

$$\begin{cases} \dot{x}_p = \begin{bmatrix} 3 & 1 \\ -2 & 2 \end{bmatrix} x_p + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w \\ y = \begin{bmatrix} 3 & 1 \end{bmatrix} x_p + u + 2w \\ z = \begin{bmatrix} 1 & 0 \end{bmatrix} x_p + 2w \end{cases} \quad - (1)$$

$$\begin{cases} \dot{x}_c = -4x_c + 2z \\ u = x_c - 2z \end{cases} \quad - (2)$$

Closed loop system equations:

$$\dot{x}_{cl} = A_{cl} x_{cl} + B_{cl} w \quad - (3)$$

$$y = C_{cl} x_{cl} + D_{cl} w \quad - (4)$$

We know that:

$$\text{System } S: \begin{cases} \dot{x}_p = A_p x_p + B_p u + D_p w \\ y = C_p x_p + B_y u + D_y w \\ z = M_p x_p + \cancel{B_z}^0 u + D_z w \end{cases} \quad - (5)$$

$$\text{Controller } C: \begin{cases} \dot{x}_c = A_c x_c + B_c z \\ u = C_c x_c + D_c z \end{cases} \quad - (6)$$

Comparing (1) & (2) with (5) and (6),

$$\left. \begin{array}{lll} A_p = \begin{bmatrix} 3 & 1 \\ -2 & 2 \end{bmatrix} & B_p = \begin{bmatrix} 0 \\ 1 \end{bmatrix} & D_p = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ C_p = \begin{bmatrix} 3 & 1 \end{bmatrix} & B_y = 1 & D_y = 2 \\ M_p = \begin{bmatrix} 1 & 0 \end{bmatrix} & B_z = 0 & D_z = 2 \\ A_c = -4 & B_c = 2 & \\ C_c = 1 & D_c = -2 & \end{array} \right\} \quad - (7)$$

$$\left. \begin{aligned}
 A_{cl} &= \begin{bmatrix} A_p + B_p D_c M_p & B_p C_c \\ B_c M_p & A_c \end{bmatrix} \\
 B_{cl} &= \begin{bmatrix} D_p + B_p D_c D_z \\ B_c D_z \end{bmatrix} \\
 C_{cl} &= \begin{bmatrix} C_p + B_p D_c M_p & B_p C_c \end{bmatrix} \\
 D_{cl} &= D_y + B_p D_c D_z
 \end{aligned} \right\} \text{--- (8)}$$

Substitute (7) in (8) to find (3) and (4)

To find H_∞ norm:

min γ

$$\text{s.t.} \quad \begin{bmatrix} PA + A^T P & PB & C^T \\ B^T P & -\gamma I & D^T \\ C & D & -\gamma I \end{bmatrix}$$

$$P > 0$$

↓

Solve (Python)

↓
 γ^*

H_∞ norm of system
(energy to energy gain $\|T_{cc}\|$)
(induced L_2 gain) $= \gamma^*$

Python Code:

```
import numpy as np
import control as ctrl
import cvxpy as cp

# Define plant matrices
Ap = np.array([[3, 1], [-2, 2]])
Bp = np.array([[0], [1]])
Dp = np.array([[0], [1]])
Cp = np.array([[3, 1]])
By = np.array([[1]])
Dy = np.array([[2]])
Mp = np.array([[1, 0]])
Bz = np.array([[0]])
Dz = np.array([[2]])

# Define controller matrices
Ac = np.array([[ -4]])
Bc = np.array([[2]])
Cc = np.array([[1]])
Dc = np.array([[ -2]])

# Compute closed-loop matrices
Acl = np.vstack((np.hstack((Ap+(Bp@Dc@Mp), Bp@Cc)),
                    np.hstack((Bc@Mp, Ac))))
Bcl = np.vstack((Dp+(Bp@Dc@Dz),
                    Bc@Dz))
Ccl = np.hstack((Cp+(By@Dc@Mp), By@Cc))
Dcl = Dy+(By@Dc@Dz)

# Display closed-loop matrices
print('Closed-loop system matrices:\n')
print("Acl:")
print(Acl)
print("\nBcl:")
print(Bcl)
print("\nCcl:")
print(Ccl)
print("\nDcl:")
print(Dcl)

# Convert to state space form
sys_cl = ctrl.ss(Acl, Bcl, Ccl, Dcl)
```

```

# Check stability
eigenvalues = np.linalg.eigvals(Acl)
if all(np.real(eig) < 0 for eig in eigenvalues):
    print("\nThe closed-loop system is stable")
else:
    print("\nThe closed-loop system is unstable")

# Calculate H $\infty$  norm
P = cp.Variable((3, 3), symmetric=True)
gamma = cp.Variable(1)
M11 = P@Acl + Acl.T@P
M12 = P@Bcl
M13 = Ccl.T
M21 = Bcl.T@P
M22 = cp.multiply(-gamma, np.eye(1))
M23 = Dcl.T
M31 = Ccl
M32 = Dcl
M33 = cp.multiply(-gamma, np.eye(1))
# LMI Problem
LMI = cp.vstack([
    cp.hstack([M11[0][0], M11[0][1], M11[0][2], M12[0][0], M13[0][0]]),
    cp.hstack([M11[1][0], M11[1][1], M11[1][2], M12[1][0], M13[1][0]]),
    cp.hstack([M11[2][0], M11[2][1], M11[2][2], M12[2][0], M13[2][0]]),
    cp.hstack([M21[0][0], M21[0][1], M21[0][2], M22[0], M23[0][0]]),
    cp.hstack([M31[0][0], M31[0][1], M31[0][2], M32[0][0], M33[0]]])
])
constraints = [LMI << 0, P >> 0]
# Set up the optimization problem
objective = cp.Minimize(gamma)
problem = cp.Problem(objective, constraints)
# Solve the LMI problem
problem.solve()

if problem.status == 'optimal':
    # Get the value of optimal gamma
    gamma_star = gamma.value[0]
    hinfinity_norm = gamma_star
else:
    hinfinity_norm = np.inf
print(f"\nThe H $\infty$  norm of the closed-loop system is: {hinfinity_norm:.4f}")

```

Output:

Closed-loop system matrices:

Acl:

$\begin{bmatrix} 3 & 1 & 0 \end{bmatrix}$

$\begin{bmatrix} -4 & 2 & 1 \end{bmatrix}$

$\begin{bmatrix} 2 & 0 & -4 \end{bmatrix}$

Bcl:

$\begin{bmatrix} 0 \end{bmatrix}$

$\begin{bmatrix} -3 \end{bmatrix}$

$\begin{bmatrix} 4 \end{bmatrix}$

Ccl:

$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$

Dcl:

$\begin{bmatrix} -2 \end{bmatrix}$

The closed-loop system is unstable

The H^∞ norm of the closed-loop system is: inf

Screenshot:

```
File Edit Selection View Go Run Terminal Help
C:\Users\csamak> OneDrive - Clemson University > Desktop > Chirmay > S2.py > ...

1 import numpy as np
2 import control as ctrl
3 import cvxpy as cp
4
5 # Define plant matrices
6 Ap = np.array([[3, 1], [-2, 2]])
7 Bp = np.array([[0], [1]])
8 Dp = np.array([[0], [1]])
9 Cp = np.array([[3, 1]])
10 By = np.array([[1]])
11 Dy = np.array([[2]])
12 Hp = np.array([[1, 0]])
13 Bz = np.array([[0]])
14 Dz = np.array([[2]])
15
16 # Define controller matrices
17 Ac = np.array([[-4]])
18 Bc = np.array([[2]])
19 Cc = np.array([[1]])
20 Dc = np.array([[-2]])
21
22 # Compute closed-loop matrices
23 Acl = np.vstack((np.hstack((Ap+Bp*Dc@Hp, Bp*Cc)),
24                  np.hstack((Bc@Hp, Ac))))
25 Bcl = np.vstack((Bp+Bp*Dc@Dz,
26                  Bc@Dz))
27 Ccl = np.hstack((Cp+By*Dc@Hp, By*Cc))
28 Dcl = Dy+By*Dc@Dz
29
30 # Display closed-loop matrices
31 print('Closed-loop system matrices:\n')
32 print("Acl:")
33 print(Acl)
34 print("\nBcl:")
35 print(Bcl)
36 print("\nCcl:")
37 print(Ccl)
38 print("\nDcl:")
39 print(Dcl)
40
41 # Convert to state space form
42 sys_cl = ctrl.ss(Acl, Bcl, Ccl, Dcl)
43
44 # Check stability
45 eigenvalues = np.linalg.eigvals(Acl)
46 if all(np.real(eig) < 0 for eig in eigenvalues):
47     print("The closed-loop system is stable")
48 else:
49     print("The closed-loop system is unstable")

Interactive - S2.py
Interrupt X Clear All Restart Variables Save Export Expand Collapse lim_for_control (Python 3.10.13)

import numpy as np

Closed-loop system matrices:

Acl:
[[ 3  1  0]
 [-4  2  1]
 [ 2  0 -4]]

Bcl:
[[ 0]
 [-3]
 [ 4]]

Ccl:
[[1 1 1]]

Dcl:
[[-2]]

The closed-loop system is unstable

The H-infinity norm of the closed-loop system is: inf

Type 'python' code here and press Shift+Enter to run
Ln 33, Col 11 Spaces 4 UTF-8 CHLF Python 3.10.13 (lim_for_control) conda
```