

$$1. \quad \dot{X} = \begin{bmatrix} \dot{q} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \ddot{q}_0(t) = -(5+2\delta)\dot{q}(t) - (4+\delta)q(t)$$

$$\dot{X} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(4+\delta) & -(5+2\delta) \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix}}_A \underbrace{\begin{bmatrix} q \\ \dot{q} \end{bmatrix}}_X + \underbrace{\begin{bmatrix} 0 & 0 \\ -1 & -2 \end{bmatrix}}_K \underbrace{\delta}_{\Delta} \underbrace{\begin{bmatrix} q \\ \dot{q} \end{bmatrix}}_X$$

$$= AX + K\phi$$

$$\phi = \Delta \psi$$

$$\psi = MX + \mu \phi^0$$

$$A = \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix} \quad K = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$M = \begin{bmatrix} -1 & -2 \end{bmatrix} \quad H = 0$$

2. To find bound γ s.t. uncertain system is stable (time varying uncertain)

$$\min \bar{\gamma} \quad \text{s.t.} \quad \begin{bmatrix} PA + A^T P & PK & M^T \\ K^T P & -\frac{1}{\gamma} I & H^T \\ M & H & -\frac{1}{\gamma} I \end{bmatrix} < 0 \quad \dots \text{(SGT LMI)}$$

$$P > 0$$

$$\equiv \min \bar{\gamma}$$

$$\text{s.t.} \quad \begin{bmatrix} PA + A^T P & PK & M^T \\ K^T P & -\bar{\gamma} I & H^T \\ M & H & -\bar{\gamma} I \end{bmatrix} < 0$$

$$P > 0$$

↓
PYTHON

$$\bar{\gamma}^* = 251.6102 \Rightarrow \gamma^* \Rightarrow |\delta(t)| < \frac{1}{\gamma^*} = \gamma^*$$

For guaranteed stability,

3. To find bound for uncertain system with time-invariant uncertainty

i.e. if δ was time invariant,

$$A = \begin{bmatrix} 0 & 1 \\ (-4-\delta) & (-5-2\delta) \end{bmatrix}$$

$$\text{char. eq}^n: \lambda^2 - (-5-2\delta)\lambda - (-4-\delta) = 0$$

$$\Rightarrow \lambda^2 + (2\delta+5)\lambda + (-\delta-4) = 0$$

$$\Rightarrow 2\delta+5 > 0 \Rightarrow \delta + 2.5 > 0$$

$$\text{or } -\delta-4 > 0 \Rightarrow \delta + 4 > 0$$

$$\Rightarrow \delta > -2.5$$

$$\Rightarrow -2.5 < \delta < 2.5$$

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 ( $\gamma^*$ ): {gamma_star:.4f}')
delta_bounds = gamma_star
print(f'Stability Guaranteed for  $|\delta(t)| < \{delta\_bounds:.4f\}$  i.e. -
{delta_bounds:.4f} <  $\delta(t)$  < {delta_bounds:.4f}')

```

Output:

Optimal Solution (γ^*): 2.4173

Stability Guaranteed for $|\delta(t)| < 2.4173$ i.e., $-2.4173 < \delta(t) < 2.4173$

Screenshot:

```

C:\Users> cd C:\Users> OneDrive - Clemson University > Desktop > S3.py
1 import numpy as np
2 import cvxpy as cp
3 from scipy import signal
4 import control as ctrl
5 import matplotlib.pyplot as plt
6
7 # Define the State-Space Model of System
8 A = np.array([[0, 1],
9               [-4, -5]])
10 K = np.array([[0],
11               [1]])
12 M = np.array([[1, -2]])
13 H = np.array([[0]])
14
15
16
17 # (2) Find Bounds on Disturbance for Uncertain System
18 # Define variables
19 P = cp.Variable((2, 2), symmetric=True)
20 gamma_bar = cp.Variable(1)
21 M11 = P@A + A.T@P
22 M12 = P@K
23 M13 = H.T
24 M21 = K.T@P
25 M22 = cp.multiply(-gamma_bar, np.eye(1))
26 M23 = H.T
27 M31 = H
28 M32 = H
29 M33 = cp.multiply(-gamma_bar, np.eye(1))
30 # LMI Problem in Small Gain Theorem (SGT)
31 LMI = cp.vstack([
32     cp.hstack([M11[0][0], M11[0][1], M21[0][0], M31[0][0]]),
33     cp.hstack([M11[1][0], M11[1][1], M21[0][1], M31[0][1]]),
34     cp.hstack([M21[0][0], M21[0][1], M22[0][0], M23[0][0]]),
35     cp.hstack([M31[0][0], M31[0][1], M32[0][0], M33[0][0]])
36 ])
37 constraints = [LMI < 0, P > 0]
38 # Set up the optimization problem
39 objective = cp.Minimize(gamma_bar)
40 problem = cp.Problem(objective, constraints)
41 # Solve the LMI problem
42 problem.solve()
43 # Get the value of gamma_bar (energy-to-energy gain)
44 gamma_bar_star = gamma_bar.value[0]
45 gamma_star = 1/gamma_bar_star
46 print(f'Optimal Solution ( $\gamma^*$ ): {gamma_star:.4f}')
47 delta_bounds = gamma_star
48 print(f'Stability Guaranteed for  $|\delta(t)| < \{delta\_bounds:.4f\}$  i.e. -
{delta_bounds:.4f} <  $\delta(t)$  < {delta_bounds:.4f}')

```