State-Space Analysis

July2022
Linear Systems

Mahsa Seraji
Dr. Shasadeghi

The desired transfer function:

$$g(s) = \frac{-2.5(s^2 + 8s + 15)}{(s - 0.4)(s - 0.5)(s^2 - 10s + 29)}$$

$$= \frac{-2.5s^2 - 20s - 37.5}{s^4 - 10.9s^3 + 38.2s^2 - 28.1s + 5.8}$$

$$g(s) = \frac{b(s)}{a(s)} = \frac{b_{n-1}s^{n-1} + b_{n-2}s^{n-2} + \cdots + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_0}$$

$$\Rightarrow \begin{cases} a_3 = -10.9 \\ a_2 = 38.2 \\ a_1 = -28.1 \\ a_0 = 5.8 \end{cases}$$

$$\Rightarrow \begin{cases} b_3 = 0 \\ b_2 = -2.5 \\ b_1 = -20 \\ b_0 = -37.5 \end{cases}$$

Obtaining the state-space using MATLAB:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

Codes:

```
>> a=conv([-2.5],[1 8 15]);
b=conv([1 -0.9 0.2],[1 -10 29]);
g1=tf(a,b);
[A,B,C,D] =tf2ss(a,b);
g1 =

        -2.5 s^2 - 20 s - 37.5
  ----------------------------------------
  s^4 - 10.9 s^3 + 38.2 s^2 - 28.1 s + 5.8
A =

   10.9000  -38.2000   28.1000   -5.8000
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
B =
```

```
     1
     0
     0
     0

C =

         0   -2.5000  -20.0000  -37.5000

D =

     0
```

Section A: State Feedback Linear Control Systems: Regulator Design

1. Minimal State-Space Realization and Analysis of Controllability, Stability, and Open-Loop Response (if stable)

Controllability:
The minimal realization of the state-space transformation is achieved when it is both controllable and observable.

Approach 1: Utilize the ctrb command to obtain the controllability matrix. If the matrix is full rank, controllability is achieved.

Codes:

```
Co=ctrb(A,B)
rankCo=rank(Co)
```

```
Co =

    1.0000   10.9000   80.6100  490.3690
         0    1.0000   10.9000   80.6100
         0         0    1.0000   10.9000
         0         0         0    1.0000


rankCo =

     4
```

➔ It is observed that since the rank of the matrix is complete, this realization is controllable.

Approach 2: Another method is to obtain the canonical controller realization of the transformation function, which is always controllable. Then, the controllability matrix can be examined, and if its rank is complete, it indicates controllability.

Canonical Controller Realization:

The matrix form of it is as follows ->

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_{n-1} \end{bmatrix} x(t)$$

$$\Phi_{cc} = [b_c \quad A_c b_c \quad \ldots . A_c^{n-1} b_c]$$

As a result, we have:

```
>> A=[0 1 0 0;0 0 1 0;0 0 0 1;-5.8 28.1 -38.2 10.9];
B=[0;0;0;1];
C=[-37.5 -20 -2.5 0];
D=[0;0;0;0];
>> A

A =

       0    1.0000         0         0
       0         0    1.0000         0
       0         0         0    1.0000
   -5.8000   28.1000  -38.2000   10.9000

>> B

B =

    0
    0
    0
    1

>> C

C =

  -37.5000  -20.0000   -2.5000         0

>> D

D =

    0
    0
    0
    0
phicc=[B,A*B,(A^2)*B,(A^3)*B];
```

```
rankphicc=rank(phicc)

rankphicc =

    4
```

➔ It is evident that since the rank of the matrix is complete, this realization is controllable.

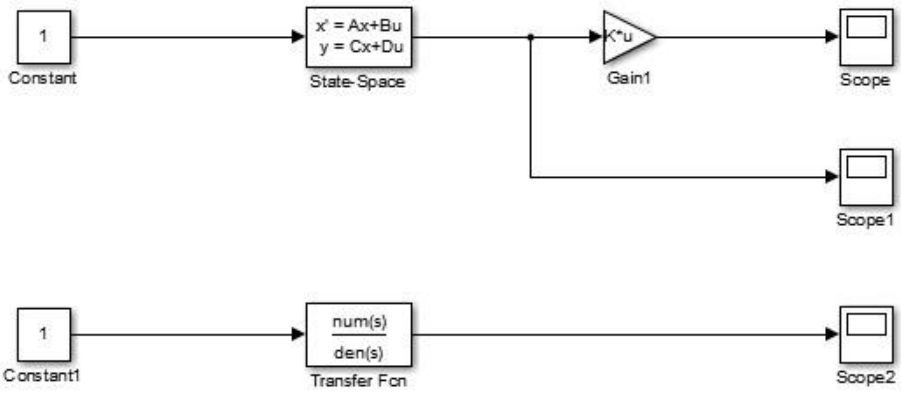
Stability:
The system is stable if all the eigenvalues of matrix A are negative, i.e.:

```
>> eig_A=eig(A)
eig_A =

  5.0000 + 2.0000i
  5.0000 - 2.0000i
  0.5000 + 0.0000i
  0.4000 + 0.0000i
```

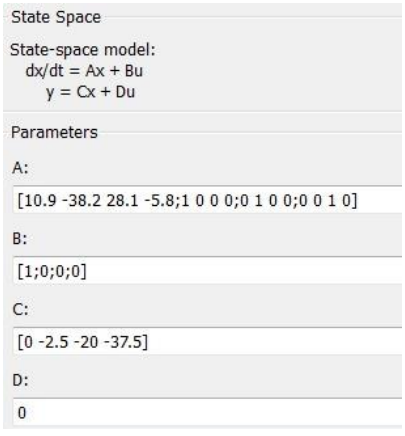
➔ Because the eigenvalues are positive, the system is unstable.

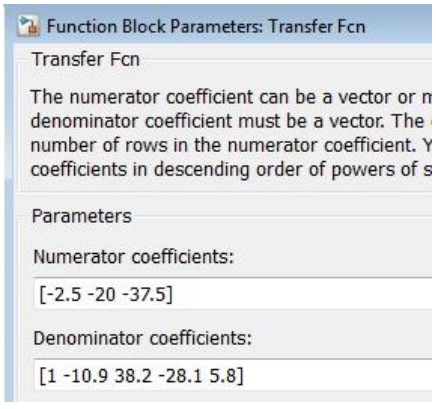
In the case of stability, the open-loop response is as follows:
The system is unstable.

In the simulation section, we have:

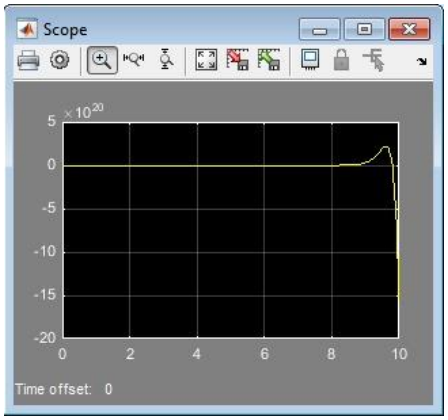


Settings:



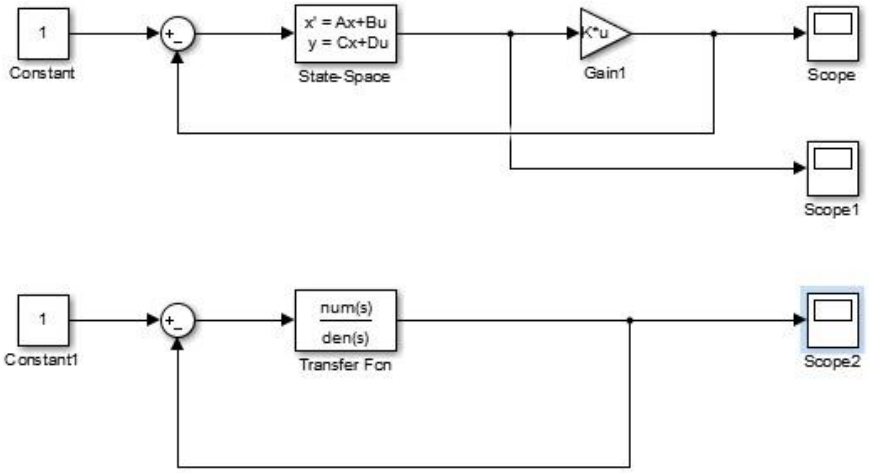


Input:

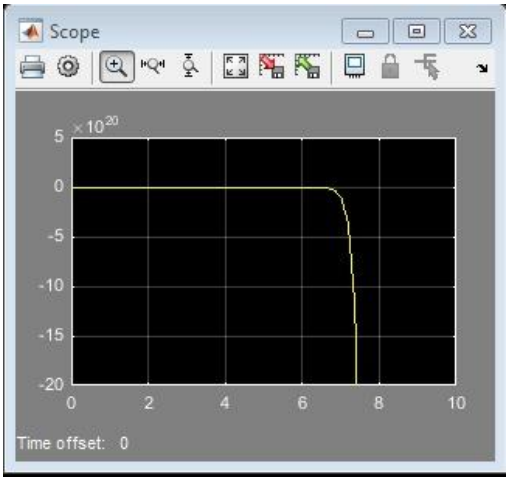


➔ We observe that the system is unstable.

2. The closed-loop behaviour of the system (unity feedback)



Input:



➔ We observe that the system is unstable again.

3&4. State Feedback Gain for Two Categories of Fast and Slow Poles Using 3 Methods and Analyzing the Closed-Loop Response, etc.

Slow Poles:
Bass-Gura Method:

$$a = [a_{n-1} \;\; a_{n-2} \dots \; a_0] = [-10.9 \;\; 38.2 \;\; -28.1 \;\; 5.8]$$

$$\psi = \begin{bmatrix} 1 & a_3 & a_2 & a_1 \\ 0 & 1 & a_3 & a_2 \\ 0 & 0 & 1 & a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -10.9 & 38.2 & -28.1 \\ 0 & 1 & -10.9 & 38.2 \\ 0 & 0 & 1 & -10.9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If we consider the closed-loop poles to be from -1 to -4:

$$\alpha(s) = (s+1)(s+2)(s+3)(s+4) = s^4 + 10s^3 + 35s^2 + 50s + 24$$

$$\Longrightarrow \alpha = [10 \;\; 35 \;\; 50 \;\; 24]$$

Then, with the following commands, we can obtain (k) using the Bass-Gura method:

```
>> a=conv([-2.5],[1 8 15]);
b=conv([1 -0.9 0.2],[1 -10 29]);
g1=tf(a,b);
[A,B,C,D] =tf2ss(a,b);
a=[-10.9 38.2 -28.1 5.8];
sai=[1 -10.9 38.2 -28.1;0 1 -10.9 38.2;0 0 1 -10.9;0 0 0 1];
phic=[B,A*B,(A^2)*B,(A^3)*B];
alpha=[10 35 50 24];
k1=(alpha-a)*(inv(sai))*(inv(phic))

k1 =

  20.9000   -3.2000   78.1000   18.2000
```

Ackermann's Method:

We can obtain (k) using the acker command:

```
k2=acker(A,B,[-1 -2 -3 -4])
k2 =


18.2000   78.1000   -3.2000   20.9000
```

Canonical Controller Realization Method:

We can obtain (k) using the canonical controller realization method:

$$k = \alpha - a \Longrightarrow$$

```
k_3=alpha-a;
k3=zeros(1,4);
for i=1:4
   k3(1,i)=k_3(1,5-i);
end
k3;
eig1=eig(A-B*k1);
>> eig1

eig1 =

  -4.0000
  -3.0000
  -2.0000
  -1.0000
k3 =

20.9000   3.2000-   78.1000   18.2000
k_3 =

18.2000   78.1000   3.2000-   20.9000
```

For fast poles:

If we consider the closed-loop poles to be from -20 to -23:

$$\alpha(s) = (s+20)(s+21)(s+22)(s+23)$$
$$= s^4 + 86s^3 + 2771s^2 + 39646s + 212520$$

We have:

Bass-Gura method:

```
a=conv([-2.5],[1 8 15]);
b=conv([1 -0.9 0.2],[1 -10 29]);
g1=tf(a,b);
[A,B,C,D] =tf2ss(a,b);
a=[-10.9 38.2 -28.1 5.8];
sai=[1 -10.9 38.2 -28.1;0 1 -10.9 38.2;0 0 1 -10.9;0 0 0 1];
phic=[B,A*B,(A^2)*B,(A^3)*B];
alpha=[86 2771 39646 212520];
k1=(alpha-a)*(inv(sai))*(inv(phic))
k1 =

  1.0e+05 *

   0.0010    0.0273    0.3967    2.1251
```

Ackermann's Method:

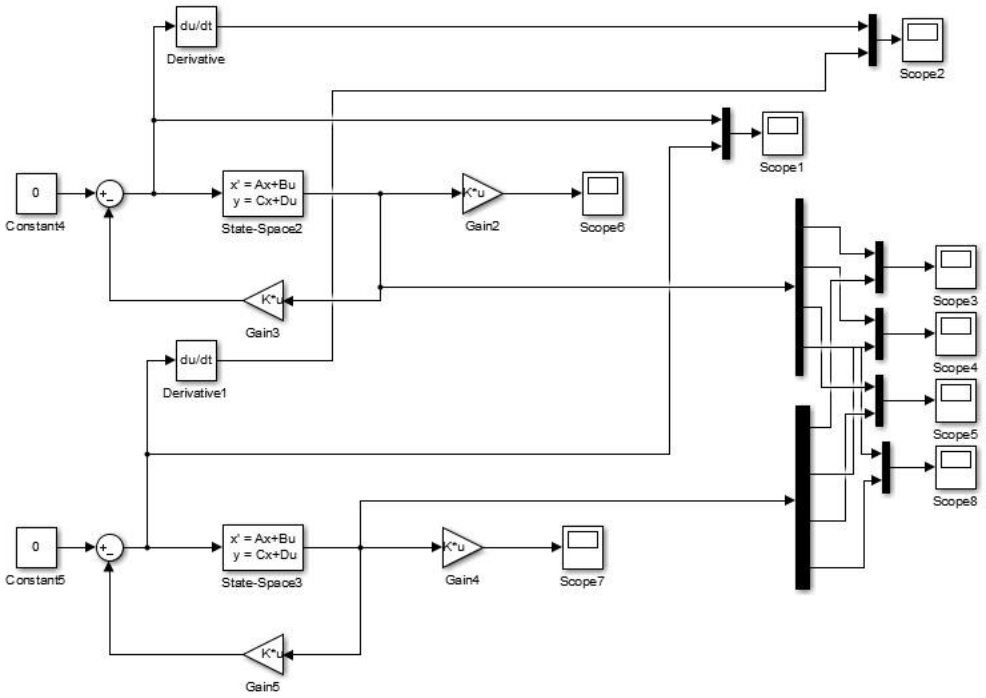
```
K2=acker(A,B,[-20 -21 -22 -23])
k2 =

  1.0e+05 *

2.1251    0.3967    0.0273    0.0010
```

Canonical Controller Realization Method:

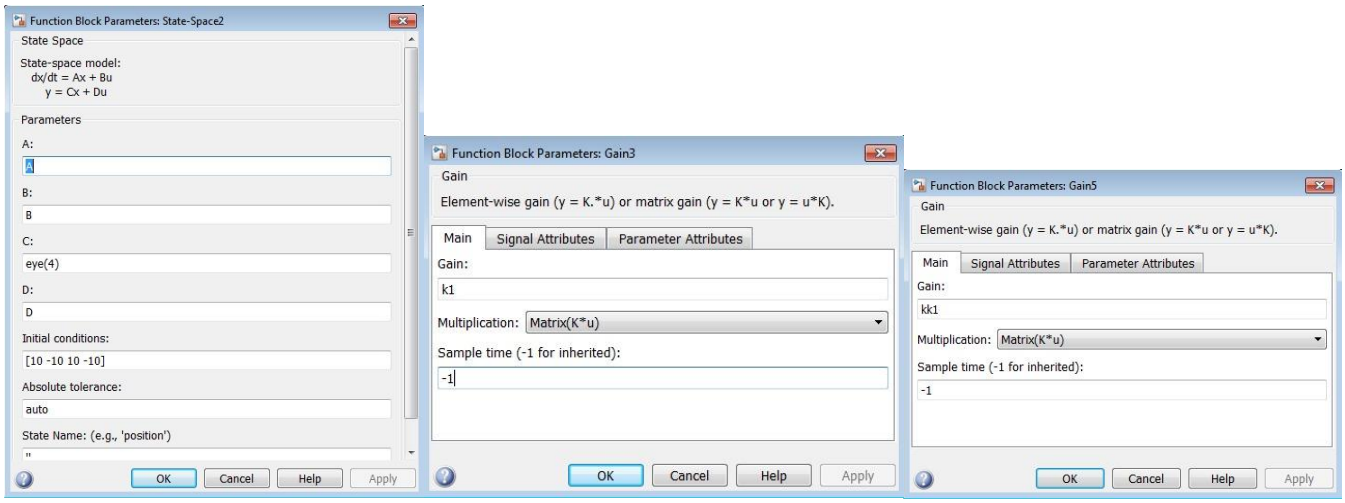
```
k3=zeros(1,4);
for i=1:4
   k3(1,i)=k_3(1,5-i);
end
k3;
eig1=eig(A-B*k1);
>> k_3

k_3 =

  1.0e+05 *

   0.0010    0.0273    0.3967    2.1251

>> eig1

eig1 =

 -23.0000
 -22.0000
 -21.0000
- 20.0000
```

➔ It is observed that they have become stable.

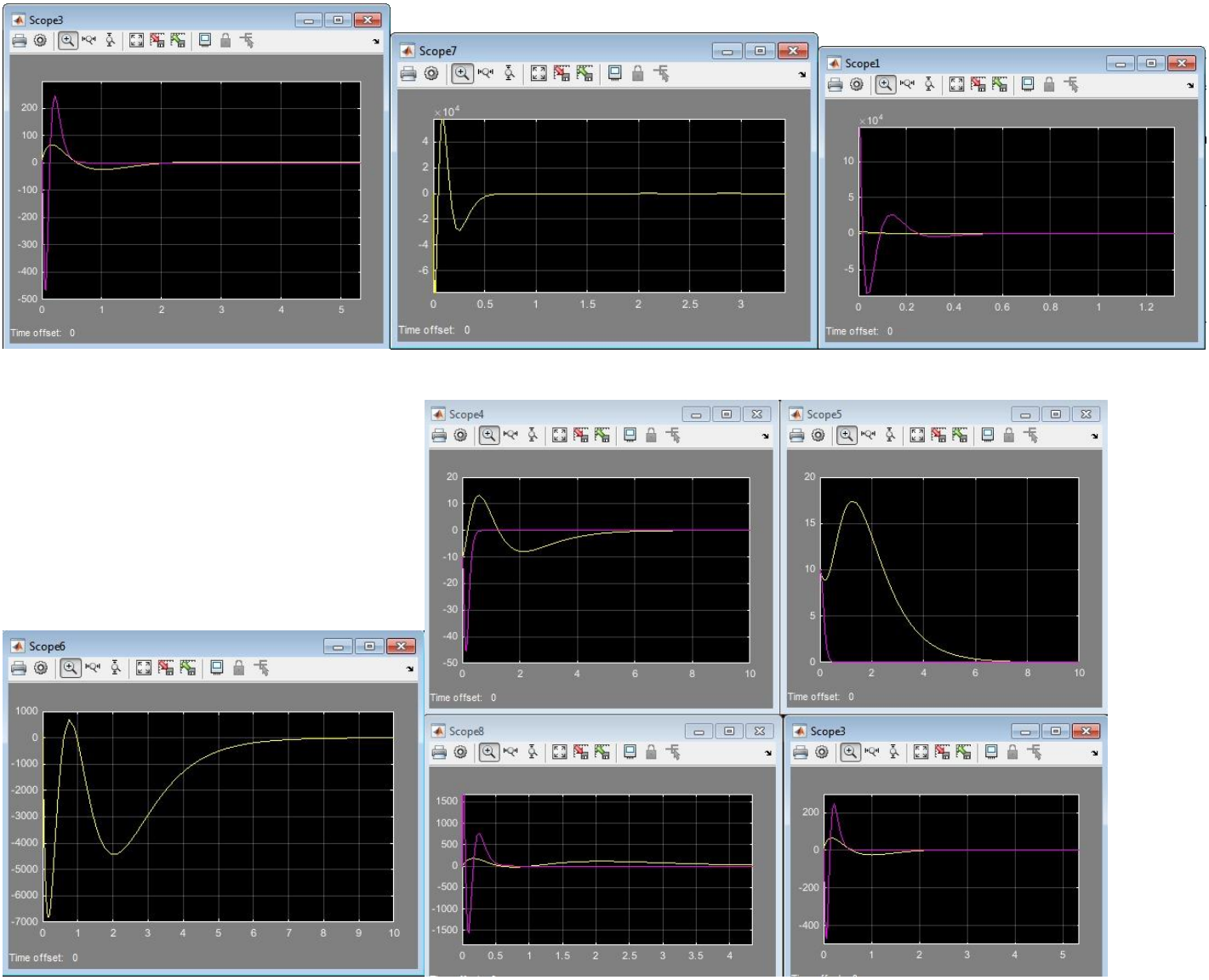
In the simulation section, we have:



Settings:



Outputs:



Scope 6: Output for Slow Pole

Scope 7: Output for Fast Pole

Scopes 3, 4, 5, 8: Comparison of States 1 to 4

Scope 1: Comparison of Command Signal

Scope 2: Comparison of Command Signal Derivative

➔ Result: It is observed that stabilization has been achieved, and the convergence of fast pole graphs is faster than that of slow poles.

5. Transforming the system into a system with two inputs and calculating the state feedback matrix, etc.

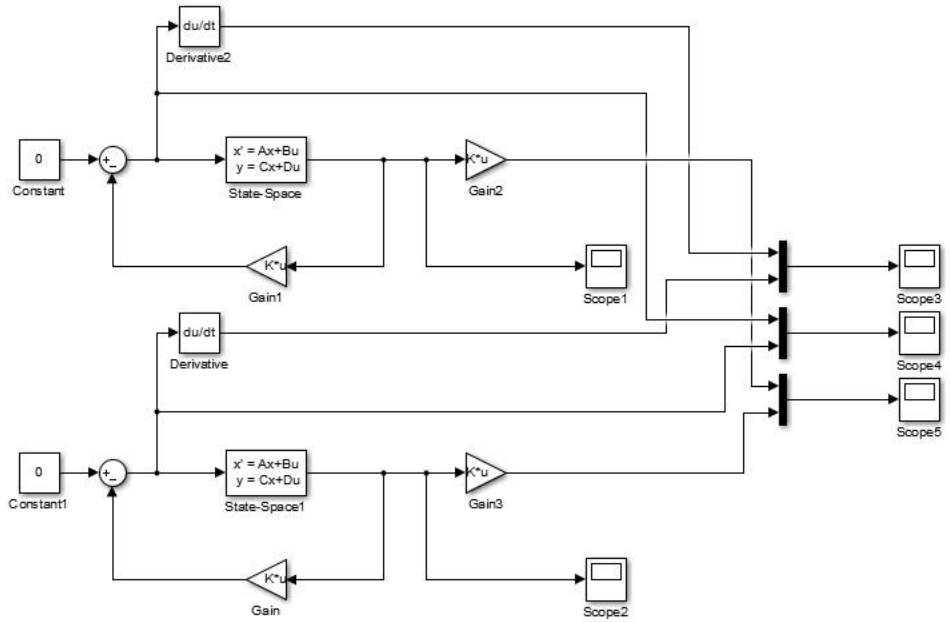
```
>> A=[0 1 0 0;0 0 1 0;0 0 0 1;-5.8 28.1 -38.2 10.9];
BB=[0 1;0 0;0 0;1 0];
DD=[0 0;0 0;0 0;0 0];
PHIC=[BB,A*BB,(A^2)*BB,(A^3)*BB];
F=[-1 0 0 0;0 -2 0 0;0 0 -3 0;0 0 0 -4];
Kbar1=[1 1 0 0;0 0 1 1];
Kbar2=[1 1 1 0;0 0 1 1];
phio1=[Kbar1;Kbar1*F;Kbar1*(F^2);Kbar1*(F^3)];
phio2=[Kbar2;Kbar2*F;Kbar2*(F^2);Kbar2*(F^3)];
rankphio1=rank(phio1);
rankphio2=rank(phio2);
T1=lyap(A,-F,-BB*Kbar1);
T2=lyap(A,-F,-BB*Kbar2);
K1=Kbar1*inv(T1);
K2=Kbar2*inv(T2);
eig1=eig(A-BB*K1);
eig2=eig(A-BB*K2);
>> rankphio1

rankphio1 =

    4
rankphio2 =

    4

>> K1

K1 =

  -5.8899   38.1171  -21.0171   18.9760
   1.9240   53.5661   76.9821   25.3400
>> K2

K2 =

  -6.1131   43.1238  -13.1163   21.6467
  -0.7467  113.4745  171.5185   57.2973

>> eig1

eig1 =

  -4.0000
  -3.0000
  -2.0000
  -1.0000

>> eig2
```

eig2 =

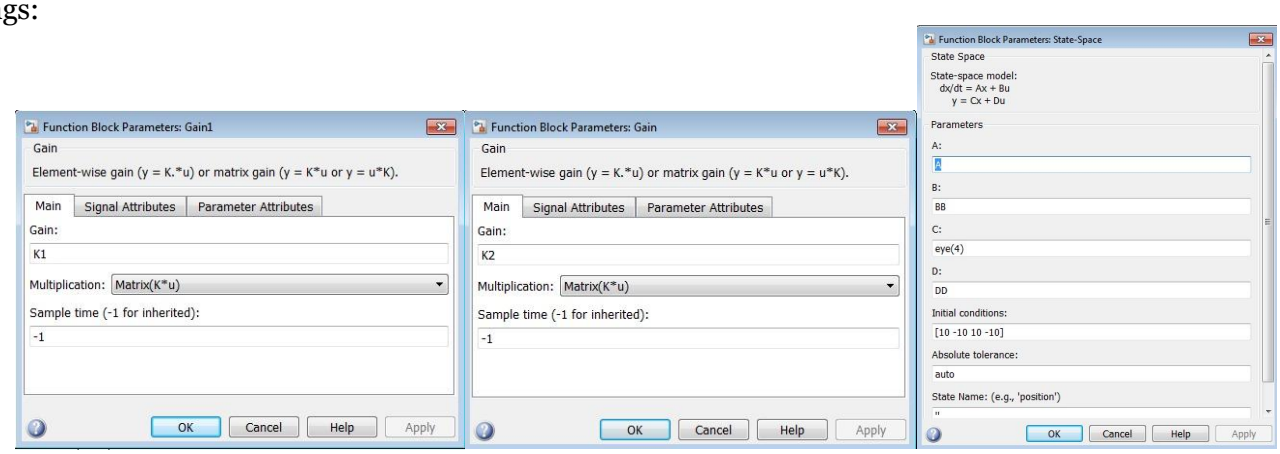
-4.0000
-3.0000
-2.0000
-1.0000

➔ The rank of matrices Phi1 and Phi2 is complete so we can use matrices kbar and F.

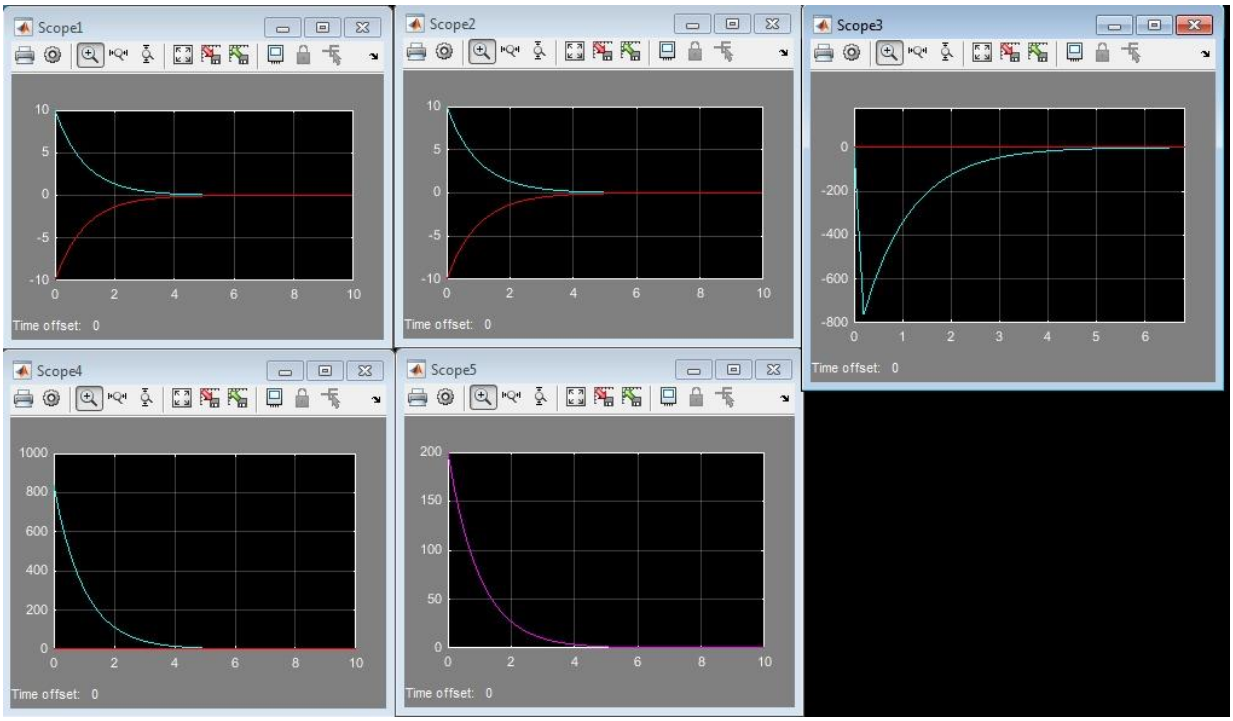
In the simulation section, we have:



Settings:



Outputs:



Scope 1 and 2: States 1 and 2

Scope 5: Output Comparison

Scope 4: Control Signal Comparison

Scope 3: Comparison of Control Signal Derivatives

➔ It is observed that stabilization has been achieved, and for each state feedback matrix 2, we have control signals.

Section B: Linear Feedback Control Systems: Tracking Design

1, 2, 3) Designing Static Precompensators for Tracking with Integral Control for the System and Analyzing the Tracking Performance and Robustness in the Presence of Model Parameter Changes and Closed-Loop System Performance by Introducing Constant Disturbances:

```
> A=[10.9 -38.2 28.1 -5.8;1 0 0 0;0 1 0 0;0 0 1 0];
B=[1;0;0;0];
C=[0 -2.5 -20 -37.5];
D=[0;0;0;0];
Aa=[10.95 -38.22 28.1 -5.82;1.01 -0.01 0.02 0;-0.02 1.02 0.01 0;0.03 -0.01 1.01 0];
Bb=[-0.01;0.001;-0.001;1];
Cc=[0.02 -2.51 -20.001 -37.49];
%static compensator
K1=[18.2 78.1 -3.2 20.9];
invGclz=inv(-C*inv(A-B*K1)*B);
%INTEGRAL CONTROL SINGLE INPUT
Abar=[A zeros(4,1);-C 0];
Bbar=[B;0];
Cbar=[C 0];
phic=[A B;0 -C];
rankphic=rank(phic);
eig(Abar);
K2=acker(Abar,Bbar,[-1 -2 -3 -4 -5]);
Acl=Abar-Bbar*K2;
eig1=eig(Acl)
```

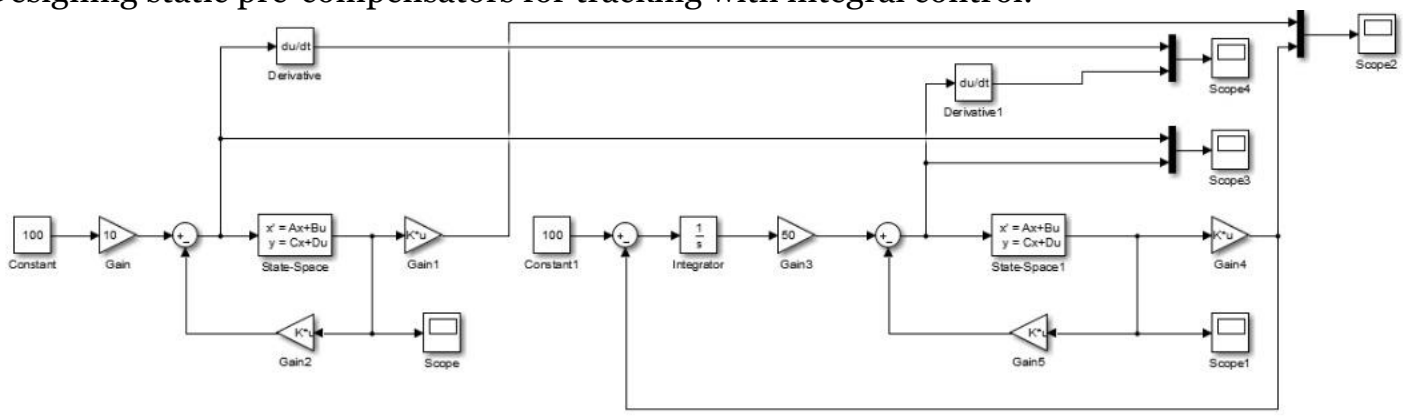
Result:

```
eig1 =

  -5.0000
  -4.0000
  -3.0000
  -2.0000
  -1.0000

>> invGclz

invGclz =

  -0.7120

>> rankphic

rankphic =

    5


>> eig1
```

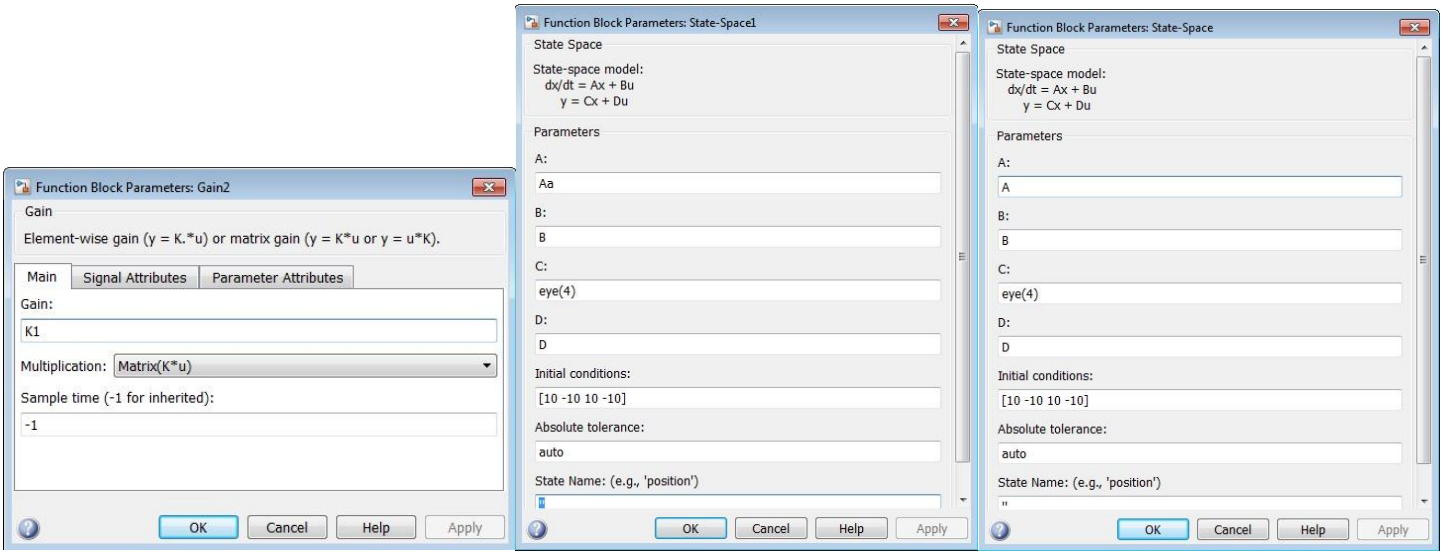
```
eig1 =

  -5.0000
  -4.0000
  -3.0000
  -2.0000
  -1.0000

>> >> K2

K2 =

  25.9000   46.8000  245.1000  204.2000    3.2000
```

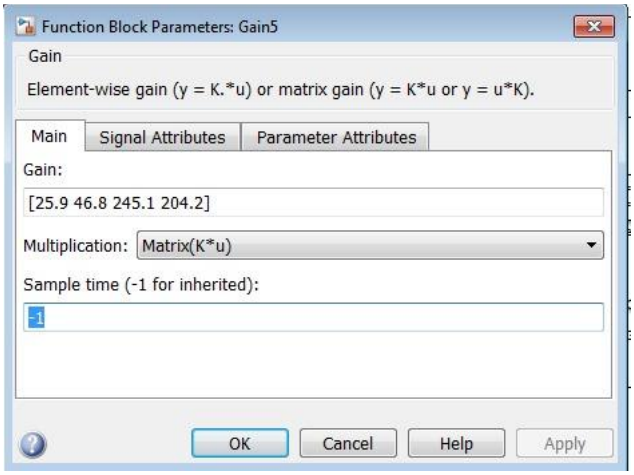
➔ The rank of the completed matrix is full, so we can use integral control for gain calculation, and since we have 5 states, we also have 5 eigenvalues.

Simulation:
Designing static pre-compensators for tracking with integral control:

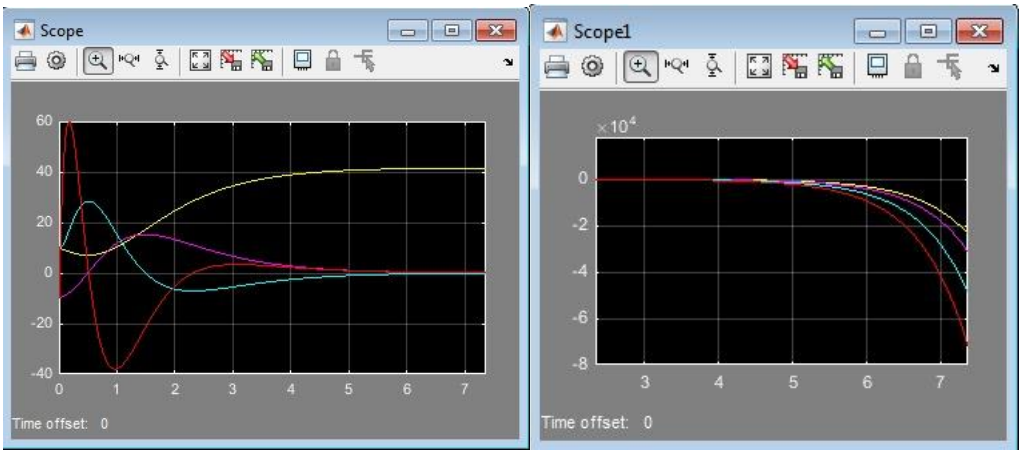


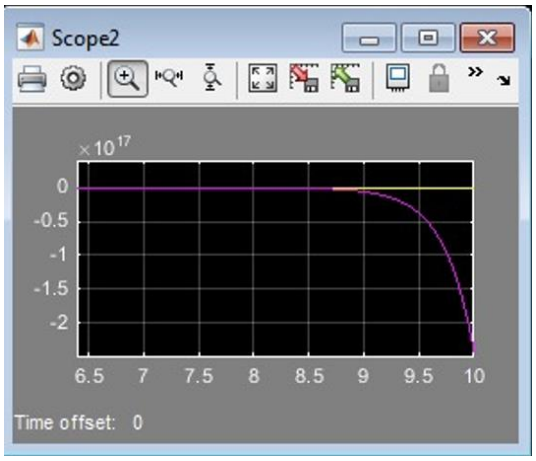
Setting:

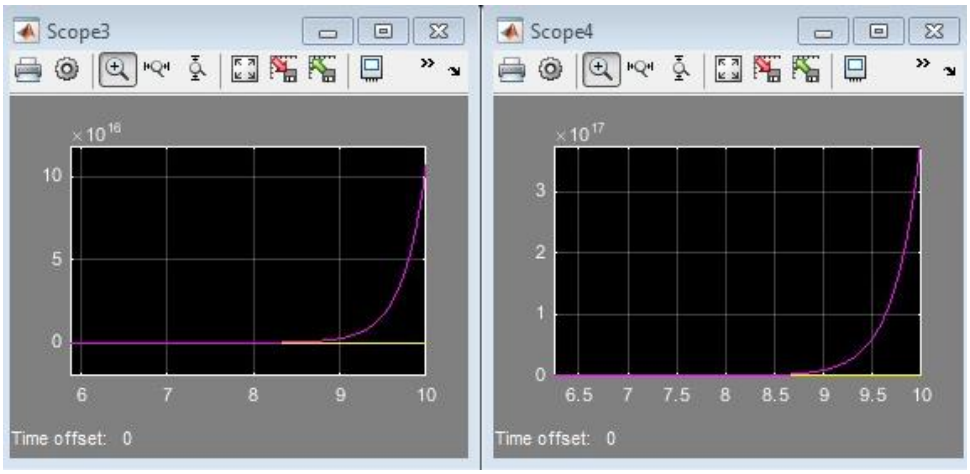



Function Block Parameters: Gain5
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
[25.9 46.8 245.1 204.2]
Multiplication:
Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply

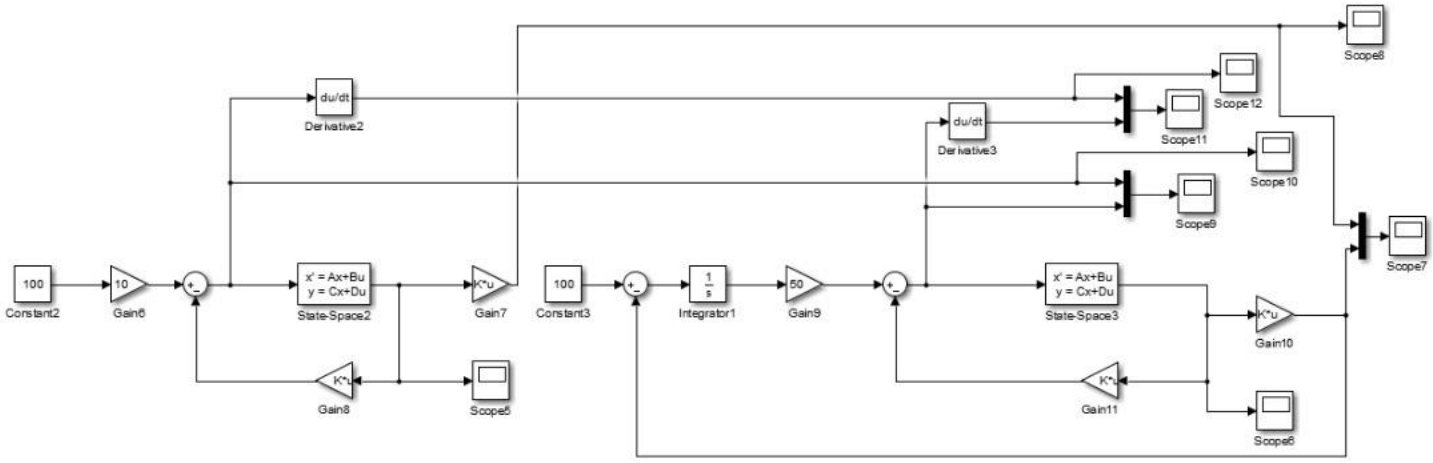
Output:


Scope
Time offset: 0
Scope1
×10^4
Time offset: 0

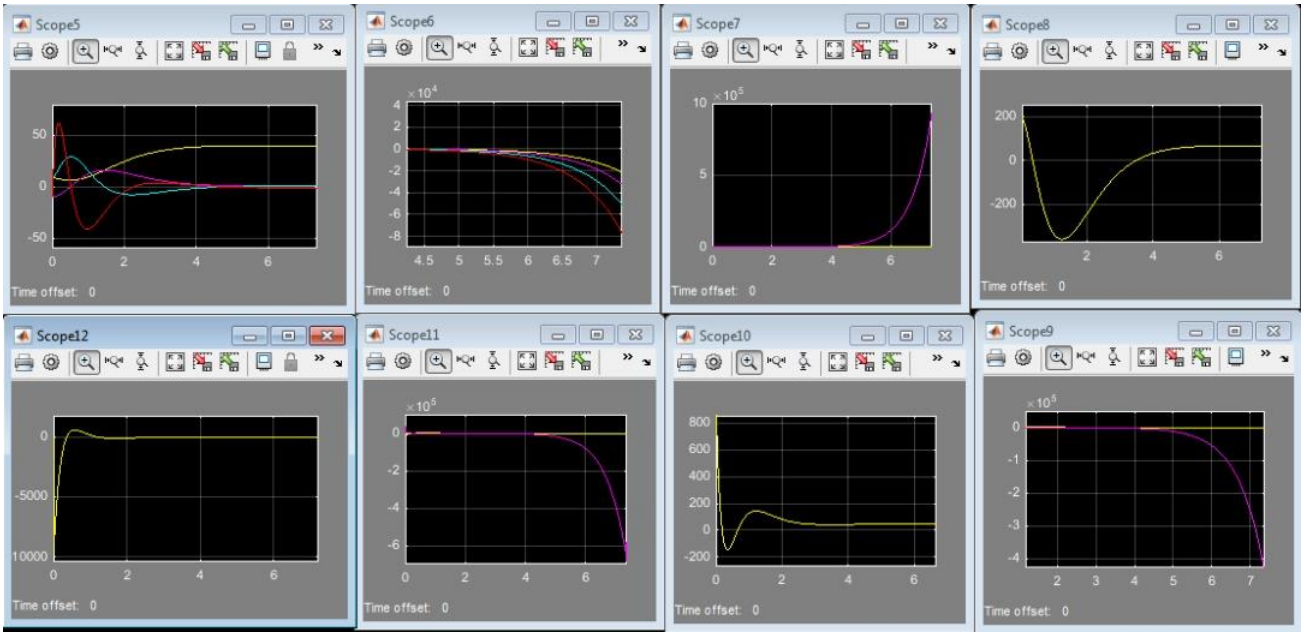

Scope2
×10^17
Time offset: 0



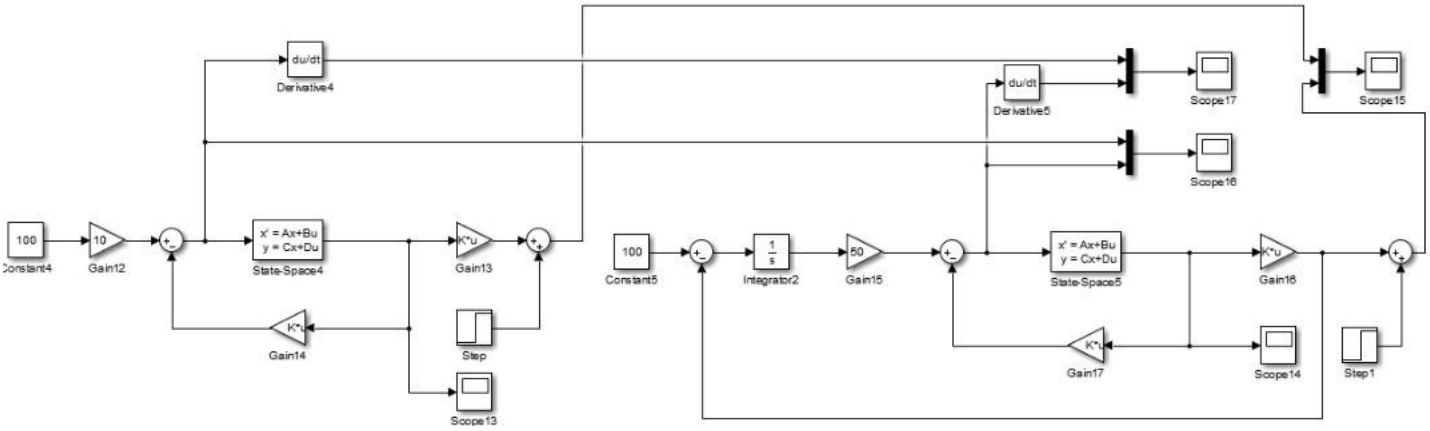
Robustness against parameter changes:



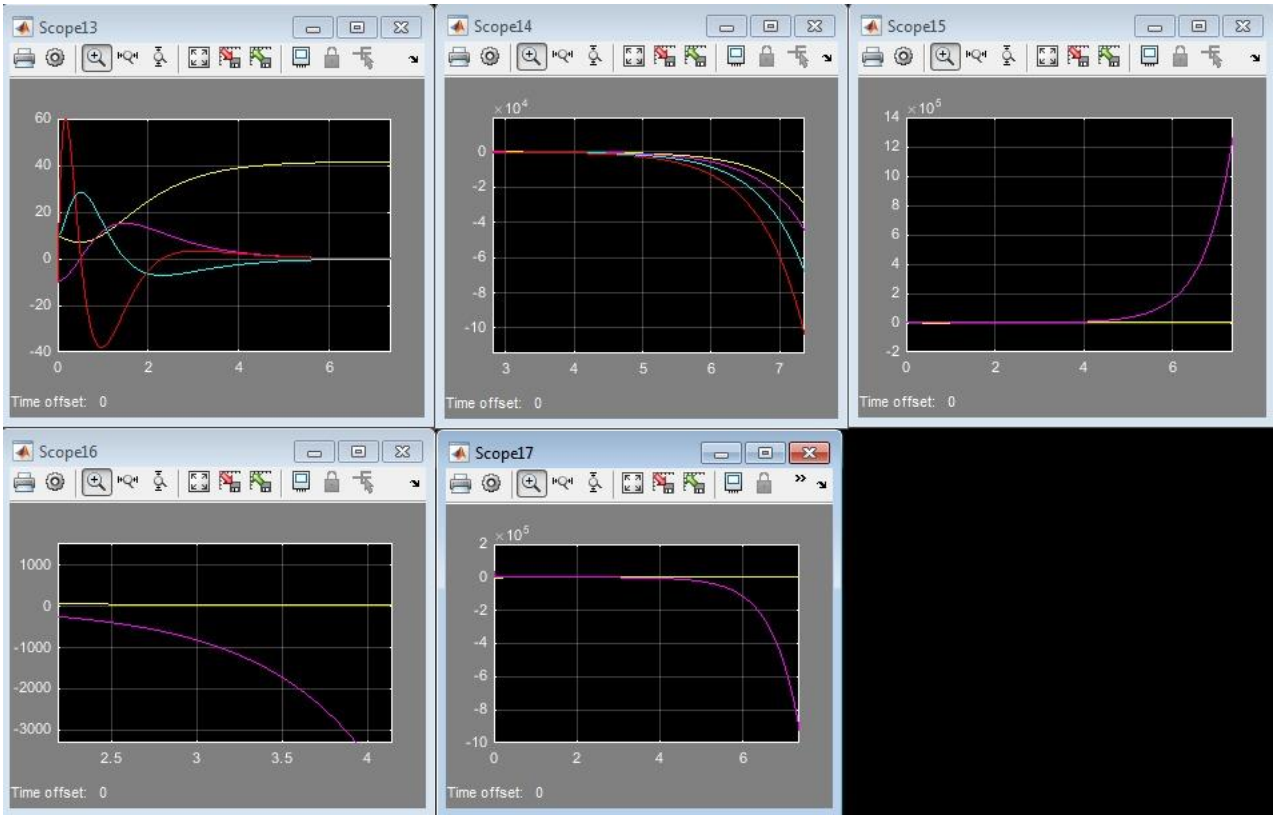
Output:



Closed-loop system performance with the introduction of constant disturbances:



Output:



➔ Conclusion:

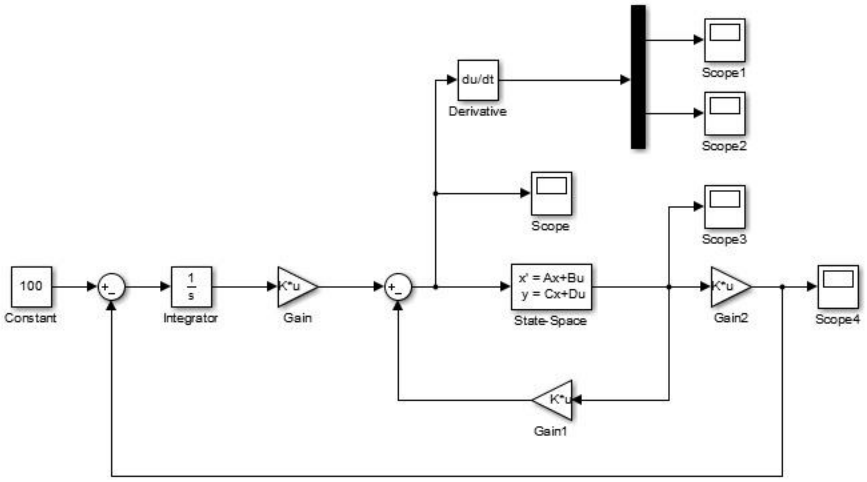
The state plots, output, command signal, and command signal variations obtained from the static precompensator method are better than the integral control method.

4. Designing a tracker for a two-input system:

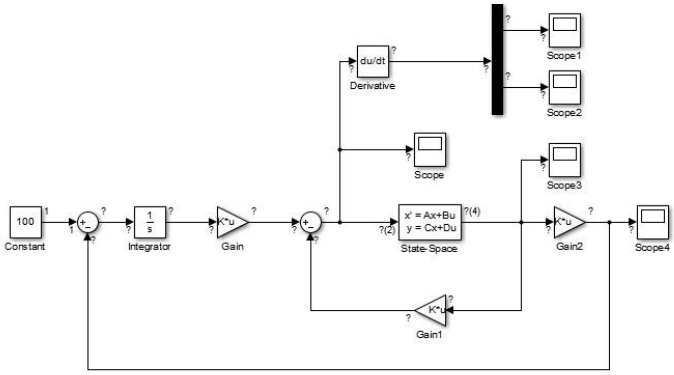
```
A=[10.9 -38.2 28.1 -5.8;1 0 0 0;0 1 0 0;0 0 1 0];
BB=[1 0;0 0;0 0;0 1];
DD=[0 0;0 0;0 0;0 0];
ABAR=[A zeros(4,1);-C 0];
BBAR=[BB;0 0];
CBAR=[C 0];
PHIC=[BB A;0 0 -C];
RANKPHIC=rank(PHIC);
F=[-1 0 0 0 0;0 -2 0 0 0;0 0 -3 0 0;0 0 0 -4 0;0 0 0 0 -5];
kbar=[1 1 1 0 0;0 0 0 1 1];
PHIO=[kbar;kbar*F;kbar*(F^2);kbar*(F^3);kbar*(F^4)];
rankPHIO=rank(PHIO);
T=lyap(ABAR,-F,-BBAR*kbar);
K=kbar*inv(T);
eig2=eig(ABAR-BBAR*K);
>> rankPHIC
Undefined function or variable 'rankPHIC'.

>> RANKPHIC

RANKPHIC =

    5

>> rankPHIO

rankPHIO =

    5

>> T
T =

    0.0119    0.0252    0.0334    0.0551    0.0469
   -0.0119   -0.0126   -0.0111   -0.0138   -0.0094
    0.0119    0.0063    0.0037    0.0034    0.0019
   -0.0119   -0.0031   -0.0012    0.2491    0.1996
    0.2381    0.0118   -0.0000   -2.3443   -1.5000

>> K

K =

   16.3641  -29.8897   31.1490   -4.9149    0.0841
   -3.1360  -15.1811  -14.1407    9.5359    0.5816
>> eig2

eig2 =

   -1.0000
   -2.0000
```

-5.0000
-3.0000
-4.0000

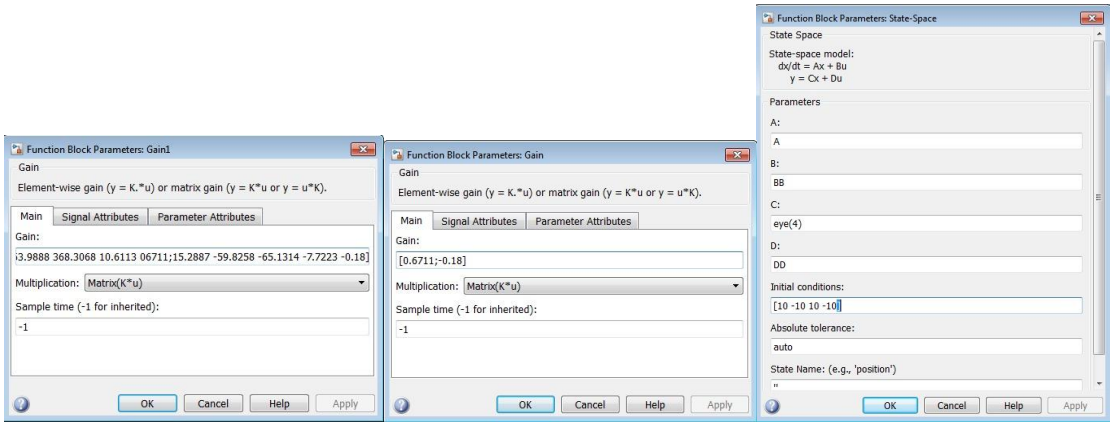
Simulink:



Error:



Settings:



➔ We are unable to find the tracker.

5. Adding (s=0) as a transfer function:

```
s=tf('s');
gg=(-2.5*(s^2+8*s+15)*(s))/((s-0.4)*(s-0.5)*(s^2-10*s+29));
A=[10.9 -38.2 28.1 -5.8;1 0 0 0;0 1 0 0;0 0 1 0];
B=[0;0;0;01];
CC=[-37.5 -2.5 -20 0];
Abarr=[A zeros(4,1);-CC 0];
Bbarr=[B;0];
Cbarr=[CC 0];
phicc=[B A;0 -CC];
rankphicc=rank(phicc)
eig(Abarr);
K3=acker(Abarr,Bbarr,[-1 -2 -3 -4 -5]);
ACL=Abarr-Bbarr*K3;
eig3=eig(ACL)

rankphicc =

    5


eig3 =

  -5.0000
  -4.0000
  -3.0000
  -2.0000
  -1.0000

>> K3 =

 -56.7431  165.7379 -169.1362   25.9000   -1.0345
```

➔ We observe that the rank of the completed matrix is full, so we can use the integral control method for gain calculation.

Section C: Observer

1. Observability:

```
>> A=[10.9 -38.2 28.1 -5.8;1 0 0 0;0 1 0 0;0 0 1 0];
B=[0;0;0;1];
C=[0 -2.5 -20 -37.5];
D=[0;0;0;0];
phio=obsv(A,C);
Rankphio=rank(phio);
>> Rankphio

Rankphio =

4
```

➔ The rank of the completed matrix is full, so this realization is observable.

2 and 3) Selecting 2 sets of poles and obtaining the gains and variable state response, etc.

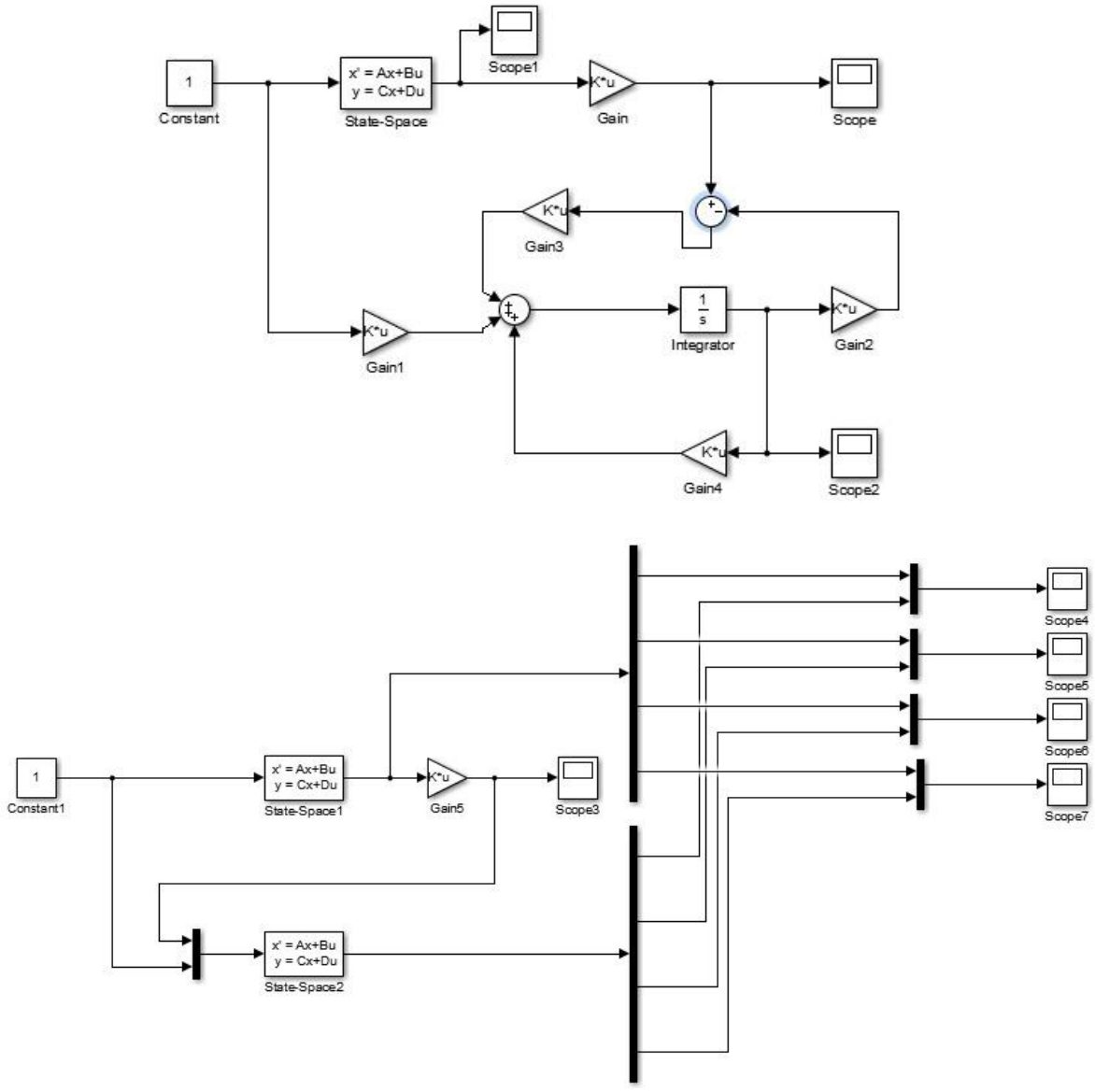
```
>> A=[10.9 -38.2 28.1 -5.8;1 0 0 0;0 1 0 0;0 0 1 0];
C=[0 -2.5 -20 -37.5];
DD=[0 0;0 0;0 0;0 0];
L1=acker(A',C',[-1 -2 -3 4])';
L2=acker(A',C',[-50 -51 -52 -53])';
```

Result:

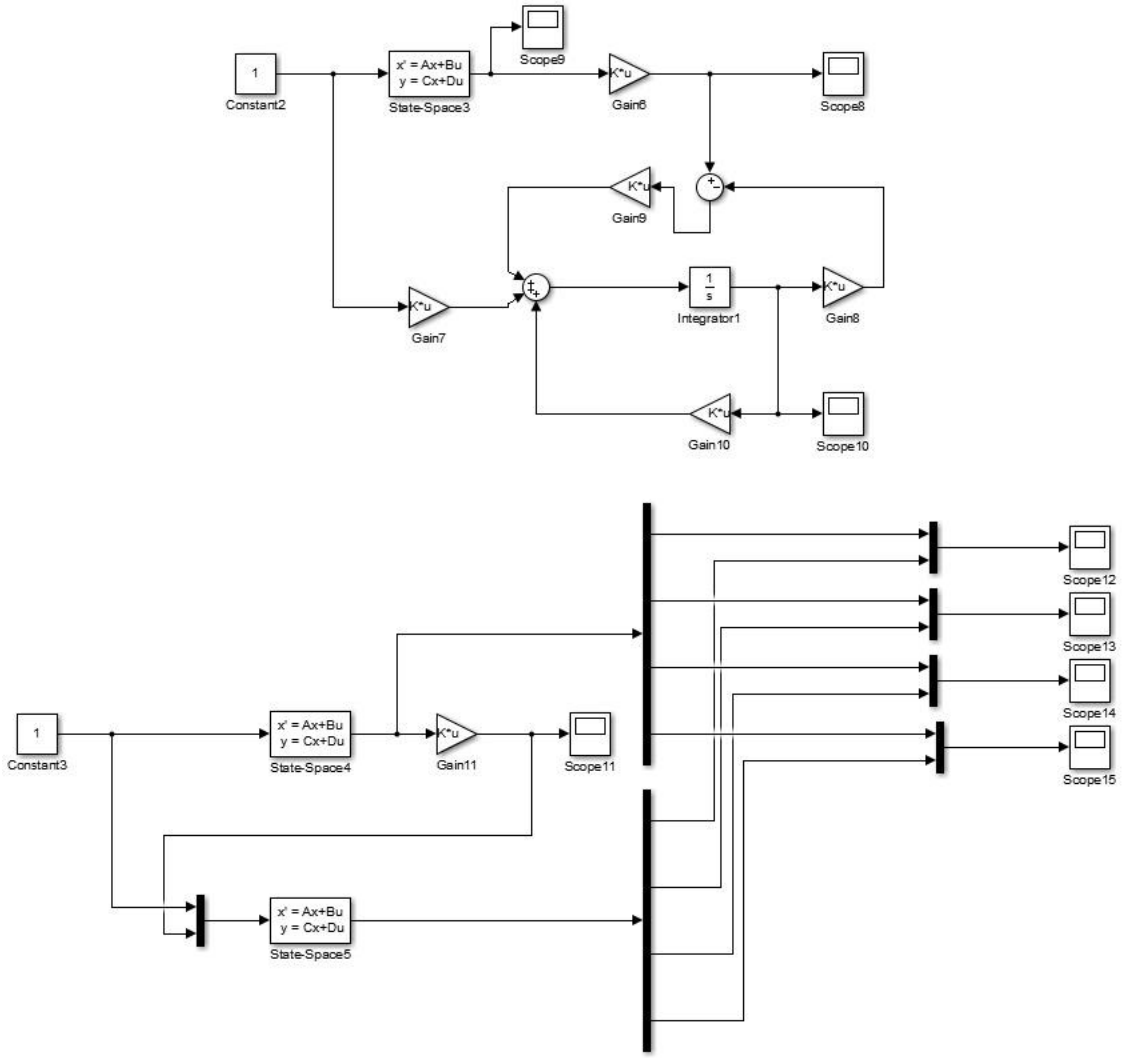
```
>> L1
L1 =

  -42.3697
   -5.1989
   -0.3922
   -0.0016
>> L2

L2 =

   1.0e+03 *

   -5.7076
    4.6518
   -2.5868
1.0637
```

➔ So, we were able to obtain the observer gains for slow and fast poles.

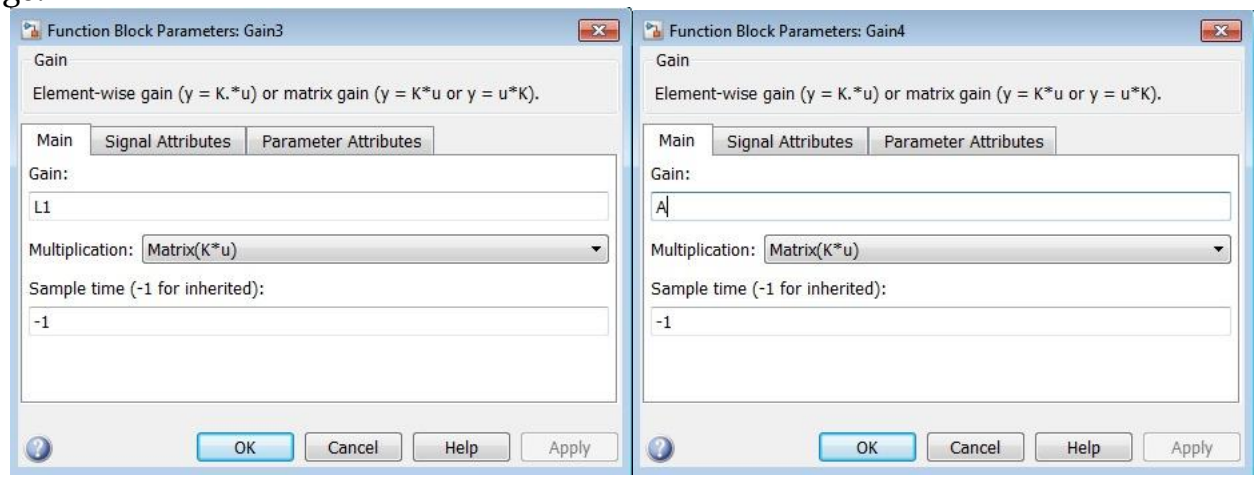
Simulink:
For slow & fast poles:
L1

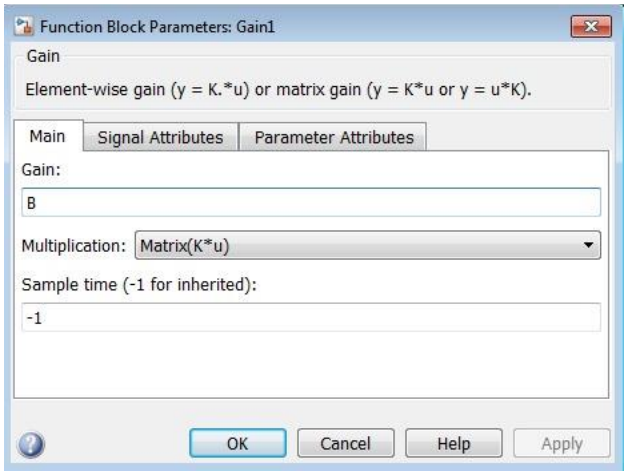


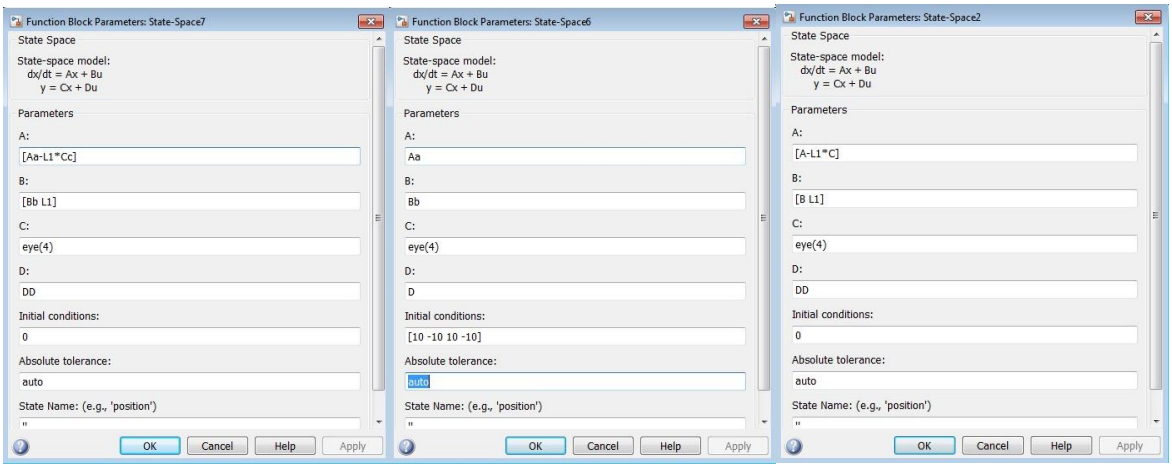
L2


1
Constant2
x' = Ax+Bu
y = Cx+Du
State-Space3
Scope9
K*u
Gain6
Scope8
K*u
Gain9
1
s
Integrator1
K*u
Gain8
K*u
Gain7
K*u
Gain10
Scope10
Scope12
Scope13
Scope14
Scope15
1
Constant3
x' = Ax+Bu
y = Cx+Du
State-Space4
K*u
Gain11
Scope11
x' = Ax+Bu
y = Cx+Du
State-Space5

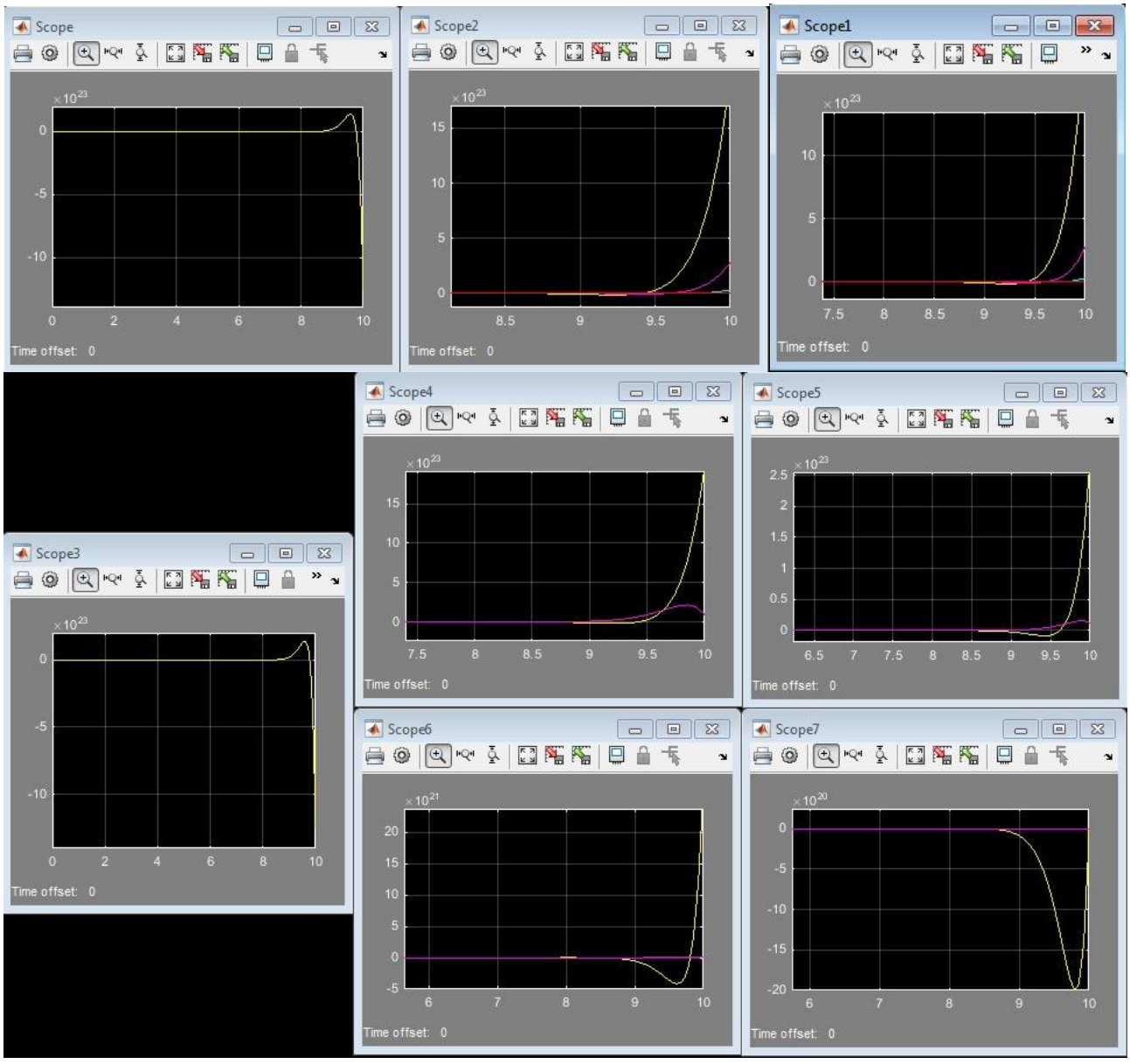
Settings:


Function Block Parameters: Gain3
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
L1
Multiplication: Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply
Function Block Parameters: Gain4
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
A
Multiplication: Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply


Function Block Parameters: Gain1
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
B
Multiplication:
Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply

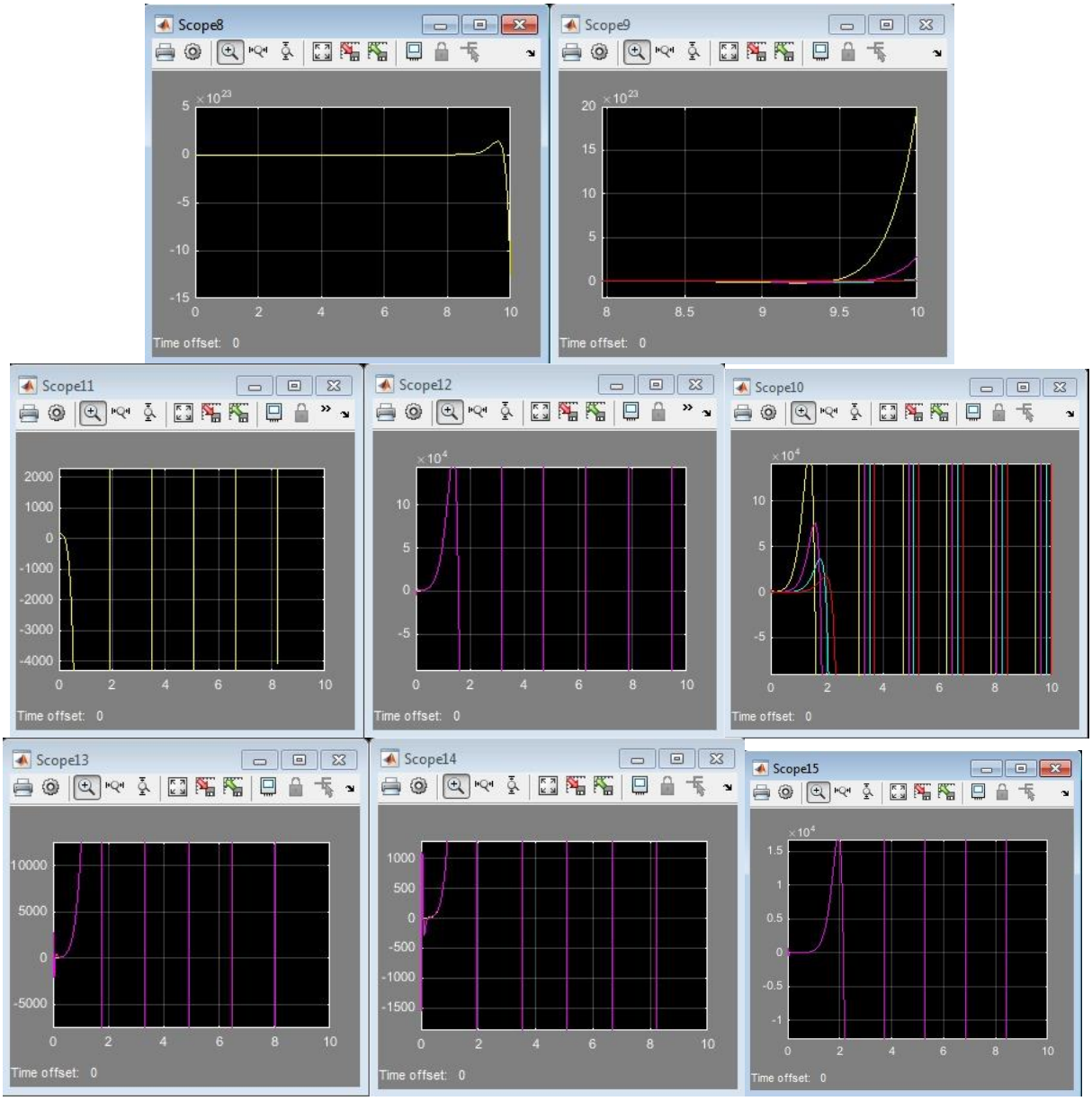

Function Block Parameters: State-Space7
State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
[Aa-L1*Cc]
B:
[Bb L1]
C:
eye(4)
D:
DD
Initial conditions:
0
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply
Function Block Parameters: State-Space6
State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
Aa
B:
Bb
C:
eye(4)
D:
D
Initial conditions:
[10 -10 10 -10]
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply
Function Block Parameters: State-Space2
State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
[A-L1*C]
B:
[B L1]
C:
eye(4)
D:
DD
Initial conditions:
0
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply

Output L1:



Scopes:
3: Output
4: Comparison of the first state of the system and full-order observer for slow poles
5: Comparison of the second state of the system and full-order observer for slow poles
6: Comparison of the third state of the system and full-order observer for slow poles
7: Comparison of the fourth state of the system and full-order observer for slow poles

Output L2:



Scopes:
11: Output
12: Comparison of the first state of the system and full-order observer for fast poles
13: Comparison of the second state of the system and full-order observer for fast poles
14: Comparison of the third state of the system and full-order observer for fast poles

➔ Result:
The convergence of states for the full-order observer for fast poles is faster compared to slow poles.

6. Robustness against parameter changes and using the same observer for estimating the changed system state.

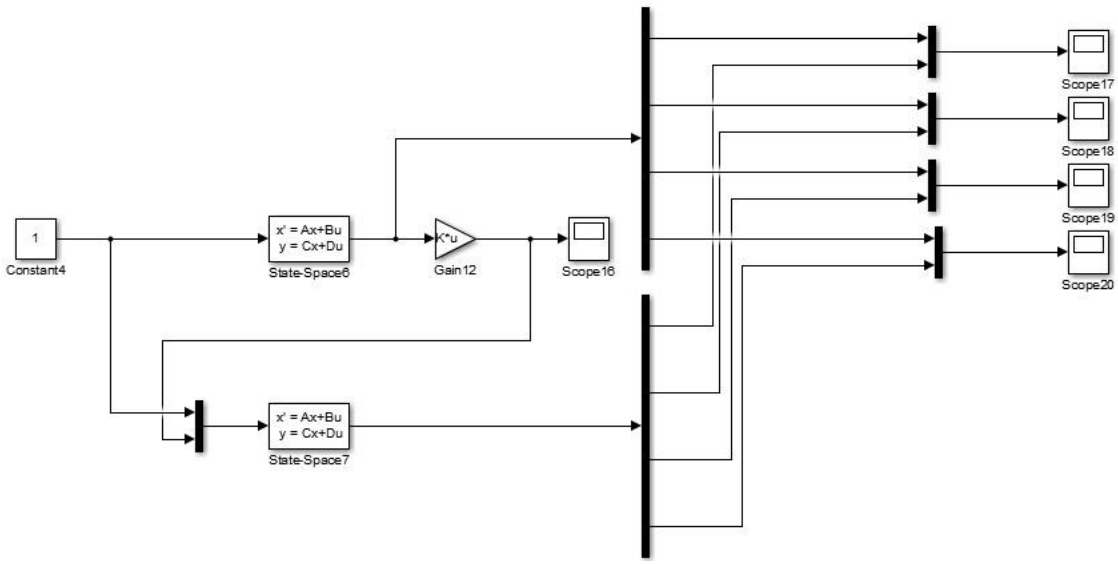
Coding section:

```
D=[0;0;0;0];
DD=[0 0;0 0;0 0;0 0];
Aa=[10.95 -38.22 28.1 -5.82;1.01 -0.01 0.02 0;-0.02 1.02 0.01 0;0.03 -0.01 1.01 0];
Bb=[1;0.001;-0.001;-0.01];
Cc=[0.02 -2.51 -20.001 -37.49];
L1=acker(A',C',[-1 -2 -3 -4])';
```

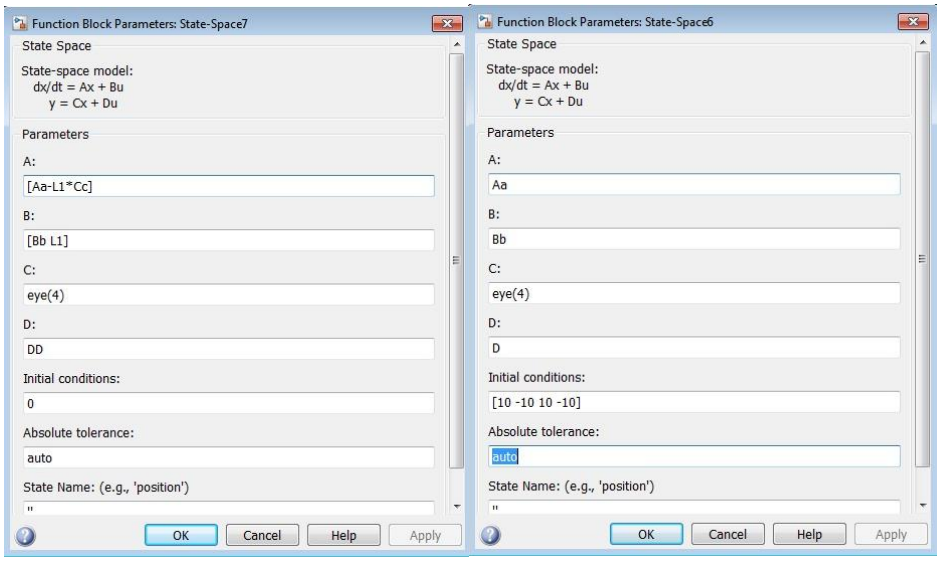
Result:

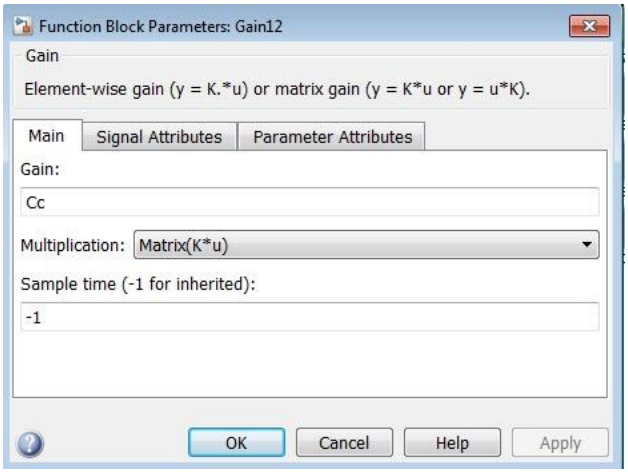
```
>> L1

L1 =

  -42.3697
   -5.1989
   -0.3922
   -0.0016
```

Simulink:

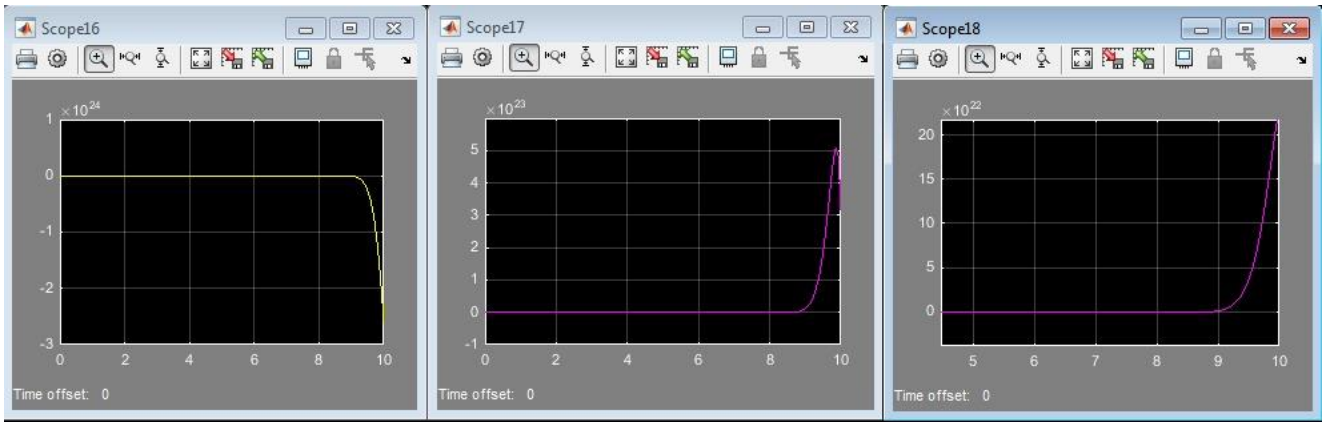


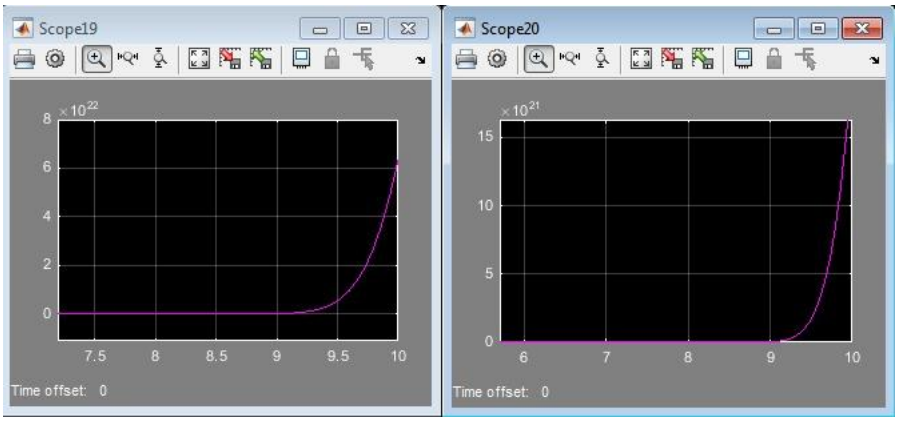
Setting:


Function Block Parameters: State-Space7
State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
[Aa-L1*Cc]
B:
[Bb L1]
C:
eye(4)
D:
DD
Initial conditions:
0
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply
Function Block Parameters: State-Space6
State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
Aa
B:
Bb
C:
eye(4)
D:
D
Initial conditions:
[10 -10 10 -10]
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply


Function Block Parameters: Gain12
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
Cc
Multiplication:
Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply

Outputs:


Scope16
×10^24
Time offset: 0
Scope17
×10^23
Time offset: 0
Scope18
×10^22
Time offset: 0



16: Output
17: Comparison of the first state of the system and observer
18: Comparison of the second state of the system and observer
19: Comparison of the third state of the system and observer
20: Comparison of the fourth state of the system and observer

➔ Result:

It is observed that the convergence of observer states to the system states has been achieved.

Section D: Feedback Control System with Observer

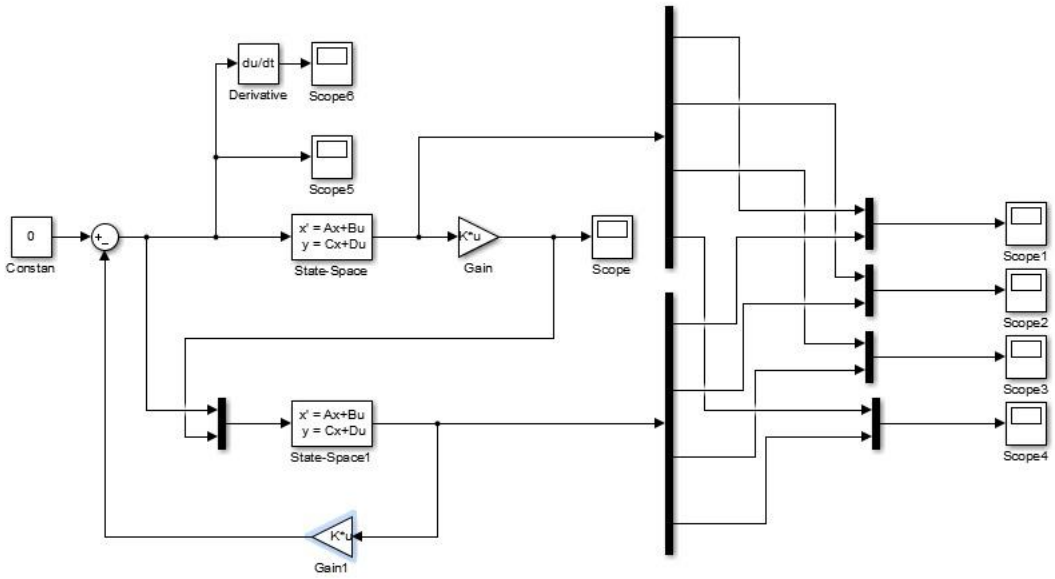
1) Feedback control system with full-order observer

```
>> A=[10.9 -38.2 28.1 -5.8;1 0 0 0;0 1 0 0;0 0 1 0];
B=[1;0;0;0];
C=[0 -2.5 -20 -37.5];
D=[0;0;0;0];
DD=[0 0;0 0;0 0;0 0];
L=acker(A',C',[-50 -51 -52 -53])';
k=[18.2 78.1 -3.2 20.9]
```

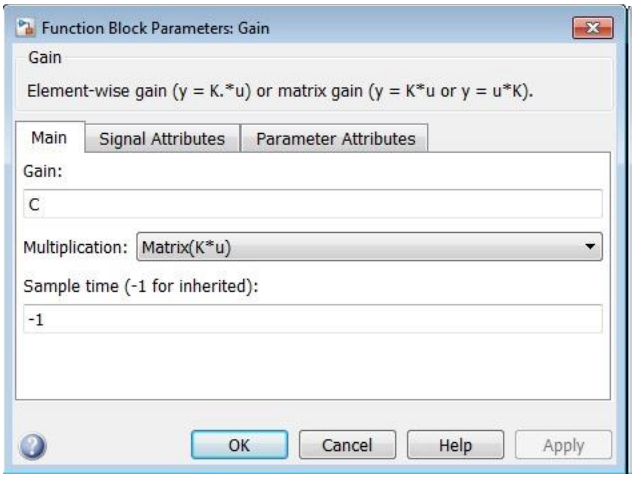
result:

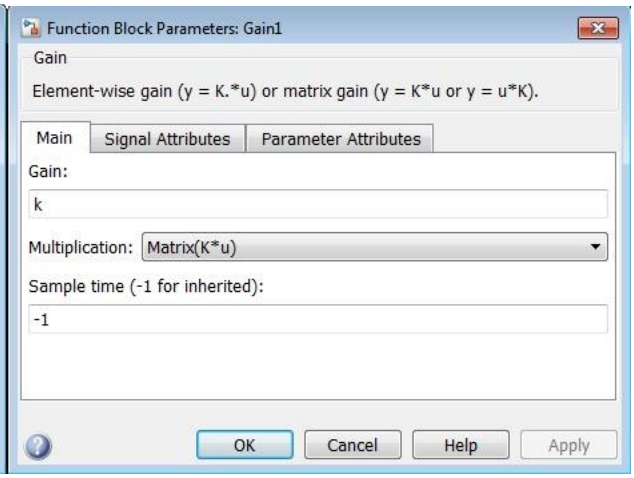
```
L =

  1.0e+03 *

  -5.7076
   4.6518
  -2.5868
   1.0637
```

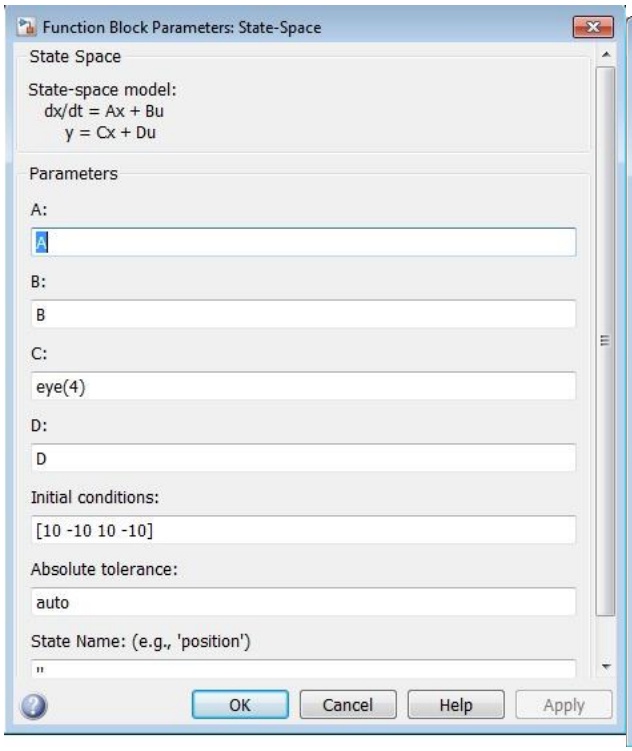
Simulink:

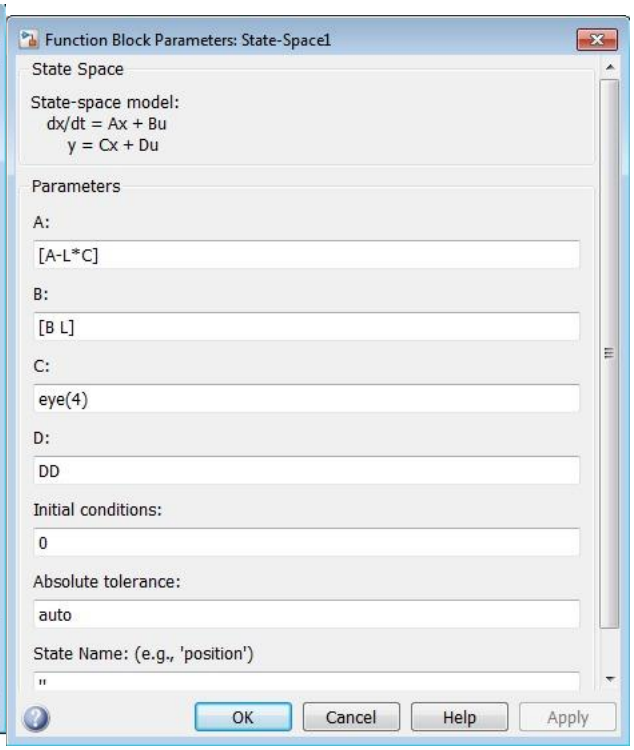


Settings:

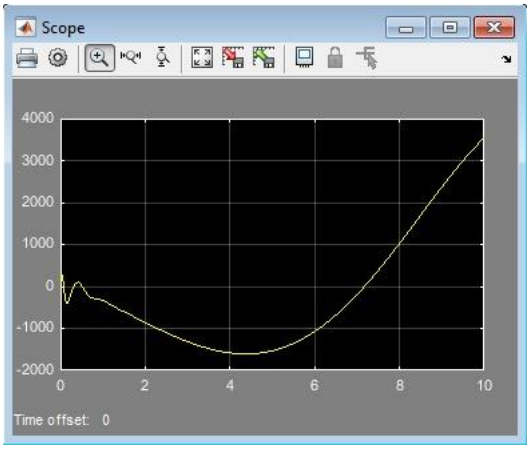

Function Block Parameters: Gain
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
C
Multiplication:
Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply


Function Block Parameters: Gain1
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
k
Multiplication:
Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply

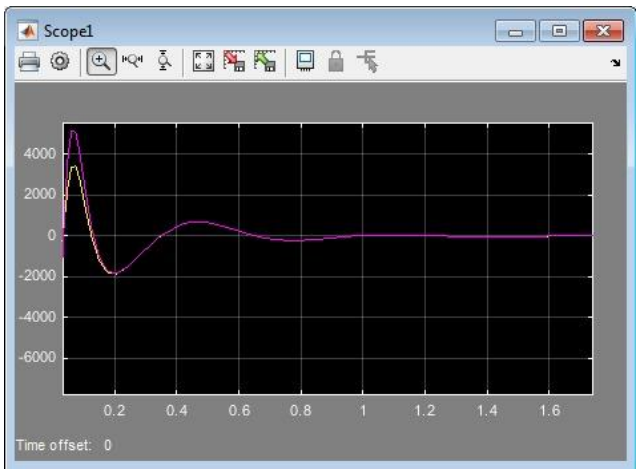

Function Block Parameters: State-Space
State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
A
B:
B
C:
eye(4)
D:
D
Initial conditions:
[10 -10 10 -10]
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply


Function Block Parameters: State-Space1
State Space
State-space model:
dx/dt = Ax + Bu
y = Cx + Du
Parameters
A:
[A-L*C]
B:
[B L]
C:
eye(4)
D:
DD
Initial conditions:
0
Absolute tolerance:
auto
State Name: (e.g., 'position')
OK
Cancel
Help
Apply

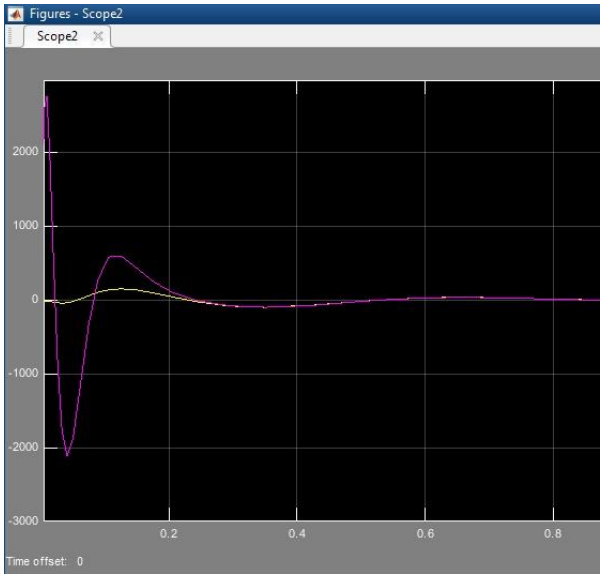
Outputs:


Scope
4000
3000
2000
1000
0
-1000
-2000
0
2
4
6
8
10
Time offset: 0

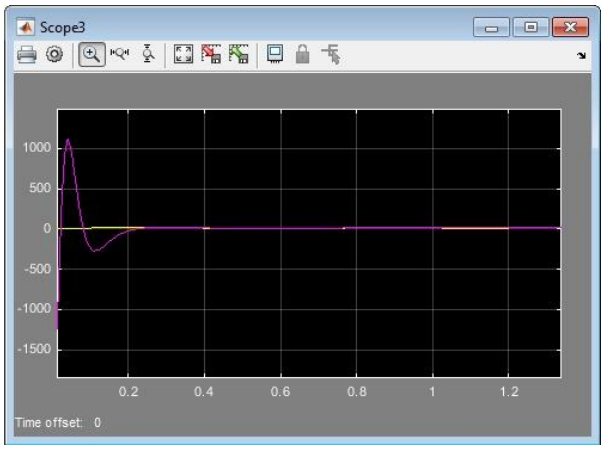
Comparison of the first state of the system and observer:



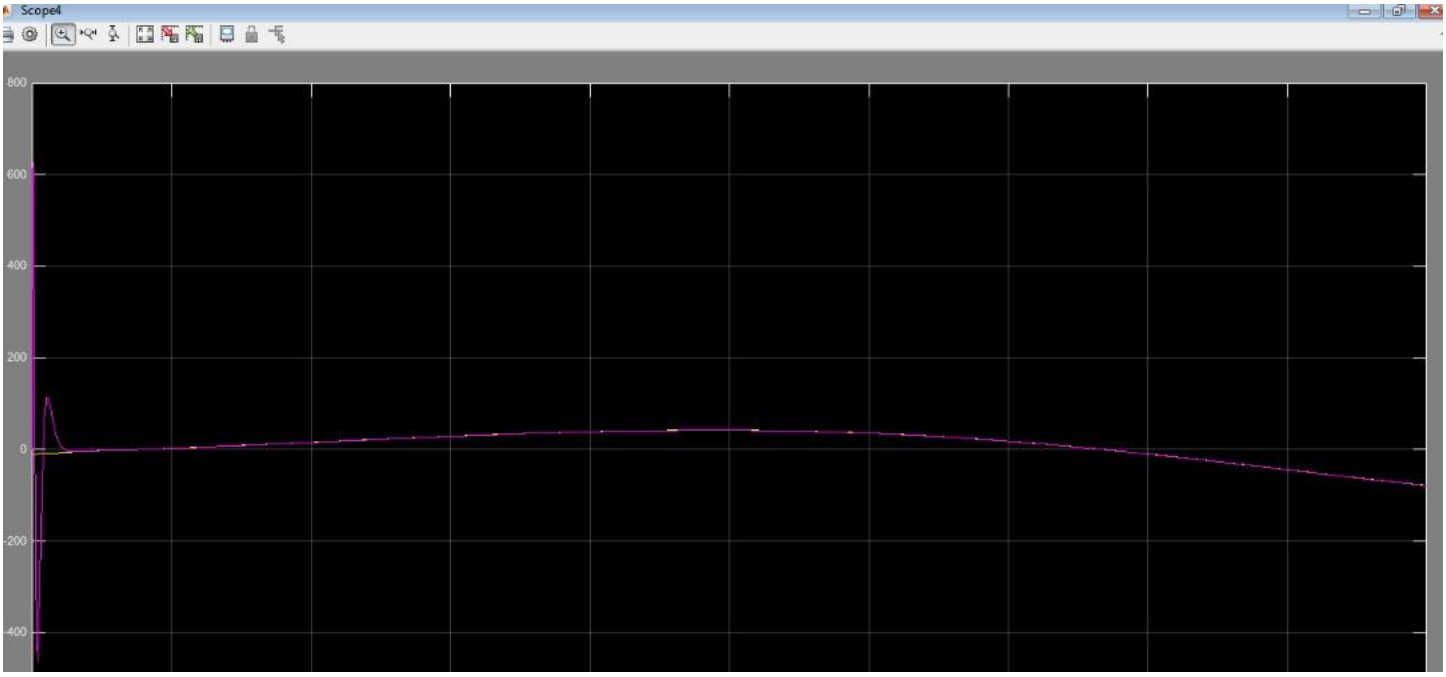
Second state:



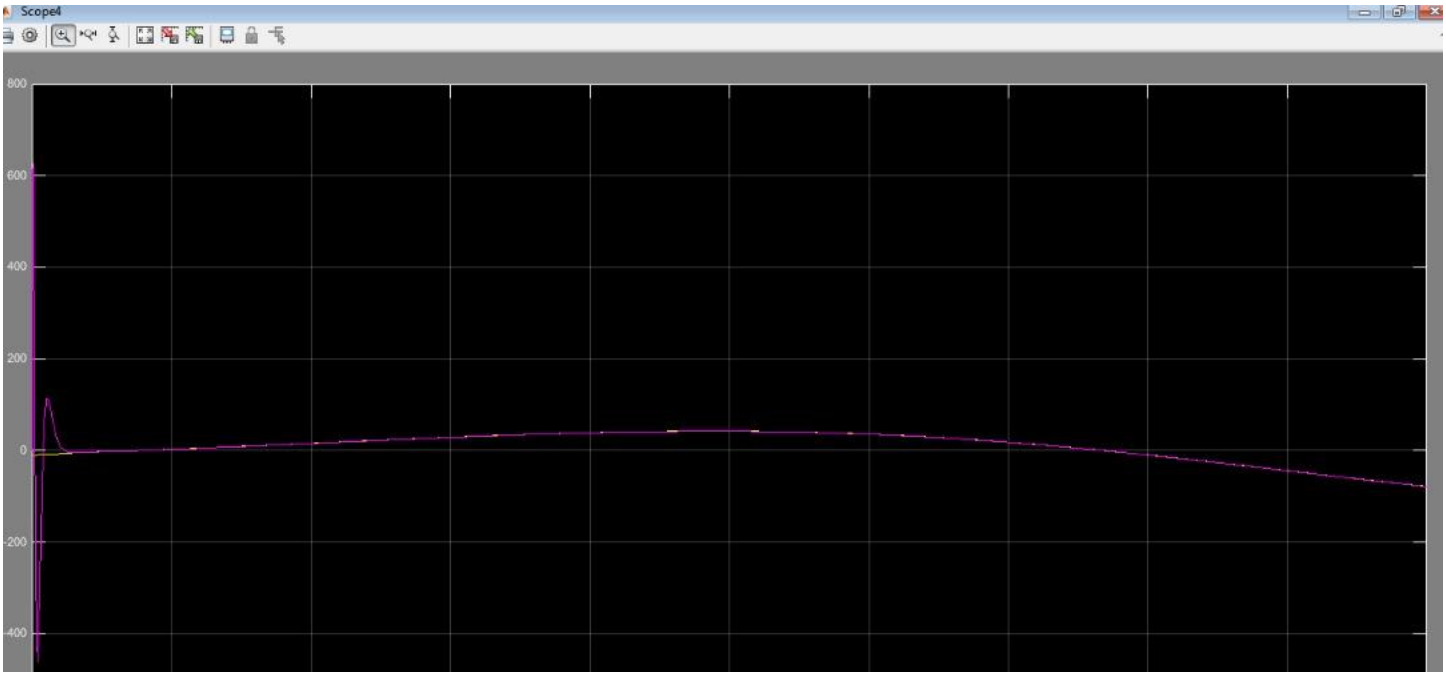
Third state:



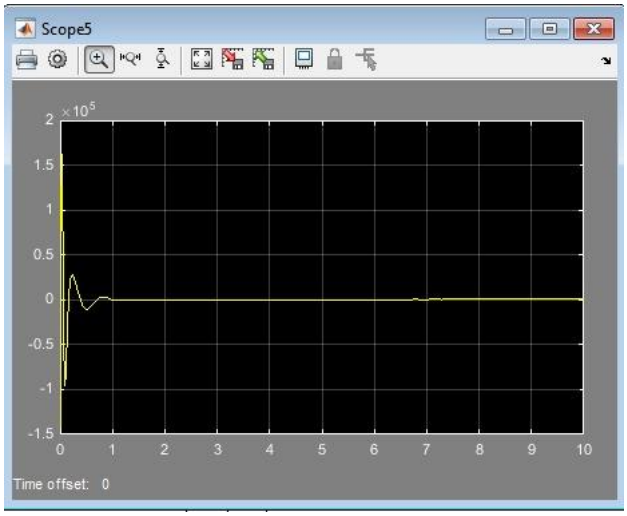
Forth state:

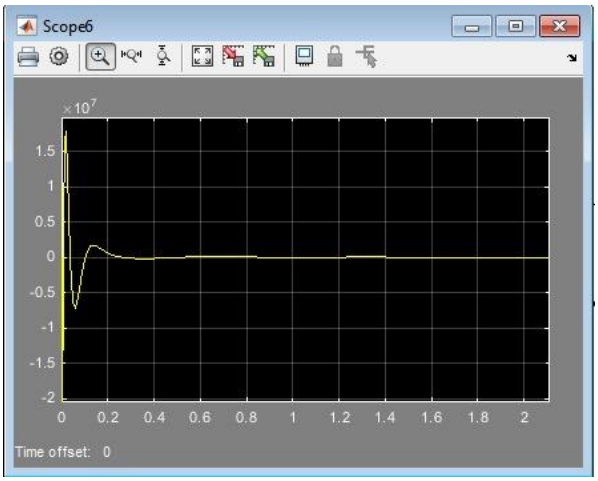


Control Signal:



Control signal derivative:



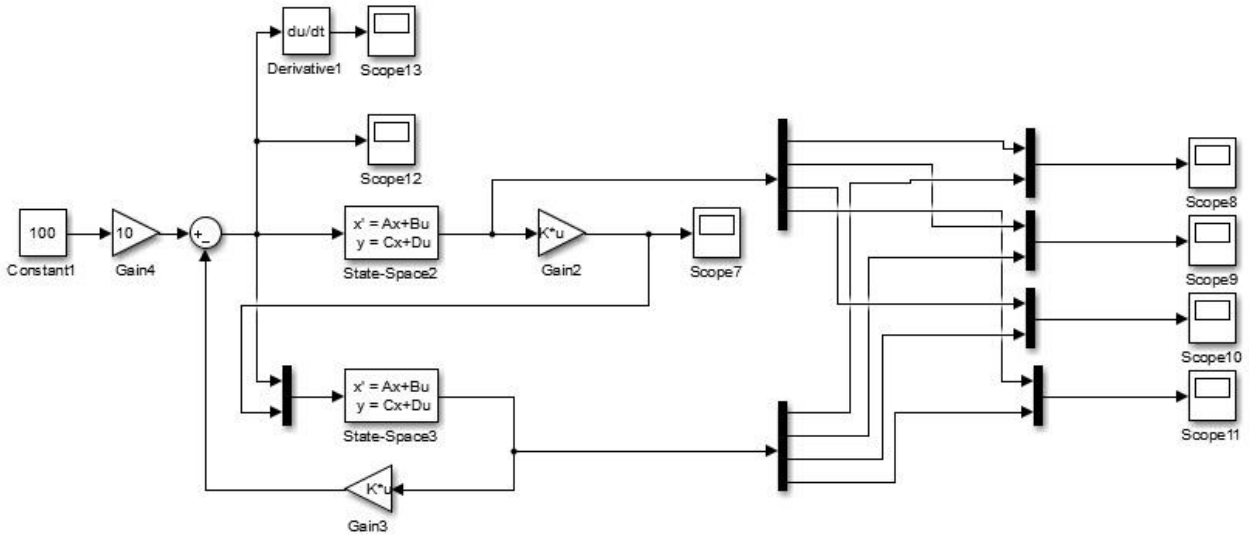


➔ Result:

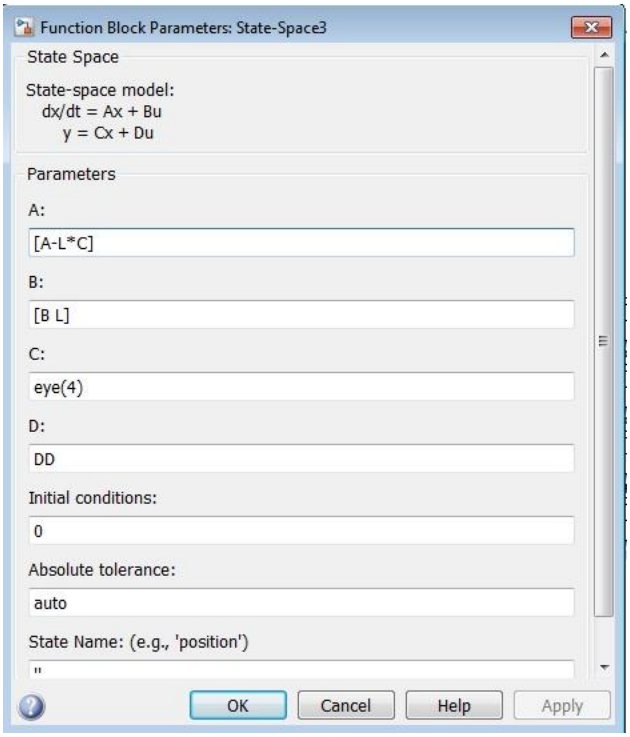
It is observed that stabilization has been achieved.

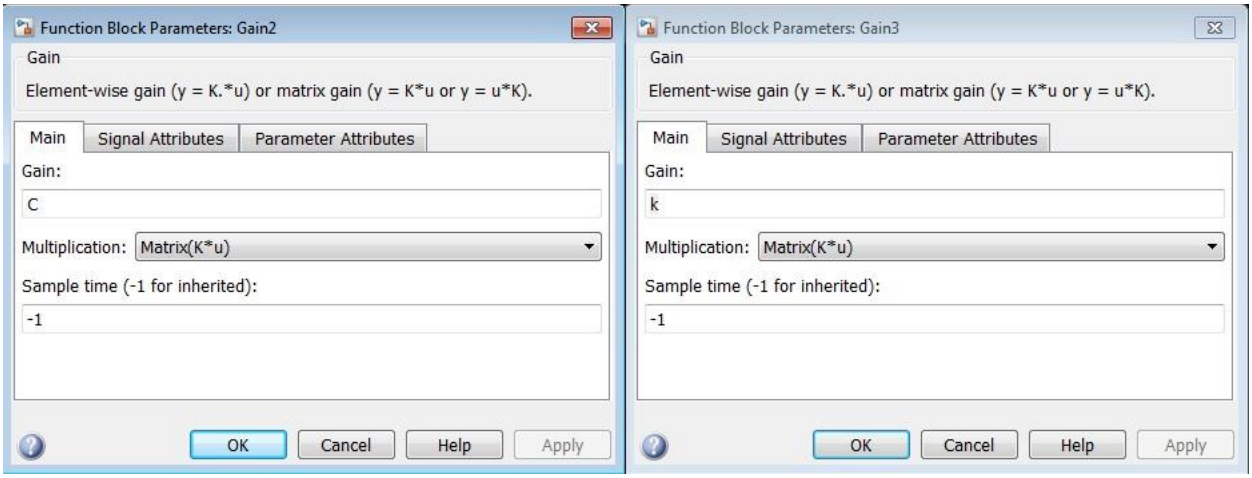
3. Static Precompensator Tracking - State Feedback - Full-Order Observer:

Simulink:

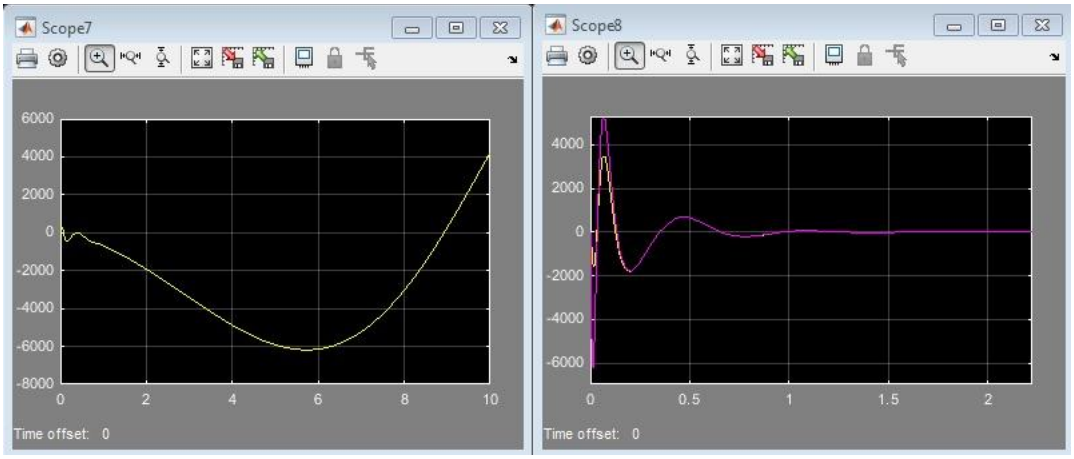


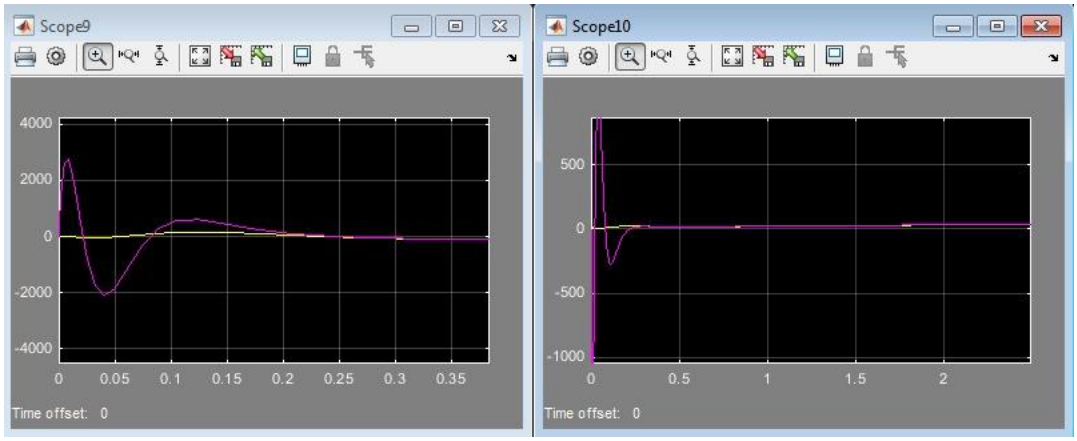
Settings:

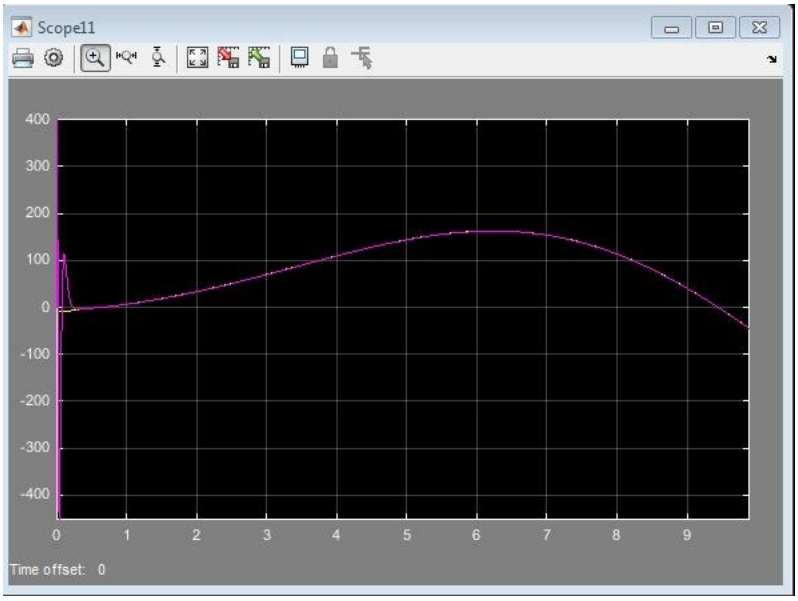


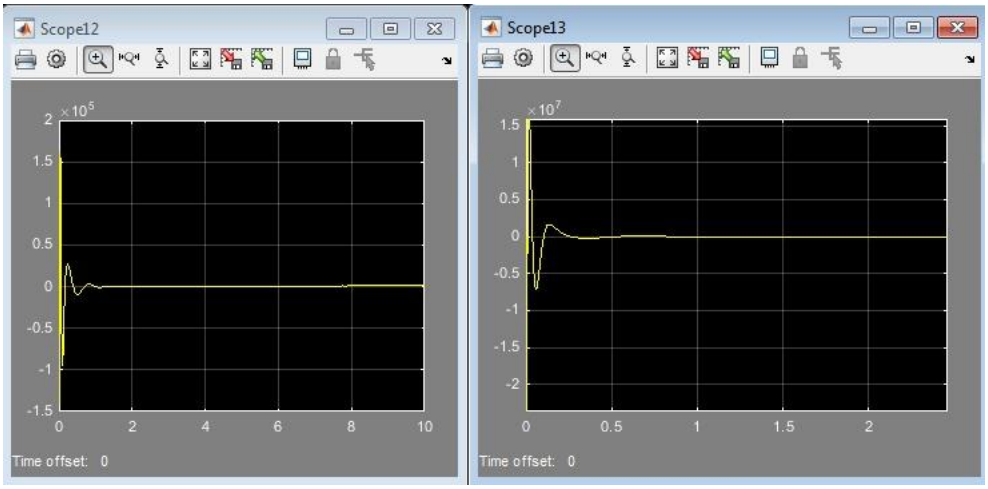

Function Block Parameters: Gain2
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
C
Multiplication:
Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply
Function Block Parameters: Gain3
Gain
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).
Main
Signal Attributes
Parameter Attributes
Gain:
k
Multiplication:
Matrix(K*u)
Sample time (-1 for inherited):
-1
OK
Cancel
Help
Apply

Outputs:


Scope7
Time offset: 0
Scope8
Time offset: 0


Scope9
Time offset: 0
Scope10
Time offset: 0





Scopes:
7: Output
8: Comparison of the first state of the system and observer
9: Comparison of the second state of the system and observer
10: Comparison of the third state of the system and observer
11: Comparison of the fourth state of the system and observer
12: Control Signal
13: Control Signal Derivative

➔ Result:
It is observed that tracking has been achieved.