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)
```

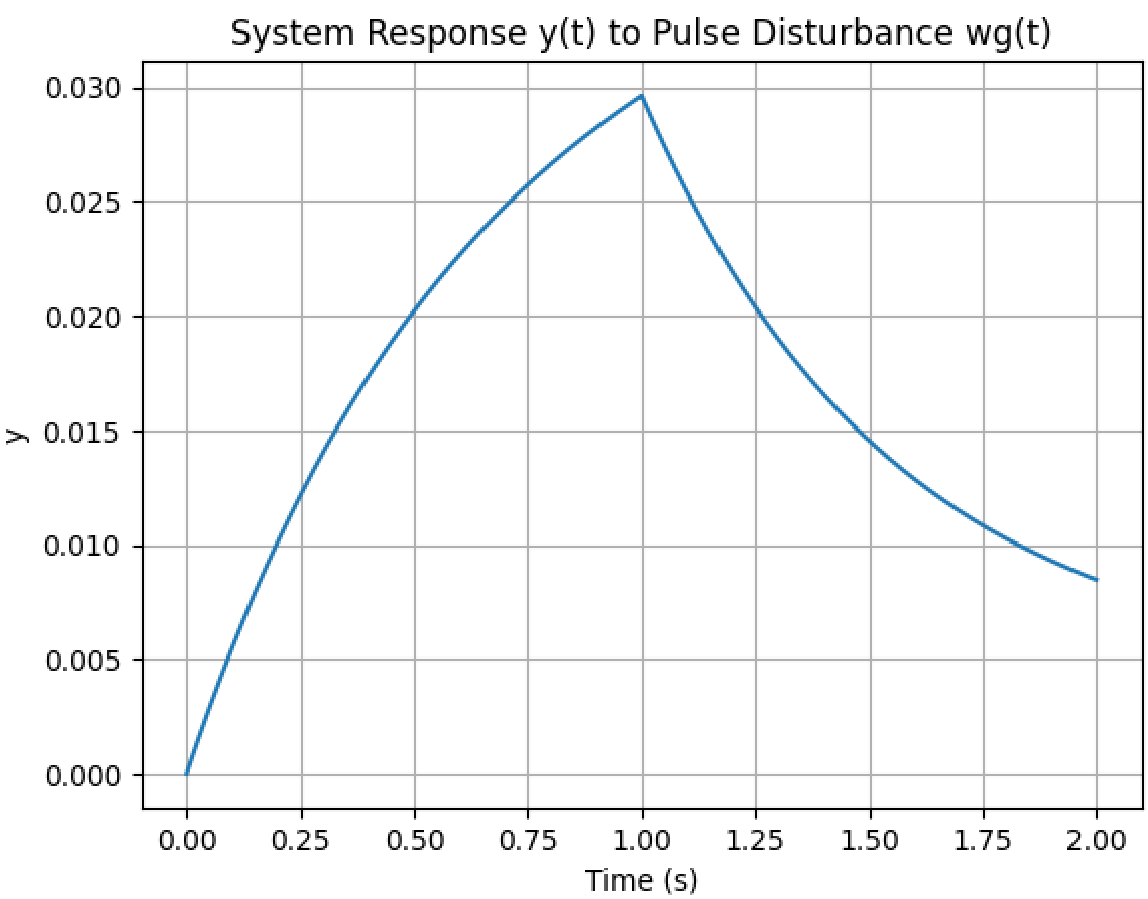
```
M11 = P@A + A.T@P
M12 = P@B
M13 = C.T
M21 = B.T@P
M22 = cp.multiply(-gamma,np.eye(1))
M23 = D.T
M31 = C
M32 = D
M33 = cp.multiply(-gamma,np.eye(1))
# LMI Problem in Bounded Real Lemma
LMI = cp.vstack([
    cp.hstack([M11[0][0], M11[0][1], M21[0][0], M31[0][0]]),
    cp.hstack([M11[1][0], M11[1][1], 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:

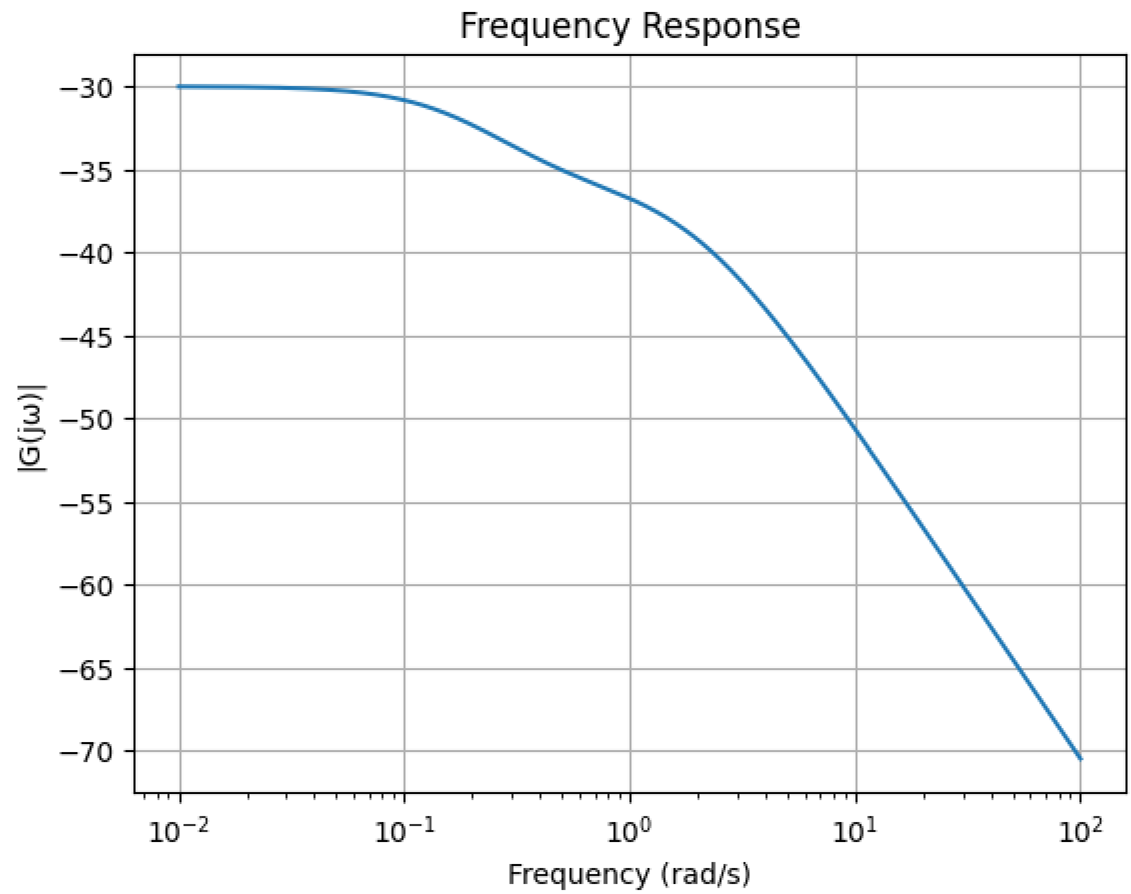
a) Energy-to-Peak Gain (Γep): 0.0116

b) 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)

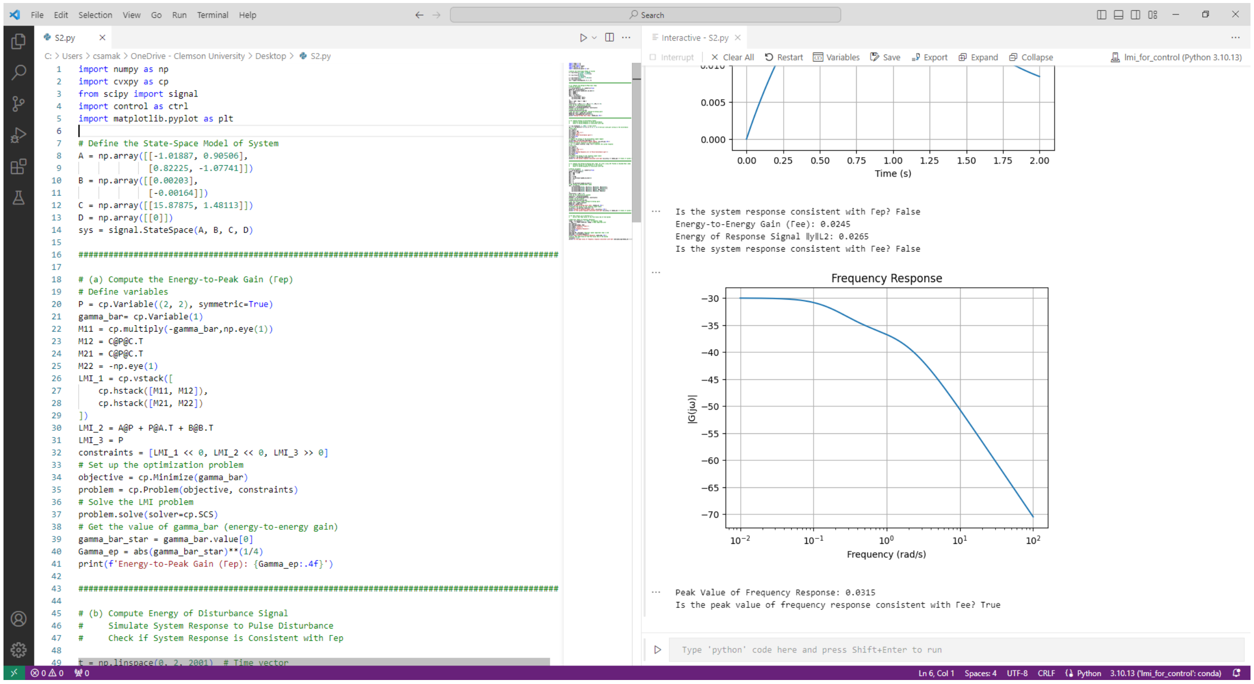


c) Energy-to-Energy Gain (Γee): 0.0245
Energy of Response Signal $\|y\|L2$: 0.0265
Is the system response consistent with Γee? False
(this is because although the system is quite robust, as indicated by the small magnitude of Γee, the disturbance is quite high for the system to handle)

d) Peak Value of Frequency Response: 0.0315
Is the peak value of frequency response consistent with Γee? True



Screenshot:



File Edit Selection View Go Run Terminal Help
S2.py
C: > Users > csamak > OneDrive - Clemson University > Desktop > S2.py
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
Interactive - S2.py
Interrupt Clear All Restart Variables Save Export Expand Collapse
lmi_for_control (Python 3.10.13)
0.005
0.000
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
Time (s)
Is the system response consistent with Γep? False
Energy-to-Energy Gain (Γee): 0.0245
Energy of Response Signal ‖y‖L2: 0.0265
Is the system response consistent with Γee? False
Frequency Response
−30
−35
−40
−45
−50
−55
−60
−65
−70
|G(jω)|
10⁻² 10⁻¹ 10⁰ 10¹ 10²
Frequency (rad/s)
Peak Value of Frequency Response: 0.0315
Is the peak value of frequency response consistent with Γee? True
Type 'python' code here and press Shift+Enter to run
Ln 6, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.13 ('lmi_for_control': conda)