# Python Code:

import numpy as np

import cvxpy as cp

import matplotlib.pyplot as plt

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

# (1) Define the State-Space Model of System

A = np.array([[-4,  2],

              [1,  -7]])

K = np.array([[1],

              [1]])

M = np.array([[1, 0]])

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

print('System:\nA = \n{} \nK = \n{}\nM = \n{} \nH = \n{}'.format(A, K, M, H))

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

# (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], M12[0][0], M13[0][0]]),

    cp.hstack([M11[1][0], M11[1][1], M12[1][0], M13[1][0]]),

    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 optimal value of gamma\_bar

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

gamma\_star = 1/gamma\_bar\_star

delta\_bounds = gamma\_star

print(f'Stability Guaranteed for |δ| < {delta\_bounds:.4f} i.e. -{delta\_bounds:.4f} < δ < {delta\_bounds:.4f}')

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

# (3) Plot Root Locus of the System as a Function of δ

delta\_values = np.linspace(-3, 3, 100000) # Range of δ values

real\_parts = [] # List to store real parts of eigenvalues for each δ

# Loop through each δ value

for delta in delta\_values:

    A\_delta = A+(K\*M\*delta) # Update the system matrix with the current δ

    eigenvalues = np.linalg.eigvals(A\_delta) # Calculate eigenvalues

    real\_parts.append(np.real(eigenvalues)) # Store the real part of the eigenvalues

# Plot the root locus

plt.figure()

plt.plot(delta\_values, real\_parts)

plt.title('Root Locus as a Function of δ')

plt.xlabel('δ')

plt.ylabel('Re(λi)')

plt.legend(['λ1', 'λ2'])

plt.grid(False)

plt.show()

# Find the maximum value of |δ| for negative real part eigenvalues

max\_delta = max(delta\_values[np.max(real\_parts, axis=1) < 0])

print(f"Maximum |δ| for Negative Real Part of Eigenvalues is: {max\_delta:.4f}")

# Determine if this solution is consistent with SGT LMI result

tolerance = 0.001

print(f'Is this Solution Consistent with SGT LMI Result? {abs(max\_delta-delta\_bounds) <= tolerance}')

print('\n-----------------------------------------------------------------------------------\n')

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

# (4) Repeat (1)-(3) for Different Uncertain System

# Define the state-space model of system

A = np.array([[0,      1],

              [-1,  -0.2]])

K = np.array([[0],

              [0.5]])

M = np.array([[1, 0]])

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

print('System:\nA = \n{} \nK = \n{}\nM = \n{} \nH = \n{}'.format(A, K, M, H))

# 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], M12[0][0], M13[0][0]]),

    cp.hstack([M11[1][0], M11[1][1], M12[1][0], M13[1][0]]),

    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 optimal value of gamma\_bar

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

gamma\_star = 1/gamma\_bar\_star

delta\_bounds = gamma\_star

print(f'Stability Guaranteed for |δ| < {delta\_bounds:.4f} i.e. -{delta\_bounds:.4f} < δ < {delta\_bounds:.4f}')

# Plot root locus of the system as a function of δ

delta\_values = np.linspace(-3, 3, 100000) # Range of δ values

real\_parts = [] # List to store real parts of eigenvalues for each δ

# Loop through each δ value

for delta in delta\_values:

    A\_delta = A+(K\*M\*delta) # Update the system matrix with the current δ

    eigenvalues = np.linalg.eigvals(A\_delta) # Calculate eigenvalues

    real\_parts.append(np.real(eigenvalues)) # Store the real part of the eigenvalues

# Plot the root locus

plt.figure()

plt.plot(delta\_values, real\_parts)

plt.title('Root Locus as a Function of δ')

plt.xlabel('δ')

plt.ylabel('Re(λi)')

plt.legend(['λ1', 'λ2'])

plt.grid(False)

plt.show()

# Find the maximum value of |δ| for negative real part eigenvalues

max\_delta = max(delta\_values[np.max(real\_parts, axis=1) < 0])

print(f"Maximum |δ| for Negative Real Part of Eigenvalues is: {max\_delta:.4f}")

# Determine if this solution is consistent with SGT LMI result

tolerance = 0.005

print(f'Is this Solution Consistent with SGT LMI Result? {abs(max\_delta-delta\_bounds) <= tolerance}')

print('\n-----------------------------------------------------------------------------------\n')

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

# (5) Simulate the System with Time-Varying Uncertainty

# Plot root locus of the system as a function of δ(t)

time\_values = np.linspace(0, 10, 100000) # Time values

delta\_values = np.cos(2\*time\_values) # Range of δ values

real\_parts = [] # List to store real parts of eigenvalues for each δ

# Loop through each δ value

for delta in delta\_values:

    A\_delta = A+(K\*M\*delta) # Update the system matrix with the current δ

    eigenvalues = np.linalg.eigvals(A\_delta) # Calculate eigenvalues

    real\_parts.append(np.real(eigenvalues)) # Store the real part of the eigenvalues

# Plot the root locus

plt.figure()

plt.plot(delta\_values, real\_parts)

plt.title('Root Locus as a Function of δ(t)')

plt.xlabel('δ(t)')

plt.ylabel('Re(λi)')

plt.legend(['λ1', 'λ2'])

plt.grid(False)

plt.show()

# Find the maximum value of |δ| for negative real part eigenvalues

max\_delta = max(delta\_values[np.max(real\_parts, axis=1) < 0])

print(f"Maximum |δ| for Negative Real Part of Eigenvalues is: {max\_delta:.4f}")

# Is the system stable or unstable for the given δ(t)?

print('Is the System Stable or Unstable for Given δ(t)=cos(2t)?\n'

      '    In general, the system stability cannot be guaranteed for given δ(t).\n'

      '    However, for specific time-frozen instances where |δ(t)| < 0.3980,\n'

      '    and withhout prior destabilization, the system may exhibit stable\n'

      '    behavior.')

# Determine if this solution is consistent with SGT LMI result

tolerance = 0.005

print(f'Is this Solution Consistent with SGT LMI Result? {abs(max\_delta-delta\_bounds) <= tolerance}')

if not abs(max\_delta-delta\_bounds) <= tolerance:

    print('WHY?:\n    The results from eigenvalue test and SGT LMI are not consistent\n'

          '    since the former cannot be applied to time-varying systems while\n'

          '    the latter can. Hence in this case, results of SGT analysis should\n'

          '    be trusted.')

print('\n-----------------------------------------------------------------------------------\n')

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

# (6) Comment on Stability of the System to Time-Invariant and Time-Varying Perturbations

print('The results from stability analysis indicate that the system is guaranteed\n'

          'to be stable for time-invariant perturbations (where |δ| < 2.8890 i.e.,\n'

          '-2.8890 < δ < 2.8890), but is NOT guaranteed to be stable for the given\n'

          'time-varying perturbations of δ(t) = cos(2t).\n'

          'In case of systems with time-varying perturbations (uncertainty), the\n'

          'results from eigenvalue test and SGT LMI may/will NOT be consistent.\n'

          'This is because the former (i.e., eigenvalue test) can ONLY be applied\n'

          'to time-invariant systems or if the system is slowly varying (i.e., if\n'

          'the uncertainty is much slower than system dynamics) while the latter\n'

          '(i.e., small gain theorem) being generalization of the Nyquist criterion\n'

          'to non-linear time-varying MIMO systems, can be applied to systems with\n'

          'time-invariant as well as time-varying perturbations (uncertainty). Hence,\n'

          'in general, results of SGT analysis can be trusted but the results of\n'

          'eigenvalue test cannot be trusted blindly.')

# Output:

1. System:

A =

[[-4 2]

[ 1 -7]]

K =

[[1]

[1]]

M =

[[1 0]]

H =

[[0]]

1. Stability Guaranteed for |δ| < 2.8890 i.e. -2.8890 < δ < 2.8890
2. Maximum |δ| for Negative Real Part of Eigenvalues is: 2.8889

Is this Solution Consistent with SGT LMI Result? True

A graph of a function

Description automatically generated

1. System:

A =

[[ 0. 1. ]

[-1. -0.2]]

K =

[[0. ]

[0.5]]

M =

[[1 0]]

H =

[[0]]

Stability Guaranteed for |δ| < 0.3980 i.e. -0.3980 < δ < 0.3980

Maximum |δ| for Negative Real Part of Eigenvalues is: 2.0000

Is this Solution Consistent with SGT LMI Result? False

A graph of a function

Description automatically generated

1. Maximum |δ| for Negative Real Part of Eigenvalues is: 1.0000

Is the System Stable or Unstable for Given δ(t)=cos(2t)?

In general, the system stability cannot be guaranteed for given δ(t).

However, for specific time-frozen instances where |δ(t)| < 0.3980,

and withhout prior destabilization, the system may exhibit stable

behavior.

Is this Solution Consistent with SGT LMI Result? False

WHY?:

The results from eigenvalue test and SGT LMI are not consistent

since the former cannot be applied to time-varying systems while

the latter can. Hence in this case, results of SGT analysis should

be trusted.

A graph of a function

Description automatically generated

1. The results from stability analysis indicate that the system is guaranteed

to be stable for time-invariant perturbations (where |δ| < 2.8890 i.e.,

-2.8890 < δ < 2.8890), but is NOT guaranteed to be stable for the given

time-varying perturbations of δ(t) = cos(2t).

In case of systems with time-varying perturbations (uncertainty), the

results from eigenvalue test and SGT LMI may/will NOT be consistent.

This is because the former (i.e., eigenvalue test) can ONLY be applied

to time-invariant systems or if the system is slowly varying (i.e., if

the uncertainty is much slower than system dynamics) while the latter

(i.e., small gain theorem) being generalization of the Nyquist criterion

to non-linear time-varying MIMO systems, can be applied to systems with

time-invariant as well as time-varying perturbations (uncertainty). Hence,

in general, results of SGT analysis can be trusted but the results of

eigenvalue test cannot be trusted blindly.

# Screenshot:

