# Python Code:

import numpy as np

import cvxpy as cp

from scipy import signal

import control as ctrl

import matplotlib.pyplot as plt

# Define the State-Space Model of System

A = np.array([[0,    1],

              [-4,  -5]])

K = np.array([[0],

              [1]])

M = np.array([[-1, -2]])

H = np.array([[0]])

#################################################################################

# (2) Find Bounds on Disturbance for Uncertain System

# Define variables

P = cp.Variable((2, 2), symmetric=True)

gamma\_bar = cp.Variable(1)

M11 = P@A + A.T@P

M12 = P@K

M13 = M.T

M21 = K.T@P

M22 = cp.multiply(-gamma\_bar,np.eye(1))

M23 = H.T

M31 = M

M32 = H

M33 = cp.multiply(-gamma\_bar,np.eye(1))

# LMI Problem in Small Gain Theorem (SGT)

LMI = cp.vstack([

    cp.hstack([M11[0][0], M11[0][1], M21[0][0], M31[0][0]]),

    cp.hstack([M11[1][0], M11[1][1], M21[0][1], M31[0][1]]),

    cp.hstack([M21[0][0], M21[0][1], M22[0], M23[0][0]]),

    cp.hstack([M31[0][0], M31[0][1], M32[0][0], M33[0]])

])

constraints = [LMI << 0, P >> 0]

# Set up the optimization problem

objective = cp.Minimize(gamma\_bar)

problem = cp.Problem(objective, constraints)

# Solve the LMI problem

problem.solve()

# Get the value of gamma\_bar (energy-to-energy gain)

gamma\_bar\_star = gamma\_bar.value[0]

gamma\_star = 1/gamma\_bar\_star

print(f'Optimal Solution (γ\*): {gamma\_star:.4f}')

delta\_bounds = gamma\_star

print(f'Stability Guaranteed for |δ(t)| < {delta\_bounds:.4f} i.e. -{delta\_bounds:.4f} < δ(t) < {delta\_bounds:.4f}')

# Output:

Optimal Solution (γ\*): 2.4173

Stability Guaranteed for |δ(t)| < 2.4173 i.e., -2.4173 < δ(t) < 2.4173

# Screenshot:

