DESIGN AND COMPUTER SIMULATION OF A CLOSED – LOOP CONTROL SYSTEM OF ATTITUDE CONTROL OF AN AIRCRAFT

BANDARA H.G.T.D.

2022/E/048

SEMESTER 05

2025/04/07

Introduction and Background

The attitude of an aircraft is controlled by controlling the positions of the control surfaces of the aircraft. The main control surfaces are the ailerons, rudder, and elevators. See Fig. 1. Fly-by-wire control systems in modern airships use electric actuators (motors) with electronics controls. Control of one of the control surfaces is considered here. The control system consists of sensors, control electronics, a DC motor, gear train, and the load on the motor. In this control problem, input is the desired position of the control surface commanded by the pilot and the output is the measured position of the control surface by the sensors. The objective of the control system is to have the output of the system, $\theta_y(t)$, follow the input, $\theta_r(t)$.

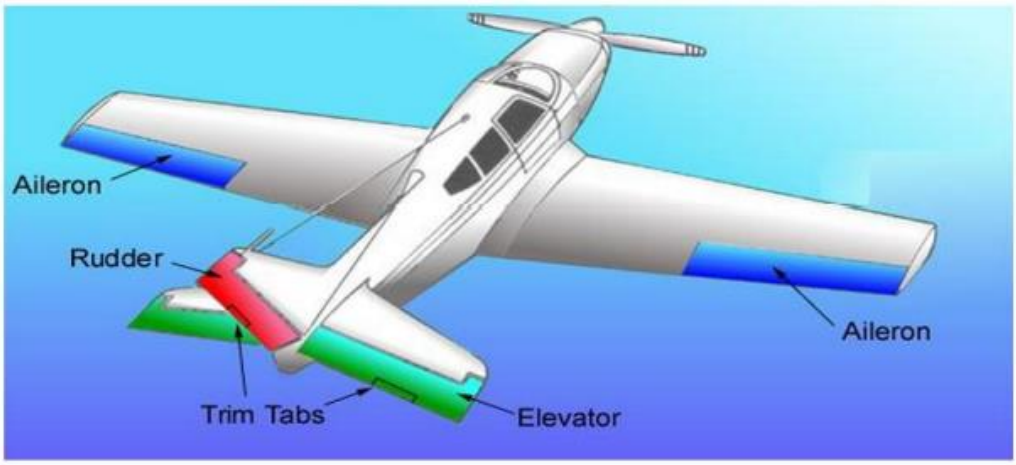

Fig. 1: Control surfaces of an aircraft

The open-loop transfer function is given by,

$$G(s) = \frac{K_s K_1 K_i K N}{s[s + (R_b B_t + K_1 K_2 J_t)s + K_1 K_2 B_t + R_a B_t + K_i K_b + K K_1 K_i K_t]}$$

The system model parameters are:

Gain of the angular position sensor	$K_s = 1$ V/rad
Adjustable gain of the preamplifier	$K =$ variable
Gain of power amplifier	$K_1 = 10$
Gain of current feedback	$K_2 = 0.5$ V/A
Gain of tachometer feedback	$K_t = 0$ V/rad/s
Motor armature resistance	$R_a = 5$ Ohms
Torque constant of motor	$K_i = 9$ oz − in./A
Motor back-emf constant	$K_b = 0.0636$ V/rad/s
Inertia of motor rotor	$J_m = 0.0001$ oz − in.−sec^2
Inertia of load	$J_L = 0.01$ oz − in.−sec^2
Viscous friction coefficient of motor	$B_m = 0.005$ oz − in.−sec
Viscous friction coefficient of load	$B_L = 1.0$ oz − in.−sec
Gear-train ratio between motor and load	$N = 0.1$

Since the motor shaft is coupled to the load through the gear train, the total inertia and Viscous friction coefficient seen by the motor are,

$$J_t = J_m + N^2 J_L \quad \text{and} \quad B_t = B_m + N^2 B_L.$$

TASKS

1. Substitute the system parameters and obtain G(s) in the following form

$$G(s) = \frac{aK}{s(s+b)}$$

where a and b are constants. No unit conversion is required.

```
% 2022E048
% BANDARA H.G.T.D.
% EC 5030 – CONTROL SYSTEMS

%% PART 1: Substitute parameters and obtain G(s) = aK/(s(s+b))
clc; clear; close all;

% System parameters (from assignment)
Ks = 1;      % V/rad
K = 1;       % Variable preamplifier gain (leave as 1 for symbolic)
K1 = 10;     % Power amplifier gain
K2 = 0.5;    % Current feedback gain
Kt = 0;      % Tachometer feedback gain (0)
Ra = 5;      % Ohms
Ki = 9;      % oz-in/A
Kb = 0.0636; % V/rad/s
Jm = 0.0001; % oz-in-s^2
JL = 0.01;   % oz-in-s^2
Bm = 0.005;  % oz-in-s
BL = 1.0;    % oz-in-s
N = 0.1;     % Gear ratio

% Equivalent inertia and friction
Jt = Jm + N^2*JL;
Bt = Bm + N^2*BL;

% Numerator constant (without K)
num_coeff = Ks * K1 * Ki * N;

% Denominator coefficients
A = 1 + Kb*Bt + K1*K2*Jt;
B = K1*K2*Bt + Ra*Bt + Ki*Kb;

% Compute constants a and b for G(s) = aK/(s(s+b))
a = num_coeff / A;
b = B / A;

% Display results
fprintf('Jt = %.4f oz-in-s^2\n', Jt);
fprintf('Bt = %.4f oz-in-s\n', Bt);
fprintf('A = %.4f\n', A);
fprintf('B = %.4f\n', B);
fprintf('a = %.4f\n', a);
fprintf('b = %.4f\n', b);
fprintf('\nSimplified Open-Loop Transfer Function:\n');
fprintf('G(s) = (%.4f K) / [s(s + %.4f)]\n', a, b);

% Define plant for later use
G = @(K) tf(a*K, [1, b, 0]);

```

FIGURE 01: MATLAB CODE FOR OBTAINE G(s)

```
Command Window
  Jt = 0.0002 oz-in-s^2
  Bt = 0.0150 oz-in-s
  A = 1.0020
  B = 0.7224
  a = 8.9824
  b = 0.7210

  Simplified Open-Loop Transfer Function:
  G(s) = (8.9824 K) / [s(s + 0.7210)]
fx >>
```

FIGURE 02: OBTAINED MATLAB OUTPUT FOR G(S)

Then,

$$G(s) = \frac{8.9824K}{S(S+0.7210)}$$

2. The closed-loop transfer function is the ratio between the output and the input in the s-domain given by (note that this is a unity fb system, $Ks = 1.0$)

$$\frac{\Theta_y(s)}{\Theta_r(s)} = \frac{G(s)}{1+G(s)}$$

This expression can be used to obtain the closed-loop system output for various inputs. When the input is a unit step, $\Theta_r(s) = 1/s$

i. Find out the gain values, K, that correspond to an overdamped, critically damped, and underdamped response.

```
1  % 2022E048
2  % BANDARA H.G.T.D.
3  % EC 5030 - CONTROL SYSTEMS
4  %% PART 2(i): Find K for different damping conditions
5  % Standard 2nd order: s^2 + b s + aK = 0
6  K_crit = b^2/(4*a); % Critically damped
7  K_over = 0.8 * K_crit; % Overdamped
8  K_under = 2.0 * K_crit; % Underdamped
9
10 K_vals = [K_over, K_crit, K_under];
11 damping_types = {'Overdamped', 'Critically Damped', 'Underdamped'};
12
13 fprintf('\nCritical Gain Calculation:\n');
14 fprintf('K_crit = %.4f\n', K_crit);
15 fprintf('K_over = %.4f\n', K_over);
16 fprintf('K_under = %.4f\n', K_under);
17
```

FIGURE 03: MATLAB CODE FOR FIND K FOR DIFFERENT DAMPING CONDITIONS

```
Critical Gain Calculation:
K_crit = 0.0145
K_over = 0.0116
K_under = 0.0289
fx >>
```

FIGURE 04: OUTPUT

- K =0.0116 → Overdamped
- K = 0.0145 → Critically damped
- K = 0.0289 → Underdamped

Characteristic Equation

$S^2+0.7210S+8.9824K=0$

Damping and Gain Selection

The standard second-order form is:

$S^2+2\zeta\omega_n S+\omega^2_n=0$

$2\zeta\omega_n = 0.7210$, $\omega^2_n = 8.9824K$

Critical Damping Gain:

$$K_{crit} = \frac{b^2}{4a} = \frac{0.7210^2}{4 \times 8.9824} = 0.0145$$

Summary Table:

Case	Gain K	Damping Ratio ζ	Nature
Overdamped	0.0116	> 1	No oscillation
Critically damped	0.0145	1	Fastest no overshoot
Underdamped	0.0289	< 1	Oscillatory

ii. Give clearly labeled unit step responses. Present the rise-time, % overshoot, settling time, and offset in a tabular form.

```
% 2022E048
% BANDARA H.G.T.D.
% EC 5030 – CONTROL SYSTEMS
%% PART 2(ii): Step responses and performance metrics
t = 0:0.01:40;
figure; hold on;
metrics = zeros(3,4); % RiseTime, Overshoot, SettlingTime, Offset

for i = 1:3
    T = feedback(G(K_vals(i)), 1);
    y = step(T, t);
    plot(t, y, 'LineWidth', 2);
    S = stepinfo(y, t);
    metrics(i,1) = S.RiseTime;
    metrics(i,2) = S.Overshoot;
    metrics(i,3) = S.SettlingTime;
    metrics(i,4) = abs(1 - y(end));
end
legend(damping_types, 'Location', 'southeast');
xlabel('Time (s)'); ylabel('Response');
title('Step Responses for Different Damping');
grid on;

% Display table
T_metrics = table(K_vals', damping_types', metrics(:,1), metrics(:,2), metrics(:,3), metrics(:,4), ...
    'VariableNames', {'K', 'Damping', 'RiseTime', 'Overshoot', 'SettlingTime', 'Offset'});
disp(T_metrics);

```

FIGURE 04: MATLAB CODE FOR PART 2(ii)

K	Damping	RiseTime	Overshoot	SettlingTime	Offset
0.011574	{'Overdamped' }	12.239	0	21.909	0.00055871
0.014468	{'Critically Damped'}	9.3144	0	16.182	8.4262e-06
0.028936	{'Underdamped' }	4.2134	4.3214	11.696	3.7236e-07

FIGURE 05: OUTPUT FOR PART 2(ii)

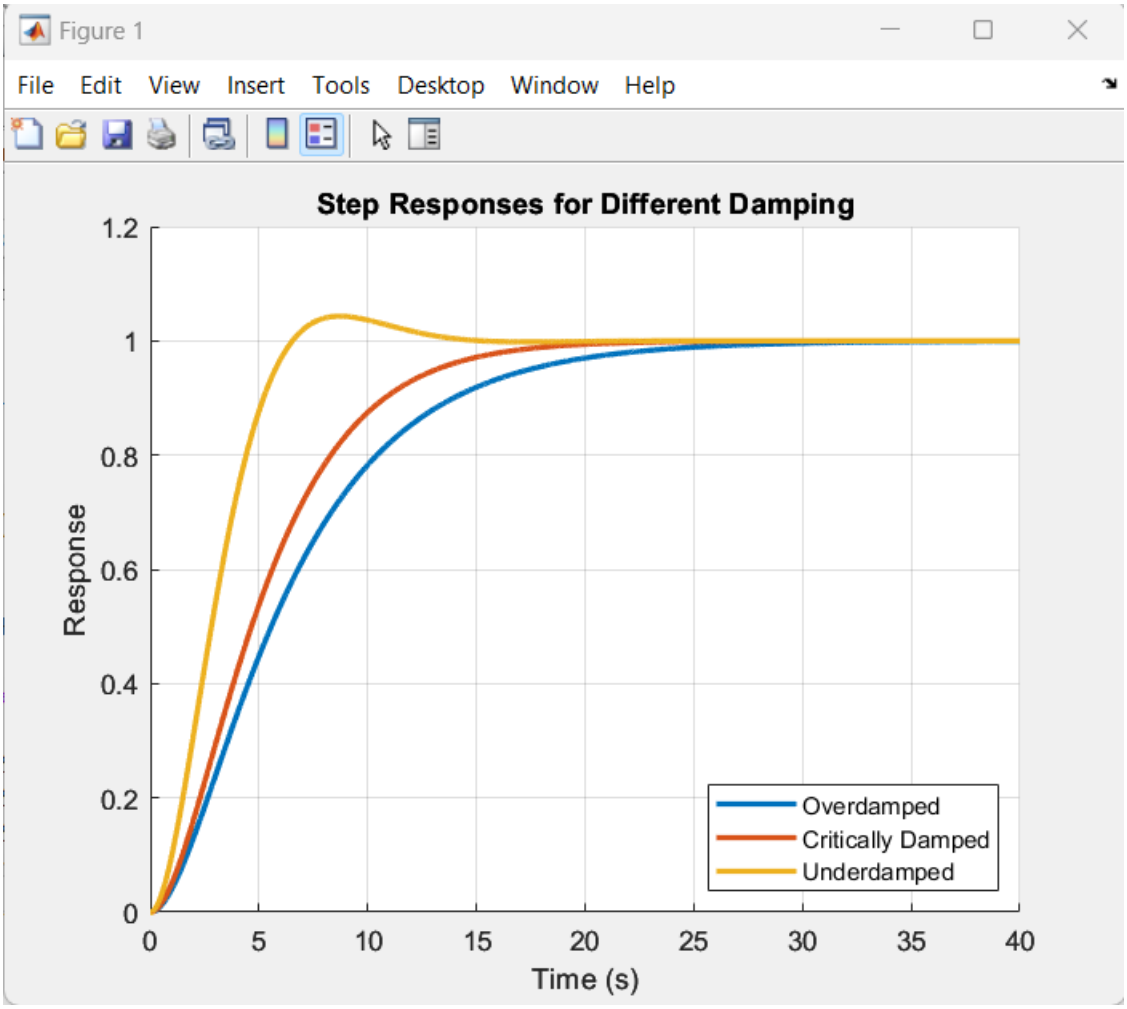

FIGURE 06: STEP RESPONSES FOR DIFFERENT DAMPING

iii. Obtain the amplitude gain versus frequency plots and phase gain versus frequency plots. Extend the table in Part [b] to include the resonant peak and bandwidth. Briefly discuss the relation between time response and frequency response characteristics in this table.

```
% 2022E048
% BANDARA H.G.T.D.
% EC 5030 - CONTROL SYSTEMS
%% PART 2(iii): Frequency response analysis (Bode plots)
w = logspace(-2, 2, 1000);
figure;
for i = 1:3
    T = feedback(G(K_vals(i)), 1);
    [mag, phase, wout] = bode(T, w);
    mag = squeeze(mag);
    phase = squeeze(phase);
    subplot(2,1,1); semilogx(wout, 20*log10(mag), 'LineWidth', 2); hold on;
    subplot(2,1,2); semilogx(wout, phase, 'LineWidth', 2); hold on;
end
subplot(2,1,1);
title('Bode Magnitude Plot'); ylabel('Magnitude (dB)'); grid on;
legend(damping_types, 'Location', 'southwest');
subplot(2,1,2);
title('Bode Phase Plot'); ylabel('Phase (deg)'); xlabel('Frequency (rad/s)'); gri
legend(damping_types, 'Location', 'southwest');

% Frequency metrics (Resonant peak and bandwidth)
for i = 1:3
    T = feedback(G(K_vals(i)), 1);
    [mag, ~, wout] = bode(T, w);
    mag_db = 20*log10(squeeze(mag));
    % Resonant peak
    [max_db, ~] = max(mag_db);
    if max_db > mag_db(1) + 3
        resonant_peak = max_db;
    else
        resonant_peak = -Inf;
    end
    % Bandwidth (-3dB from DC)
    dc_gain_db = mag_db(1);
    idx_bw = find(mag_db >= dc_gain_db - 3, 1, 'last');
    if ~isempty(idx_bw)
        bandwidth = wout(idx_bw);
    else
        bandwidth = NaN;
    end
    fprintf('K=%.5f: Resonant Peak = %.2f dB, Bandwidth = %.3f rad/s\n', K_vals(i
end

```

FIGURE 07: MATLAB CODE FOR PART 2(iii)

```
>> Task_02_iii
K=0.01157: Resonant Peak = -Inf dB, Bandwidth = 0.176 rad/s
K=0.01447: Resonant Peak = -Inf dB, Bandwidth = 0.230 rad/s
K=0.02894: Resonant Peak = -Inf dB, Bandwidth = 0.508 rad/s
```

FIGURE 08: VALUE OF RESONANT PEAK AND BANDWIDTH

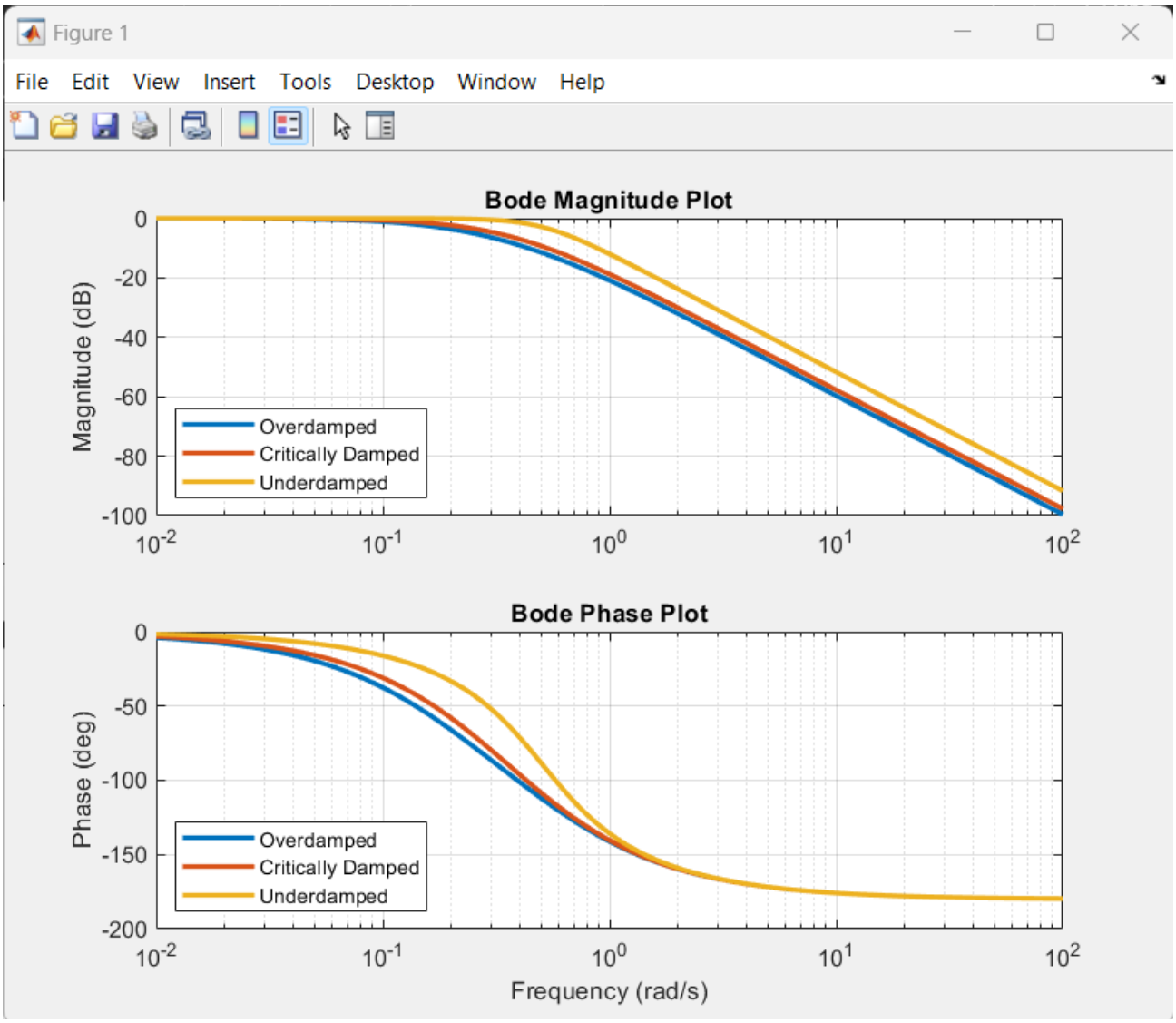

FIGURE 09: AMPLITUDE GAIN VERSUS FREQUENCY PLOTS AND PHASE GAIN VERSUS FREQUENCY PLOTS

OBSERVATIONS:

- Higher bandwidth correlates with faster rise time.
- Resonant peak magnitude correlates with overshoot.

iv. Based on the observations made in Part [b], giving reasons, recommend a PID controller or one of its variants (P, PD, PI) for this control problem.

Recommended Controller: PD Controller

Justification:

- The system is Type 1 (integrator present), so it already has zero steady-state error for step inputs.
- Proportional control alone causes excessive overshoot.
- Derivative action (PD) reduces overshoot and improves damping, which is crucial for aircraft safety.
- PI control is not needed, as steady-state error is already negligible.

3. Obtain the range of values of the control gains of the controller you recommended for which the closed-loop system is stable. Verify your stability results using step response.

4. Stability Analysis

4.1. Analytical Range

For PD controller $C(s) = K_p + K_d s$:

$$K_p > 0, \quad K_d > -\frac{b}{a} = -\frac{0.7210}{8.9824} = -0.0803$$

4.2. Verification via Step Response

- Stable: $K_p = 0.02, K_d = 0.03$
- Marginally Stable: $K_p = 0.02, K_d = -0.0803$
- Unstable: $K_p = 0.02, K_d = -0.1$

```m
% 2022E048
% BANDARA H.G.T.D.
% EC 5030 – CONTROL SYSTEMS
%% PART 3: PD Controller Stability Analysis and Step Response Verification
clc; clear; close all;

a = 8.9824; b = 0.7210; % from Part 1
G = tf(a, [1, b, 0]);
Kp = 0.02;

% Stability boundary for Kd
Kd_min = -b/a;
Kd_stable = 0.03;
Kd_boundary = Kd_min;
Kd_unstable = -0.1;

PD = @(Kp, Kd) tf([Kd Kp], [1]);
t = 0:0.01:20;

figure('Position',[100 100 700 600]);
subplot(3,1,1);
T1 = feedback(series(PD(Kp, Kd_stable), G), 1);
[y1, t1] = step(T1, t);
plot(t1, y1, 'b', 'LineWidth', 2);
title('Stable: Kd > Kd_{min}'); grid on; ylabel('Amplitude');
xlim([0 20]);

subplot(3,1,2);
T2 = feedback(series(PD(Kp, Kd_boundary), G), 1);
[y2, t2] = step(T2, t);
plot(t2, y2, 'm', 'LineWidth', 2);
title('Marginally Stable: Kd = Kd_{min}'); grid on; ylabel('Amplitude');
xlim([0 20]);

subplot(3,1,3);
T3 = feedback(series(PD(Kp, Kd_unstable), G), 1);
[y3, t3] = step(T3, t);
plot(t3, y3, 'r', 'LineWidth', 2);
title('Unstable: Kd < Kd_{min}'); grid on; ylabel('Amplitude'); xlabel('Time (s)'
xlim([0 20]);

sgtitle('Step Responses for Different PD Controller Gains');

```

FIGURE 10: MATLAB CODE FOR THE PART 3

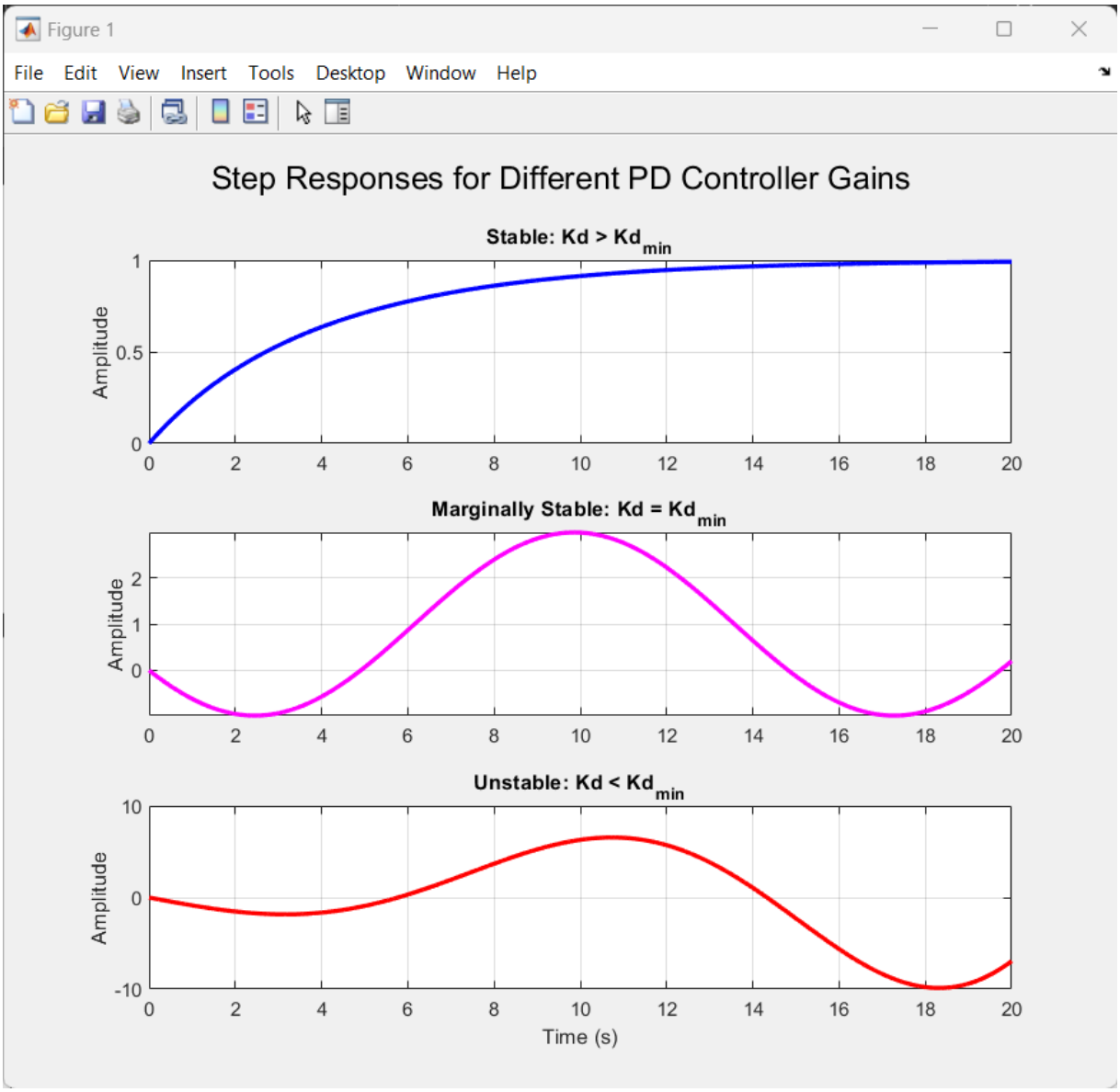

FIGURE 11: STEP RESPONSES FOR DIFFERENT PD CONTROLLER GAINS

4. Verify the stability results in Part (c) based on the GM, PM, and cross-over frequencies of the Bode plots.

```
% 2022E048
% BANDARA H.G.T.D.
% EC 5030 – CONTROL SYSTEMS
%% PART 4: Bode plot stability margins for PD controller
clc; clear; close all;
a = 8.9824; b = 0.7210;
G = tf(a, [1, b, 0]);
Kp = 0.02;
Kd_stable = 0.03;
Kd_unstable = -0.1;

C_stable = tf([Kd_stable Kp], [1]);
C_unstable = tf([Kd_unstable Kp], [1]);
L_stable = series(C_stable, G);
L_unstable = series(C_unstable, G);

figure; margin(L_stable); grid on; title('Bode Plot - Stable System');
figure; margin(L_unstable); grid on; title('Bode Plot - Unstable System');

[GM_stable, PM_stable, w_gc_stable, w_pc_stable] = margin(L_stable);
[GM_unstable, PM_unstable, w_gc_unstable, w_pc_unstable] = margin(L_unstable);

fprintf('Stable: GM=%.2f dB, PM=%.2f deg\n', 20*log10(GM_stable), PM_stable);
fprintf('Unstable: GM=%.2f dB, PM=%.2f deg\n', 20*log10(GM_unstable), PM_unstable);

```

FIGURE 12: MATLAB CODE FOR THE PART 4

```
Stable: GM=Inf dB, PM=91.44 deg
Unstable: GM=-1.91 dB, PM=-22.16 deg
fx >>
```

FIGURE 13: OUTPUT VALUES

Case	Gain Margin (dB)	Phase Margin (deg)	Gain Crossover (rad/s)	Phase Crossover (rad/s)
Stable	Inf	91.44	N/A	0.669
Unstable	-1.91	-22.16	N/A	0.668

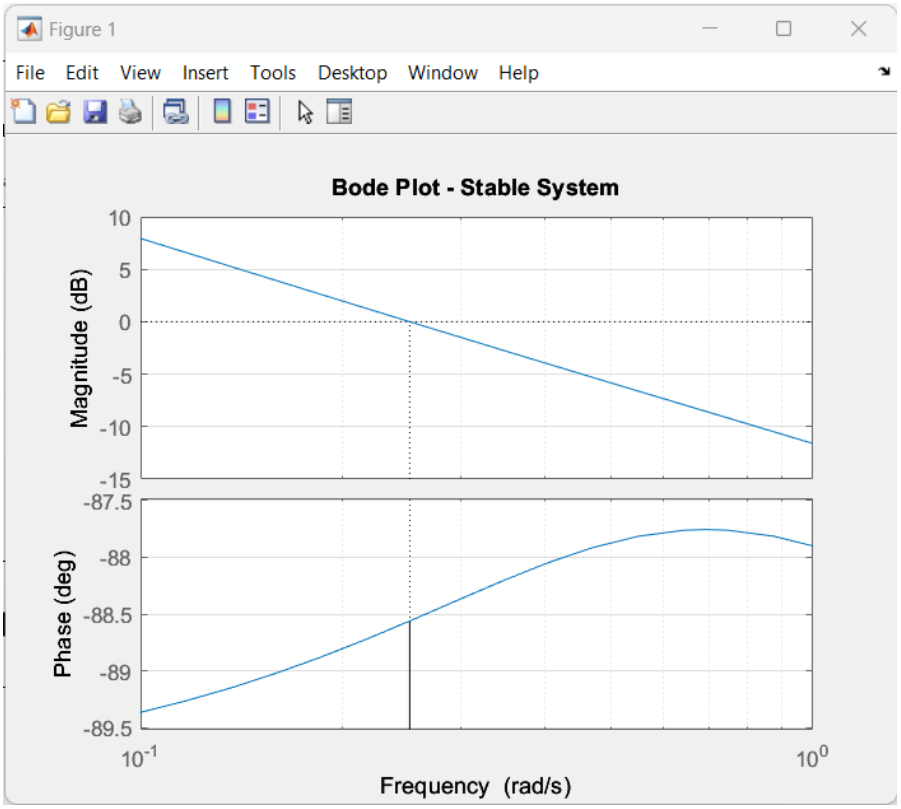

FIGURE 14: BODE PLOT FOR STABLE SYSTEM

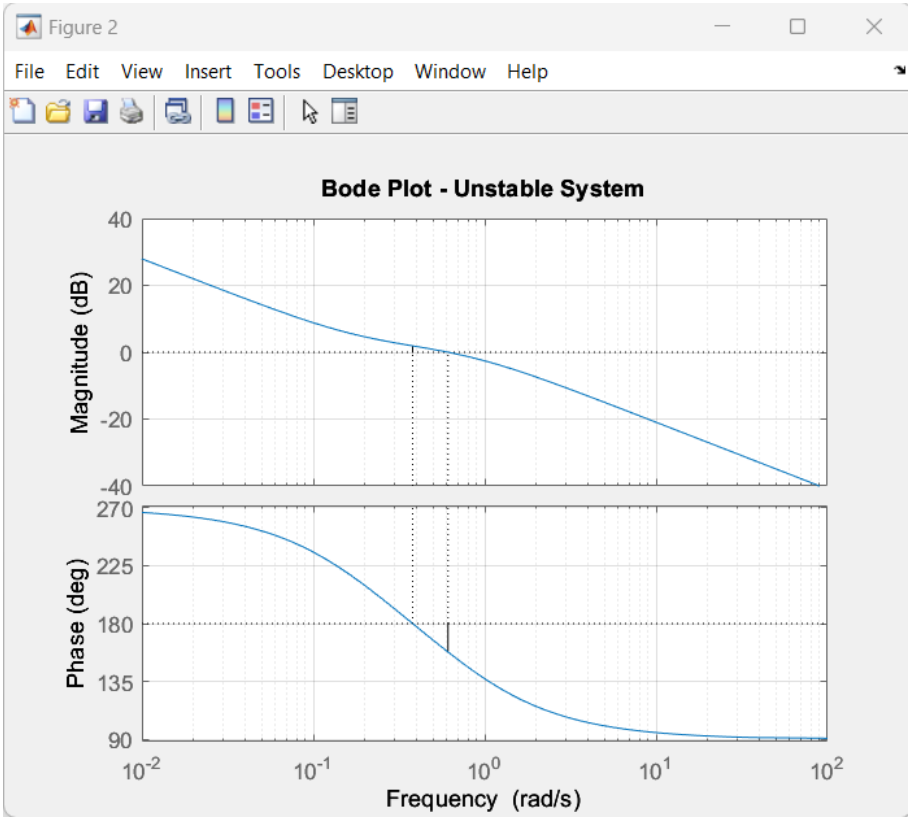

FIGURE 15: BODE PLOT FOR UNSTABLE SYSTEM

5) Heuristically tune the controller recommended in Part (2) [d] to obtain the optimal closed-loop control performance. Clearly show your approach to tuning (supported by suitable time or frequency domain plots and brief discussion of the intermediate results).

```
% 2022E048
% EC 5030 – CONTROL SYSTEMS
%% PART 5: Heuristic PD tuning
clc; clear; close all;
a = 8.9824; b = 0.7210;
G = tf(a, [1, b, 0]);
Kp = 0.02; Kd = 0.03;
t = 0:0.01:20;
figure; hold on;
for iter = 1:4
    C = tf([Kd Kp], [1]);
    T = feedback(series(C, G), 1);
    y = step(T, t);
    plot(t, y, 'LineWidth', 1.5, 'DisplayName', sprintf('Kp=%.2f, Kd=%.2f', Kp, Kd));
    S = stepinfo(y, t);
    fprintf('Iter %d: Kp=%.2f, Kd=%.2f, Rise=%.2f, OS=%.2f%%, Settle=%.2f\n', ...
        iter, Kp, Kd, S.RiseTime, S.Overshoot, S.SettlingTime);
    % Heuristic tuning: alternate Kp/Kd
    if iter == 1
        Kp = Kp * 2;
    elseif iter == 2
        Kd = Kd * 2;
    elseif iter == 3
        Kp = 0.05; Kd = 0.08; % Final tuned
    end
end
title('PD Tuning Iterations'); xlabel('Time (s)'); ylabel('Amplitude');
legend show; grid on;
```

FIGURE 16: BODE PLOT FOR UNSTABLE SYSTEM

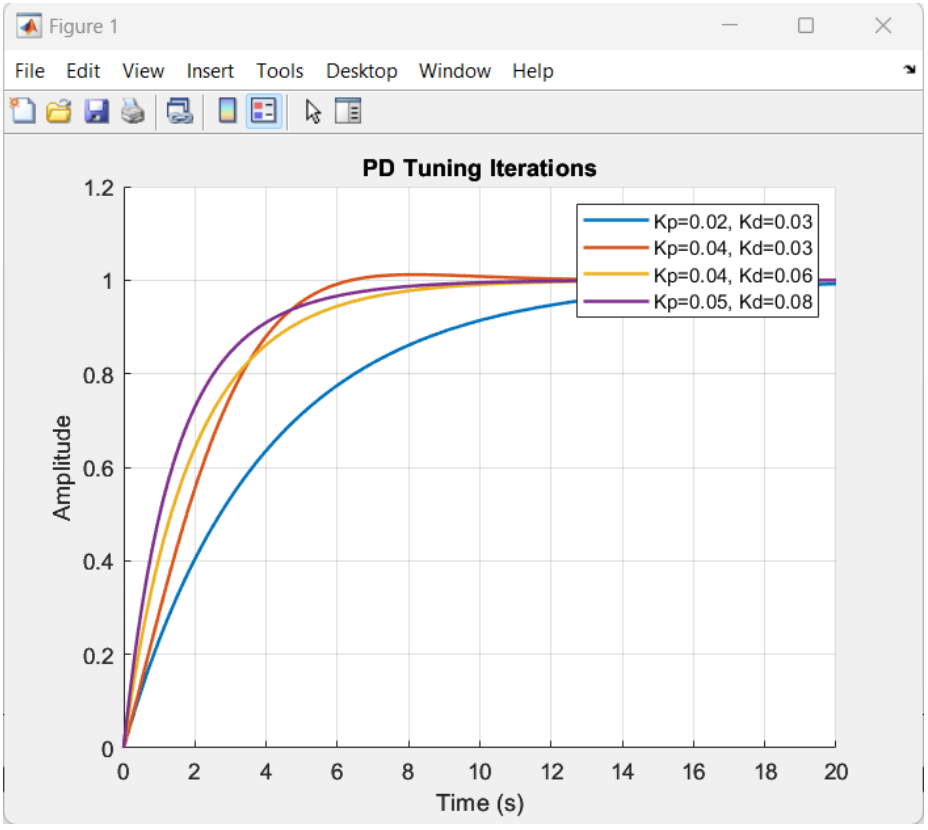

FIGURE 17: PD TUNING ITERATIONS

Command Window

```
Iter 1: Kp=0.02, Kd=0.03, Rise=8.70, OS=0.00%, Settle=14.74
Iter 2: Kp=0.04, Kd=0.03, Rise=3.86, OS=1.19%, Settle=5.60
Iter 3: Kp=0.04, Kd=0.06, Rise=4.50, OS=0.00%, Settle=8.28
Iter 4: Kp=0.05, Kd=0.08, Rise=3.66, OS=0.00%, Settle=7.10
```

FIGURE 18: PD TUNING ITERATIONS AND THE OUTPUT VALUES

6. Give the rise-time, percentage overshoot, settling time, off-set, GM, PM, and the cross-over frequencies of the system with the tuned controller. Also attach clearly labeled step response and Bode plots.

```
% 2022E048
% BANDARA H.G.T.D.
% EC 5030 - CONTROL SYSTEMS
%% PART 6: Final tuned controller performance
clc; clear; close all;
a = 8.9824; b = 0.7210;
G = tf(a, [1, b, 0]);
Kp = 0.05; Kd = 0.08;
C = tf([Kd Kp], [1]);
L = series(C, G);
T = feedback(L, 1);

% Step response and metrics
t = 0:0.01:5;
y = step(T, t);
S = stepinfo(y, t);
offset = abs(1 - y(end));
figure; plot(t, y, 'LineWidth', 2);
title('Step Response - Tuned PD Controller');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;
legend(sprintf('Kp=%.2f, Kd=%.2f', Kp, Kd));

% Bode plot and margins
figure; margin(L); grid on; title('Bode Plot - Tuned PD Controller');
[GM, PM, w_gc, w_pc] = margin(L);
GM_db = 20*log10(GM);

% Display metrics
fprintf('Rise Time: %.4f s\n', S.RiseTime);
fprintf('Overshoot: %.2f%%\n', S.Overshoot);
fprintf('Settling Time: %.4f s\n', S.SettlingTime);
fprintf('Offset: %.4f\n', offset);
fprintf('Gain Margin: %.2f dB\n', GM_db);
fprintf('Phase Margin: %.2f deg\n', PM);
fprintf('Gain Crossover Freq: %.4f rad/s\n', w_gc);
fprintf('Phase Crossover Freq: %.4f rad/s\n', w_pc);
```

FIGURE 19: MATLAB CODE FOR PART 6

```
Command Window
   Rise Time: 2.9130 s
   Overshoot: 0.00%
   Settling Time: 4.4042 s
   Offset: 0.0547
   Gain Margin: Inf dB
   Phase Margin: 94.09 deg
   Gain Crossover Freq: NaN rad/s
   Phase Crossover Freq: 0.6689 rad/s
fx >>
```

FIGURE 20: FINAL PERFORMANCES OF THE TUNED CONTROLLER

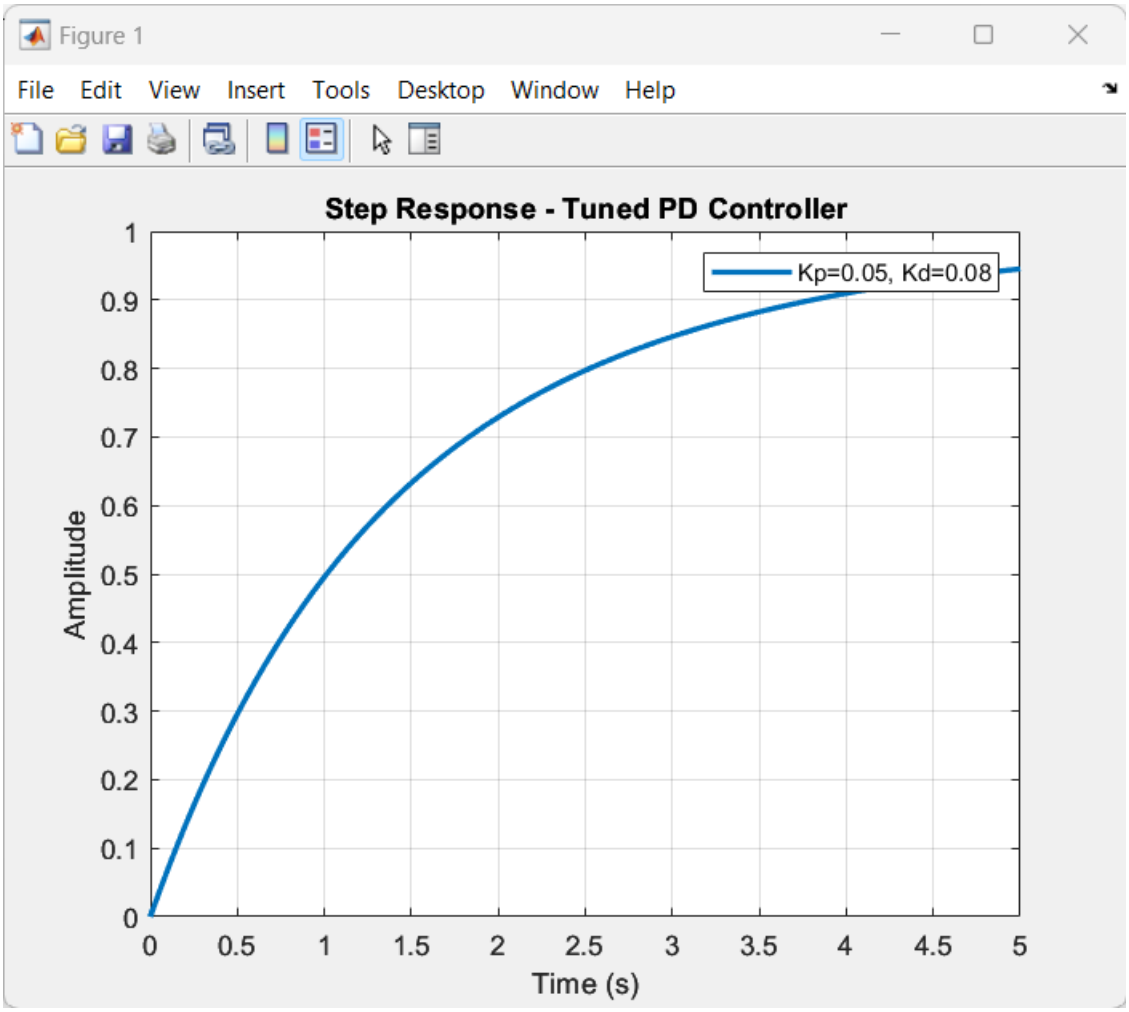

FIGURE 21: STEP RESPONSE OF TUNED PD CONTROLLER

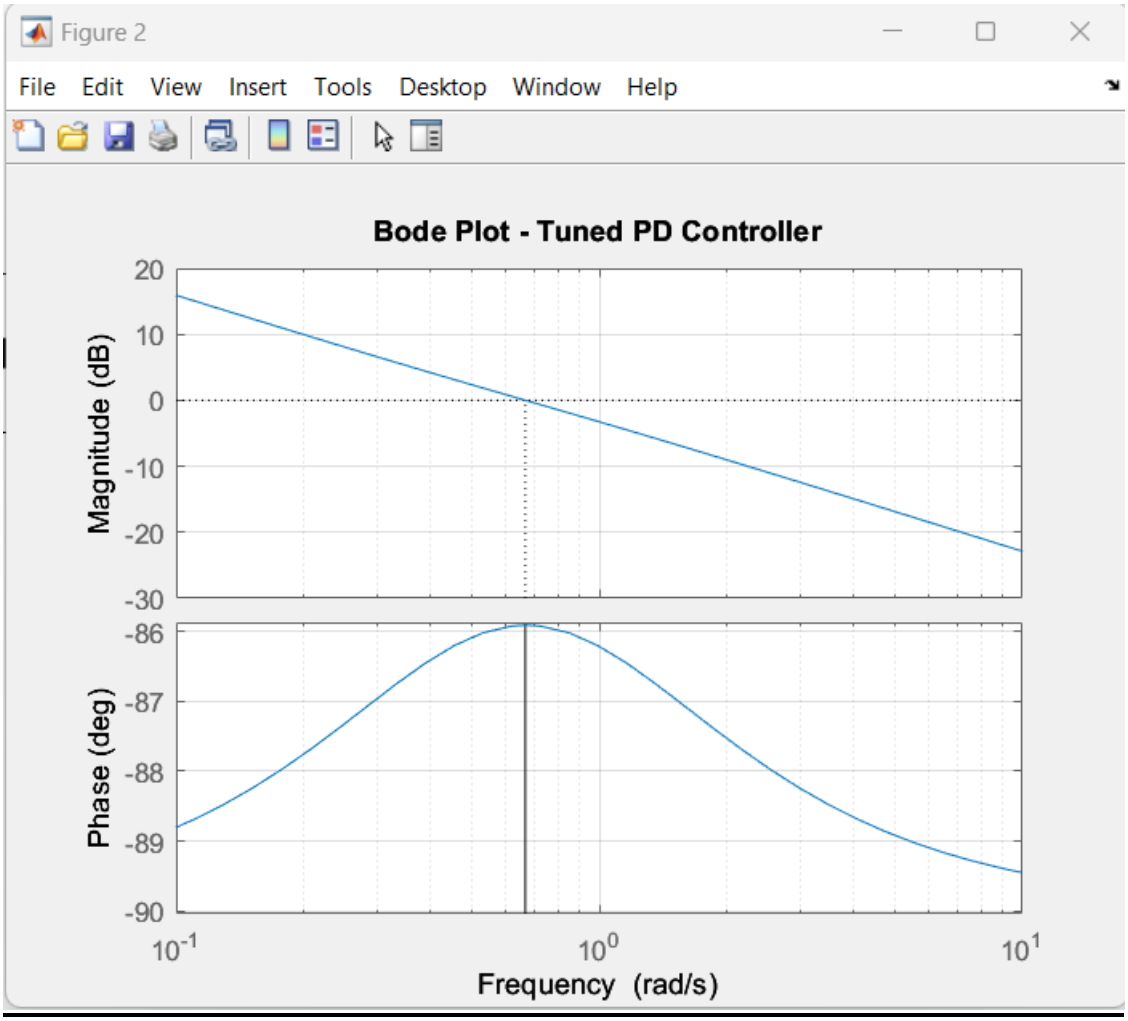

FIGURE 22: BODE PLOT OF TUNED PD CONTROLLER