# 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∞ 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∞ norm of the closed-loop system is: {hinfinity\_norm:.4f}")

# Output:

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∞ norm of the closed-loop system is: inf

# Screenshot:

