# **CODE:**

% Clear workspace

clear

clc

% Add parser and solver to path

addpath(genpath('C:\Users\tsamak\Downloads\MathWorks\Toolboxes\archives\required\YALMIP'))

addpath(genpath('C:\Users\tsamak\Downloads\MathWorks\Toolboxes\archives\required\SeDuMi'))

% Define the system matrices

A1 = [-4, 1; 0, 2];

B1 = [1; 0];

A2 = [-3, 2; 4, 1];

B2 = [0; 1];

% Define decision variables

n = size(A1, 1); % Number of states

P = sdpvar(n, n);

K1 = sdpvar(1, n);

K2 = sdpvar(1, n);

% Define the LMI constraints for SYS1

Constraints1 = [P >= 0.001\*eye(n), A1\*P + P\*A1' + B1\*K1 + K1'\*B1' <= 0];

% Define the LMI constraints for SYS2

Constraints2 = [P >= 0.001\*eye(n), A2\*P + P\*A2' + B2\*K2 + K2'\*B2' <= 0];

% Solve the LMI for SYS1

options = sdpsettings('verbose', 0);

sol1 = optimize(Constraints1, [], options);

% Solve the LMI for SYS2

sol2 = optimize(Constraints2, [], options);

% Check the feasibility of the LMIs and compute K1 and K2 if feasible

if sol1.problem == 0

disp('SYS1 is stabilizable.');

K1\_value = value(K1);

disp('Stabilizing control law for SYS1:');

disp(K1\_value);

else

disp('SYS1 is not stabilizable.');

end

if sol2.problem == 0

disp('SYS2 is stabilizable.');

K2\_value = value(K2);

disp('Stabilizing control law for SYS2:');

disp(K2\_value);

else

disp('SYS2 is not stabilizable.');

end

# **OUTPUT:**

SYS1 is not stabilizable.

SYS2 is stabilizable.

Stabilizing control law for SYS2:

-3.3333 -2.4912

# **SCREENSHOT:**

A screenshot of a computer

Description automatically generated