0         0   -0.0000         0         0
a = 1×5
    1.0000         0    0.0025         0         0
ans =

  From input 1 to output...
   1:  0

   2:  0

   3:  0

   4:  0

  From input 2 to output...
            2 s
   1:  --------------
       s^3 + 0.0025 s

           2 s^2
   2:  --------------
       s^3 + 0.0025 s

           -4e-05 s
   3:  ----------------
       s^4 + 0.0025 s^2

          -4e-05 s
   4:  --------------
       s^3 + 0.0025 s

Continuous-time transfer function.
Co = 4×8
         0         0         0    2.0000         0         0         0   -
0.0050
         0    2.0000         0         0         0   -0.0050         0
0
         0         0         0         0         0   -0.0000         0
0
         0         0         0   -0.0000         0         0         0
0.0000
unco = 1
Ob = 16×4
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000
         0    1.0000         0         0
```

```
    0.0075         0         0  500.0000
         0         0         0    1.0000
         0   -0.0000         0         0
    0.0075         0         0  500.0000
         0   -0.0025         0         0
unob = 0
```

We change the value of T due to uncontrollability.

```
B=[0 0;
    0 2;
    0 0;
    -0.01 0]
```

```
Co = 4×8
         0         0         0    2.0000   -5.0000         0         0   -
0.0050
         0    2.0000   -5.0000         0         0   -0.0050    0.0125
0
         0         0   -0.0100         0         0   -0.0000    0.0001
0
   -0.0100         0         0   -0.0000    0.0001         0         0
0.0000
unco = 0
Ob = 16×4
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000
         0    1.0000         0         0
    0.0075         0         0  500.0000
         0         0         0    1.0000
         0   -0.0000         0         0
    0.0075         0         0  500.0000
         0   -0.0025         0         0
unob = 0
```

We provide controllability and observability with changing T.

We can find poles of the system find eigenvalues of matrix A.

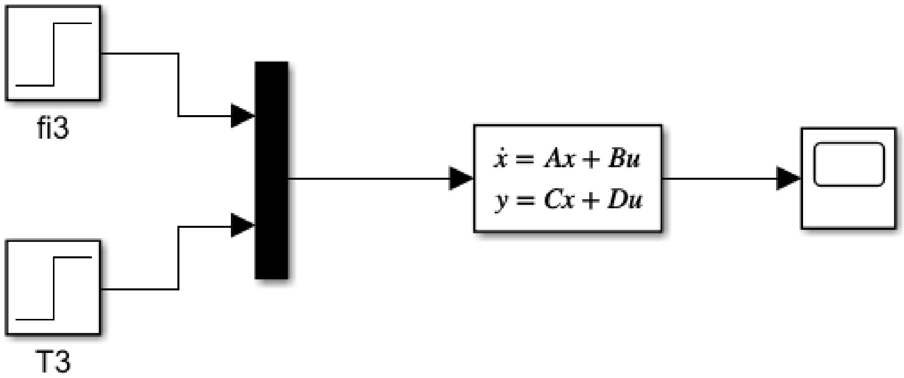
```
% P = pole(sys);
Apoles = eig(A)
```

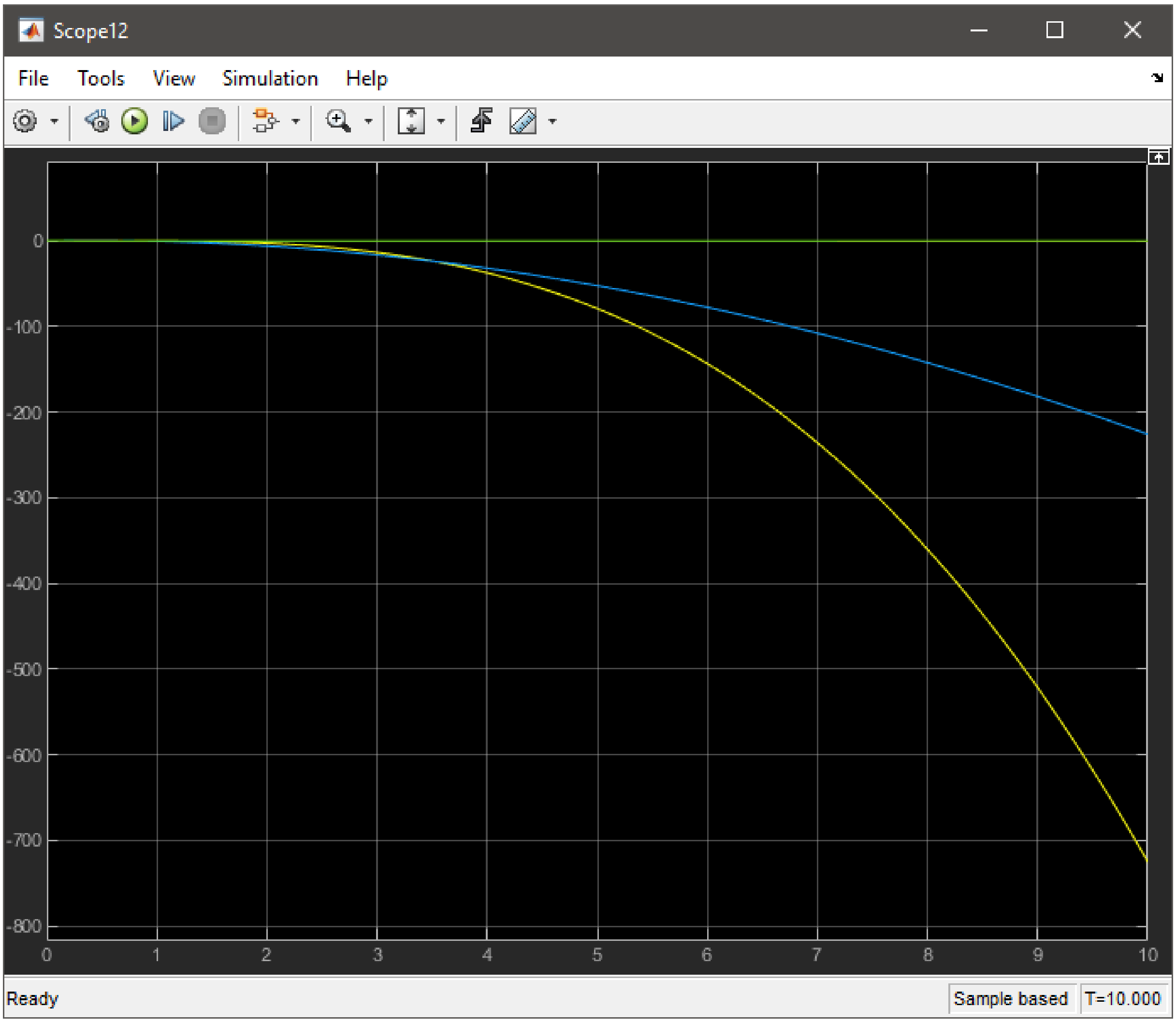
```
Apoles = 4×1 complex
   0.0000 + 0.0000i
   0.0000 + 0.0500i
   0.0000 - 0.0500i
   0.0000 + 0.0000i
```

Full-State Feedback

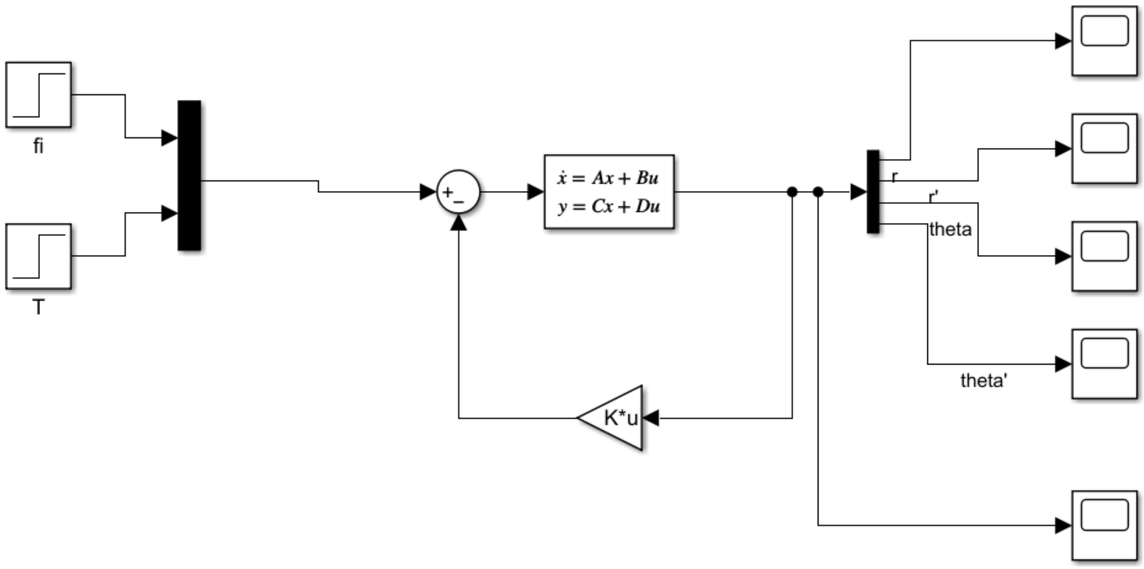
```
% State-Feedback Gain Selection
p=[-2+i*2;-2-i*2;-1;-1]
K = place(A,B,p)
P = pole(sys)
plant = (A-B*K)
poles_cl = eig(plant)
```

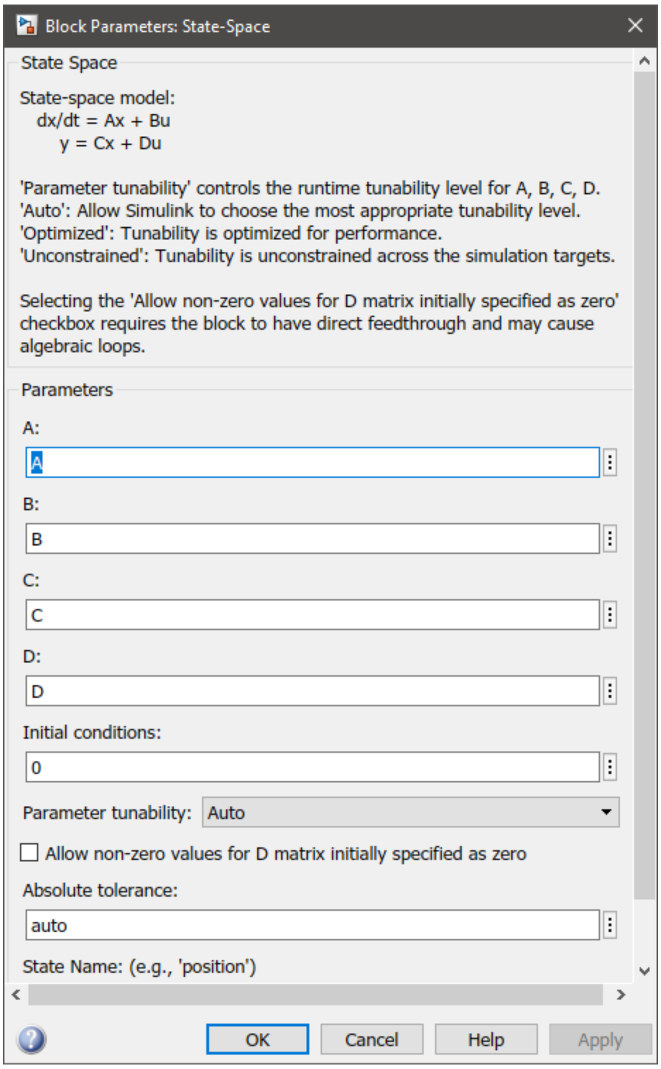
```
p = 4×1 complex
  -2.0000 + 2.0000i
  -2.0000 - 2.0000i
  -1.0000 + 0.0000i
  -1.0000 + 0.0000i
K = 2×4
 -800.0000 -799.9980 -400.0000 -500.0000
    0.0037    0.5000   -0.5000  249.5000
P = 4×1 complex
   0.0000 + 0.0000i
   0.0000 + 0.0500i
   0.0000 - 0.0500i
   0.0000 + 0.0000i
plant = 4×4
         0    1.0000         0         0
    0.0000   -1.0000    1.0000    1.0000
         0         0         0    1.0000
   -8.0000   -8.0000   -4.0000   -5.0000
poles_cl = 4×1 complex
  -2.0000 + 2.0000i
  -2.0000 - 2.0000i
  -1.0000 + 0.0000i
  -1.0000 + 0.0000i
```

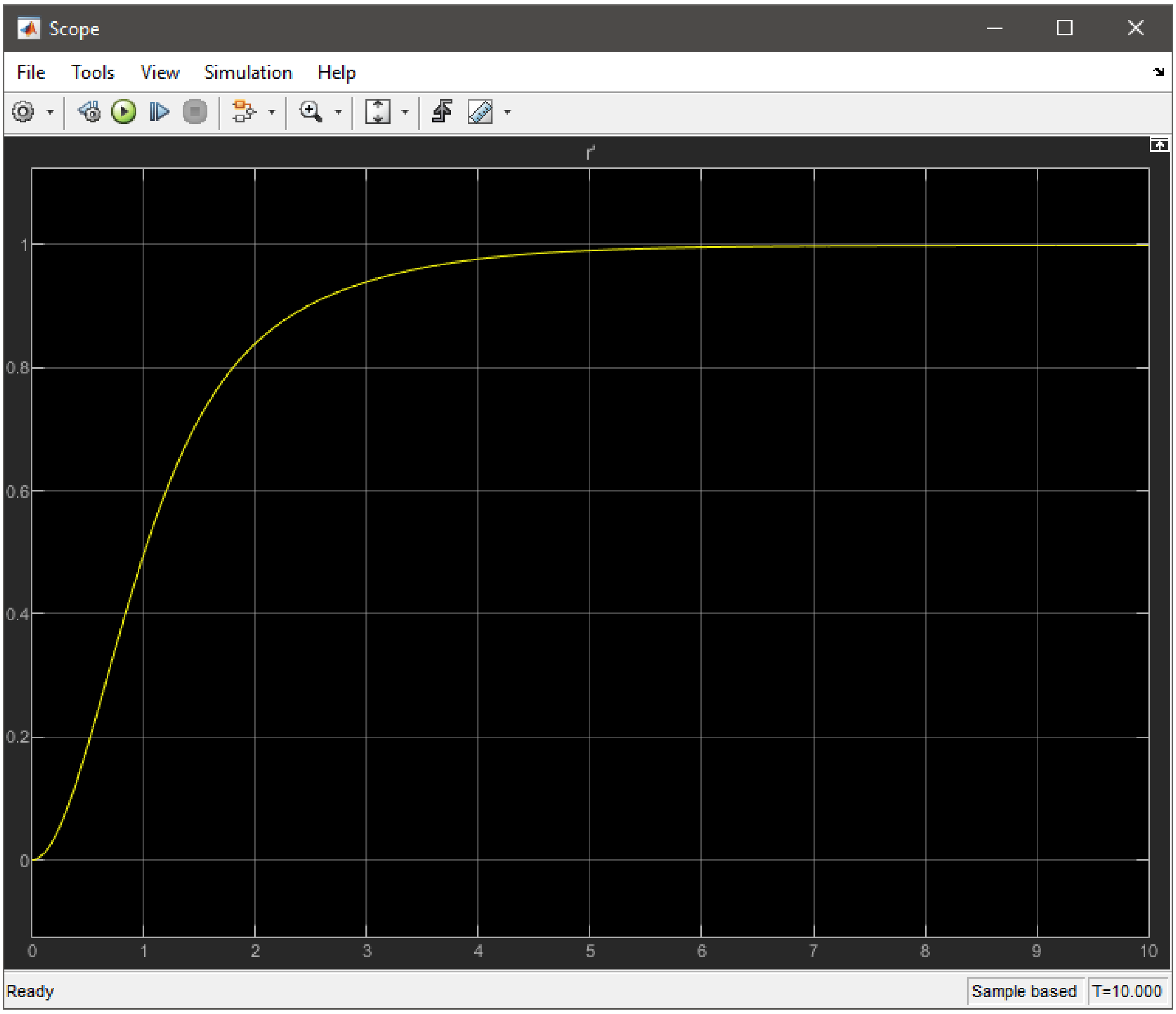

fi3
T3
$\dot{x} = Ax + Bu$
$y = Cx + Du$


Scope12
File Tools View Simulation Help
0
-100
-200
-300
-400
-500
-600
-700
-800
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
T=10.000

Pole Placement



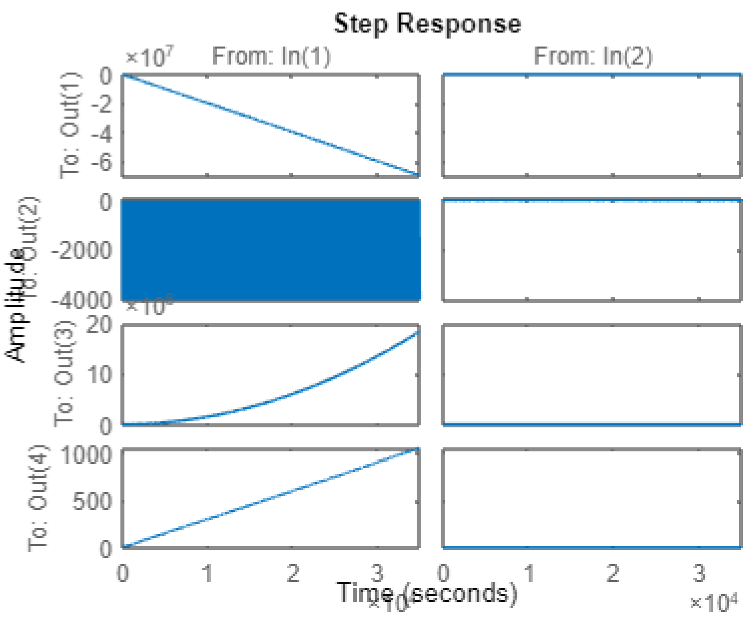


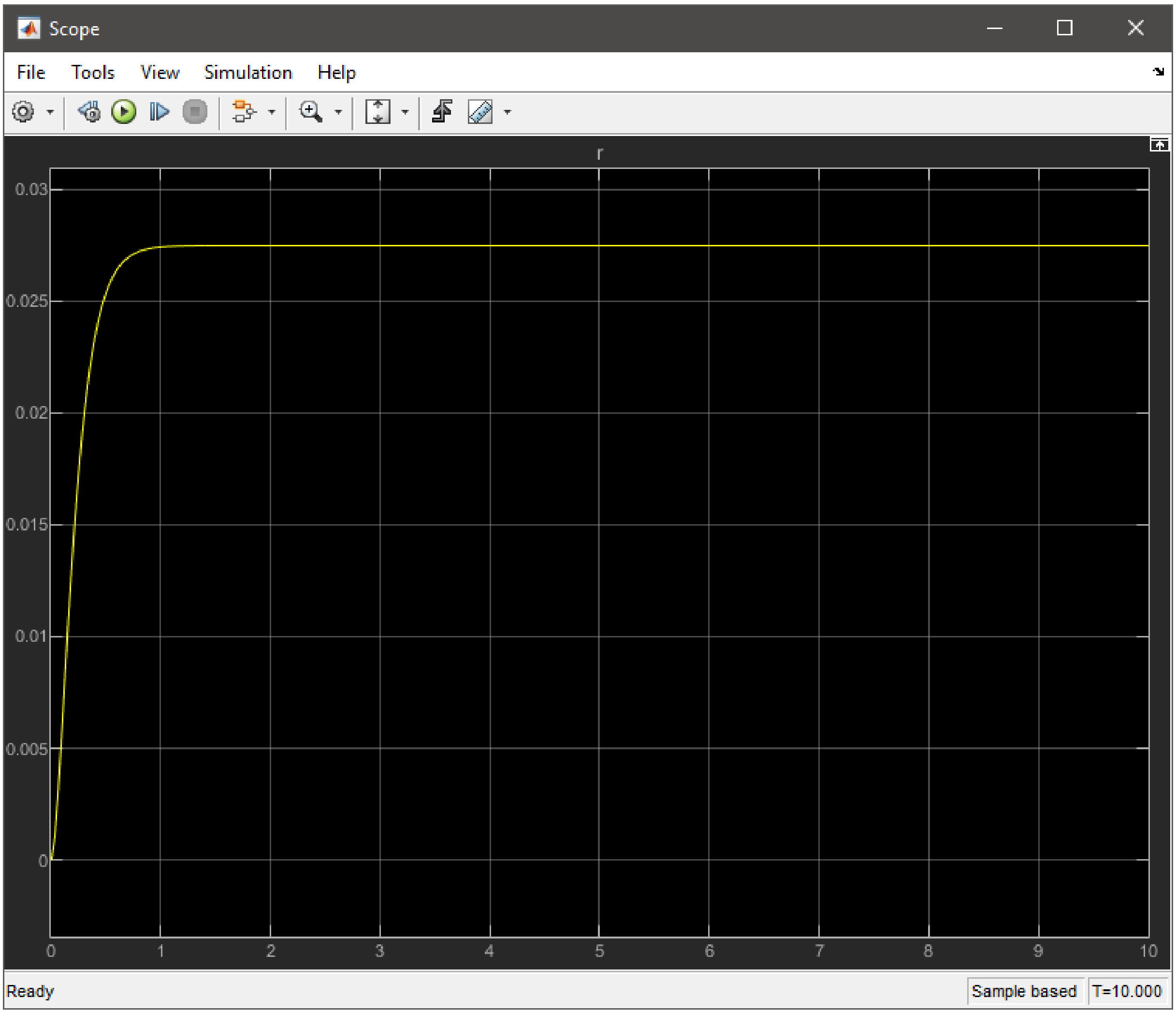


We use 3 times faster roots

```
des_poles = (min(real(poles_cl))*3)-(1:4);
des_poles
K = place(A,B,des_poles)
step(sys);
```

```
des_poles = 1×4
   -7.0000   -8.0000   -9.0000  -10.0000
K = 2×4
10³ ×
    0.7791    0.0918   -6.9333   -1.6740
    0.0368    0.0086   -0.0043    0.2495
```

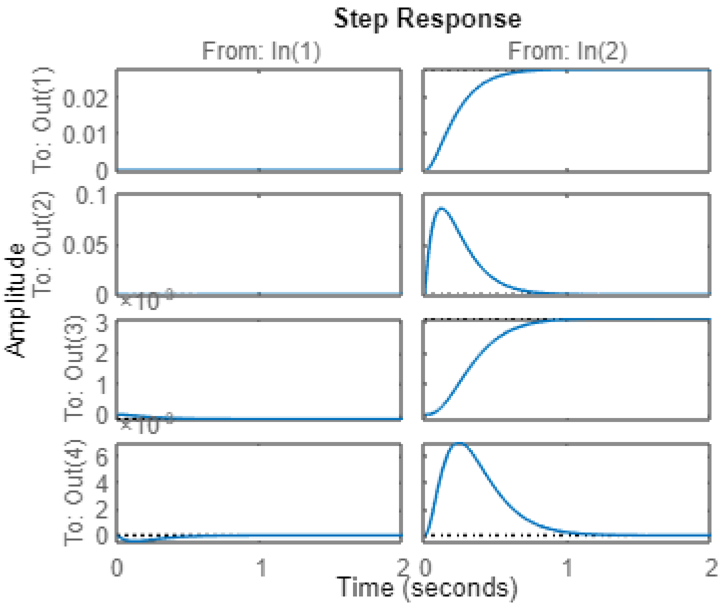




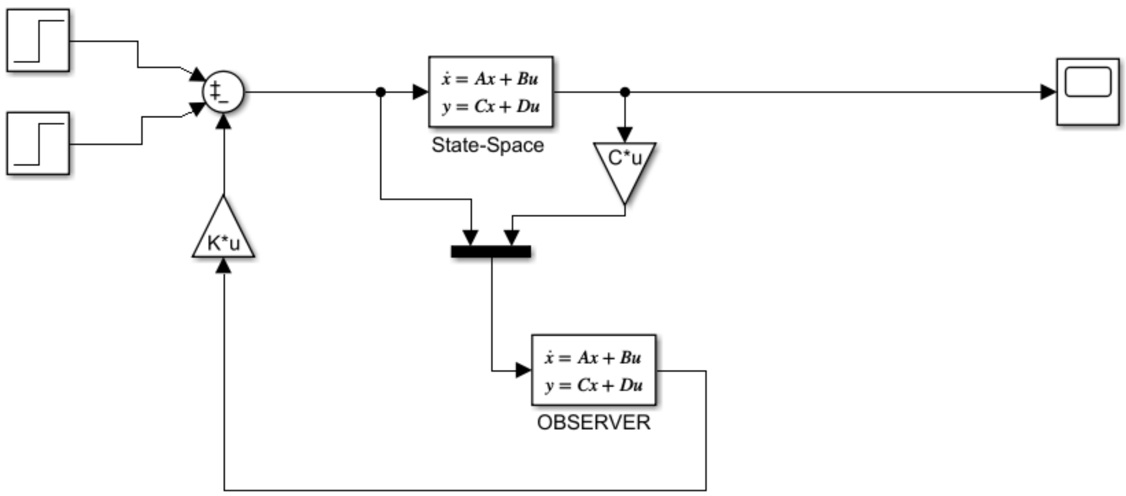
As we can see from plot, our settling time and overshoot percentage is decreased.

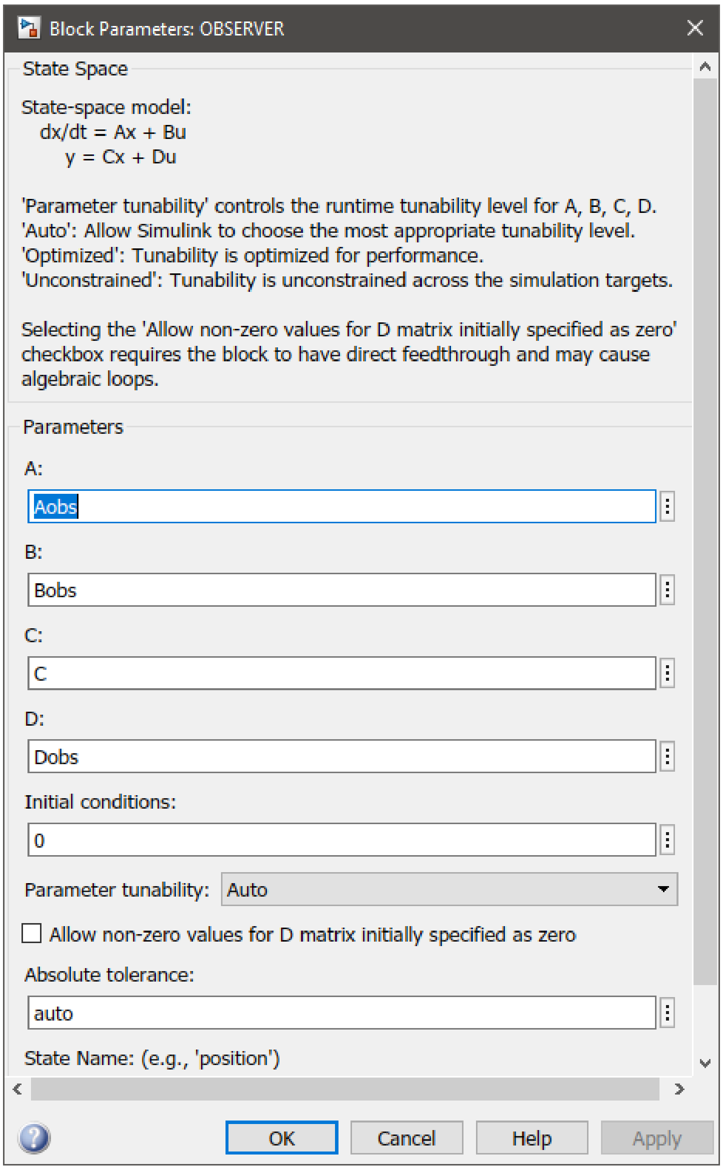
```
sys2=ss(A-B*K,B,C,D);


tf(sys2)
step(sys2)
Kdc=dcgain(sys2);
```

```
ans =

  From input 1 to output...
              -0.01013 s - 0.08653
   1:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

        -0.01013 s^2 - 0.08653 s - 2.918e-17
   2:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

           -0.01 s^2 - 0.1726 s - 0.7367
   3:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

         -0.01 s^3 - 0.1726 s^2 - 0.7367 s
   4:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

  From input 2 to output...
             2 s^2 + 33.48 s + 138.7
   1:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

            2 s^3 + 33.48 s^2 + 138.7 s
   2:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

                  1.837 s + 15.58
   3:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

          1.837 s^2 + 15.58 s + 3.684e-15
   4:  --------------------------------------
       s^4 + 34 s^3 + 431 s^2 + 2414 s + 5040

Continuous-time transfer function.
```


Step Response
From: In(1)
From: In(2)
Amplitude
To: Out(1)
To: Out(2)
To: Out(3)
To: Out(4)
Time (seconds)

Observer


State-Space
C*u
K*u
OBSERVER



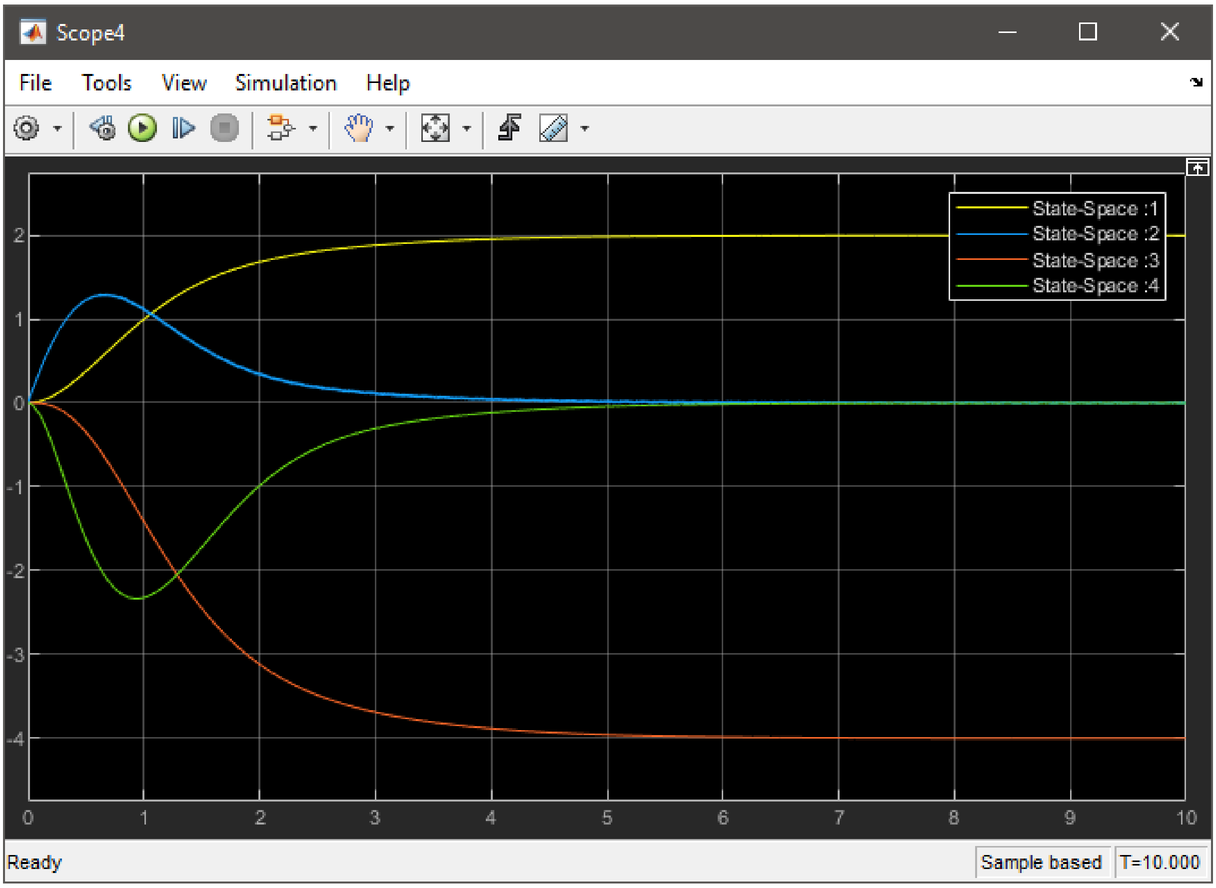

```
L=place(A.',C.',des_poles)
obsplant=(A-L*C)
poles_obs=eig(obsplant)
des_poles1 = (min(real(poles_obs))*3)-(1:4); %This is better
des_poles1
```

```
L = 4×4
    7.0000    0.0075         0         0
    1.0000    8.0000         0   -0.0000
         0         0    9.0000         0
```

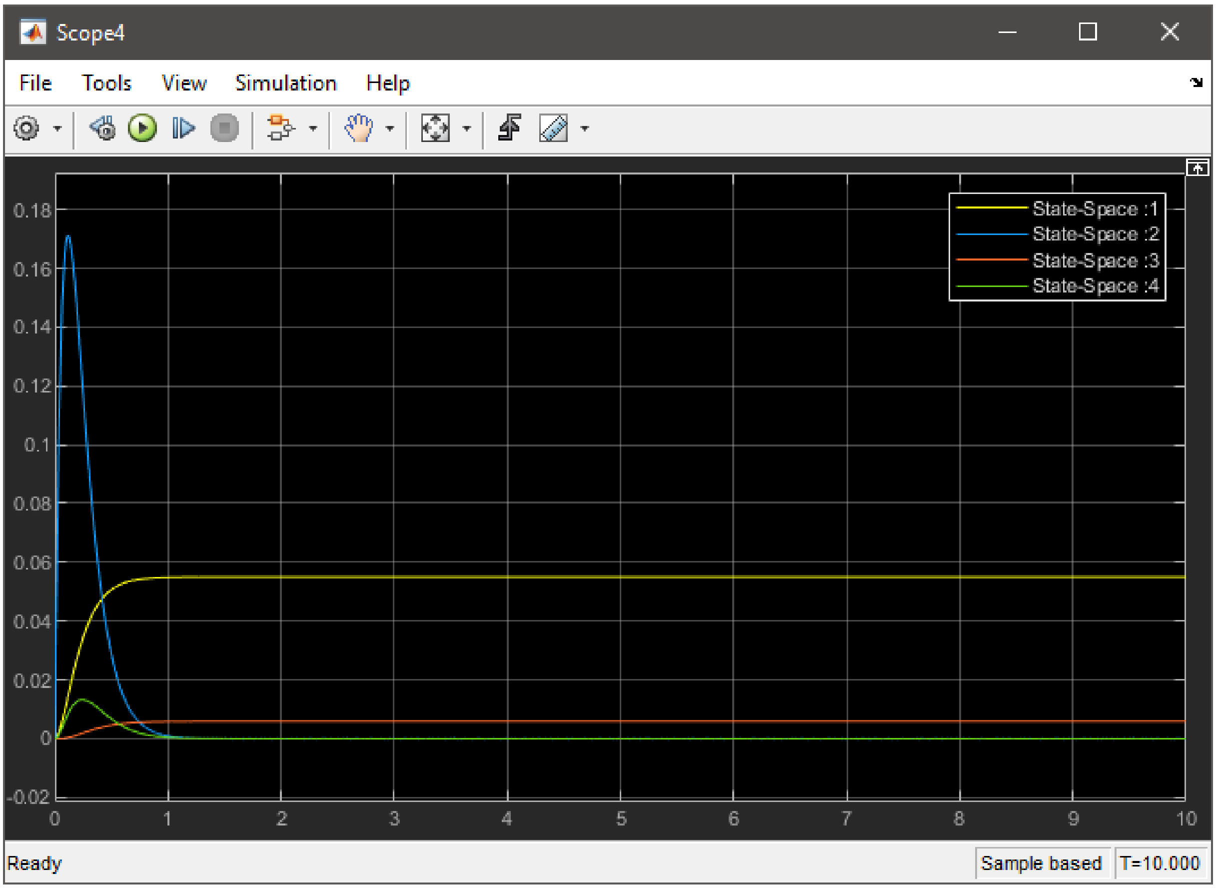
```
         0  500.0000    1.0000   10.0000
obsplant = 4×4
   -7.0000    0.9925         0         0
   -0.9925   -8.0000         0  500.0000
         0         0   -9.0000    1.0000
         0 -500.0000   -1.0000  -10.0000
poles_obs = 4×1 complex
10² ×
  -0.0900 + 5.0000i
  -0.0900 - 5.0000i
  -0.0700 + 0.0000i
  -0.0900 + 0.0000i
des_poles1 = 1×4
  -28.0000  -29.0000  -30.0000  -31.0000
```

```
Aobs = [A-L*C]
Bobs = [B L]
Cobs=[0 0 0 0;
    0 0 0 0;
    0 0 1 0;
    0 0 0 0];


Cobs1=[1.1 0 0 0;
    0 1 0 0;
    0 0 1 0;
    0 0 0 1];


Dobs = zeros(4,6);
```

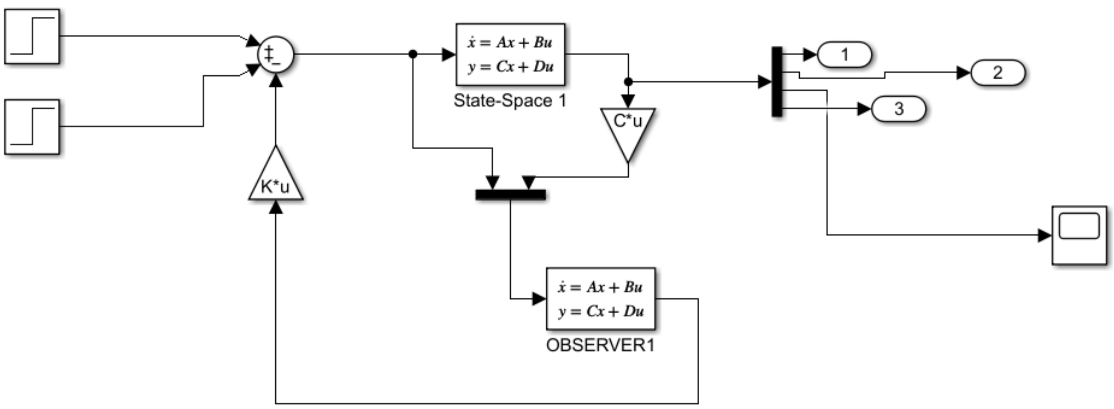
```
Aobs = 4×4
   -7.0000    0.9925         0         0
   -0.9925   -8.0000         0  500.0000
         0         0   -9.0000    1.0000
         0 -500.0000   -1.0000  -10.0000
Bobs = 4×6
         0         0    7.0000    0.0075         0         0
         0    2.0000    1.0000    8.0000         0   -0.0000
         0         0         0         0    9.0000         0
   -0.0100         0         0  500.0000    1.0000   10.0000
```

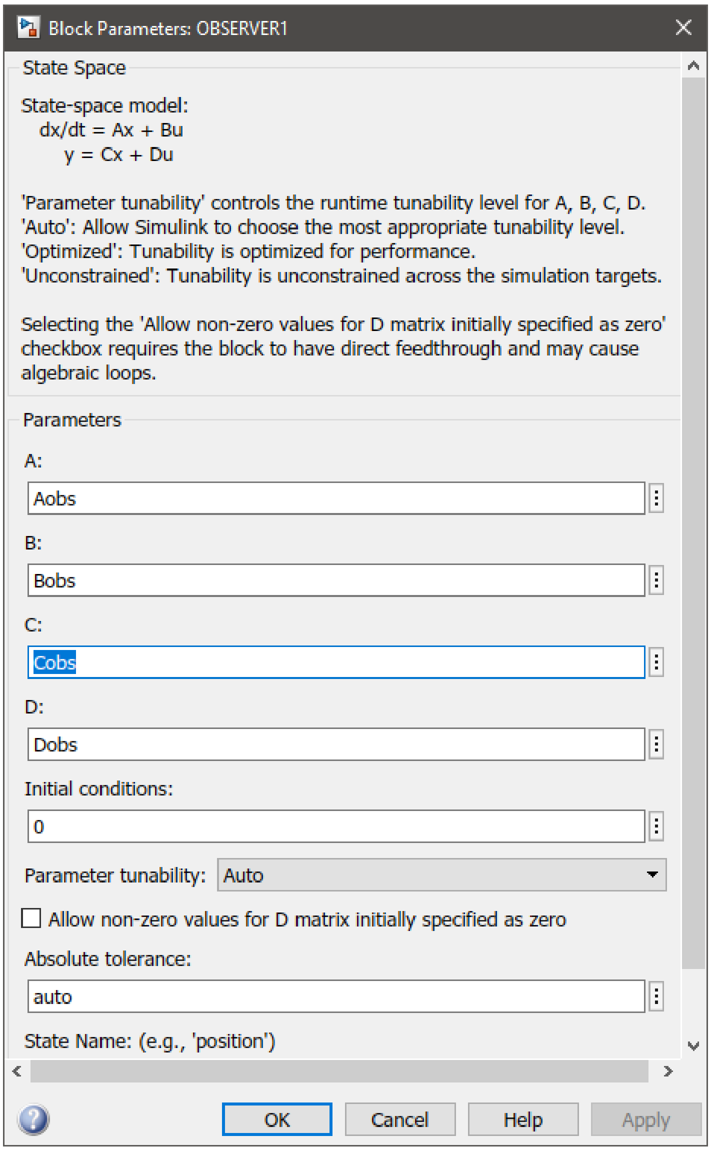


In first attempt we tried slower roots and then we increased its speed. Therefore we observe close performance that we measure before.



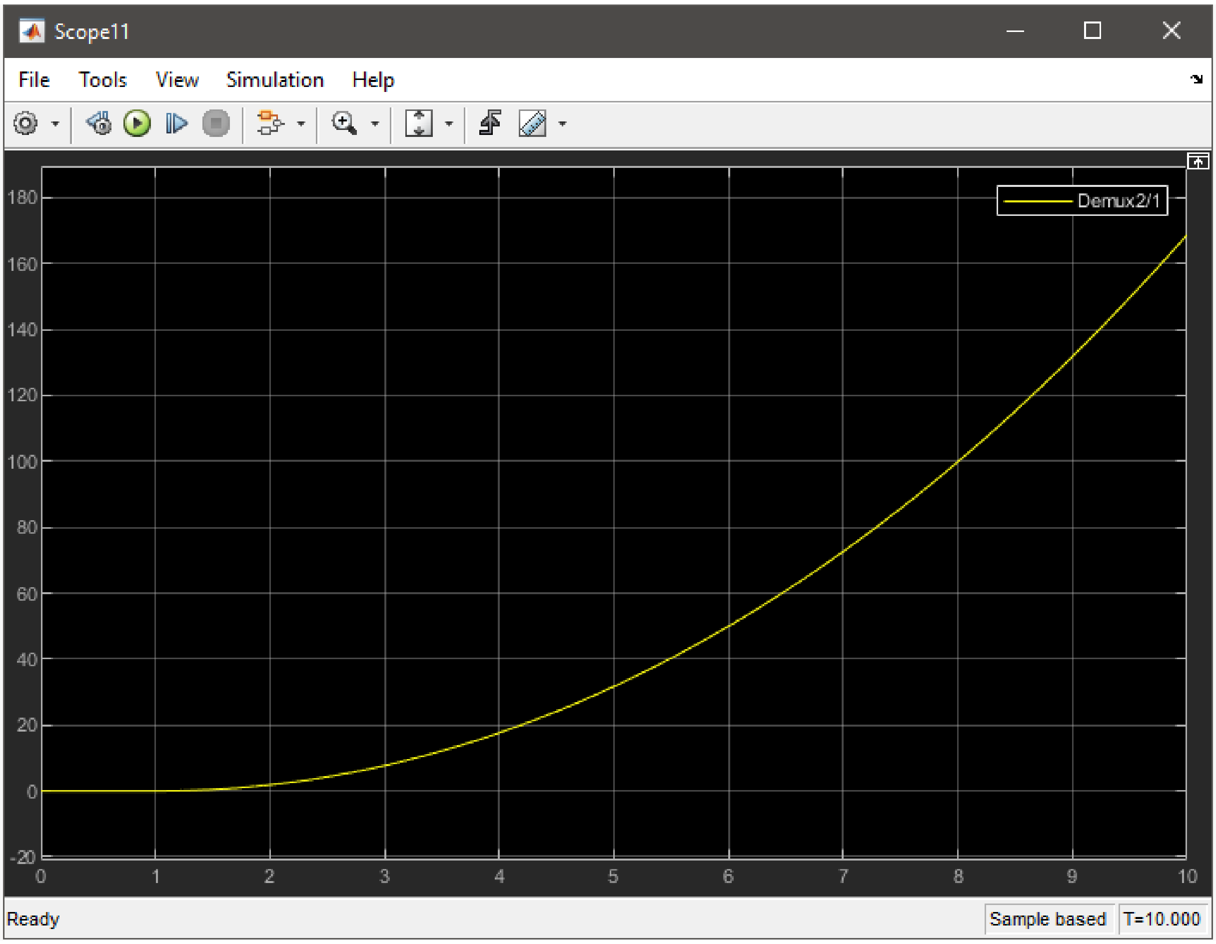
Using observer with faster roots Show us same result with greater magnitude than Full-State Feedback controller.





Only the member third columns of third row is equal to 1. So only this point permit transmitting the signal.

```
Cobs=[0 0 0 0;
    0 0 0 0;
    0 0 1 0;
    0 0 0 0];
```



This is plot of r measure when the only the angle theta is measured as output

LQR

```
%%%LQR 1%%%
Q =[1 0 0 0;
 0 1 0 0;
 0 0 1 0;
 0 0 0 1]
R =eye(2)
N=0;
[Klqr,S,E]=lqr(A,B,Q,R,N)


Nbar= -pinv(C*inv(A-B*Klqr)*B);
G=Nbar;


plant_lqr = (A-B*Klqr)
```

```
poles_lqr = eig(plant_lqr)
```

```
Q = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
R = 2×2
     1     0
     0     1
Klqr = 2×4
   -0.4230   -0.7254   -0.9061 -227.3233
    0.9099    1.1750   -0.4230  145.0888
S = 4×4
10⁴ ×
    0.0001    0.0000    0.0000    0.0042
    0.0000    0.0001   -0.0000    0.0073
    0.0000   -0.0000    0.0250    0.0091
    0.0042    0.0073    0.0091    2.2732
E = 4×1 complex
  -0.0040 + 0.0000i
  -0.9789 + 0.0000i
  -1.8202 + 1.3397i
  -1.8202 - 1.3397i
plant_lqr = 4×4
         0    1.0000         0         0
   -1.8123   -2.3501    0.8460  209.8224
         0         0         0    1.0000
   -0.0042   -0.0073   -0.0091   -2.2732
poles_lqr = 4×1 complex
  -1.8202 + 1.3397i
  -1.8202 - 1.3397i
  -0.9789 + 0.0000i
  -0.0040 + 0.0000i
```

In first LQR model we are used Identity matrix for Q.

But we need to observe r values of system. Significance of this of this value more important for us. Because we are trying to control its Radius to mass center.

Therefore we are going to increase fisrt member of Q matrix.

```
%%%LQR 2%%%
Q2 =[100 0 0 0;
 0 1 0 0;
 0 0 1 0;
 0 0 0 1]
R2 =eye(2)
N2=0;
[Klqr2,S2,E2]=lqr(A,B,Q2,R2,N2)


Nbar2= -pinv(C*inv(A-B*Klqr2)*B);
```

```
G=Nbar;


plant_lqr2 = (A-B*Klqr2)
poles_lqr2 = eig(plant_lqr2)
```

```
Q2 = 4×4
   100     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
R2 = 2×2
     1     0
     0     1
Klqr2 = 2×4
   -1.4855   -1.0648   -0.9889 -247.6417
    9.8928    3.1233   -0.1485  212.9547
S2 = 4×4
10⁴ ×
    0.0032    0.0005    0.0001    0.0149
    0.0005    0.0002   -0.0000    0.0106
    0.0001   -0.0000    0.0250    0.0099
    0.0149    0.0106    0.0099    2.4764
E2 = 4×1 complex
  -0.0040 + 0.0000i
  -2.3972 + 0.0000i
  -3.1609 + 3.2965i
  -3.1609 - 3.2965i
plant_lqr2 = 4×4
         0    1.0000         0         0
  -19.7781   -6.2465    0.2971   74.0905
         0         0         0    1.0000
   -0.0149   -0.0107   -0.0099   -2.4764
poles_lqr2 = 4×1 complex
  -3.1609 + 3.2965i
  -3.1609 - 3.2965i
  -2.3972 + 0.0000i
  -0.0040 + 0.0000i
```

We can see that some poles are getting faster.

```
%%%LQR 3%%%
Q3 =[1000 0 0 0;
 0 1 0 0;
 0 0 100 0;
 0 0 0 1]
R3 =eye(2)
N3=0;
[Klqr3,S3,E3]=lqr(A,B,Q3,R3,N3)


Nbar3= -pinv(C*inv(A-B*Klqr3)*B);
G=Nbar;
```

```
plant_lqr3 = (A-B*Klqr3)
poles_lqr3 = eig(plant_lqr3)
```

```
Q3 = 4×4
        1000           0           0           0
           0           1           0           0
           0           0         100           0
           0           0           0           1
R3 = 2×2
     1     0
     0     1
Klqr3 = 2×4
   -2.0614   -1.1686   -9.9787 -253.4403
   31.5593    5.5847   -0.6519  233.7282
S3 = 4×4
10⁴ ×
    0.0179    0.0016    0.0008    0.0206
    0.0016    0.0003   -0.0000    0.0117
    0.0008   -0.0000    0.2540    0.0998
    0.0206    0.0117    0.0998    2.5344
E3 = 4×1 complex
  -0.0400 + 0.0000i
  -2.4877 + 0.0000i
  -5.5880 + 5.6854i
  -5.5880 - 5.6854i
plant_lqr3 = 4×4
         0    1.0000         0         0
  -63.1110  -11.1694    1.3037   32.5436
         0         0         0    1.0000
   -0.0206   -0.0117   -0.0998   -2.5344
poles_lqr3 = 4×1 complex
  -5.5880 + 5.6854i
  -5.5880 - 5.6854i
  -2.4877 + 0.0000i
  -0.0400 + 0.0000i
```

We can understand that the poles placed upside are decreasing faster.

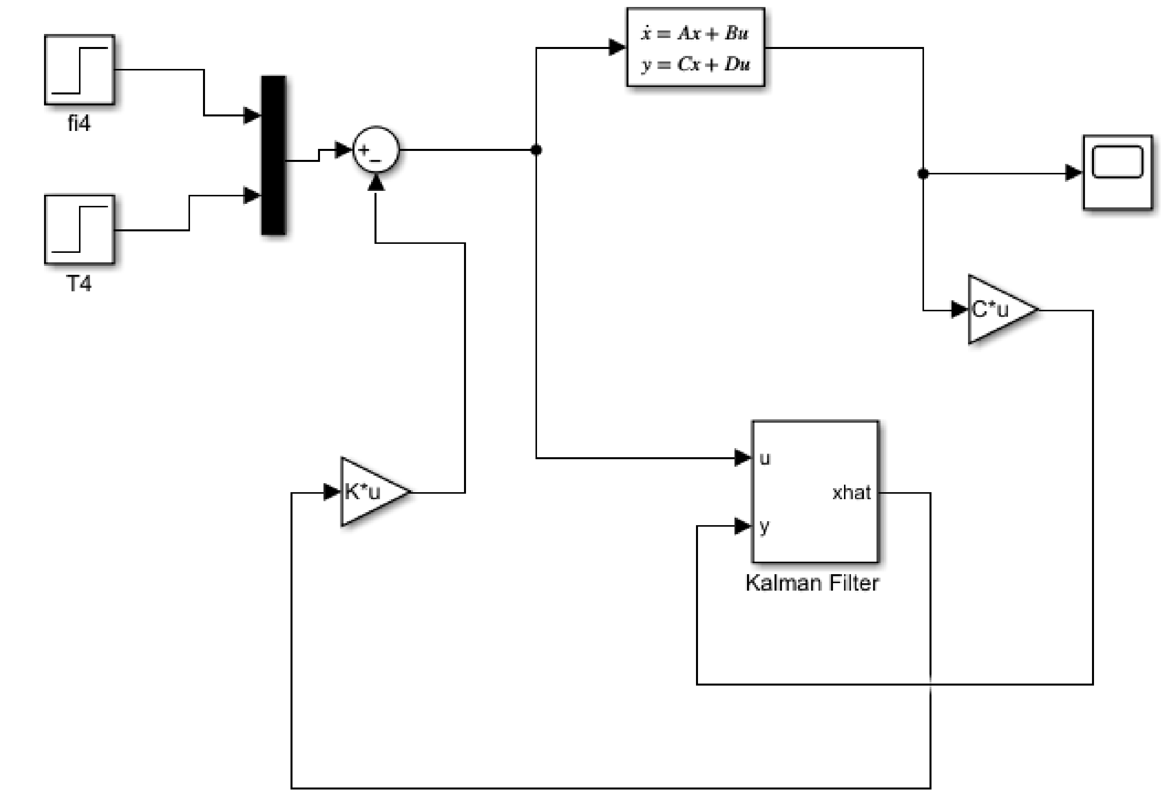
```
%%%LQR 4%%%
Q4 =[1000 0 0 0;
 0 1 0 0;
 0 0 100 0;
 0 0 0 1]
R4 =[100 0;
    0 10]
N4=0;
[Klqr4,S4,E4]=lqr(A,B,Q4,R4,N4)


Nbar4= -pinv(C*inv(A-B*Klqr4)*B);
G=Nbar;
```

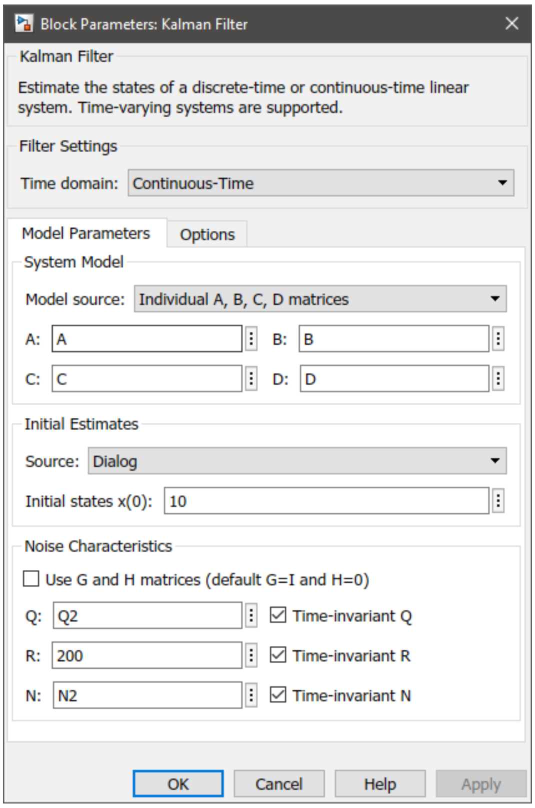
```
plant_lqr4 = (A-B*Klqr4)
poles_lqr4 = eig(plant_lqr4)
```

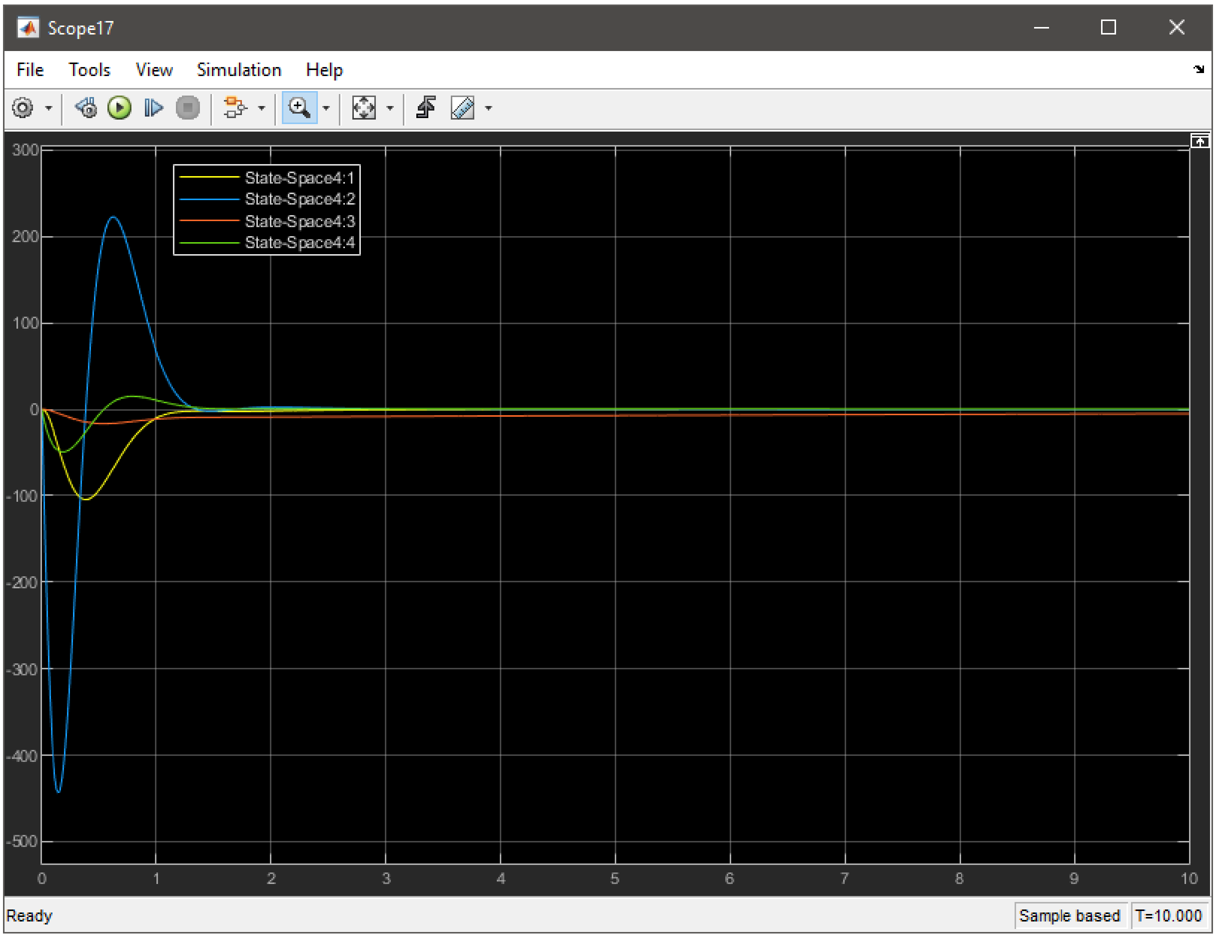
```
Q4 = 4×4
        1000           0           0           0
           0           1           0           0
           0           0         100           0
           0           0           0           1
R4 = 2×2
   100     0
     0    10
Klqr4 = 2×4
   -0.0799   -0.1219   -0.9997  -80.2874
   10.0006    3.1539   -0.0799  243.7770
S4 = 4×4
10⁵ ×
    0.0032    0.0005    0.0001    0.0080
    0.0005    0.0002   -0.0000    0.0122
    0.0001   -0.0000    0.0803    0.1000
    0.0080    0.0122    0.1000    8.0287
E4 = 4×1 complex
  -0.0127 + 0.0000i
  -0.7901 + 0.0000i
  -3.1540 + 3.1721i
  -3.1540 - 3.1721i
plant_lqr4 = 4×4
         0    1.0000         0         0
  -19.9936   -6.3078    0.1598   12.4459
         0         0         0    1.0000
   -0.0008   -0.0012   -0.0100   -0.8029
poles_lqr4 = 4×1 complex
  -3.1540 + 3.1721i
  -3.1540 - 3.1721i
  -0.7901 + 0.0000i
  -0.0127 + 0.0000i
```

When we increase value of R matrix, we observe that speed of poles decreased to previous state

KALMAN Filter

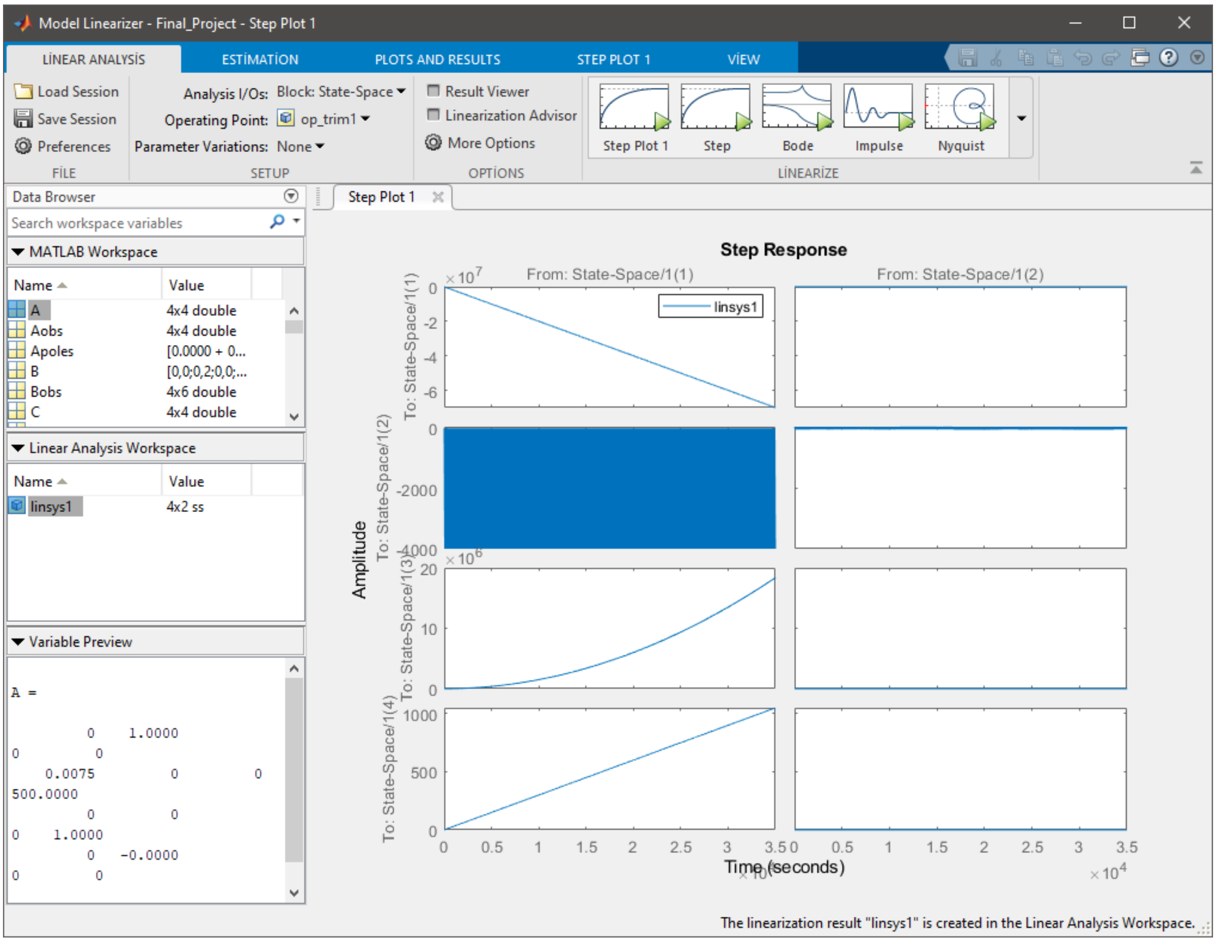


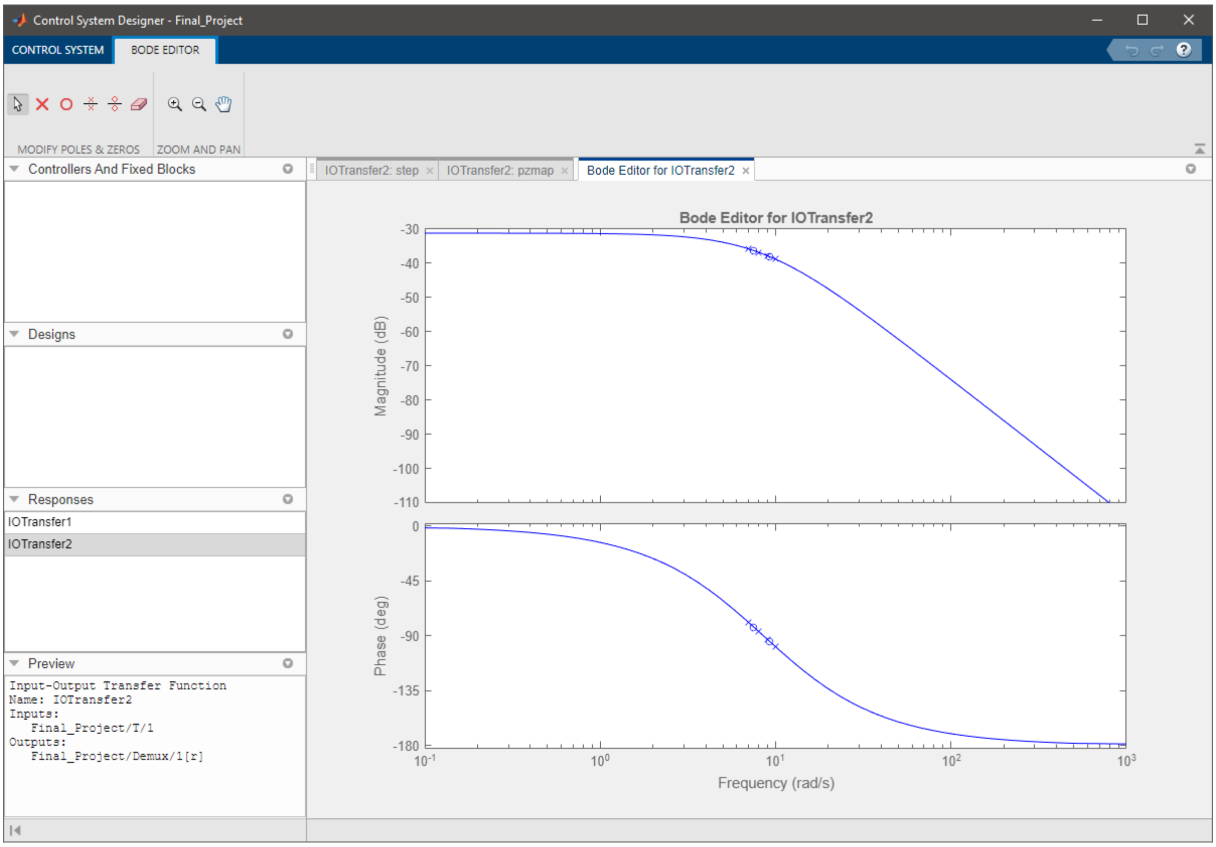




In Kalman Filter we used Q2 and N2 that we used before for LQR.

Model Linearizer




Control System Designer - Final_Project
CONTROL SYSTEM
BODE EDITOR
MODIFY POLES & ZEROS
ZOOM AND PAN
Controllers And Fixed Blocks
Designs
Responses
IOTransfer1
IOTransfer2
Preview
Input-Output Transfer Function
Name: IOTransfer2
Inputs:
Final_Project/T/1
Outputs:
Final_Project/Demux/1[r]
IOTransfer2: step
IOTransfer2: pzmap
Bode Editor for IOTransfer2
Bode Editor for IOTransfer2
Magnitude (dB)
Phase (deg)
Frequency (rad/s)