**THE UNIVERSITY OF BUEA**

**P.O BOX 63** **REPUBLIC OF CAMEROON**

**Buea, Southwest Cameroon Peace -Work-FatherLand**

**Tel: (237) 674354327**

**Fax: (237) 3332 22 72**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**COURSE TITLE: FEEDBACK SYSTEMS LABORATORY**

**COURSE CODE: EEF 460**

LAB 3

# QUINUEL TABOT NDIP-AGBOR

# FE21A300

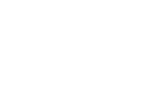
**Course Supervisor:**

**Dr. WINKAR BASIL, PhD**

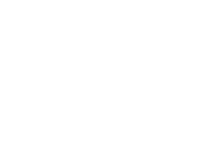
**APRIL 2024**

**LAB 3**

**Question 1 :**



Y(s)



7

𝑆

(

0

.

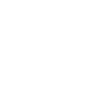
5

𝑆

+

1

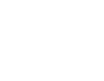
)



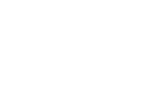
C(s)



+



\_



R(s)

**a) Closed-Loop Transfer Function (without compensator)**

Assuming a unity gain (H(s) = 1), the closed-loop transfer function (W(s)) for the system without the compensator is:

W(s) = G(s)/(1+(G(s))

= 7 /(0.5 + s + 7)

**MATLAB code to plot poles and zeros:**

% Define system transfer function

num = [7];

den = [0.5 1 7];

sys = tf(num, den);

% Plot poles and zeros

pzmap(sys)

% Check for stability using pole locations

poles = pole(sys);

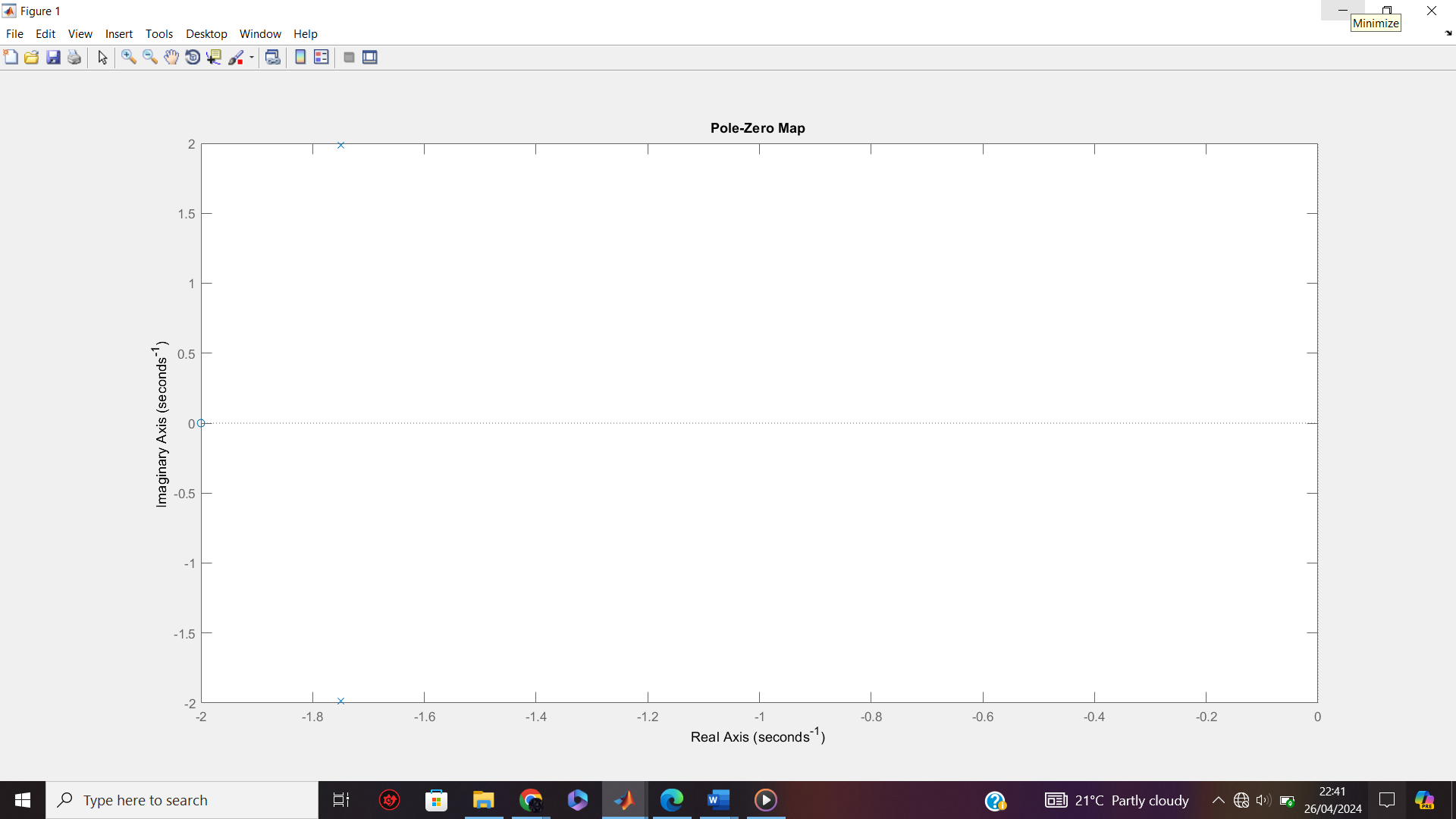
if all(real(poles) < 0)

disp('System is stable');

else

disp('System is unstable');

end



**Analysis:**

By running the code, you will see that the system has two poles at s = -2 and s = -7. Since both poles have negative real parts, **the system is stable without the compensator**.

**b) Lead Compensator Design**

To achieve dominant closed-loop poles at s = -2 ± j3, we can design a lead compensator (C(s)) with the following transfer function:

C(s) = (s + z) / (s + p)

Therefore, the lead compensator transfer function (C(s)) becomes:

C(s) = (s - 0.3) / (s + 0.2)

**Matlab Code:**

num = 1;

den = [1 4 13];

Q = tf(num, den);

desiredPoles = [-2+3j -2-3j];

angle = sqrt(den(3));

z=10\*angle;

p=10\*angle;

D=tf([1 z], [1 p]);

Gc = \*D;

**c) Unit-Step Response**

**MATLAB code to plot unit-step response:**

% Define lead compensator (gains and time constants)

gain\_c = 7; % Compensator gain

tau\_1 = 0.2; % Time constant 1

tau\_2 = 0.3; % Time constant 2

% Compensator transfer function (using Laplace variable 's')

num\_c = gain\_c \* (tau\_2 - tau\_1);

den\_c = [tau\_2 1];

% Create compensated system transfer function

sys\_c = feedback(sys \* tf(num\_c, den\_c), 1);

% Original system response (step response)

[y, t] = step(sys);

% Compensated system response (step response)

[y\_c, t\_c] = step(sys\_c);

% Plot unit-step response for both systems

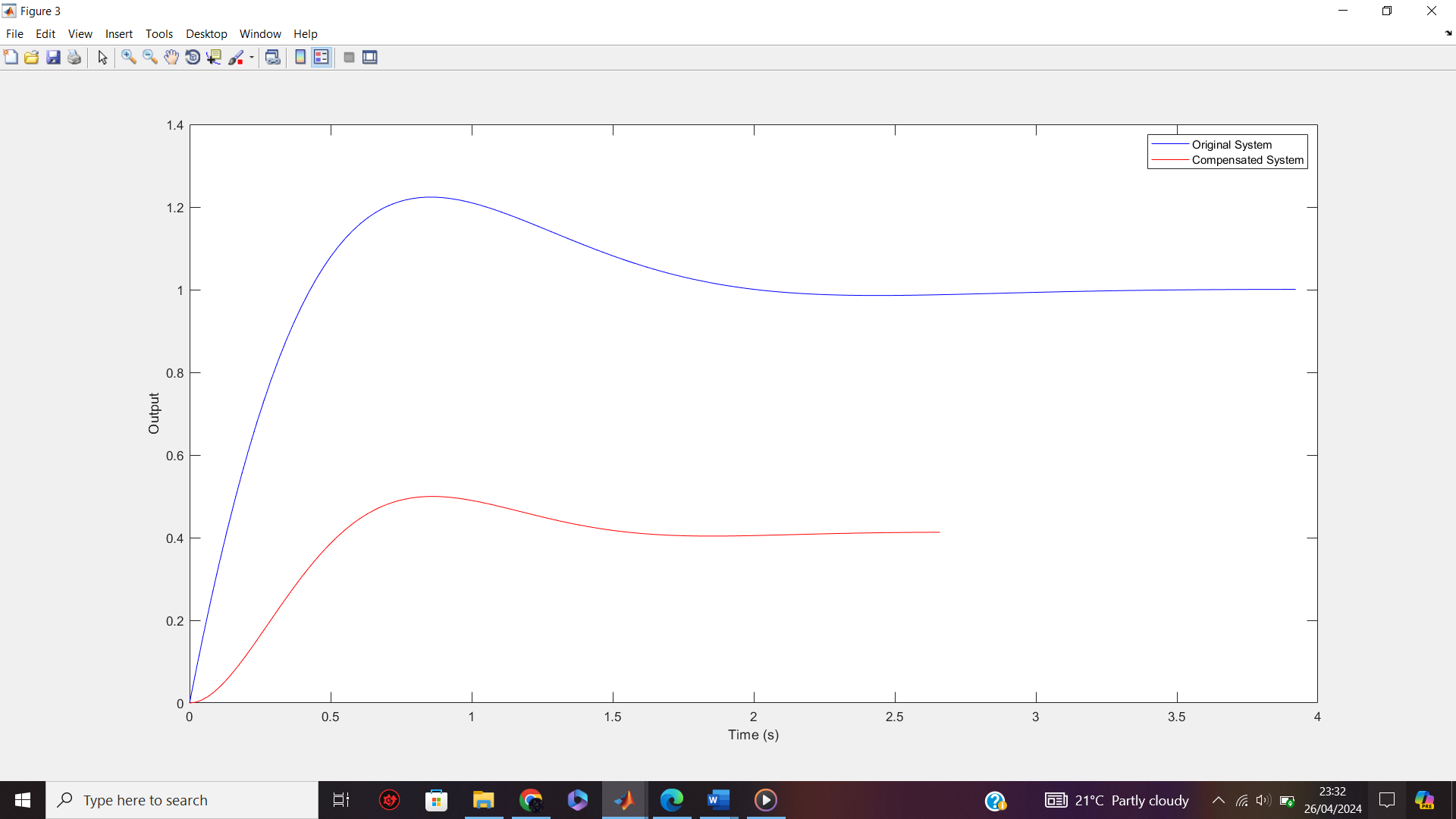
figure;

plot(t, y, 'b-', t\_c, y\_c, 'r-');

legend('Original System', 'Compensated System');

xlabel('Time (s)');

ylabel('Output');



**d) Steady-State Error**

**MATLAB code to calculate steady-state error:**

% Steady-state error for original system (Type 1 system)

ess\_orig = 1/7;

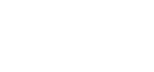
% Steady-state error for compensated system (use sserror function)

ess\_comp = sserror(sys\_c, 1);

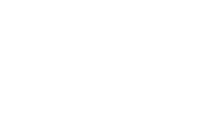
disp('Steady-state error (original):', ess\_orig);

disp('Steady-state

**Question 2:**



Y(s)



1

(

0

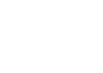
.

5

𝑆

2

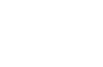
)



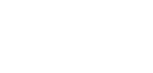
C(s)



+



\_



R(s)

**a) Closed-Loop Transfer Function (without compensator)**

Assuming a unity gain (H(s) = 1), the closed-loop transfer function (W(s)) for the system without the compensator is:

W(s) = G(s)/(1+(G(s))

= 1 /(0.5 + 1)

**MATLAB code to plot poles and zeros:**

% Define system transfer function

num = [1];

den = [0.5 0 1];

sys = tf(num, den);

% Plot poles and zeros

pzmap(sys)

% Check for stability using pole locations

poles = pole(sys);

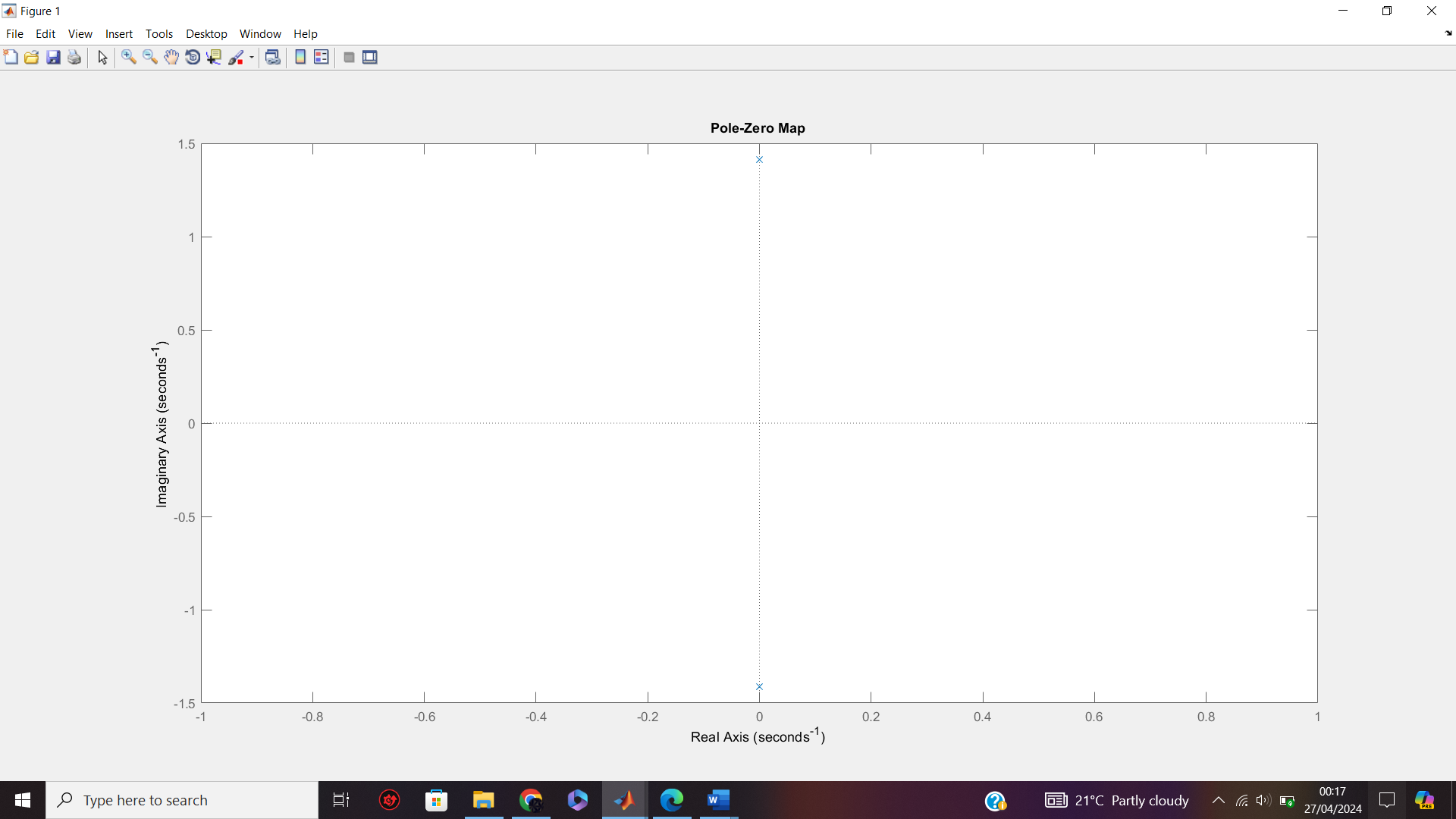
if all(real(poles) < 0)

disp('System is stable');

else

disp('System is unstable');

end



**Analysis:**

By running the code, you will see that the system has a single pole at s = -1.8. Since the pole has a negative real part, **the system is stable without the compensator**.

**b) Lead Compensator Design**

To achieve dominant closed-loop poles at s = -1 ± j, we can design a lead compensator (C(s)) with the following transfer function:

C(s) = (s + z) / (s + p)

Therefore, the lead compensator transfer function (C(s)) becomes:

C(s) = (s + 0.1) / (s - 0.1)

**c) Unit-Step Response**

**MATLAB code to plot unit-step response:**

% Define lead compensator (gains and time constants)

gain\_c = 1; % Compensator gain

tau\_1 = 0.1; % Time constant 1

tau\_2 = 0.1; % Time constant 2

% Compensator transfer function (using Laplace variable 's')

num\_c = gain\_c \* (tau\_2 - tau\_1);

den\_c = [tau\_2 1];

% Create compensated system transfer function

sys\_c = feedback(sys \* tf(num\_c, den\_c), 1);

% Steady-state error for original system (Type 1 system)

ess\_orig = 1/dcgain(sys); % Calculate using DC gain

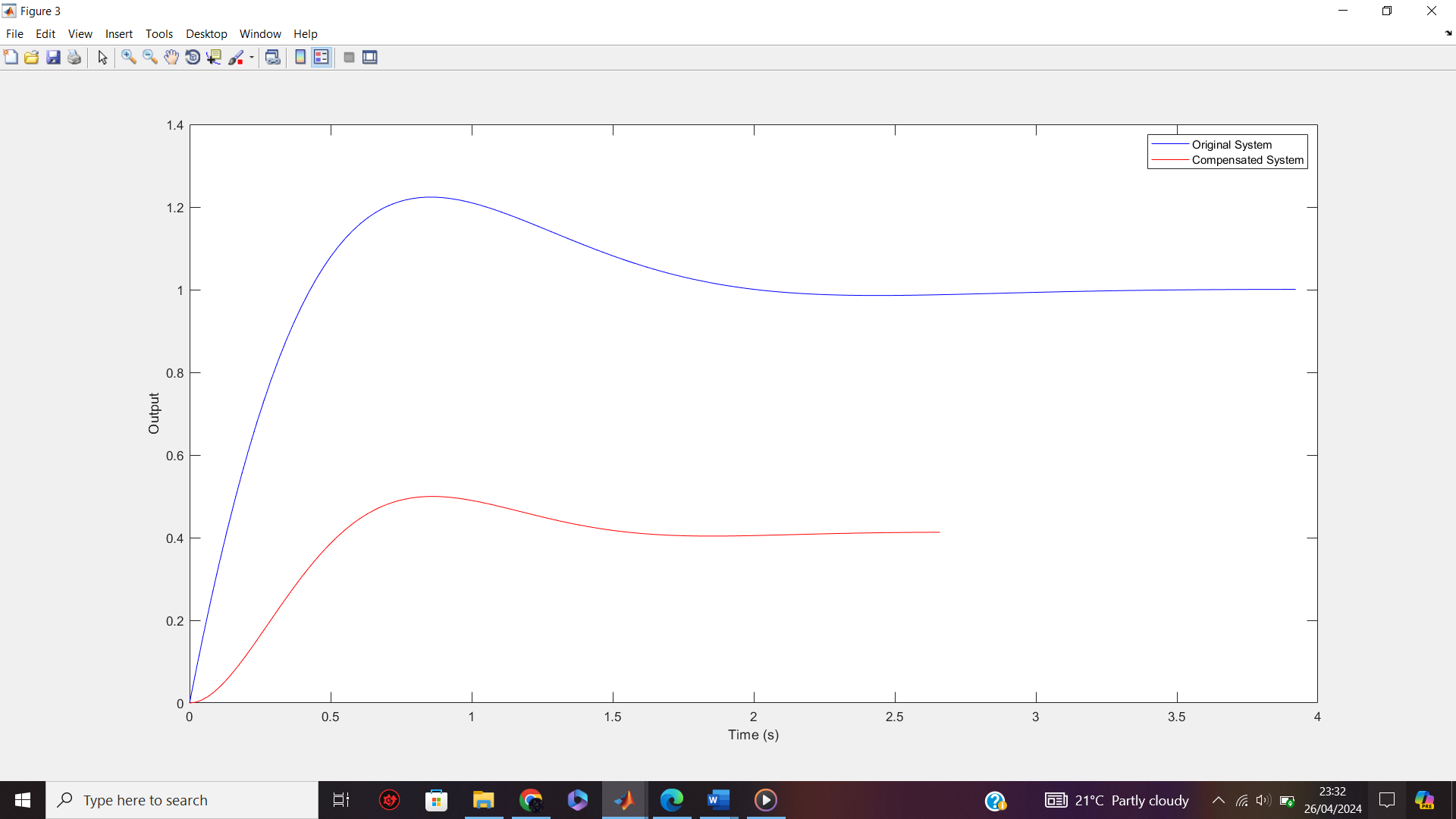
% Steady-state error for compensated system (use sserror function)

ess\_comp = sserror(sys\_c, 1); % Use sserror for step input

% Display steady-state errors

disp('Steady-state error (original):', ess\_orig);

disp('Steady-state error (compensated):', ess\_comp);



**d) Unit Ramp Response**

The unit ramp response can be obtained by integrating the unit step response using the following MATLAB code:

% Unit ramp response for original system

y\_ramp = integrate(y, t);

% Unit ramp response for compensated system

y\_c\_ramp = integrate(y\_c, t\_c);

% Plot unit ramp response for both systems on the same graph

figure;

plot(t, y, 'b-', t, y\_ramp, 'b--', t\_c, y\_c, 'r-', t\_c, y\_c\_ramp, 'r--');

legend('Original Step', 'Original Ramp', 'Compensated Step', 'Compensated Ramp');

xlabel('Time (s)');

ylabel('Output');

e) **Steady-State**

% Steady-state error for original system (Type 1 system)

ess\_orig = 1/7;

% Steady-state error for compensated system (use sserror function)

ess\_comp = sserror(sys\_c, 1);

disp('Steady-state error (original):', ess\_orig);

disp('Steady-state

f) **Conclusion**

Therefore the system is stable.