Part 1

Q1)

Code

%--------Q1--------

% Define the open-loop transfer function G(s)

num\_G = 1;

den\_G = [1 1 0]; % s(s+1) = s^2 + s

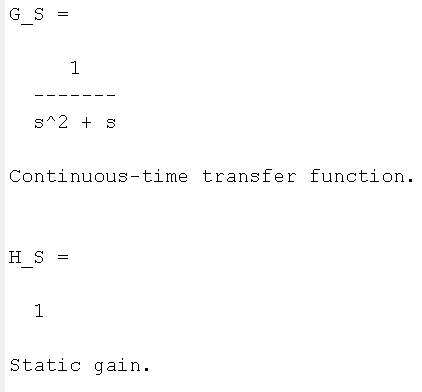
G\_S = tf(num\_G, den\_G)

% Define the feedback transfer function H(s)

num\_H = [1];

den\_H = [1]; % Unity Feedback

H\_S = tf(num\_H, den\_H)

Output:  


Q2) the 𝑠𝑡𝑒𝑝() command to plot the output of 𝐺(𝑆)

Code: We made a function that plots step time response and checks stability

%--------Q2-------- Step Response of G(s) (Open-Loop)

% Plot step response of G(s)

draw\_step(G\_S, 'Open-Loop System G(s)');  
  
  
function draw\_step(sys, sys\_name)

% Create figure

figure;

% Plot step response

step(sys);

title(['Step Response of ', sys\_name]);

grid on;

% Display poles

poles = pole(sys);

disp(['Poles of ', sys\_name, ':']);

disp(poles);

% Check stability

if all(real(poles) < 0)

disp('System is stable (all poles in LHP)');

elseif any(real(poles) > 0)

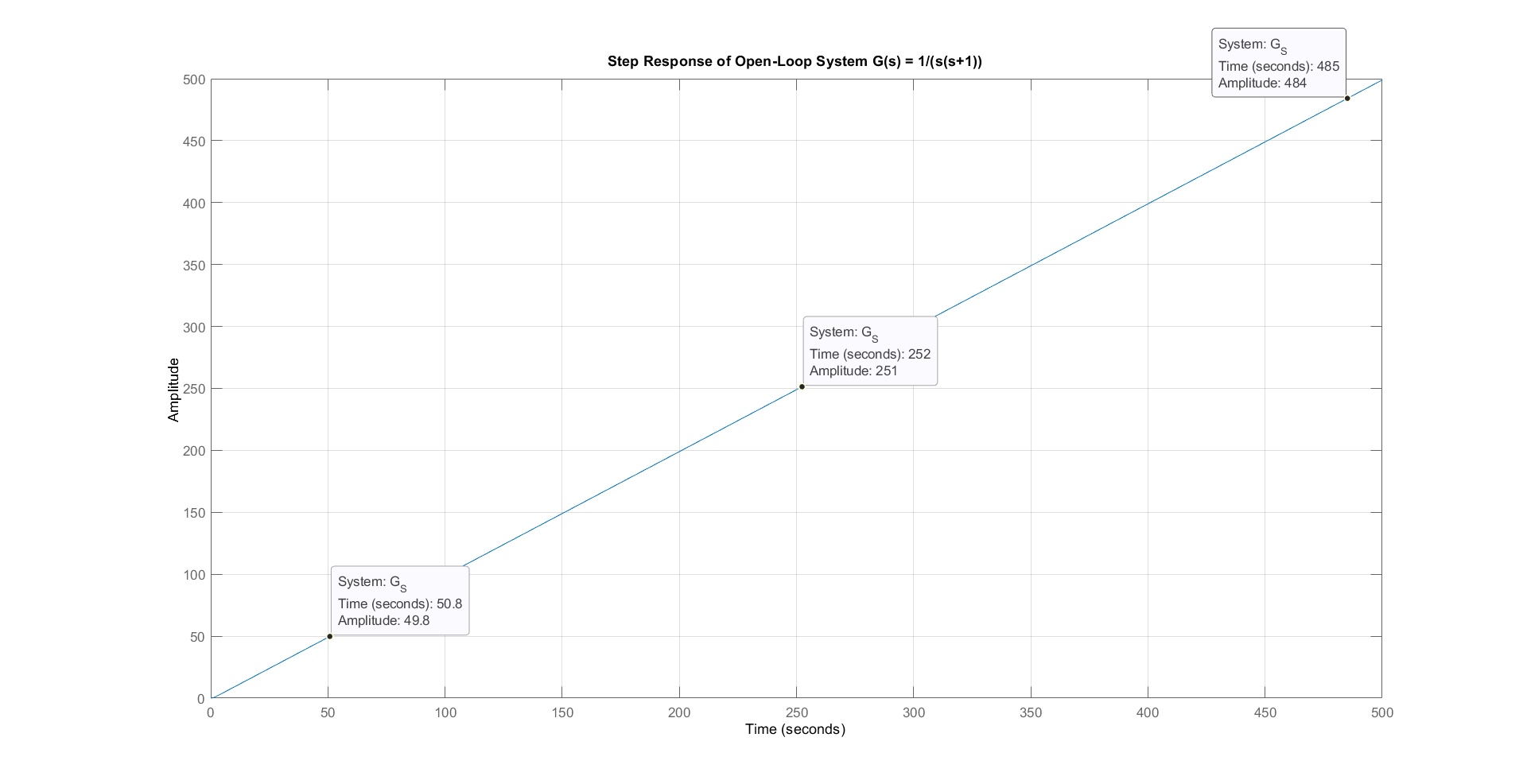
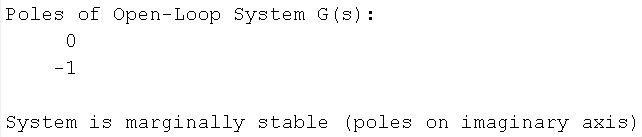
disp('System is unstable (at least one pole in RHP)');

else

disp('System is marginally stable (poles on imaginary axis)');

end

end

Output:

Q3)

Code:

%--------Q3-------- Closed-Loop Analysis

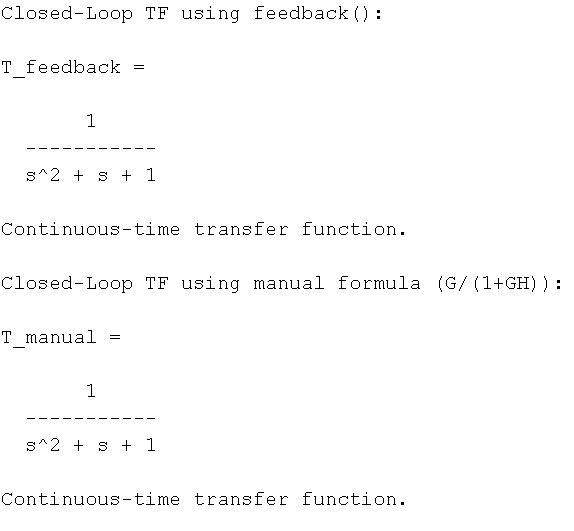
disp('Closed-Loop TF using feedback():');

T\_feedback = feedback(G\_S, H\_S)

disp('Closed-Loop TF using manual formula (G/(1+GH)):');

T\_manual = (1 / (1 + G\_S \* H\_S)) \* G\_S; % Equivalent to T(s) = G/(1+GH)

T\_manual = minreal(T\_manual) % Cancel common terms

Output:

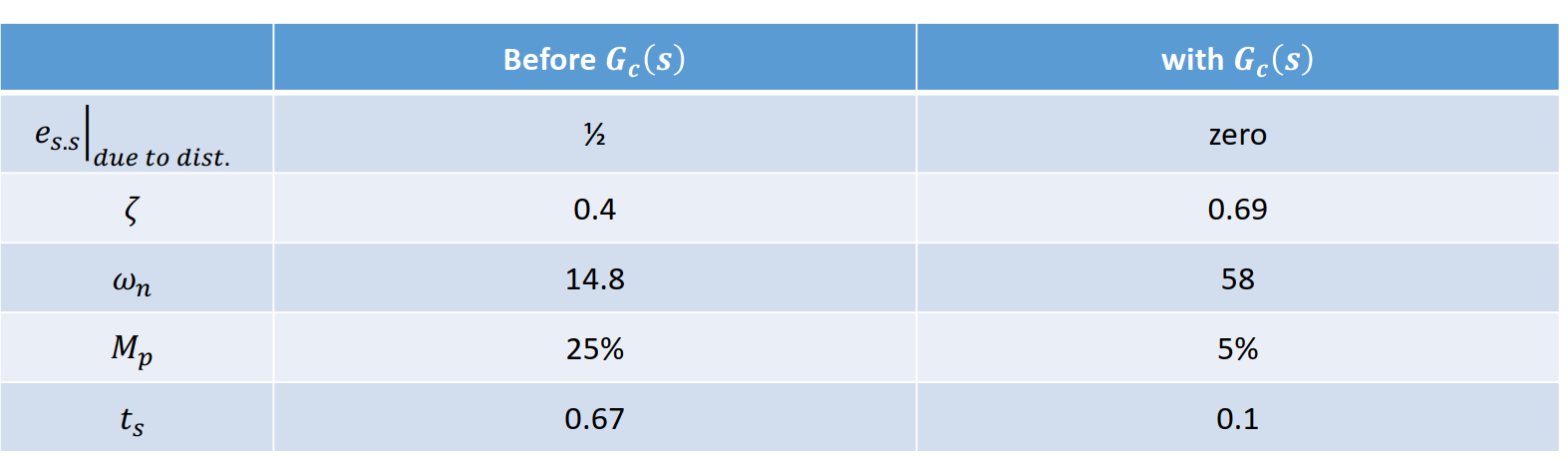
Q4)

Code:

%--------Q4-------- Step Response of T(s) (Closed-Loop)

% Plot step response of T(s)

draw\_step(T\_feedback, 'Closed-Loop System T(s)');

Output:

Ouput:

System: Closed-Loop System T(s)

Poles: -0.5-0.86603i -0.5+0.86603i

Stability: stable (all poles in LHP)

Over shoot MP: 16.2929% at t = 3.592 sec

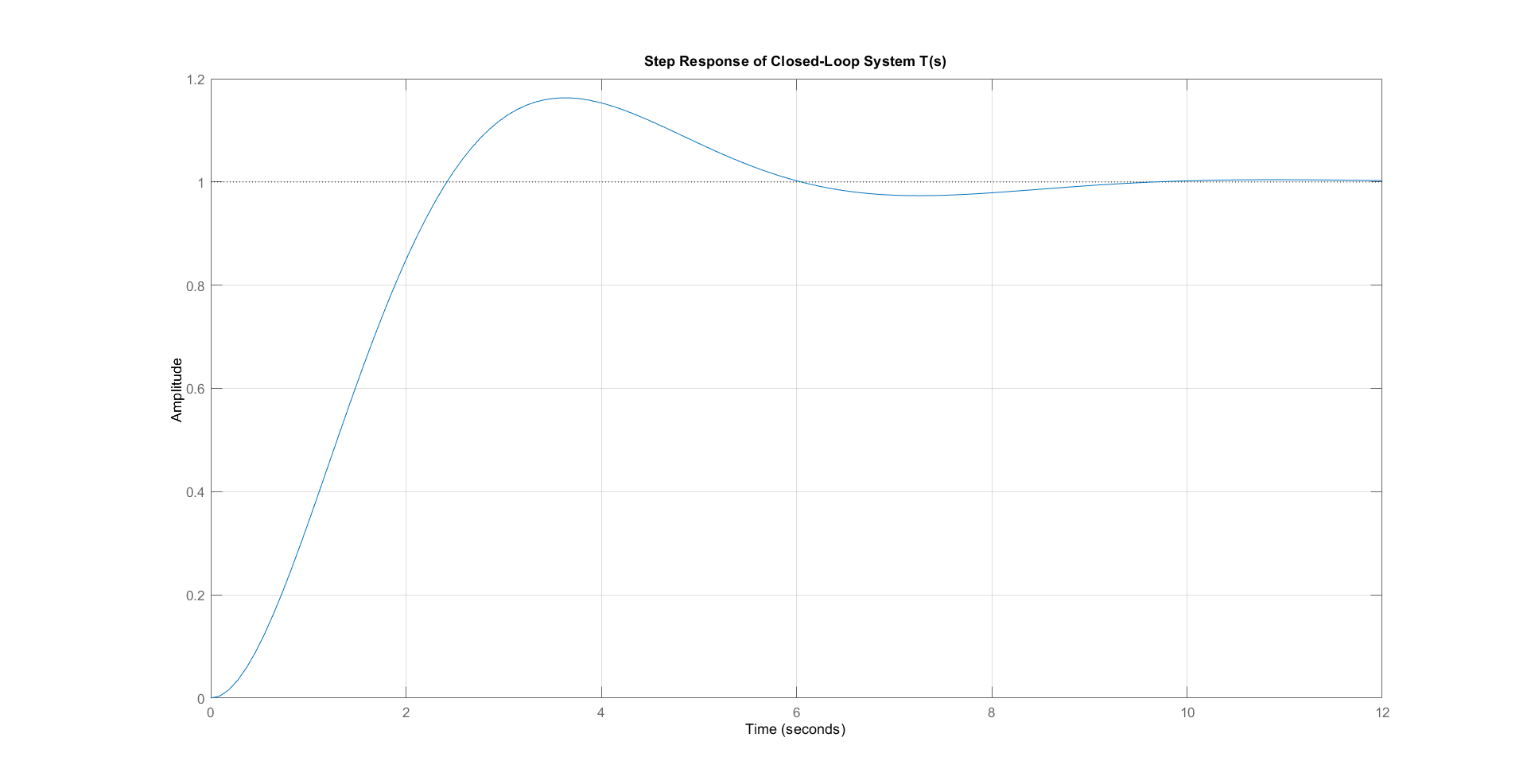
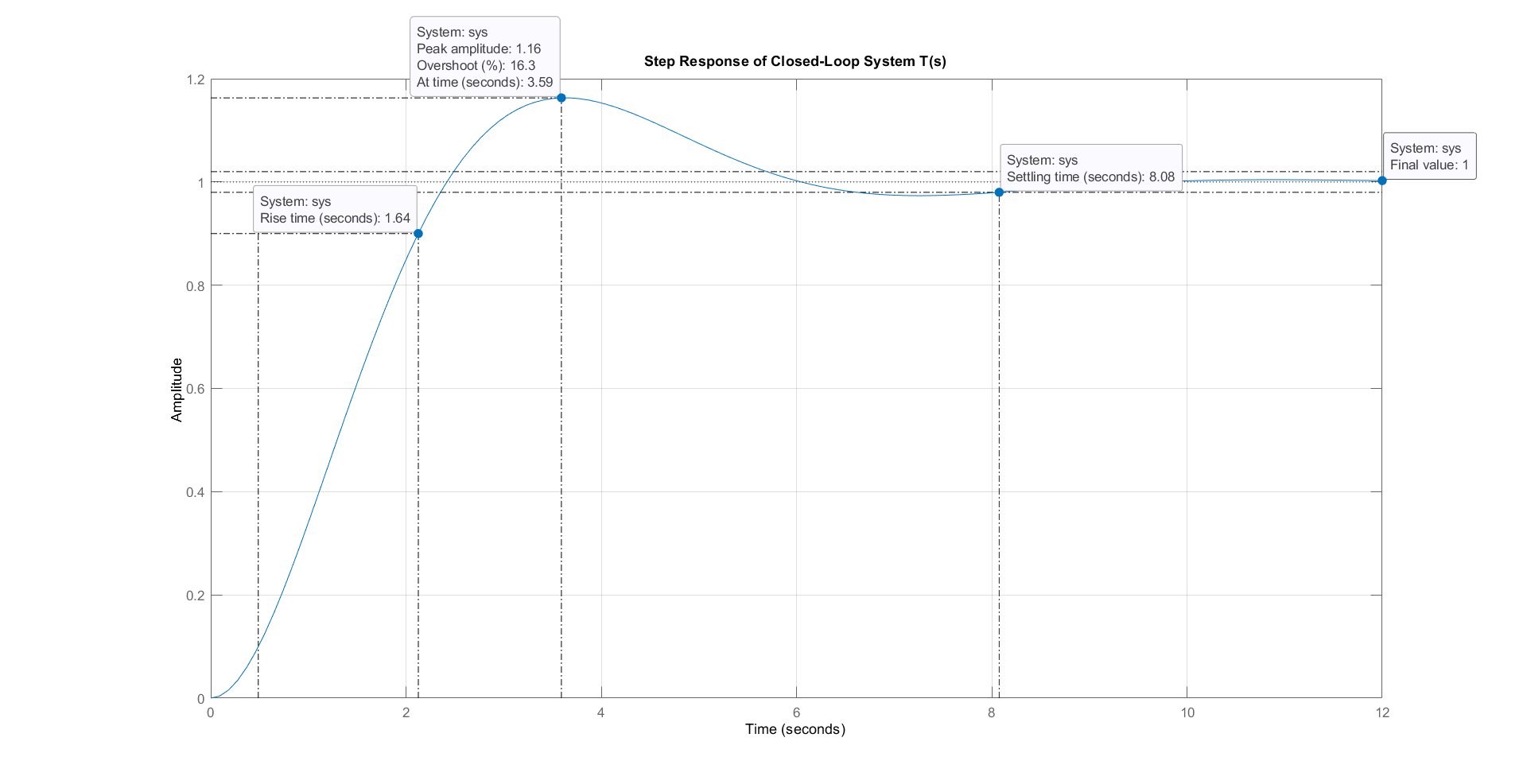
Damping ratio (ζ): 0.500

Natural frequency (ωn): 1.000 rad/s

Settling time (2%): 8.1051 sec

Rise time (10-90%): 1.6579 sec

Steady-state value: 1.0014



Q5)

Code:

function [poles] = draw\_poles(sys)

% Create figure

figure;

% Plot pole-zero map

pzmap(sys);

title(['Pole-Zero Map of: ' inputname(1)]);

grid on;

% Get poles

poles = pole(sys);

% Display poles

disp(['Poles of ' inputname(1) ':']);

disp(poles);

% Damping characteristics (for complex poles)

if ~isreal(poles)

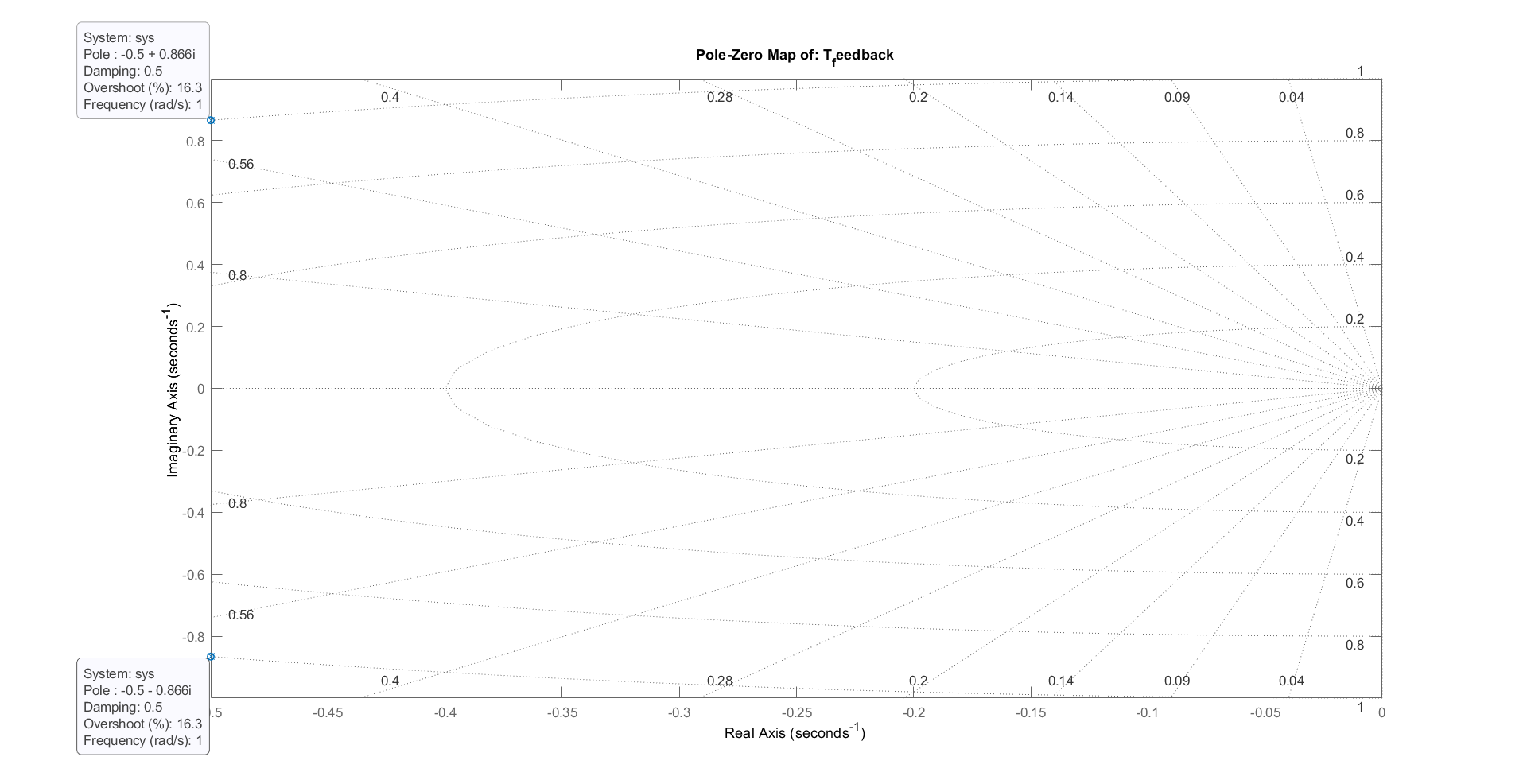
[wn, zeta] = damp(sys);

fprintf('Damping ratio (?): %.3f\n', zeta(1));

fprintf('Natural frequency (?n): %.3f rad/s\n', wn(1));

end

end

Output:

Poles of T\_feedback:

-0.5000 + 0.8660i

-0.5000 - 0.8660i

Damping ratio (ζ): 0.500

Natural frequency (ωn): 1.000 rad/s

Q6,Q7 is done

Q8

Code:

function [ess, t\_out, y\_out] = draw\_ramp(sys, t\_end, zoom\_time)

{

% Set defaults if not provided

if nargin < 2

t\_end = 100;

end

if nargin < 3

zoom\_time = 700;

end

% Create time vector

t = 0:0.1:t\_end;

%getting the ramp

ramp = tf(1,[1 0]);

% Get response data

[y\_sys, t\_sys] = step(sys.\*ramp, t);

[y\_ideal, t\_ideal] = step(ramp, t);

% Create figure with three subplots

figure;

% Subplot 1: Ideal ramp input

subplot(2,1,1);

plot(t\_ideal, y\_ideal, 'b');

hold on;

plot(t\_sys, y\_sys, 'r--');

title('Ramp Response');

xlabel('Time (sec)');

ylabel('Amplitude');

legend('Ideal', 'System', 'Location', 'northwest');

grid on;

hold off;

% Subplot 2: Zoomed comparison

subplot(2,1,2);

plot(t\_ideal, y\_ideal, 'b');

hold on;

plot(t\_sys, y\_sys, 'r--');

xlim([zoom\_time-50 zoom\_time+50]);

title(['Zoomed Comparison at t = ', num2str(zoom\_time), ' sec']);

xlabel('Time (sec)');

ylabel('Amplitude');

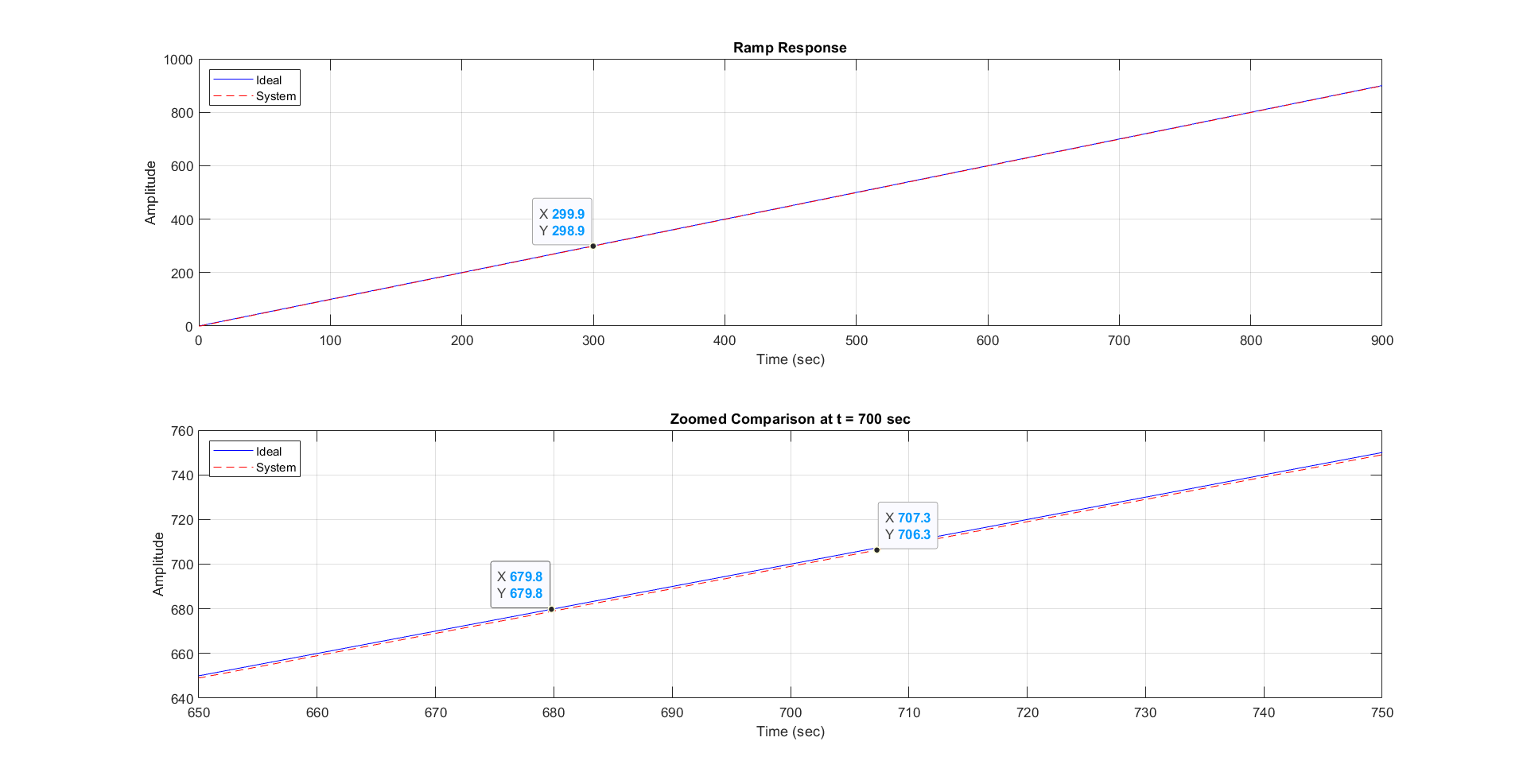
legend('Ideal', 'System', 'Location', 'northwest');

grid on;

hold off;

end

Output:



Steady-state error (ess): 1

Q9)

Code:

function [Gm, Pm, Wgc, Wpc] = draw\_Bode\_Plot(sys)

% BODE\_PLOT Analyzes system stability margins and compares margin()

% Bode\_Plot(sys)

%

% Input:

% sys - Transfer function (tf object or state-space model)

% Outputs:

% Gm - Gain margin (dB)

% Pm - Phase margin (degrees)

% Wgc - Gain crossover frequency (rad/sec)

% Wpc - Phase crossover frequency (rad/sec)

% Create margin plot

figure;

margin(sys);

grid on;

% Get stability margins

[Gm, Pm, Wgc, Wpc] = margin(sys);

% Display results

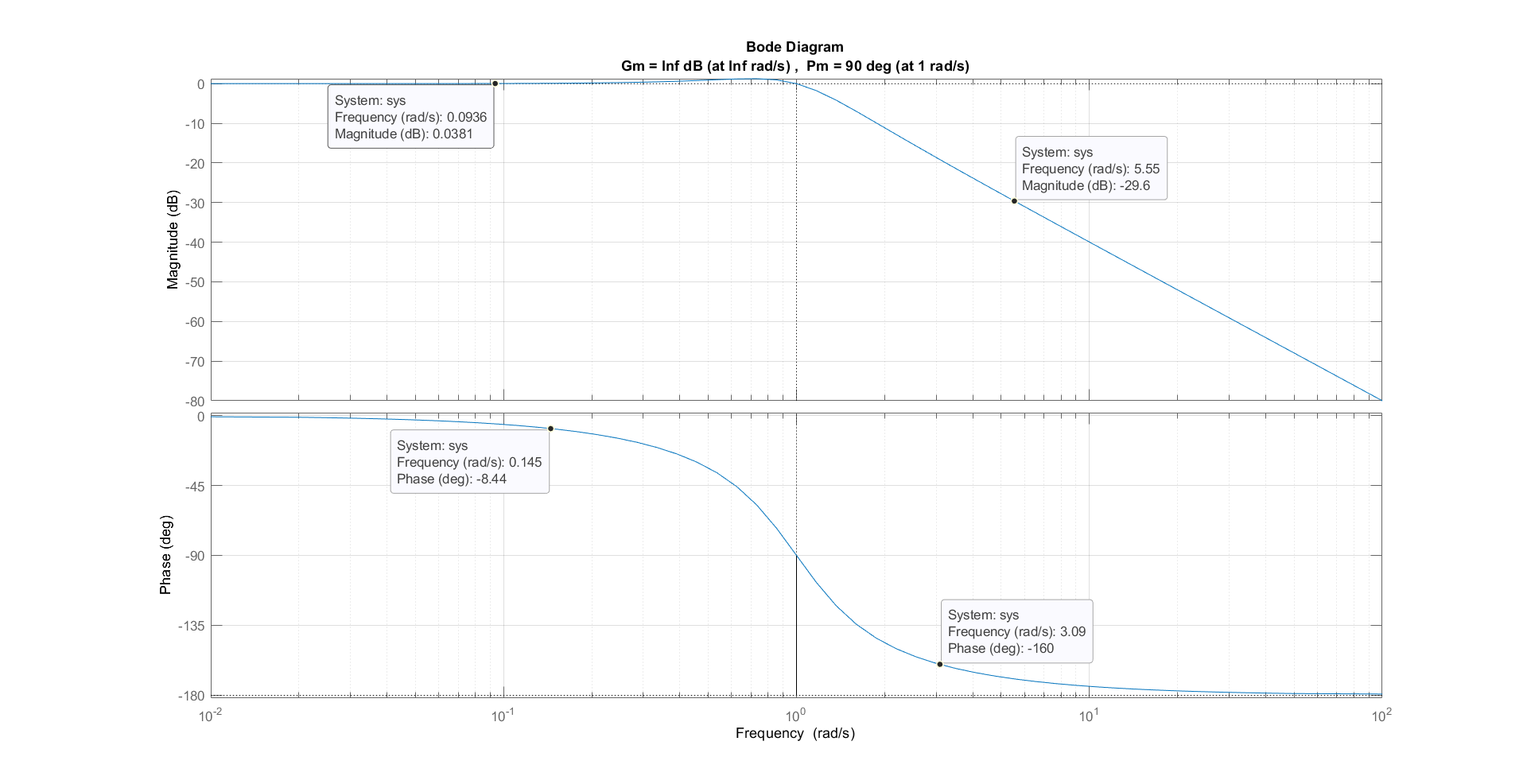
disp(['=== Stability Margins for ' inputname(1) ' ===']);

disp(['Gain Margin: ', num2str(Gm), ' dB at ', num2str(Wgc), ' rad/s']);

disp(['Phase Margin: ', num2str(Pm), '° at ', num2str(Wpc), ' rad/s']);

end

Output:



=== Stability Margins for T\_feedback ===

Gain Margin: Inf dB at Inf rad/s

Phase Margin: 90° at 1 rad/s