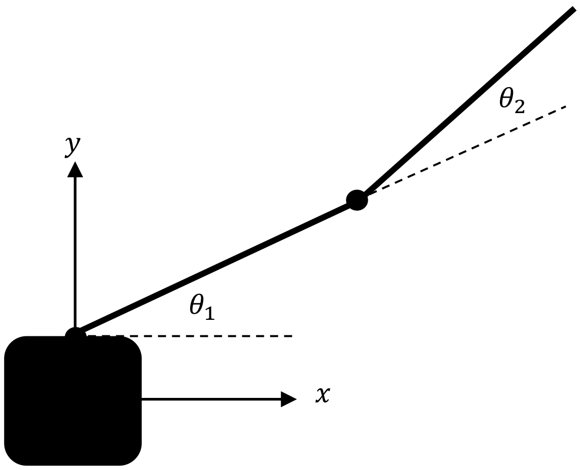
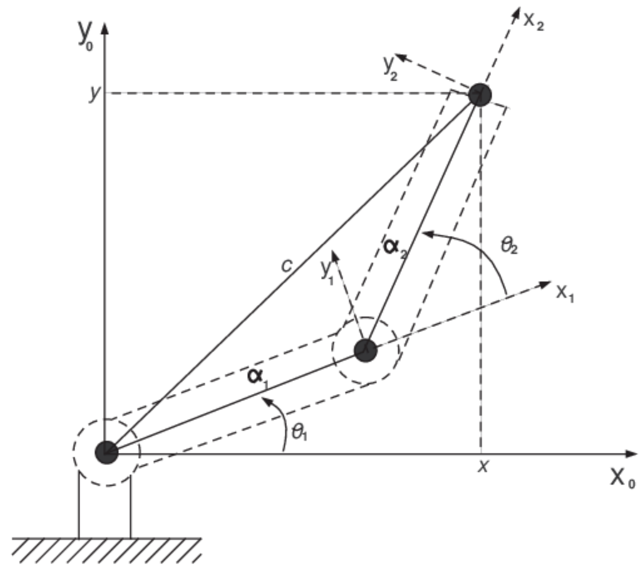
Modeling of a 2 DOF robot arm

Marzieh Ghayour Najafabadi





The system in question is a two-link arm, where the input to our system is the forces applied to each of the arms, and the output determines their final positions.

1. System's Equations

We assume that the mass and length of each arm are the same and equal to:

- $\mathbf{l_1 = l_2 = l}$

```
% Parameters---
l = 1;
m = 1;
```

$$\mathbf{m_1 = m_2 = m}$$

$$\mathbf{x_1 = l\cos\theta_1}$$

$$\mathbf{\dot{x}_1 = -l\sin\theta_1\dot{\theta}_1}$$

$$\mathbf{y_1 = l\sin\theta_1}$$

$$\mathbf{\dot{y}_1 = l\cos\theta_1\dot{\theta}_1}$$

$$\mathbf{x_2 = l\cos(\theta_1+\theta_2) + l\cos\theta_1}$$

$$\mathbf{\dot{x}_2 = -l\,[(\sin\theta_1\dot{\theta}_1) + (\sin(\theta_1+\theta_2)\,(\dot{\theta}_1+\dot{\theta}_2)]}$$

$$\mathbf{y_2 = l\sin(\theta_1+\theta_2) + l\sin\theta_1}$$

$$\mathbf{\dot{y}_2 = l\,[(\cos\theta_1\dot{\theta}_1) + (\cos(\theta_1+\theta_2)\,(\dot{\theta}_1+\dot{\theta}_2)]}$$

We use the Lagrangian equation to obtain the two terms of force:

- $\mathbf{KE = \frac{1}{2}mv_1{}^2 + \frac{1}{2}mv_2{}^2 + \frac{1}{2}I\omega^2{}_1 + \frac{1}{2}I\omega^2{}_2 = ml^2[(3.5+\cos\theta_2)\dot{\theta}_1{}^2 + \frac{3}{2}\dot{\theta}_2{}^2 + (3+\cos\theta_2)\dot{\theta}_1\dot{\theta}_2]}$

- $\mathbf{PE = mgh_1 + mgh_2 = mgl[\cos(\theta_1+\theta_2) + 2\cos\theta_1]}$

- $\mathbf{L = KE - PE =}$

$$\mathbf{ml^2[(3.5+\cos\theta_2)\dot{\theta}_1{}^2+\frac{3}{2}\dot{\theta}_2{}^2+(3+\cos\theta_2)\dot{\theta}_1\dot{\theta}_2]-\ mgl[\cos(\theta_1+\theta_2)+2\cos\theta_1]}$$

$$\mathbf{T_1 = \frac{d}{dt}\frac{\partial L}{\partial \theta_1} - \frac{\partial L}{\partial \dot{\theta}_1}}$$

$$\mathbf{T_2 = \frac{d}{dt}\frac{\partial L}{\partial \theta_2} - \frac{\partial L}{\partial \dot{\theta}_2}}$$

$$\mathbf{F = H(\ddot{\theta}) + C(\dot{\theta},\theta) + g(\theta)}$$

$$\mathbf{\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} ml^2(7+2\cos\theta_1) & ml^2(3+\cos\theta_2) \\ ml^2(3+\cos\theta_2) & 3ml^2 \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} ml^2(-\sin\theta_2)\dot{\theta}_2 & ml^2(-\sin\theta_2)\dot{\theta}_2 \\ ml^2(\sin\theta_2)\dot{\theta}_1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}}$$
$$\mathbf{+\begin{bmatrix} -mlg(\cos(\theta_1+\theta_2)+\cos\theta_1) \\ mlg\cos(\theta_1+\theta_2) \end{bmatrix}}$$

The equations obtained describe a nonlinear multi-input multi-output (MIMO) system. To analyze this system, we need to linearize it.

2. State Space Equations:

The state variables of the system are defined as follows:

$\mathbf{x_1 = \theta_1}$ $\qquad$ $\mathbf{\dot{x}_1 = \dot{\theta}_1}$

$\mathbf{x_2 = \theta_2}$ $\qquad$ $\mathbf{\dot{x}_2 = \dot{\theta}_2}$

$\mathbf{x_3 = \dot{\theta}_1}$ $\qquad$ $\mathbf{\dot{x}_3 = \ddot{\theta}_1}$

$\mathbf{x_4 = \dot{\theta}_2}$ $\qquad$ $\mathbf{\dot{x}_4 = \ddot{\theta}_2}$

First, we need to find the system's operating point so that we can linearize the equations around the operating point using the Jacobian matrix.

$\mathbf{T_1 = 0\,, T_2 = 0 \ \rightarrow}$

$$\mathbf{\begin{cases} x_1 = \frac{\pi}{2}, \\ x_2 = \frac{\pi}{2}, \\ x_3 = 0, \\ x_4 = 0 \end{cases}}$$

We form the Jacobian matrix for A, B, C, and D, and substitute the values of the operating point into it.

$$J_u = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} \\ \frac{\partial f_4}{\partial u_1} & \frac{\partial f_4}{\partial u_2} \end{bmatrix} \qquad J_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.4568 & -0.6196 & 0 & 0 \\ 0.2485 & -6.6174 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.7870 & -0.0426 \\ 0.0426 & 0.1349 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D = 0$$

```
% State space -----------------------
A = [0 0 1 0;0 0 0 1;-0.4568 -0.6196 0 0;0.2485 -6.6174 0 0];
B = [0 0;0 0;0.7870 -0.0426;0.0426 0.1349];
C = [1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];
robotarm = ss(A,B,C,0);
```

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = A \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + B \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \qquad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = C \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

---

1. Canonical Jordan form of state space matrices:

```
%Jordan canonical form matrix-------
canon(robotarm,"Modal");
```

```
A =
              x1          x2          x3          x4
   x1  -4.045e-17      0.6942           0           0
   x2     -0.6942  -4.045e-17           0           0
   x3           0           0   8.792e-17       2.568
   x4           0           0      -2.568   8.792e-17
```

```
B =
              u1          u2
   x1  -4.795e-17  -1.876e-18
   x2       1.136    -0.08159
   x3  -1.038e-17  -1.567e-17
   x4    0.008399       0.107
```

```
C =
              x1          x2          x3          x4
   y1      0.9967   4.211e-17     0.05043   5.697e-17
   y2     0.04037   5.688e-18      0.4994    7.82e-17
   y3  -6.836e-17      0.6919  -1.271e-16      0.1295
   y4  -9.911e-18     0.02802  -2.104e-16       1.282
```

2. The eigenvalues of matrix A

```
%Eigenvalues-----------
eigA_o = eig(A);
```

```
-0.0000 + 0.6942i
-0.0000 - 0.6942i
 0.0000 + 2.5675i
 0.0000 - 2.5675i
```

3. Controllability

To assess controllability, we form the controllability matrix and calculate its rank. If the rank is not full, we find the controllable and uncontrollable modes of the system and construct the controllable canonical form.

```
%Controllability--------
phi_c = ctrb(A,B);
rank(phi_c);
```

```
phi_c =

         0         0    0.7870   -0.0426         0         0   -0.3859   -0.0641
         0         0    0.0426    0.1349         0         0   -0.0863   -0.9033
    0.7870   -0.0426         0         0   -0.3859   -0.0641         0         0
    0.0426    0.1349         0         0   -0.0863   -0.9033         0         0
```

```
ans =

     4
```

Since the system rank is full, it does not have uncontrollable modes, and all modes are controllable.

4. Observability

To assess observability, we form the observability matrix and calculate its rank. If the rank is not full, we find the observable and unobservable modes of the system and construct the observable canonical form.

```
% Observability-----------
phi_o = obsv(A,C);
rank(phi_o);
```

```
phi_o =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000
         0         0    1.0000         0
         0         0         0    1.0000
   -0.4568   -0.6196         0         0
    0.2485   -6.6174         0         0
   -0.4568   -0.6196         0         0
    0.2485   -6.6174         0         0
         0         0   -0.4568   -0.6196
         0         0    0.2485   -6.6174
         0         0   -0.4568   -0.6196
         0         0    0.2485   -6.6174
    0.0547    4.3832         0         0
   -1.7579   43.6360         0         0
```

```
ans =

     4
```

Since the system rank is full, it does not have unobservable modes, and all modes are observable.

5. Realization

5.1. Canonical Controllability Realization:

```
% Realization-----------------
canon(robotarm,"Companion");
```

```
A =
        x1      x2      x3      x4
  x1     0       0       0  -3.177
  x2     1       0       0       0
  x3     0       1       0  -7.074
  x4     0       0       1       0

B =
        u1      u2
  x1     1  -1.082
  x2     0       0
  x3     0  -2.097
  x4     0       0

C =
           x1         x2         x3         x4
  y1        0      0.787          0    -0.3859
  y2        0     0.0426          0   -0.08633
  y3    0.787          0    -0.3859          0
  y4   0.0426          0   -0.08633          0
```

5.2. Canonical observability realization:

```
% Realization---------------------------
canon_cont = canon(robotarm,"Companion");
transpose(canon_cont);
```

```
A =
        x1      x2      x3      x4
  x1     0       1       0       0
  x2     0       0       1       0
  x3     0       0       0       1
  x4 -3.177      0  -7.074       0

B =
           u1         u2         u3         u4
  x1        0          0      0.787     0.0426
  x2    0.787     0.0426          0          0
  x3        0          0    -0.3859   -0.08633
  x4  -0.3859   -0.08633          0          0

C =
        x1      x2      x3      x4
  y1     1       0       0       0
  y2 -1.082      0  -2.097       0
```

By transposing the controllability realization, we can also obtain the observability realization.

6. Transfer Function

The matrix G(s) has two columns and four rows .

Four Transfer function for the first input:

```
% Transfer function
tf(robotarm);
```

Four Transfer function for the first and second input:

```
From input 2 to output...
                       -0.0426 s^2 - 0.3655
 1:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177

                       0.1349 s^2 + 0.05104
 2:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177

                      -0.0426 s^3 - 0.3655 s
 3:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177

                      0.1349 s^3 + 0.05104 s
 4:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177
```

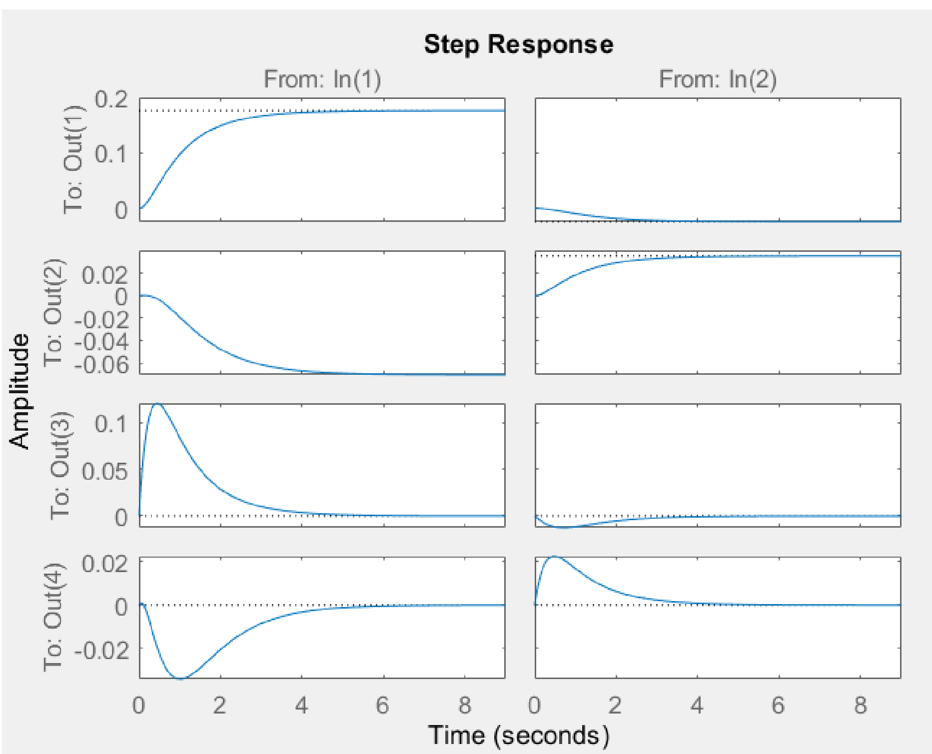
```
From input 1 to output...
                       0.787 s^2 + 5.181
 1:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177

                      0.0426 s^2 + 0.215
 2:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177

                      0.787 s^3 + 5.181 s
 3:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177

                     0.0426 s^3 + 0.215 s
 4:  -----------------------------------------------------
     s^4 - 9.494e-17 s^3 + 7.074 s^2 + 4.486e-16 s + 3.177
```

7. Designing state feedback

- **Step response of the system in the open-loop configuration :**

  This response is completely unstable and unreliable.

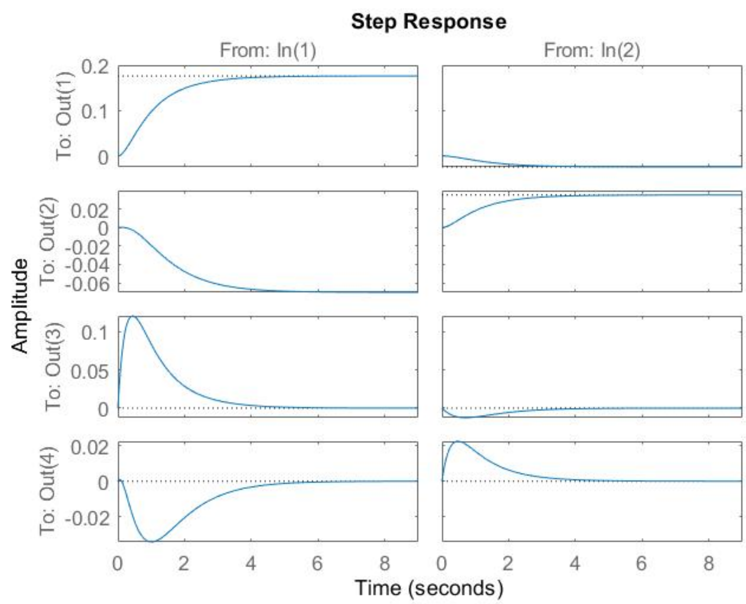
```
%% Step response of the open loop system---
figure(1)
step(robotarm)
% step(tf1)
```



We can employ state feedback and Pole assignment to stabilize this system and then analyze their responses. Once, we choose poles near the $j\omega$ axis, and once away from it. Considering that all modes of this system are controllable, we can stabilize all of them with state feedback.

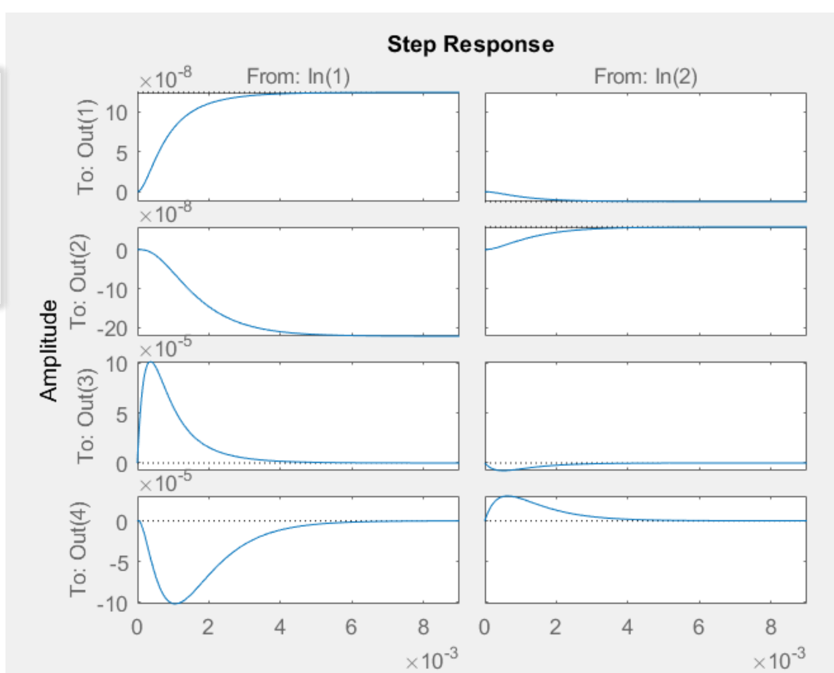
- **Near $j\omega$ Axis**:

```
%Step response of the closed loop system-
%Pole assignment
p = [-1,-2,-3,-4]; %near to zero
k = place(A,B,p);
Acl = A-B*k;
syscl = ss(Acl,B,C,0);
figure(2)
step(syscl)
```

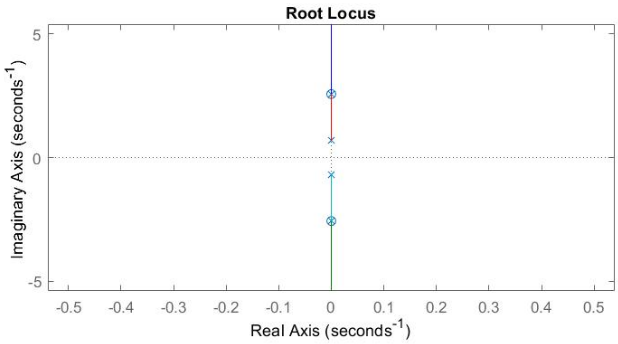


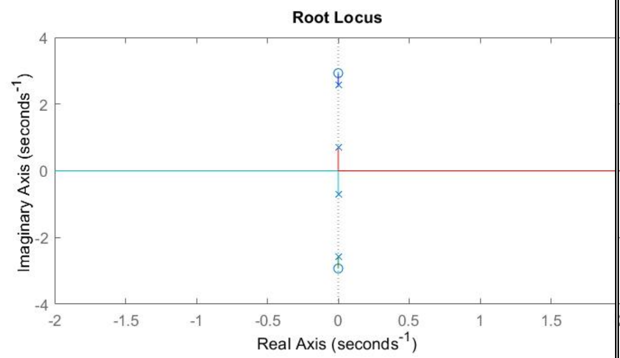
- **Far from $j\omega$ Axis**:
  The further we place the poles away from the imaginary axis towards negative infinity, the faster our response becomes stable.

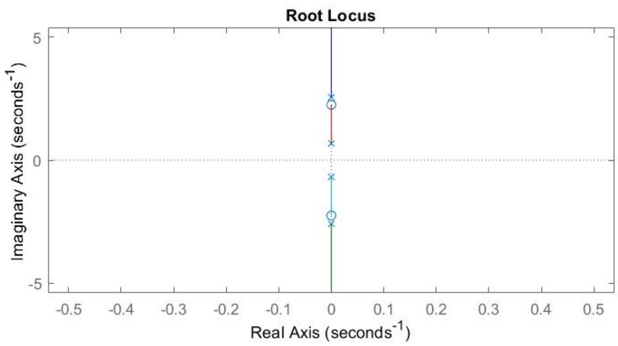
```
p = 1000*[-1,-2,-3,-4]; %far from zero
k = place(A,B,p);
Acl = A-B*k;
syscl = ss(Acl,B,C,0);
figure(3)
step(syscl)
```

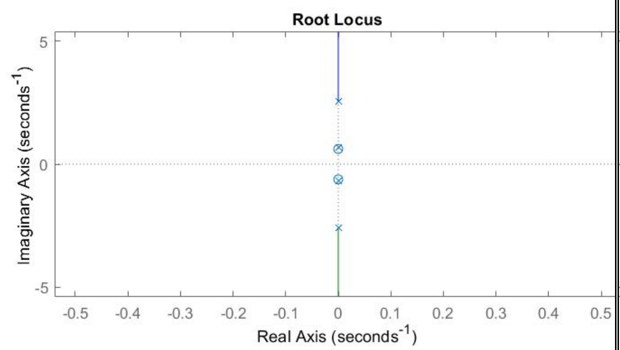


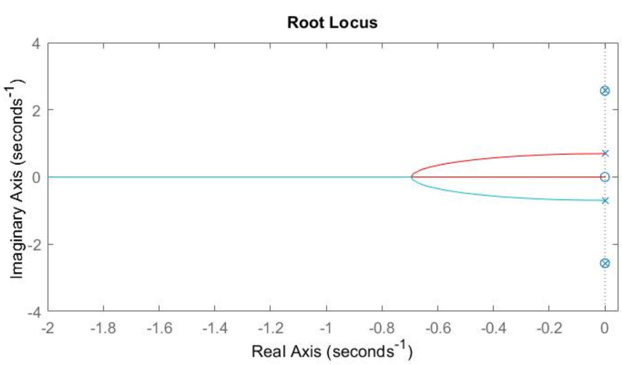
8. Root Locus

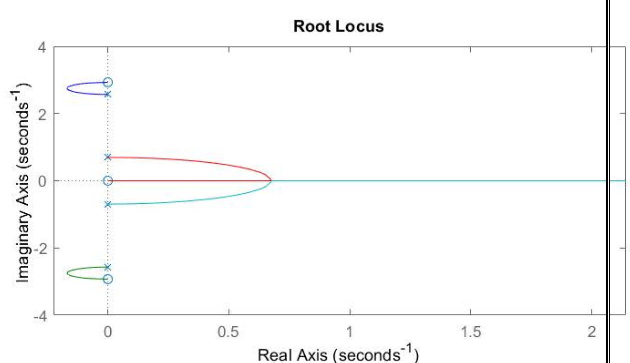

Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

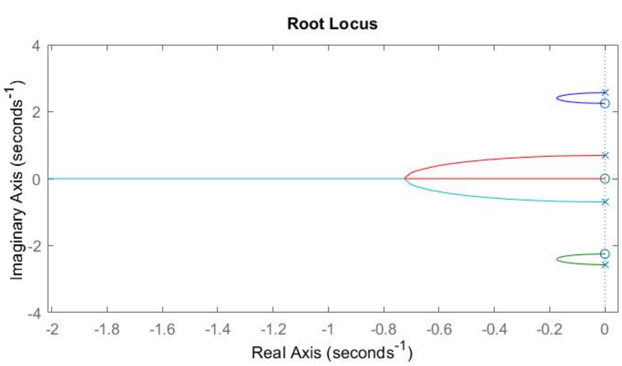

Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

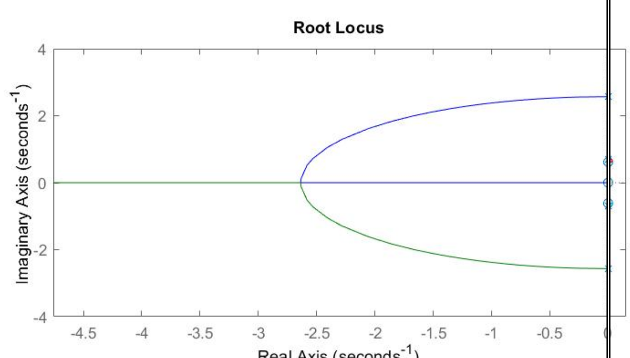

Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

9. Observer Design

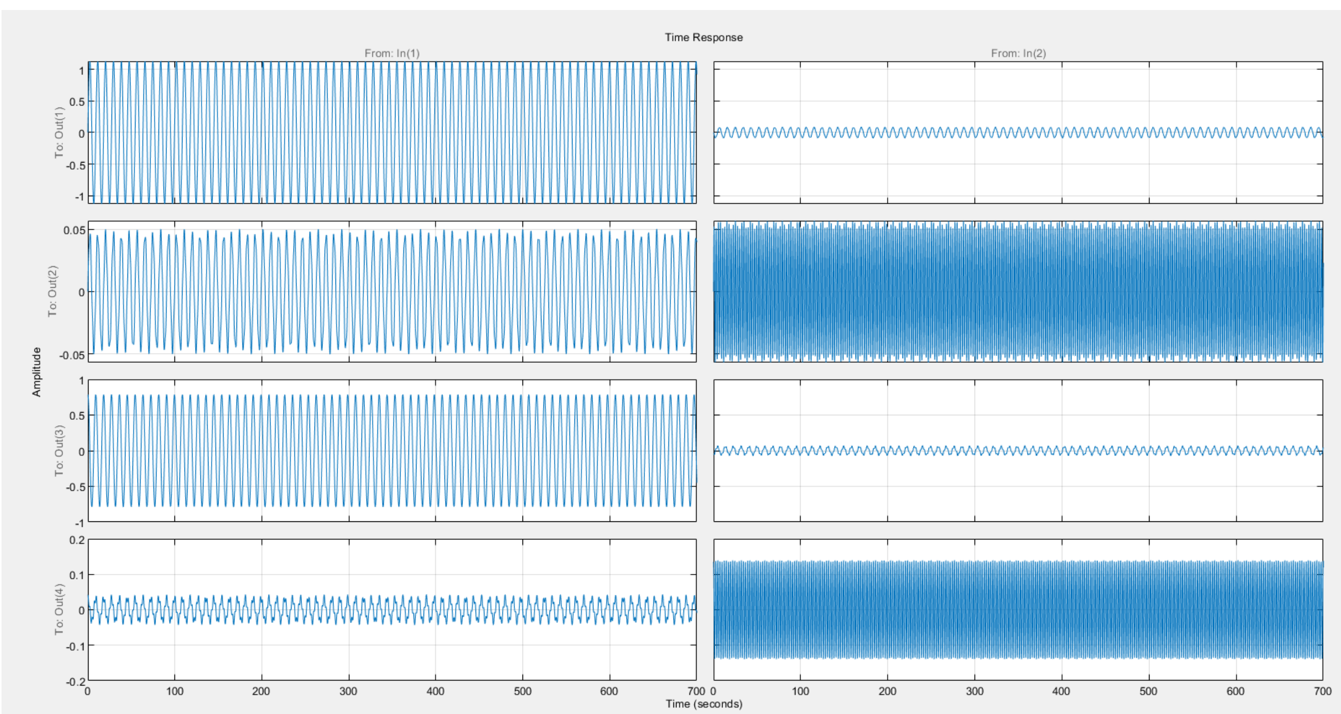
```
sysObserver =

  A =
       x1  x2  x3  x4
   x1  -1   0   0   0
   x2   0  -2   0   0
   x3   0   0  -3   0
   x4   0   0   0  -4

  B =
           u1       u2       u3       u4       u5       u6
   x1       0        0        1        0        1        0
   x2       0        0        0        2        0        1
   x3   0.787  -0.0426  -0.4568  -0.6196        3        0
   x4  0.0426   0.1349   0.2485   -6.617        0        4

  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
   y5   1   0   0   0
   y6   0   1   0   0
   y7   0   0   1   0
   y8   0   0   0   1
```

```
%Observer designing
p3 = [-1, -2, -3, -4];
L = place(A',C',p3)';
A_obs = A-L*C;
B_obs = [B, L];
C_obs = [C;eye(4)];
sysObserver = ss(A_obs,B_obs,C_obs,0);
```

10. Time domain design

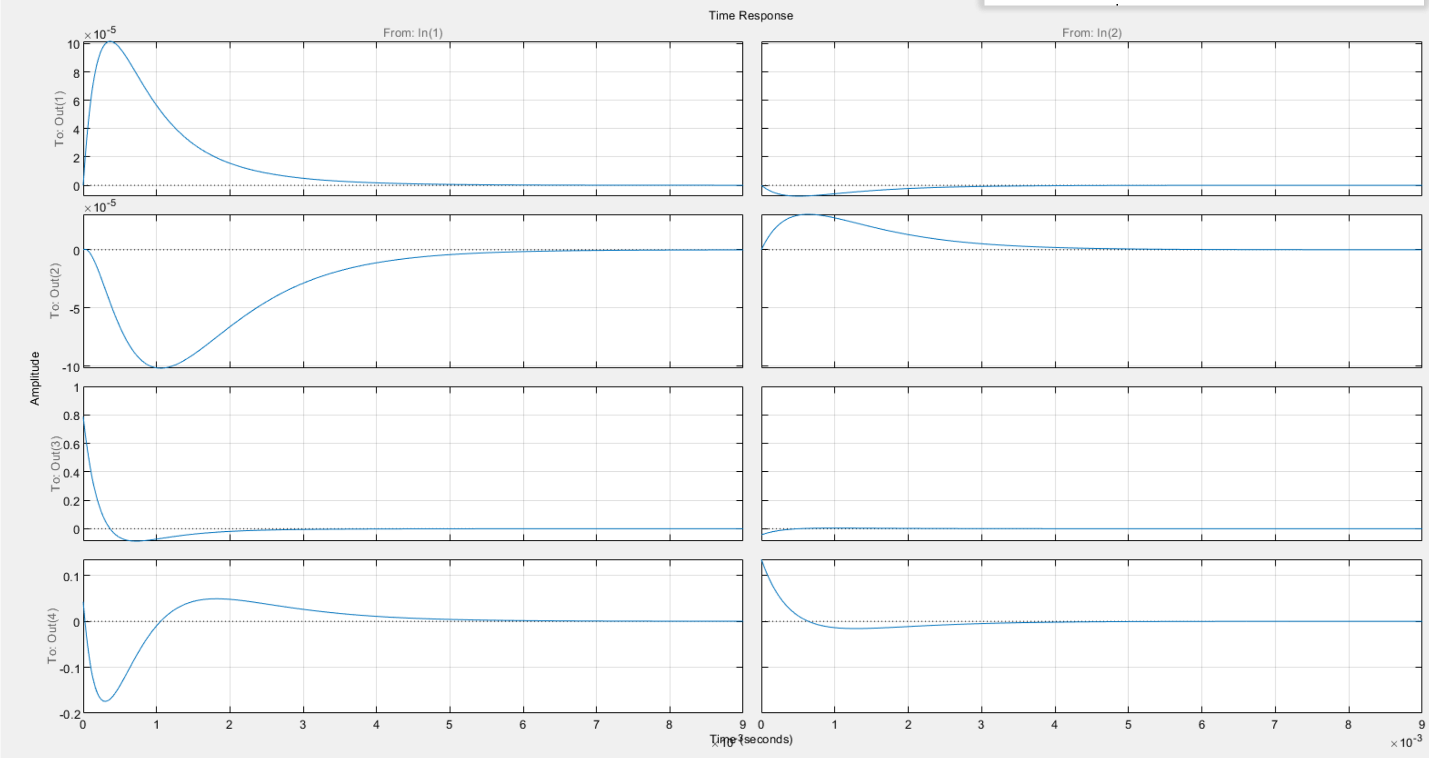
10.1. Open-Loop Time Domain Response

```
%Time response
%openloop
[y,t] = step(tf_o,8);
opts = timeoptions;
opts.Grid = 'on';
impulseplot(tf_o,opts);
```



10.2. Time response of the closed-loop system with poles away from the origin

```
%closedloop
[y,t] = step(syscl,8);
opts = timeoptions;
opts.Grid = 'on';
impulseplot(syscl,opts);
```



11. System parameters

```
%% Wn, Zeta, poles, Tr, Ts, Os, Tp
[wn,zeta] = damp(syscl);
params = stepinfo(syscl);
figure(5);
step(syscl)
stepinfo(syscl)
params(1,1)
params(2,1)
params(3,1)
params(4,1)
params(1,2)
params(2,2)
params(3,2)
params(4,2)
```

```
        RiseTime: 0.0019
   TransientTime: 0.0036
    SettlingTime: 0.0036
     SettlingMin: 1.1190e-07
     SettlingMax: 1.2411e-07
       Overshoot: 0
      Undershoot: 0
            Peak: 1.2411e-07
        PeakTime: 0.0106
```

```
        RiseTime: 0.0027
   TransientTime: 0.0049
    SettlingTime: 0.0049
     SettlingMin: -2.2189e-07
     SettlingMax: -1.9978e-07
       Overshoot: 0
      Undershoot: 0.0072
            Peak: 2.2189e-07
        PeakTime: 0.0106
```

```
        RiseTime: 0
   TransientTime: 0.0038
    SettlingTime: NaN
     SettlingMin: 0
     SettlingMax: 1.0159e-04
       Overshoot: Inf
      Undershoot: 0
            Peak: 1.0159e-04
        PeakTime: 3.6841e-04
```

```
        RiseTime: 0
   TransientTime: 0.0057
    SettlingTime: NaN
     SettlingMin: -1.0165e-04
     SettlingMax: 4.9229e-07
       Overshoot: Inf
      Undershoot: Inf
            Peak: 1.0165e-04
        PeakTime: 0.0011
```

```
      RiseTime: 0.0023
 TransientTime: 0.0042
  SettlingTime: 0.0042
   SettlingMin: -1.2347e-08
   SettlingMax: -1.1129e-08
     Overshoot: 0
    Undershoot: 0
          Peak: 1.2347e-08
      PeakTime: 0.0106
```

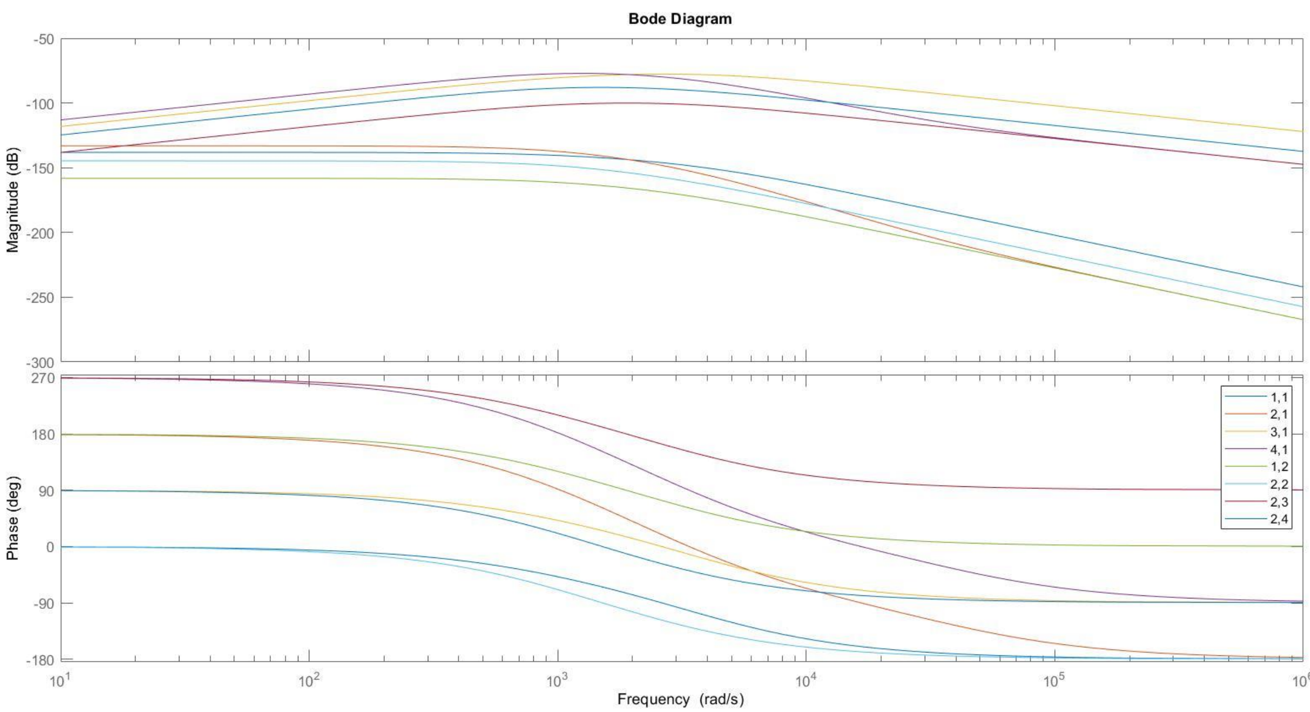
```
      RiseTime: 0.0025
 TransientTime: 0.0045
  SettlingTime: 0.0045
   SettlingMin: 5.2538e-08
   SettlingMax: 5.8323e-08
     Overshoot: 0
    Undershoot: 0
          Peak: 5.8323e-08
      PeakTime: 0.0106
```

```
      RiseTime: 0
 TransientTime: 0.0046
  SettlingTime: NaN
   SettlingMin: -7.6538e-06
   SettlingMax: 0
     Overshoot: Inf
    Undershoot: Inf
          Peak: 7.6538e-06
      PeakTime: 5.0657e-04
```

```
      RiseTime: 0
 TransientTime: 0.0052
  SettlingTime: NaN
   SettlingMin: 0
   SettlingMax: 3.0394e-05
     Overshoot: Inf
    Undershoot: 0
          Peak: 3.0394e-05
      PeakTime: 6.4472e-04
```

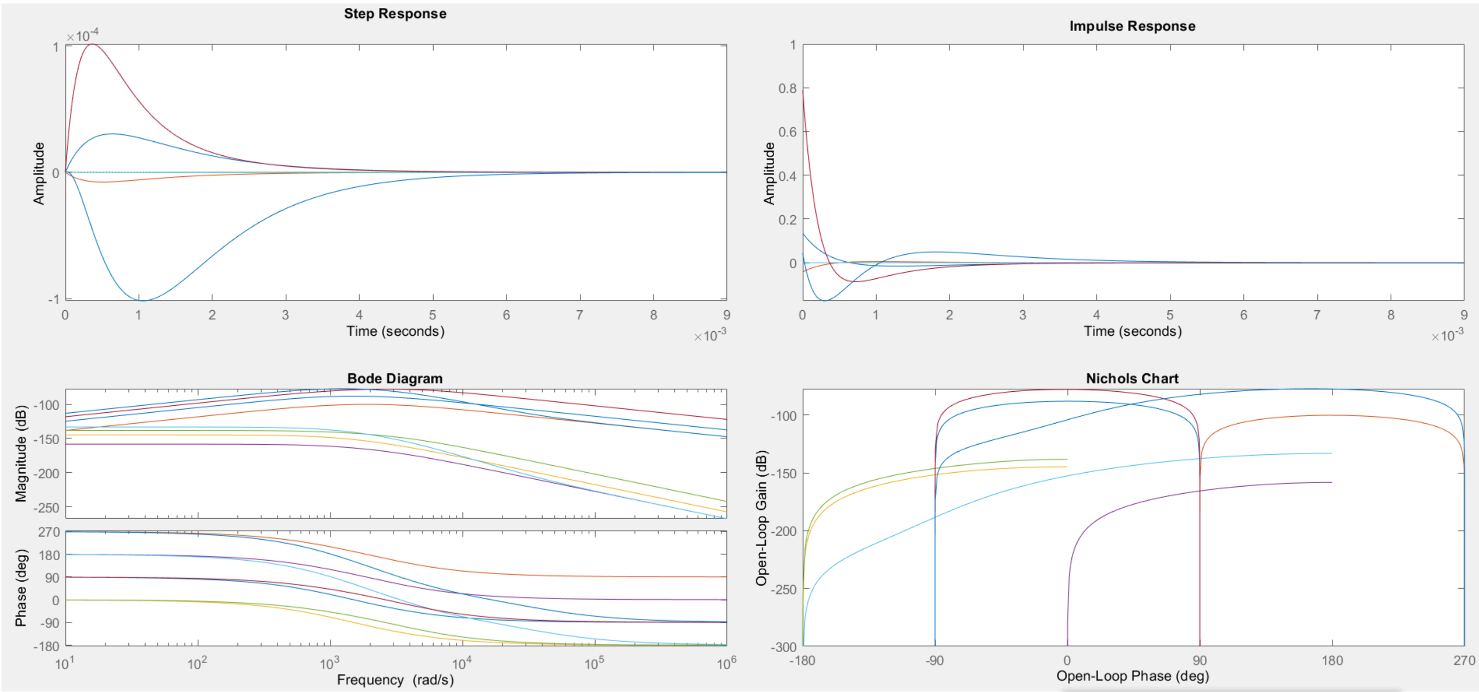
12. :Bode Diagram

```
%% Bode Diagram

for i = 1:1:2
    for j = 1:1:4
        bode(syscl(j,i));
        legend('1,1','2,1','3,1','4,1','1,2','2,2','2,3','2,4');
        hold on;
    end
end
```



13. All Diagrams in one plot:

```
%% all diagrams together
linearSystemAnalyzer({'step';'impulse';'bode';'nichols'},sysc1(4,2),sysc1(3,2),sysc1(2,2),sysc1(1,2),sysc1(1,1),sysc1(2,1),sysc1(3,1),sysc1(4,1))
```

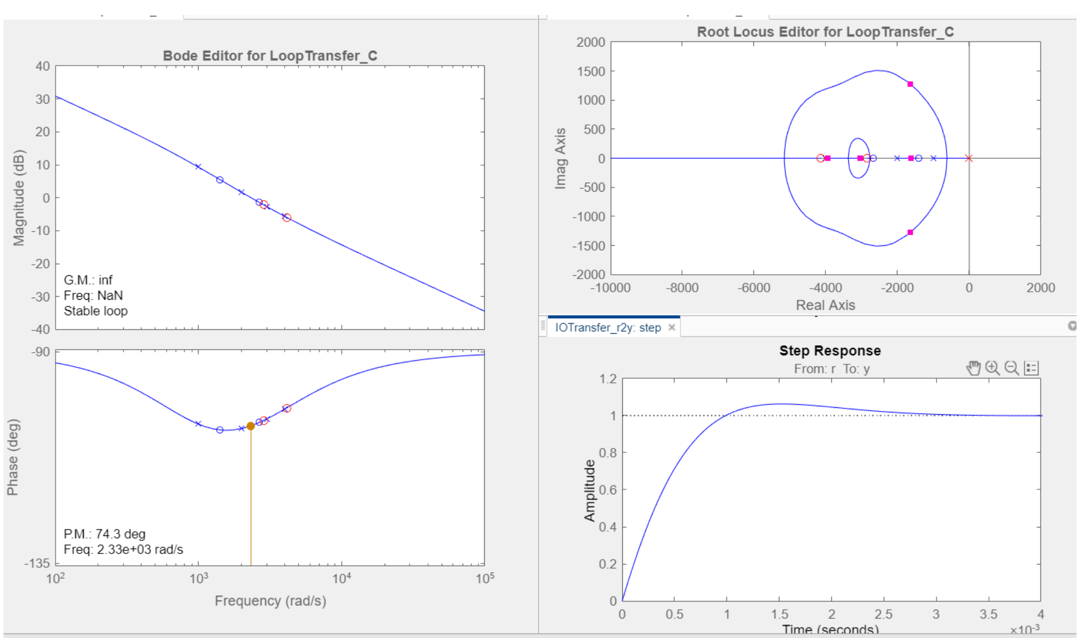


14. Sisotool:

```
%% Sisotool
sisotool(sysc1(4,2))
sisotool(sysc1(3,2))
sisotool(sysc1(2,2))
sisotool(sysc1(1,2))
sisotool(sysc1(4,1))
sisotool(sysc1(3,1))
sisotool(sysc1(2,1))
sisotool(sysc1(1,1))
```

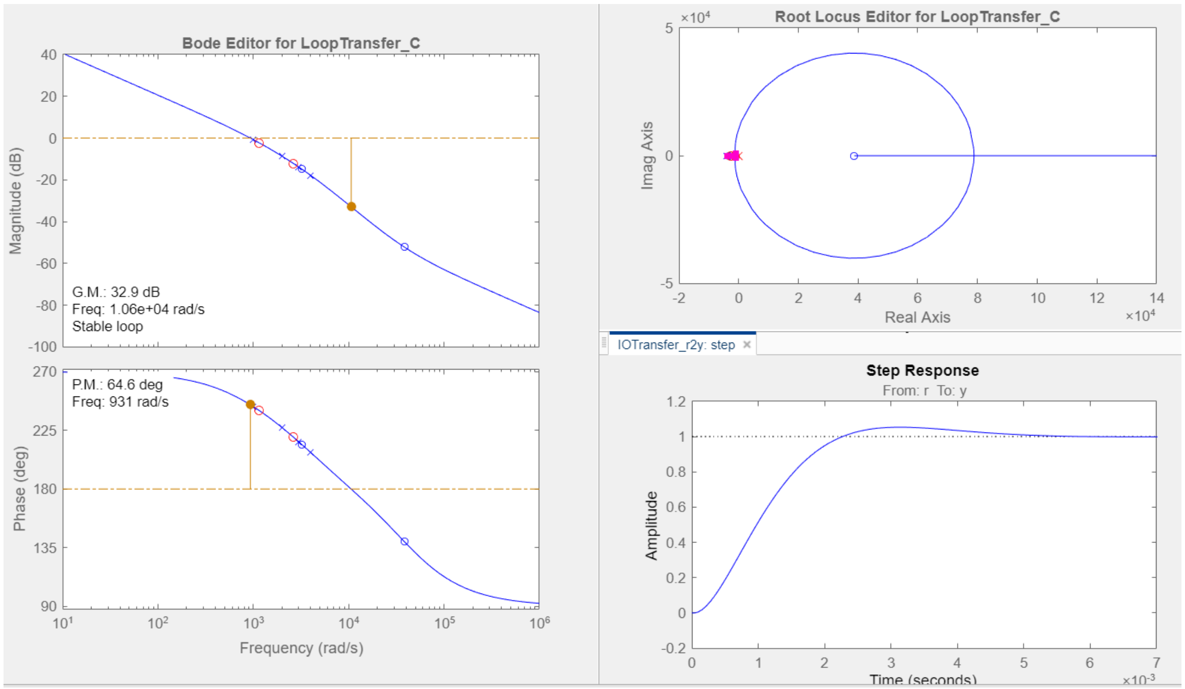
14.1. PID Controller (System 1)

```
with Kp = 1.67e+07, Ki = 2.83e+10, Kd = 2.38e+03
```



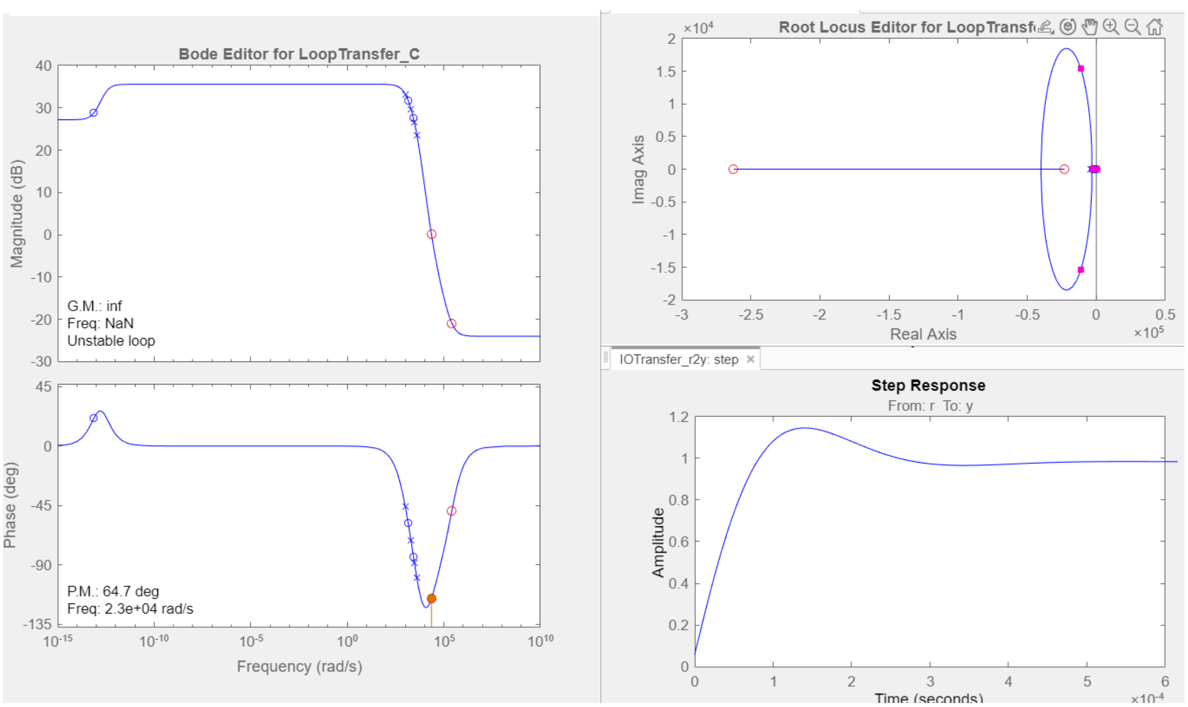
14.2. PID Controller (System 2)

```
             1
Kp + Ki * --- + Kd * s
             s

with Kp = -5.95e+06, Ki = -4.8e+09, Kd = -1.57e+03
```



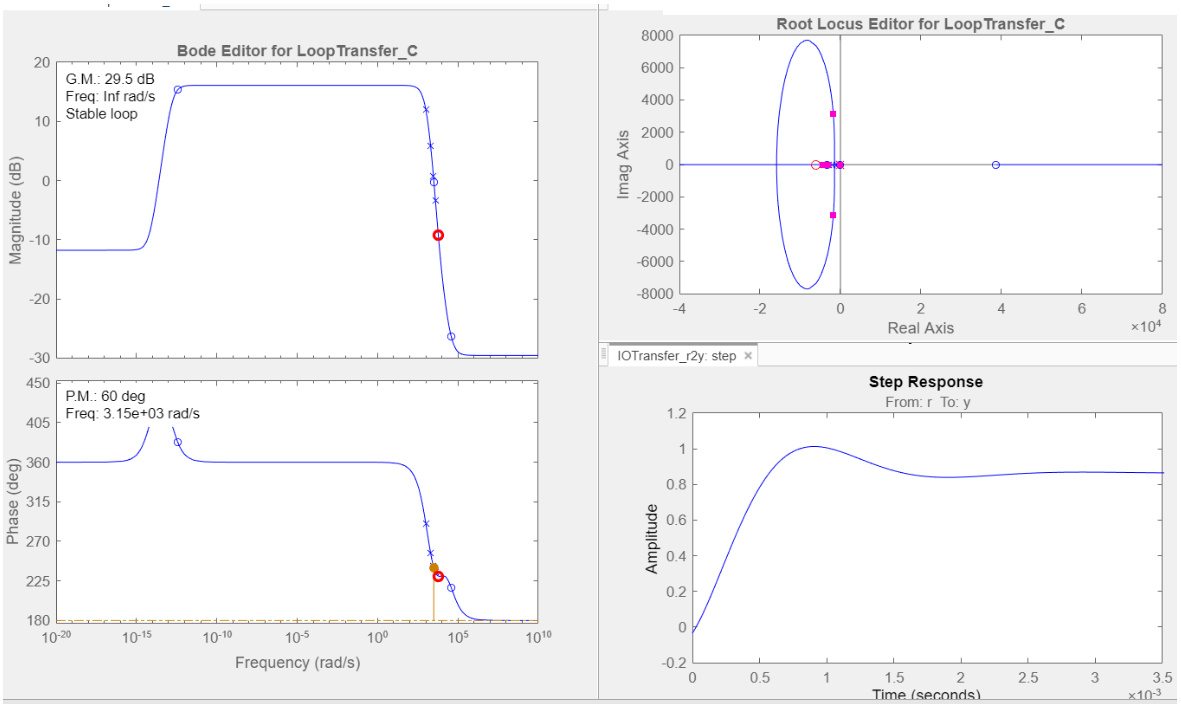
14.3. PID Controller (System 3)

```
             1
Kp + Ki * --- + Kd * s
             s

with Kp = 2.31e+04, Ki = 4.83e+08, Kd = 0.0807
```



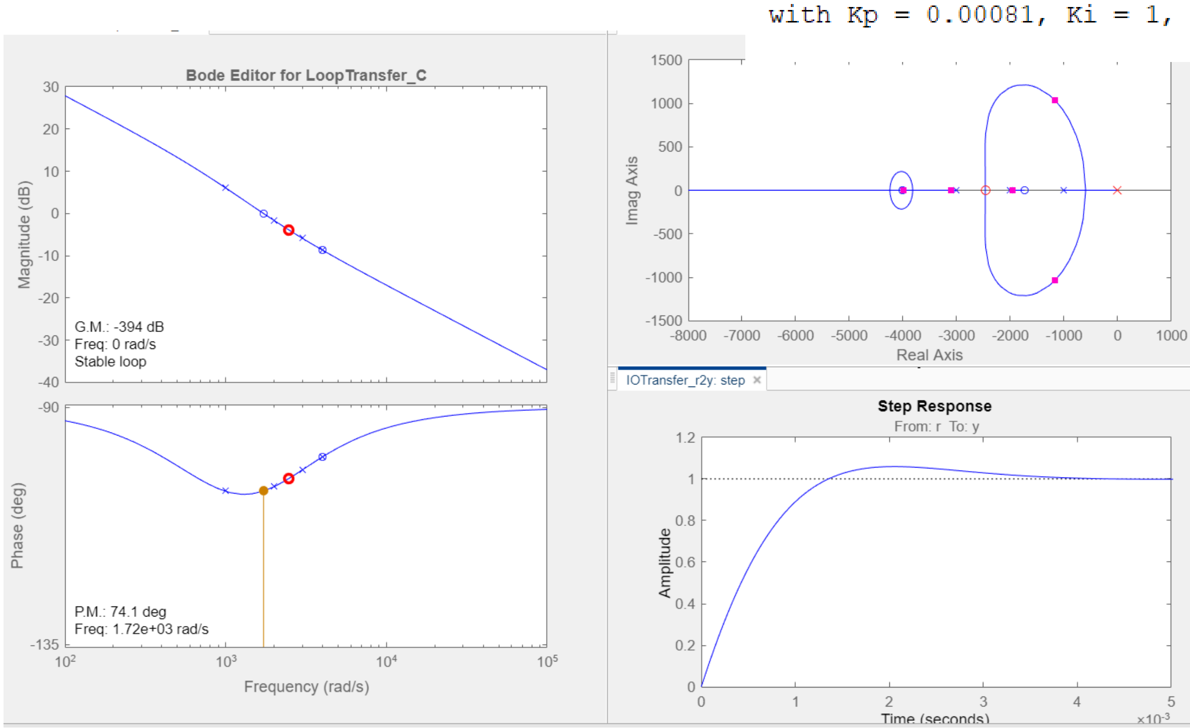
14.4. PID Controller (System 4)

```
             1
Kp + Ki * --- + Kd * s
             s

with Kp = -9.49e+03, Ki = -2.87e+07, Kd = -0.831
```



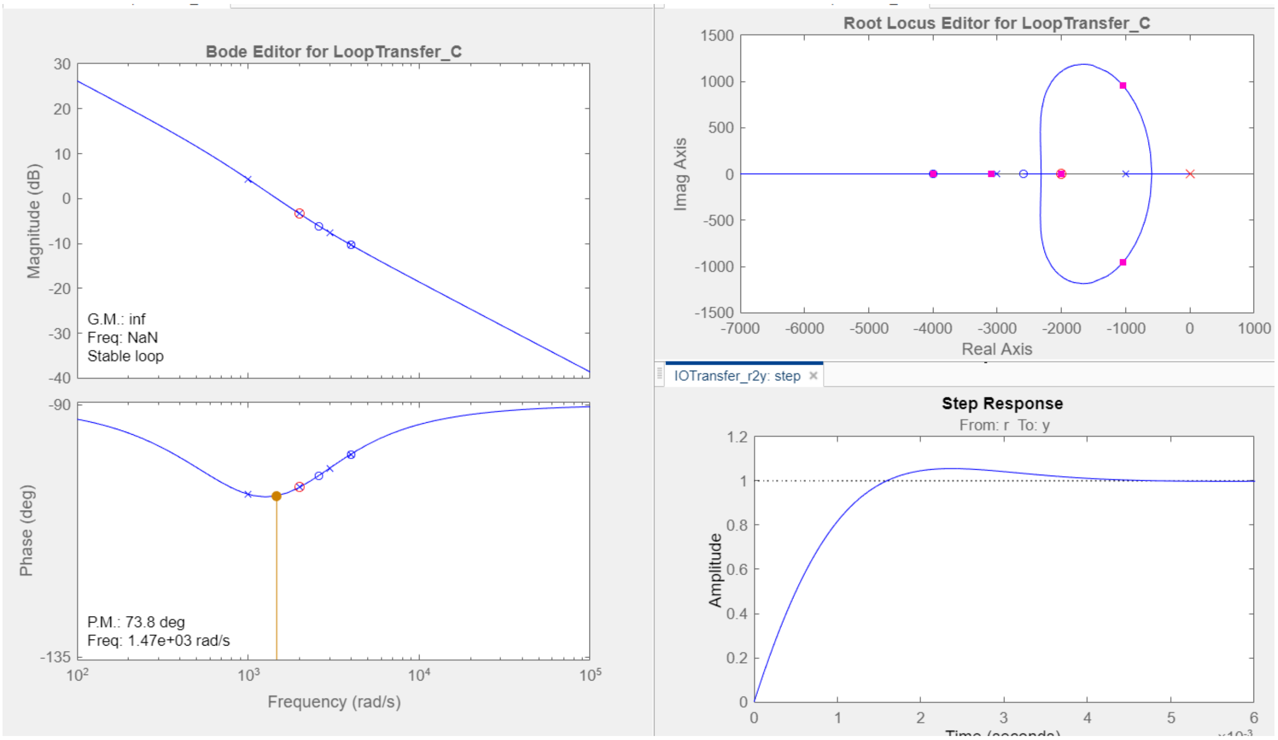
14.5. PID Controller (System 5)

```
             1
Kp + Ki * --- + Kd * s
             s

with Kp = 0.00081, Ki = 1, Kd = 1.68e-07
```



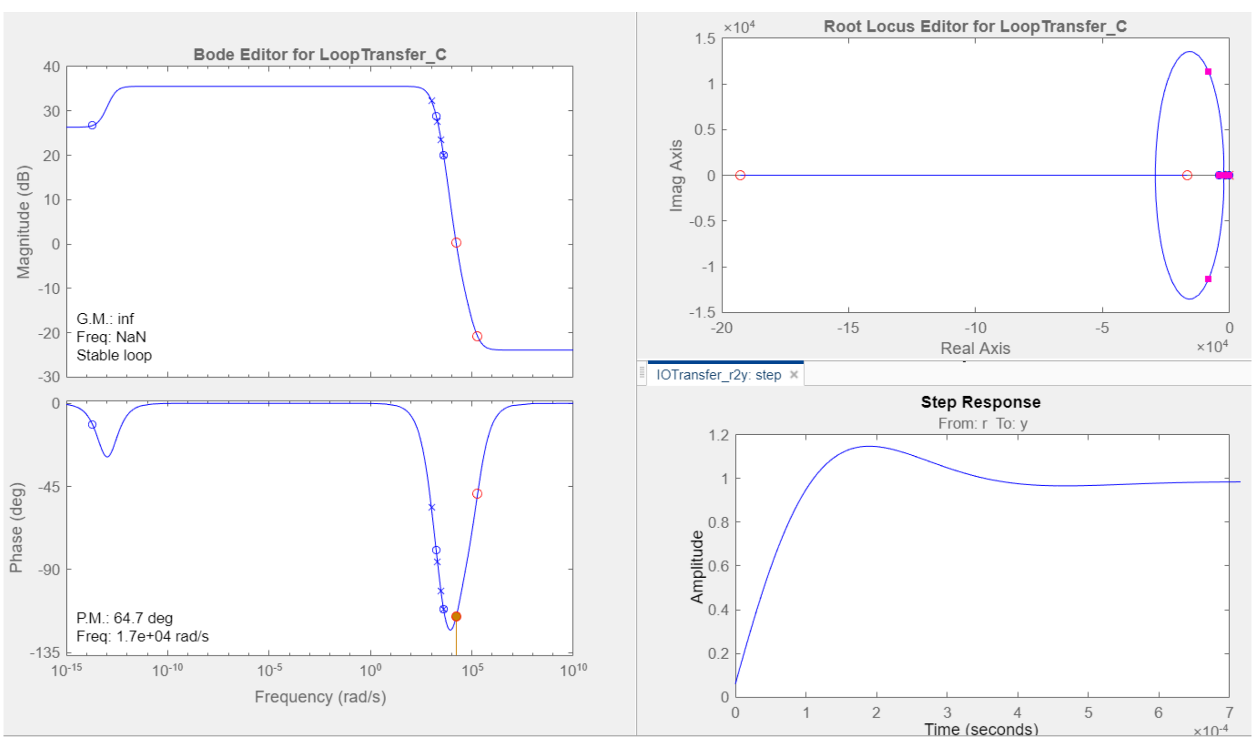
14.6. PID Controller (System 6)

```
             1
  Kp + Ki * --- + Kd * s
             s

  with Kp = 3.52, Ki = 10, Kd = 2.5e-06
```



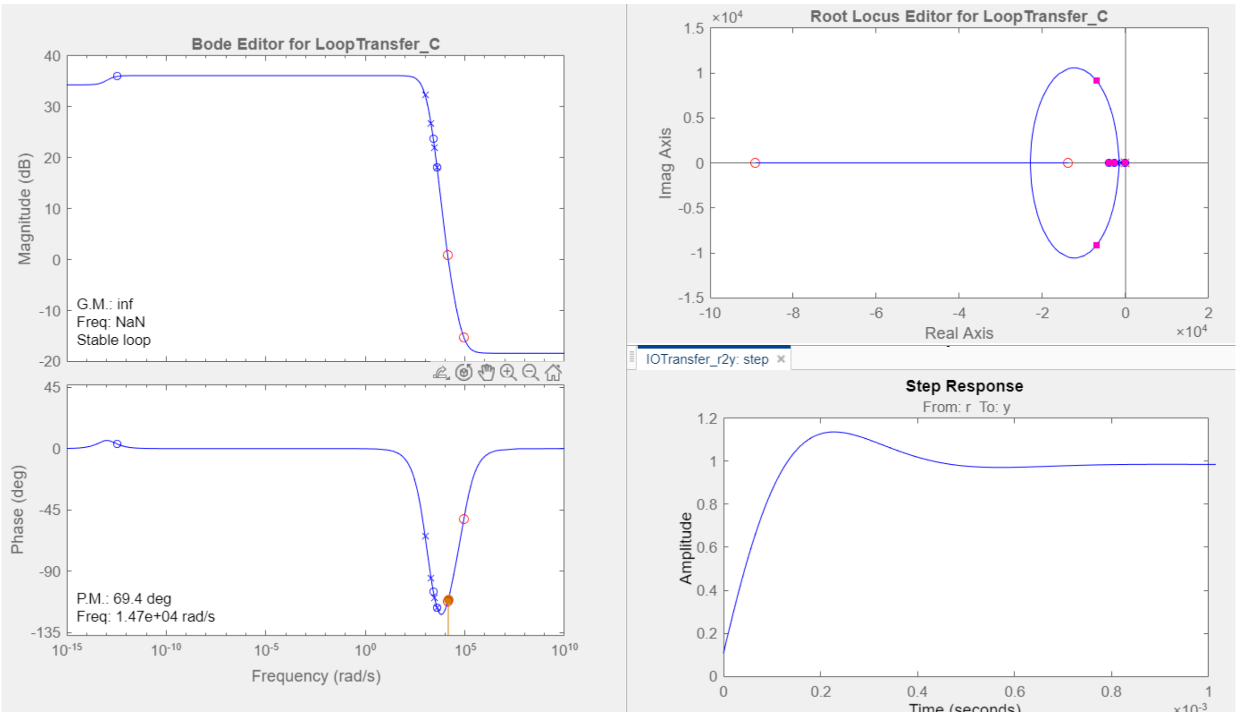
14.7. PID Controller (System 7)

```
           1
Kp + Ki * --- + Kd * s
           s

with Kp = -3.16e+05, Ki = -4.85e+09, Kd = -1.51
```

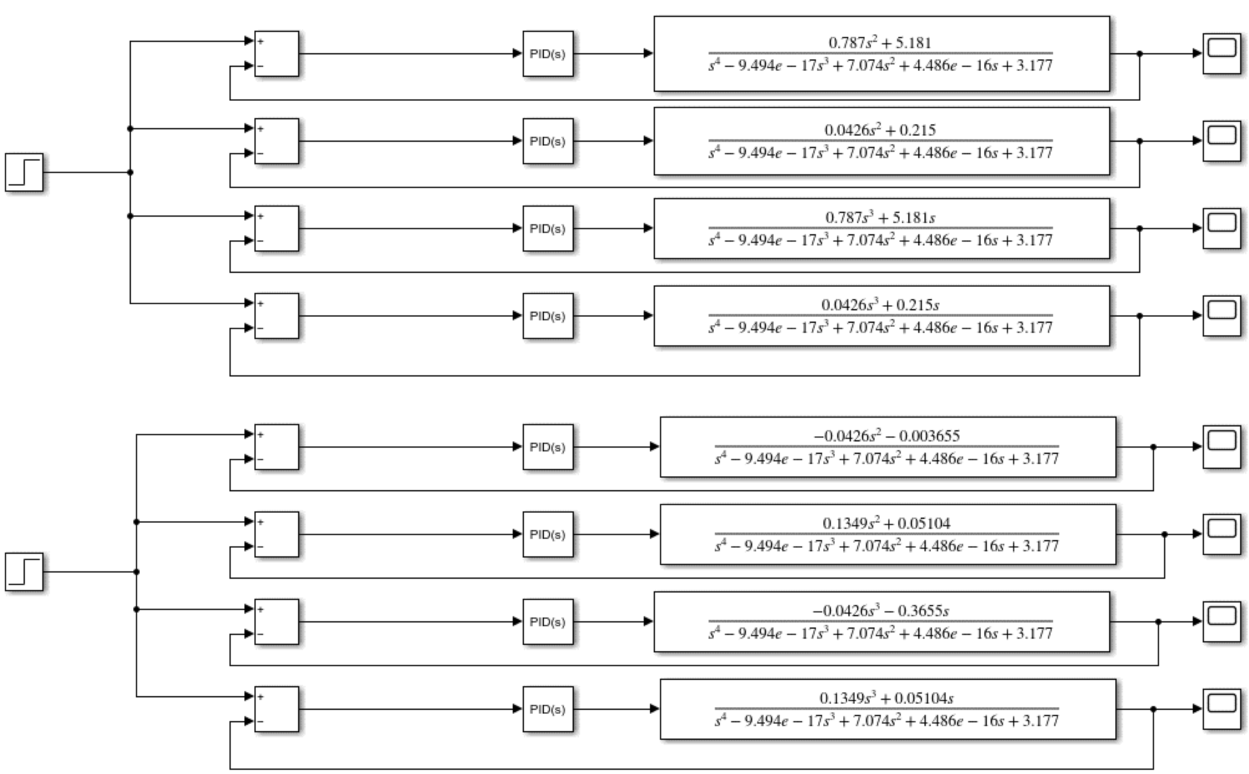


14.8. PID Controller (System 8)

```
             1
Kp + Ki * --- + Kd * s
             s

with Kp = 9.22e+04, Ki = 1.1e+09, Kd = 0.881
```

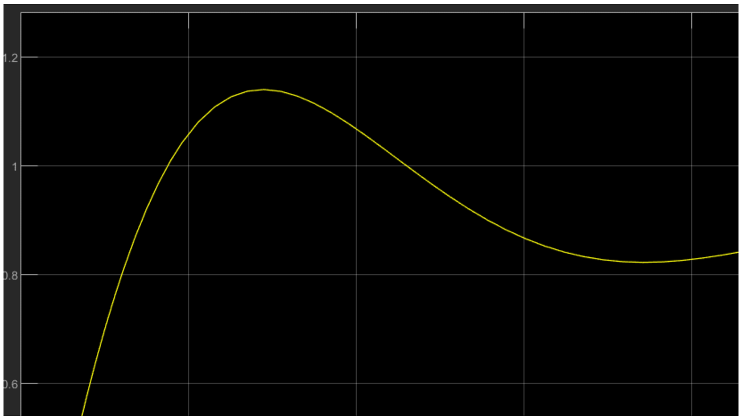


Simulink .15

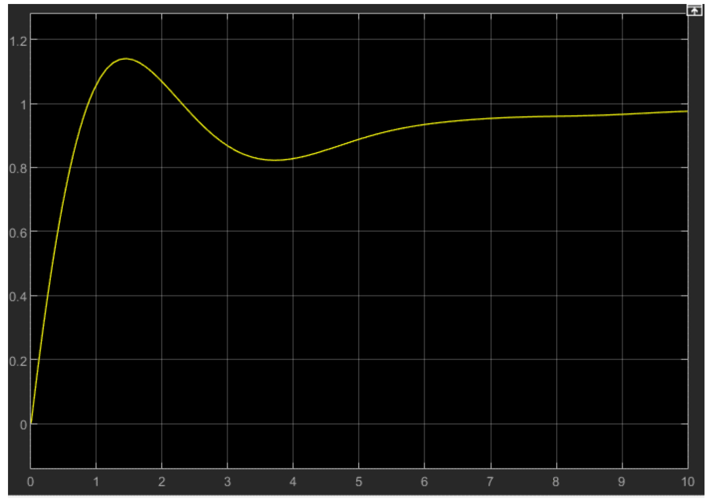


For each of the inputs, we have 4 parallel systems for which we close-loop with unity feedback and design a PID controller for each transfer function with coefficients obtained from the sisotool section. The output of the systems for the input "strp" is as follows:

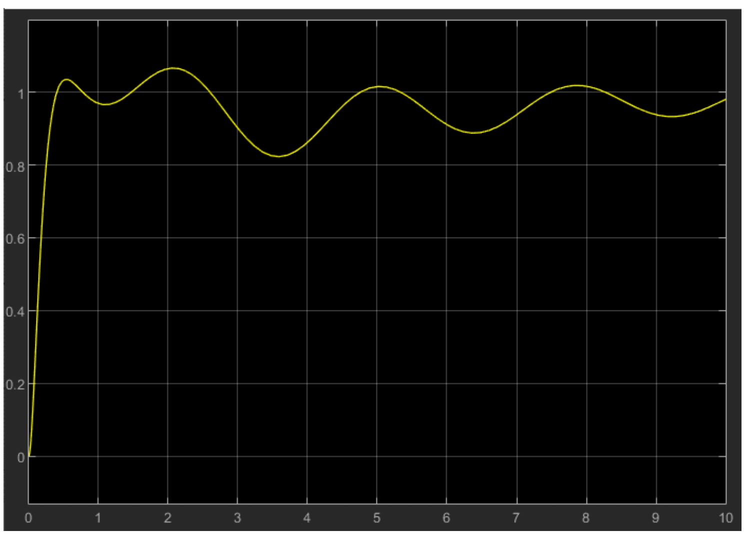
Sys1:



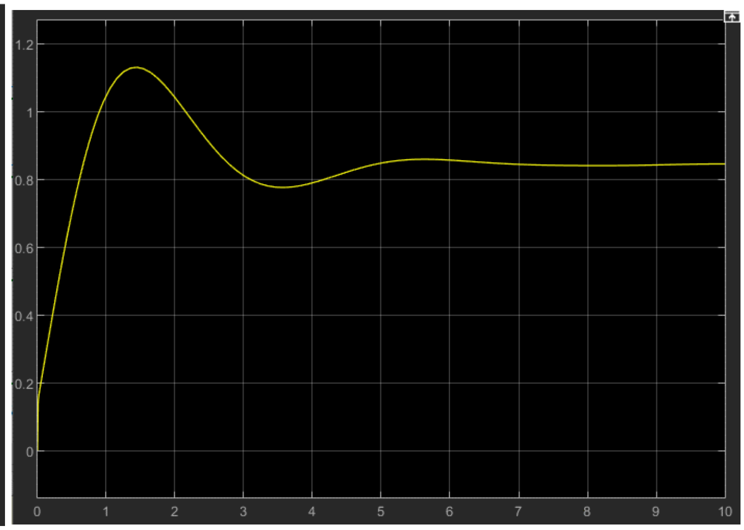
Sys2:



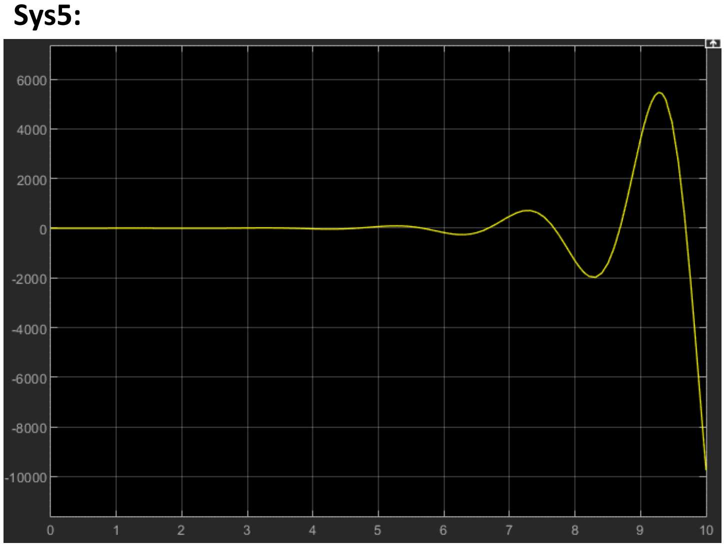
Sys3:



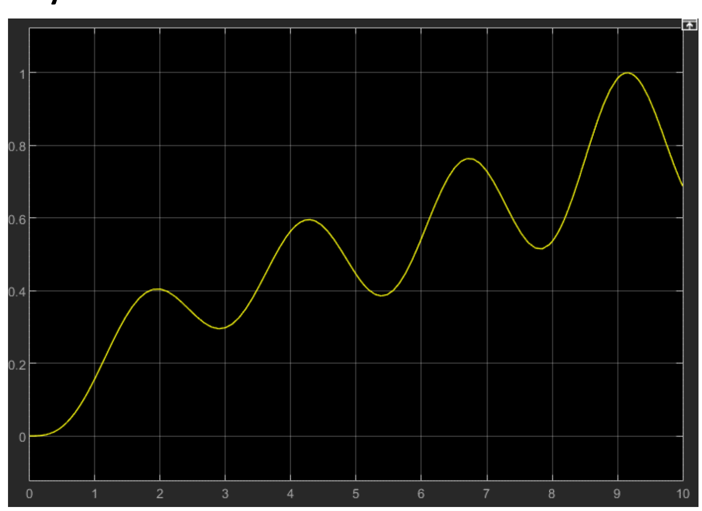
Sys4:



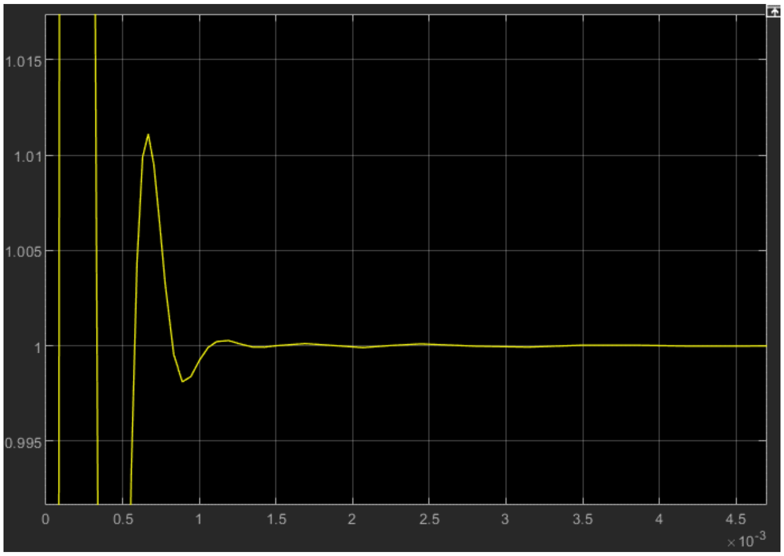
Sys5:



Sys6:

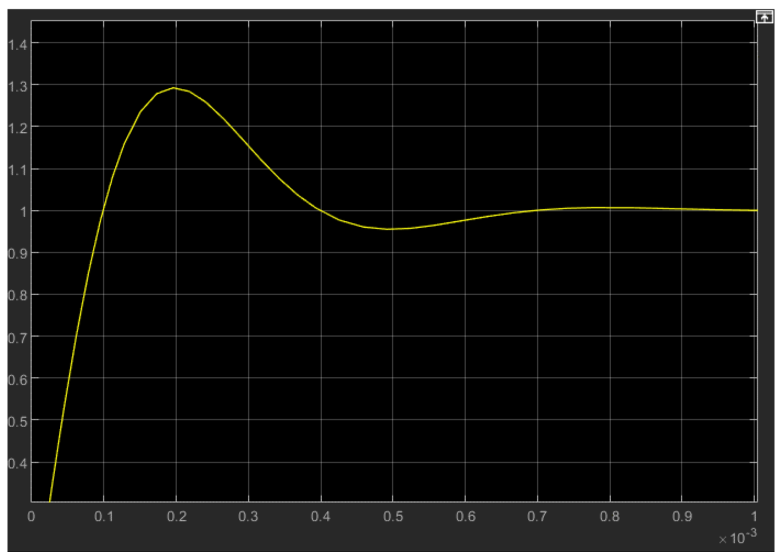


Sys7:



1.015
1.01
1.005
1
0.995
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
× 10^-3

:Sys8



1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
× 10^-3