# 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([[-1.01887, 0.90506],

              [0.82225, -1.07741]])

B = np.array([[0.00203],

              [-0.00164]])

C = np.array([[15.87875, 1.48113]])

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

sys = signal.StateSpace(A, B, C, D)

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

# (a) Compute the Energy-to-Peak Gain (Γep)

# Define variables

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

gamma\_bar= cp.Variable(1)

M11 = cp.multiply(-gamma\_bar,np.eye(1))

M12 = C@P@C.T

M21 = C@P@C.T

M22 = -np.eye(1)

LMI\_1 = cp.vstack([

    cp.hstack([M11, M12]),

    cp.hstack([M21, M22])

])

LMI\_2 = A@P + P@A.T + B@B.T

LMI\_3 = P

constraints = [LMI\_1 << 0, LMI\_2 << 0, LMI\_3 >> 0]

# Set up the optimization problem

objective = cp.Minimize(gamma\_bar)

problem = cp.Problem(objective, constraints)

# Solve the LMI problem

problem.solve(solver=cp.SCS)

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

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

Gamma\_ep = abs(gamma\_bar\_star)\*\*(1/4)

print(f'Energy-to-Peak Gain (Γep): {Gamma\_ep:.4f}')

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

# (b) Compute Energy of Disturbance Signal

#     Simulate System Response to Pulse Disturbance

#     Check if System Response is Consistent with Γep

t = np.linspace(0, 2, 2001)  # Time vector

wg = 2 \* np.where((t >= 0) & (t <= 1), 1, 0) # Vertical wind gust acting as the disturbance

# Plot wg(t)

plt.figure()

plt.plot(t, wg)

plt.xlabel('Time (s)')

plt.ylabel('wg')

plt.title('Pulse Disturbance wg(t)')

plt.grid(True)

plt.show()

# Compute the energy of the disturbance signal ∥wg∥L2

L2\_norm\_wg = np.sqrt(np.trapz(wg\*\*2, t))

print(f'Energy of Disturbance Signal ∥wg∥L2: {L2\_norm\_wg:.4f}')

# Simulate system response to pulse disturbance

t, y, \_ = signal.lsim(sys, U=wg, T=t) # Simulate the system response

# Plot y(t)

plt.figure()

plt.plot(t, y)

plt.xlabel('Time (s)')

plt.ylabel('y')

plt.title('System Response y(t) to Pulse Disturbance wg(t)')

plt.grid(True)

plt.show()

# Estimate the energy of the response signal ∥y∥L2

L2\_norm\_y = np.sqrt(np.trapz(y\*\*2, t))

print(f'Is the system response consistent with Γep? {L2\_norm\_y <= Gamma\_ep}') # Check if system response is consistent with Γep

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

# (c) Compute the Energy-to-Energy Gain (Γee) (H∞ norm) using LMI Problem in Bounded Real Lemma

#     Estimate Energy of the Response of the System

#     Check if System Response is Consistent with Γee

# Define variables

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

gamma = cp.Variable(1)

M11 = P@A + A.T@P

M12 = P@B

M13 = C.T

M21 = B.T@P

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

M23 = D.T

M31 = C

M32 = D

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

# LMI Problem in Bounded Real Lemma

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]

# Set up the optimization problem

objective = cp.Minimize(gamma)

problem = cp.Problem(objective, constraints)

# Solve the LMI problem

problem.solve(solver=cp.SCS)

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

gamma\_star = gamma.value[0]

Gamma\_ee = gamma\_star

print(f'Energy-to-Energy Gain (Γee): {Gamma\_ee:.4f}')

# Estimate the energy of the response signal ∥y∥L2

L2\_norm\_y = np.sqrt(np.trapz(y\*\*2, t))

print(f'Energy of Response Signal ∥y∥L2: {L2\_norm\_y:.4f}')

print(f'Is the system response consistent with Γee? {L2\_norm\_y <= Gamma\_ee}') # Check if system response is consistent with Γee

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

# (d) Plot |G(jω)| as a function of ω

#     Verify that Peak Value of the Plot Gives Γee of the System

# Bode Plot (Peak of Frequency Response)

omega = np.logspace(-2, 2, 1000)  # Frequency range

\_, mag, \_ = signal.bode(sys, omega) # Bode magnitude plot

plt.figure()

plt.semilogx(omega, mag)

plt.xlabel('Frequency (rad/s)')

plt.ylabel('|G(jω)|')

plt.title('Frequency Response')

plt.grid(True)

plt.show()

peak\_mag\_dB = max(mag) # Maximum (peak) magnitude (Γep) in dB

peak\_mag = 10\*\*(peak\_mag\_dB/20)

print(f'Peak Value of Frequency Response: {peak\_mag:.4f}')

# Verify that Peak Value of the Plot Gives Γee of the System

tolerance = 0.01

print(f'Is the peak value of frequency response consistent with Γee? {abs(peak\_mag-Gamma\_ee) <= tolerance}')

# Output:

1. Energy-to-Peak Gain (Γep): 0.0116
2. Energy of Disturbance Signal ∥wg∥L2: 2.0005

Is the system response consistent with Γep? False

(this is because although the system is quite robust, as indicated by the small magnitude of Γep, the disturbance is quite high for the system to handle)

A graph with a line

Description automatically generated

1. Energy-to-Energy Gain (Γee): 0.0245

Energy of Response Signal ∥y∥L2: 0.0265

Is the system response consistent with Γee? False

(this is because although the system is quite robust, as indicated by the small magnitude of Γee, the disturbance is quite high for the system to handle)

1. Peak Value of Frequency Response: 0.0315

Is the peak value of frequency response consistent with Γee? True

A graph with a blue line

Description automatically generated

# Screenshot:

