Problem 1-B

# **CODE:**

% PROBLEM 1-B

% Clear workspace

close all

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

A = [-4, 2; 1, -7];

K = [1; 1];

M = [1, 0];

H = 0;

% Define the LMI variables

P = sdpvar(2, 2);

gamma = sdpvar(1, 1);

% Define the LMI constraints

LMI1 = [P\*A+A'\*P, P\*K, M'; K'\*P, -gamma\*eye(1), H'; M, H, -gamma\*eye(1)] <= 0;

LMI2 = P >= 0;

% Set up the objective

Objective = gamma;

% Define the solver settings (use an LMI solver like YALMIP with a solver of your choice)

options = sdpsettings('verbose', 1, 'solver', 'sedumi');

% Solve the LMI problem

solution = optimize([LMI1, LMI2], Objective, options);

if solution.problem == 0

% Extract the optimal solution

optimal\_gamma = 1/value(gamma);

% Display the result

fprintf('Optimal gamma: %.4f\n', optimal\_gamma);

fprintf('Stability guaranteed: %.4f < delta(t) < %.4f\n', -optimal\_gamma, optimal\_gamma);

else

fprintf('LMI problem could not be solved.\n');

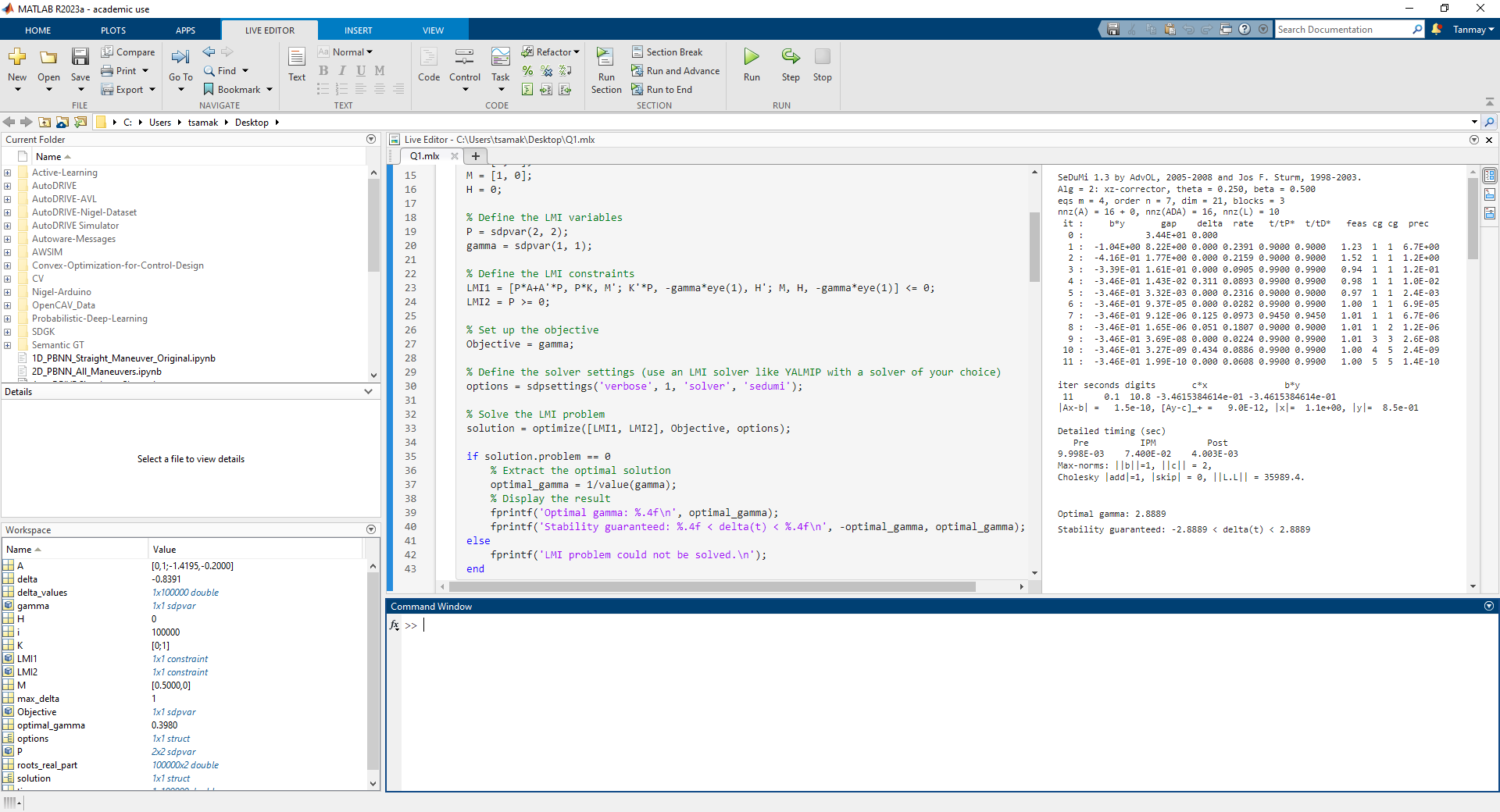
end

# **OUTPUT:**

Optimal gamma: 2.8889

Stability guaranteed: -2.8889 < delta(t) < 2.8889

# **SCREENSHOT:**



Problem 1-C

# **CODE:**

% PROBLEM 1-C

% Define delta limits and granularity

delta\_values = linspace(-5, 5, 100000);

roots\_real\_part = zeros(length(delta\_values), 2);

max\_delta = -100; % Initialize max\_delta for optimization

% Plot root locus for all delta values

for i = 1:length(delta\_values)

delta = delta\_values(i);

A = [-4 + delta, 2; 1 + delta, -7];

roots\_real\_part(i, :) = real(eig(A));

if (all(real(eig(A)) < 0)) && (delta > max\_delta)

max\_delta = delta; % Optimize max\_delta

end

end

% Create root locus plot

figure;

plot(delta\_values, roots\_real\_part);

xlabel('\delta');

ylabel('Real part of eigenvalues');

title('Root Locus of the Uncertain System');

grid on;

% Print result

fprintf('Optimal delta: %.4f\n', max\_delta);

# **OUTPUT:**

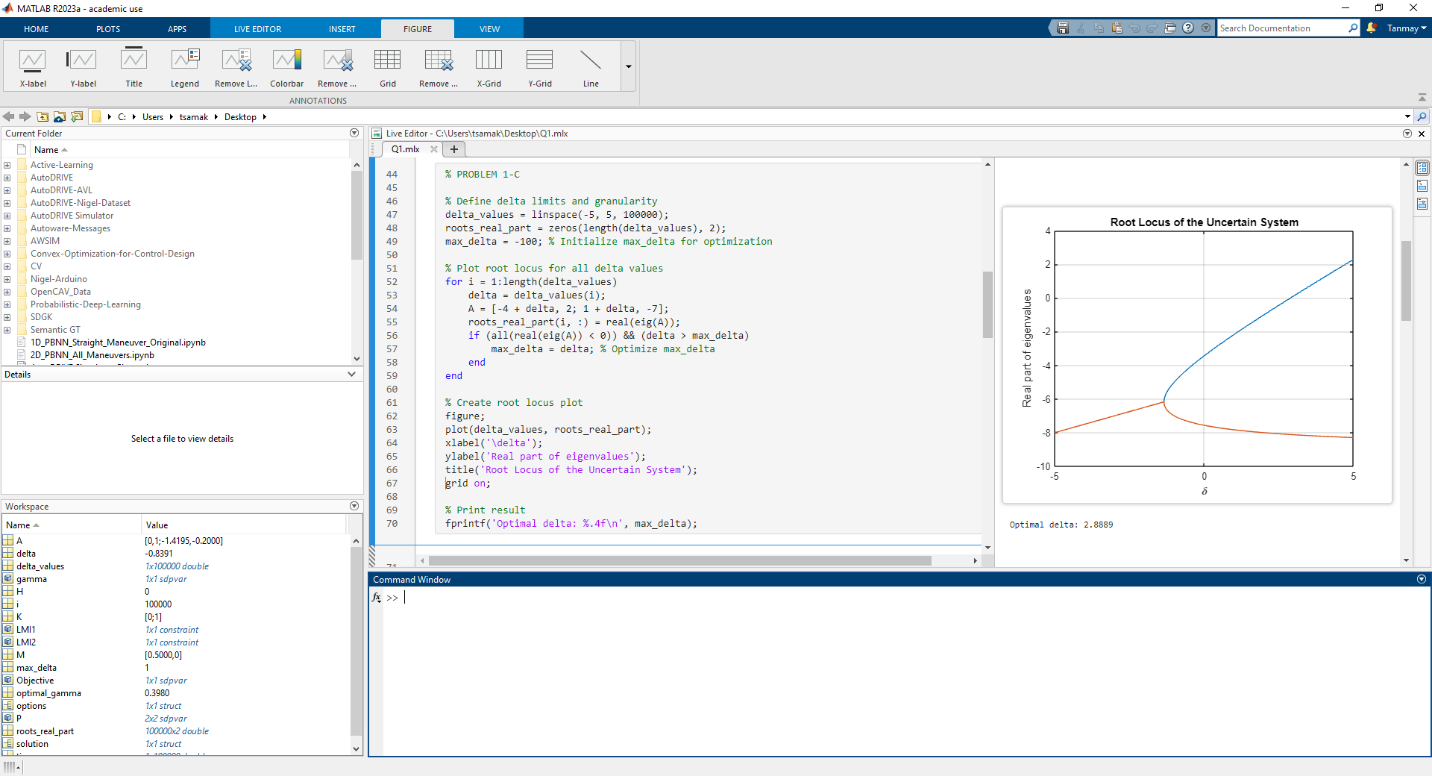
A graph of a function

Description automatically generated

Optimal delta: 2.8889

The maximum value of |δ| such that the system has eigenvalues with negative real part is consistent with the maximum bound for δ that guarantees stability of the uncertain system (based on LMI representation of SGT).

# **SCREENSHOT:**



Problem 1-D

# **CODE:**

% PROBLEM 1-D

% Define the system matrices

A = [0, 1; -1, -0.2];

K = [0; 1];

M = [0.5, 0];

H = 0;

% Define the LMI variables

P = sdpvar(2, 2);

gamma = sdpvar(1, 1);

% Define the LMI constraints

LMI1 = [P\*A+A'\*P, P\*K, M'; K'\*P, -gamma\*eye(1), H'; M, H, -gamma\*eye(1)] <= 0;

LMI2 = P >= 0;

% Set up the objective

Objective = gamma;

% Define the solver settings (use an LMI solver like YALMIP with a solver of your choice)

options = sdpsettings('verbose', 1, 'solver', 'sedumi');

% Solve the LMI problem

solution = optimize([LMI1, LMI2], Objective, options);

if solution.problem == 0

% Extract the optimal solution

optimal\_gamma = 1/value(gamma);

% Display the result

fprintf('Optimal gamma: %.4f\n', optimal\_gamma);

fprintf('Stability guaranteed: %.4f < delta(t) < %.4f\n', -optimal\_gamma, optimal\_gamma);

else

fprintf('LMI problem could not be solved.\n');

end

% Define delta limits and granularity

delta\_values = linspace(-5, 5, 100000);

roots\_real\_part = zeros(length(delta\_values), 2);

max\_delta = -100; % Initialize max\_delta for optimization

% Plot root locus for all delta values

for i = 1:length(delta\_values)

delta = delta\_values(i);

A = [0, 1; -1 + 0.5\*delta, -0.2];

roots\_real\_part(i, :) = real(eig(A));

if (all(real(eig(A)) < 0)) && (delta > max\_delta)

max\_delta = delta; % Optimize max\_delta

end

end

% Create root locus plot

figure;

plot(delta\_values, roots\_real\_part);

xlabel('\delta');

ylabel('Real part of eigenvalues');

title('Root Locus of the Uncertain System');

grid on;

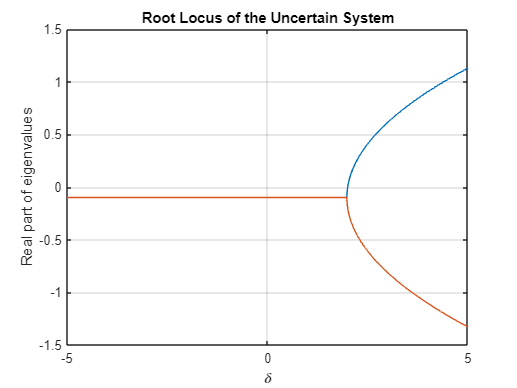
% Print result

fprintf('Optimal delta: %.4f\n', max\_delta);

# **OUTPUT:**

Optimal gamma: 0.3980

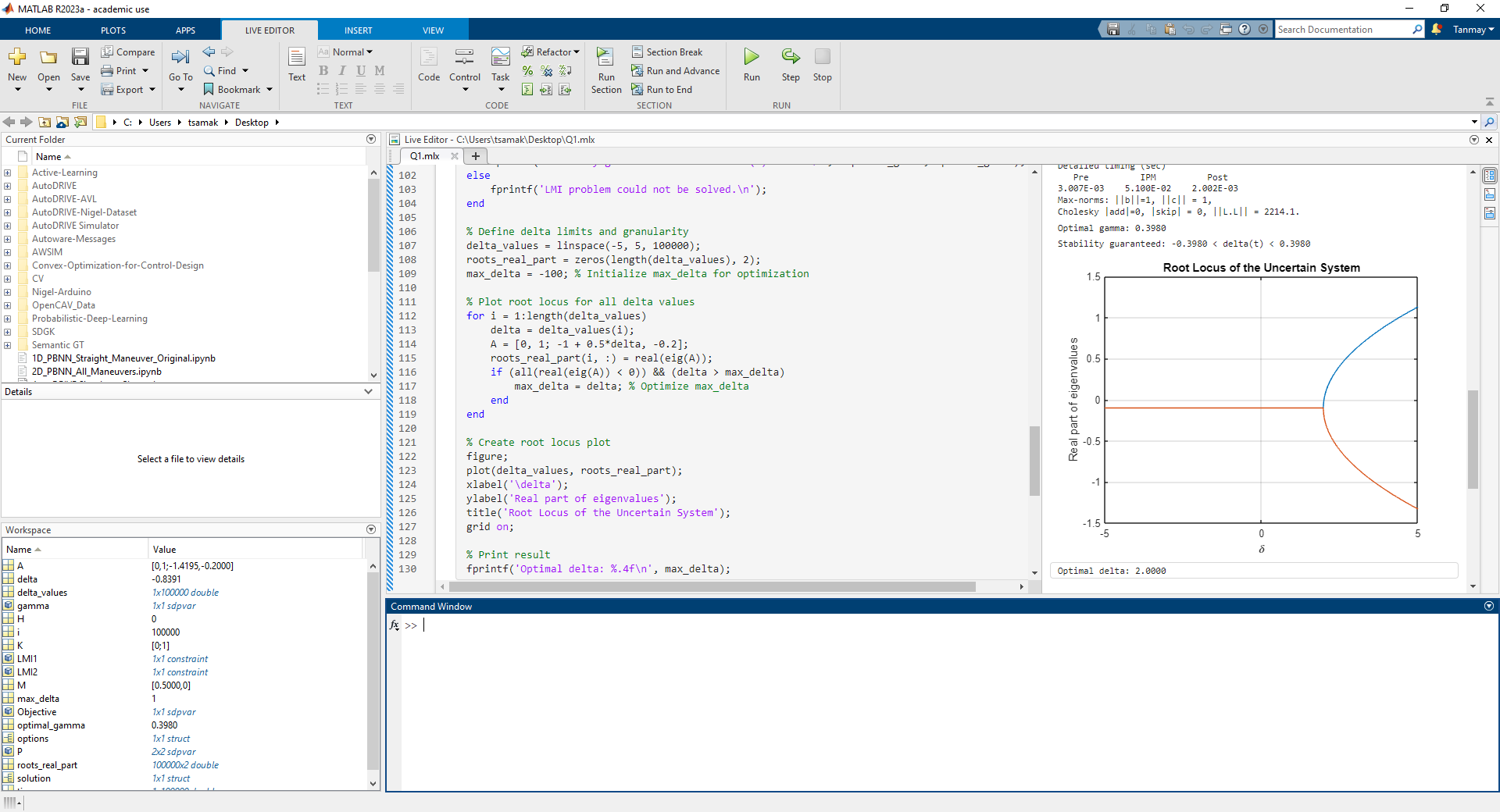
Stability guaranteed: -0.3980 < delta(t) < 0.3980



Optimal delta: 2.0000

The maximum value of |δ| such that the system has eigenvalues with negative real part is NOT consistent with the maximum bound for δ that guarantees stability of the uncertain system (based on LMI representation of SGT).

# **SCREENSHOT:**



Problem 1-E

# **CODE:**

% PROBLEM 1-E

% Define time array for simulation

time = linspace(0, 10, 100000);

roots\_real\_part = zeros(length(delta\_values), 2);

max\_delta = -100; % Initialize max\_delta for optimization

% Plot root locus for all delta values

for i = 1:length(time)

delta = cos(time(i));

A = [0, 1; -1 + 0.5\*delta, -0.2];

roots\_real\_part(i, :) = real(eig(A));

if (all(real(eig(A)) < 0)) && (delta > max\_delta)

max\_delta = delta; % Optimize max\_delta

end

end

% Create root locus plot

figure;

plot(delta\_values, roots\_real\_part);

xlabel('\delta');

ylabel('Real part of eigenvalues');

title('Root Locus of the Uncertain System');

grid on;

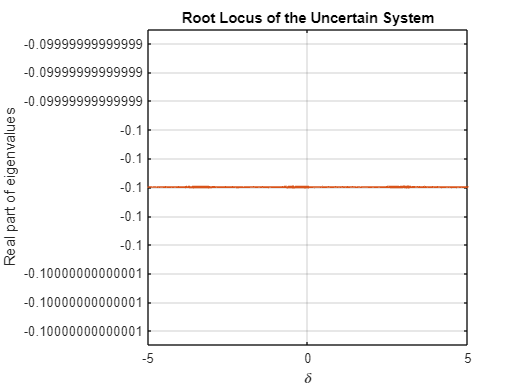
% Print result

fprintf('Optimal delta: %.4f\n', max\_delta);

# **OUTPUT:**

Optimal gamma: 0.3980

Stability guaranteed: -0.3980 < delta(t) < 0.3980

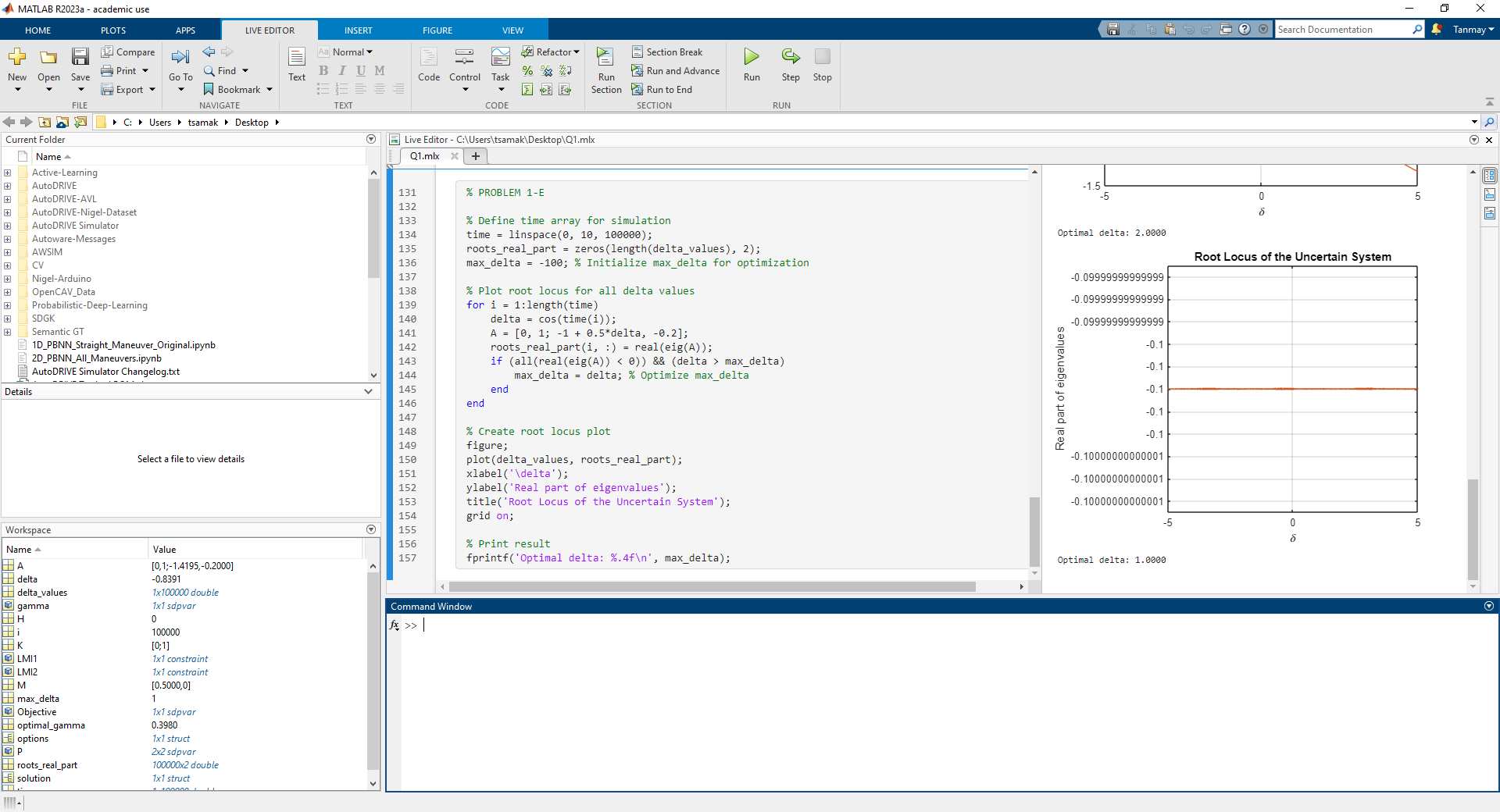


Optimal delta: 1.0000

The system is not stable throughout the given δ(t) = cos(2t). The system may be stable for -0.3980 < δ(t) < 0.3980, provided it is not already destabilized before that time instant.

The maximum value of |δ| such that the system has eigenvalues with negative real part is NOT consistent with the maximum bound for δ that guarantees stability of the uncertain system (based on LMI representation of SGT). This is likely because Eigenvalue test is not applicable to such time-varying uncertainties δ(t), it can only be applied to time-invariant or very slowly time-varying uncertainties.

# **SCREENSHOT:**



Problem 1-F

The system is guaranteed to be stable between -0.3980 < δ(t) < 0.3980 for time-invariant perturbations. The system is not guaranteed to be stable throughout any time-varying perturbation, but it may be stable for -0.3980 < δ(t) < 0.3980, provided it is not already destabilized before that time instant.

Eigenvalue conditions cannot guarantee stability to time-varying perturbations. They can only be applied to time-invariant or very slowly (slower than system dynamics) time-varying uncertainties.

SGT condition can guarantee stability to time-invariant as well as time-varying perturbations.