# Python Code:

import cvxpy as cp

import numpy as np

from numpy.linalg import eig

# Define the A matrices for each system

A1 = np.array([[-7, 5],

               [3, -4]])

A2 = np.array([[-6, 4, -2],

               [3, -8, 1],

               [-1, 5, -7]])

# Create optimization variables

n1 = A1.shape[0] # Assuming square matrices

P1 = cp.Variable((n1, n1), symmetric=True)

n2 = A2.shape[0] # Assuming square matrices

P2 = cp.Variable((n2, n2), symmetric=True)

# Define the constraints for each system

constraints = []

# A1 eigenvalues to the left of s = -2

constraint1 = [A1 @ P1 + P1 @ A1.T << -2 \* np.eye(n1)]

# A2 eigenvalues to the left of s = -2

constraint2 = [A2 @ P2 + P2 @ A2.T << -2 \* np.eye(n2)]

# Create the feasibility problem

problem1 = cp.Problem(cp.Minimize(0), constraint1)

problem2 = cp.Problem(cp.Minimize(0), constraint2)

# Solve the problem

problem1.solve()

problem2.solve()

# Check if the problem is feasible

if problem1.status == cp.OPTIMAL:

    print("System 1 has eigenvalues to the left of s = -2.")

else:

    print("System 1 does not have eigenvalues to the left of s = -2.")

if problem2.status == cp.OPTIMAL:

    print("System 2 has eigenvalues to the left of s = -2.")

else:

    print("System 2 does not have eigenvalues to the left of s = -2.")

# Confirm actual eigenvalues of both systems

w1, v1 = eig(A1)

print('Eigenvalues of System 1 are:', w1)

w2, v2 = eig(A2)

print('Eigenvalues of System 2 are:', w2)

# Output:

System 1 has eigenvalues to the left of s = -2.

System 2 has eigenvalues to the left of s = -2.

Eigenvalues of System 1 are: [-9.65331193 -1.34668807]

Eigenvalues of System 2 are: [-12.20640314+0.j -4.39679843+0.57443474j -4.39679843-0.57443474j]

# Screenshot:

