Electronics and Electrical Communications Engineering Department

Faculty of Engineering

Cairo University

Submitted to: Dr. Mina, Dr Ragia, Eng Mohamed Nady

MATLAB Control Project

Final Report submitted for course ELC4040 "Digital Control System"

4th Year

1st Semester - Academic Year 2025/2026

Prepared by:

NAME	SECTION	ID
Yousef Khaled Omar Mahmoud	4	9220984
Yousef Ashraf Mohamed Abotalib	4	9220972

Submission Date: 6 December 2025

1. Table of Contents

2. Table of Figures

3. List of Tables

4. Introduction

Modern control systems aim to achieve stability, performance, and robustness in the presence of system dynamics and external disturbances. In this project, a **digital controller** is designed for a sampled-data control system representing a practical physical process. The goal is to ensure that the closed-loop system responds accurately to changes in the reference input while maintaining stability and desirable transient characteristics.

The given plant consists of a continuous-time process and a sensor feedback element. Due to the digital implementation requirement, the system must operate with a sampling rate defined by the project specifications. Therefore, the controller design process involves:

- Modeling the continuous-time plant
- Discretizing the system using **Zero-Order Hold (ZOH)**
- Designing a **Lead compensator** in the discrete domain
- Ensuring that performance objectives such as steady-state accuracy and sufficient phase margin are satisfied

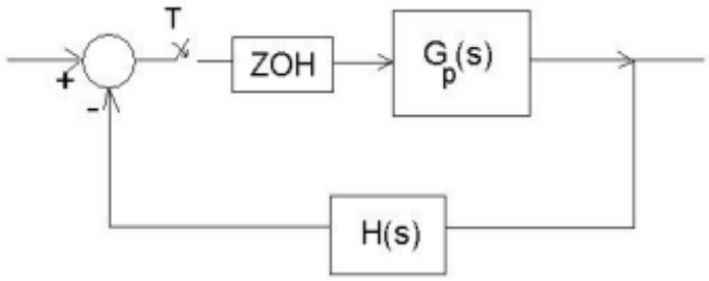
The main performance targets defined for the system are:

- A **10% steady-state error** to a unit-step input
- A **Phase Margin (PM) ≥ 50°** in the open-loop design
- A stable and smooth transient response with acceptable overshoot and settling time

To achieve these goals, the **SISO Design Tool** in MATLAB is utilized to tune the compensator directly in the discrete domain. The final validated controller is then analyzed through step responses, root locus visualization, and frequency-domain stability margins.

5. System Modeling

MATLAB Control Project



Consider the control loop shown above with:

$$G_p(s) = \frac{4}{(2s+1)(0.5s+1)} \quad , \quad H(s) = \frac{1}{0.05s+1} \quad and \quad T = 0.1s$$

Figure 1 System Block Diagram

From figure 1, The digital control system under consideration consists of a continuous-time plant Gp(s) and a sensor feedback element H(s), operating under discrete-time control with a Zero-Order Hold (ZOH) and sampling time:

$$[T = 0.1; \text{seconds}]$$

The continuous-time plant dynamics are:

$$G_p(s) = \frac{4}{(2s+1)(0.5s+1)}$$

This represents a **second-order stable plant** with two real poles located at:

$$s = -0.5 \quad \text{and} \quad s = -2$$

The feedback sensor is modeled as a first-order low-pass system:

$$H(s) = \frac{1}{0.05s+1}$$

This introduces an additional real pole at:

$$s = -20$$

Open-Loop Continuous Model

The continuous loop transfer function before sampling is:

$$G(s)H(s) = \frac{4}{(2s+1)(0.5s+1)(0.05s+1)}$$

This configuration results in a **third-order system**, and initial investigations showed that the uncompensated design lacks adequate stability margins and performance when converted to the discrete domain.

Discrete-Time Modeling

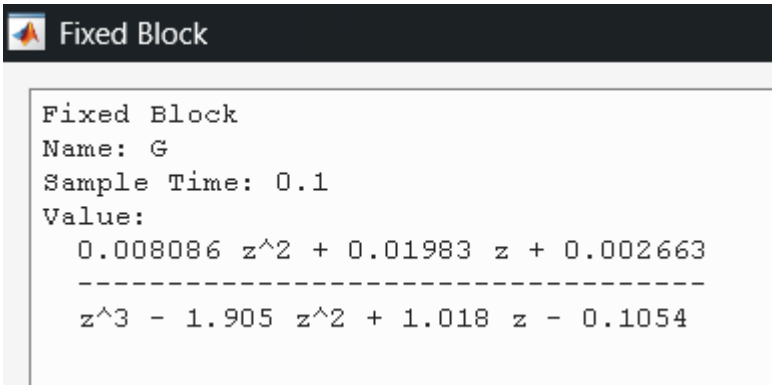


Figure 2 Discrete Open-loop plant[1]

To support digital controller implementation, the continuous-time plant and sensor are discretized using **Zero-Order Hold (ZOH)**:[3]

From Figure 2, The resulting **discrete open-loop plant** obtained from MATLAB is:

$$Gd(z)Hd(z) = \frac{0.008086z2 + 0.01983z + 0.002663}{z3 - 1.905z2 + 1.018z - 0.1054}$$

This conversion accurately preserves the effect of the digital-to-analog interface at the actuator.

Control Objectives

From the system model, the design goals are defined as:

- Maintain **closed-loop stability**
- Achieve **steady-state error ≤ 10%** for a unit-step input
- Ensure **Phase Margin ≥ 50°** in open-loop design for robustness
- Maintain smooth transient response with acceptable overshoot and settling time

This modeling stage establishes the mathematical foundation on which the compensator is designed and validated in the subsequent sections.

6. System Analysis Before Controller Design

To evaluate the baseline dynamics of the system, the continuous-time plant and feedback sensor were analyzed in open-loop configuration. The goal is to understand stability margins and performance limitations before introducing any control compensation.

Continuous-Time Open-Loop Response

the continuous open-loop transfer function:

$$L(s) = Gp(s)H(s) = \frac{4}{(2s+1)(0.5s+1)(0.05s+1)}$$

was examined. The Bode plot below is seen in Figure 3 and 4:

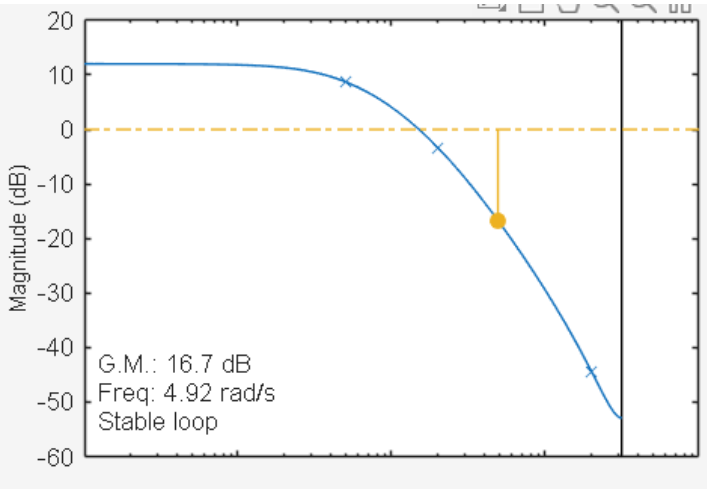


Figure 3 System Gain Margin[1]

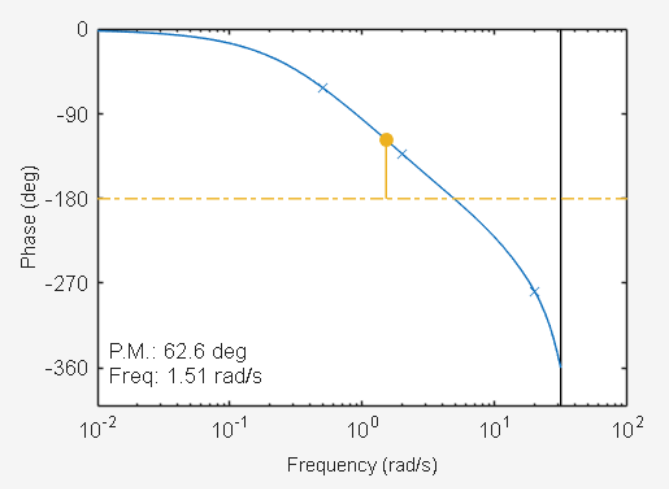


Figure 4 System Phase Margin[1]

- **Very low phase margin**
- **Low gain margin**
- **Slow response dynamics**

Performance Metric	Value	Interpretation
Steady-state gain	≈ 0.80	**20% steady-state error** (too high)
Settling time	≈ 3.9 s	Too slow
Overshoot	≈ 19%	not Acceptable
Rise time	≈ 0.7 s	Moderate

Discrete System Stability Insights

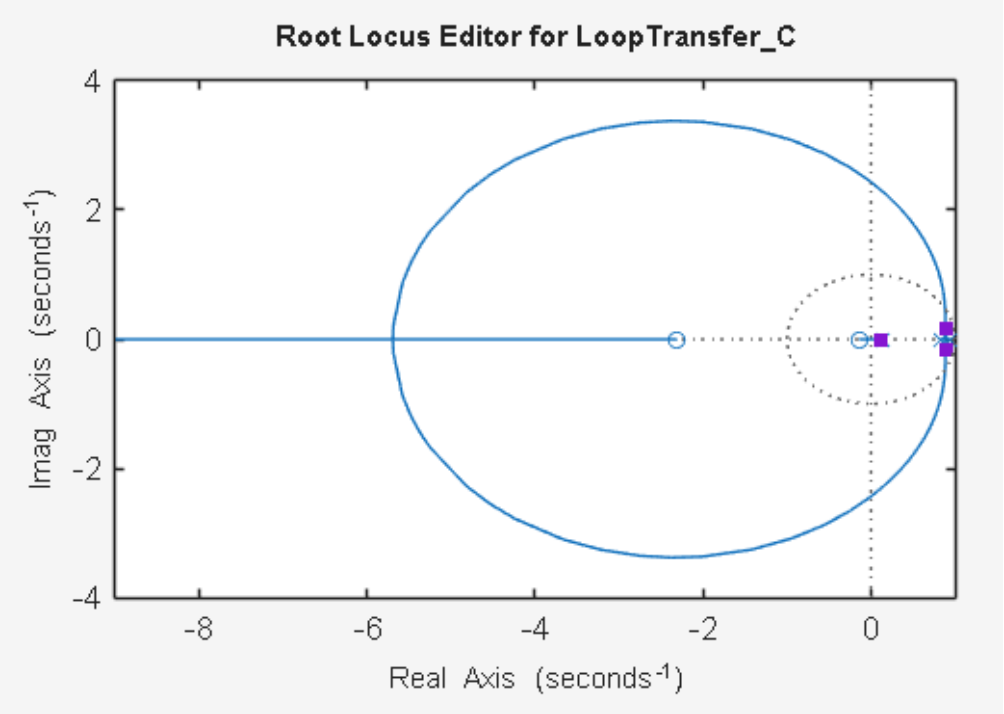


Figure 5 System Root Locus[1]

This root locus plot shows a **stable 4th-order control system** (LoopTransfer_C). All poles start in the left-half plane, and the system remains stable for all gains shown. Increasing the gain improves damping (reduces oscillation) but slows down the response. There is no risk of instability in the given gain range, offering flexibility in tuning the controller for desired performance.

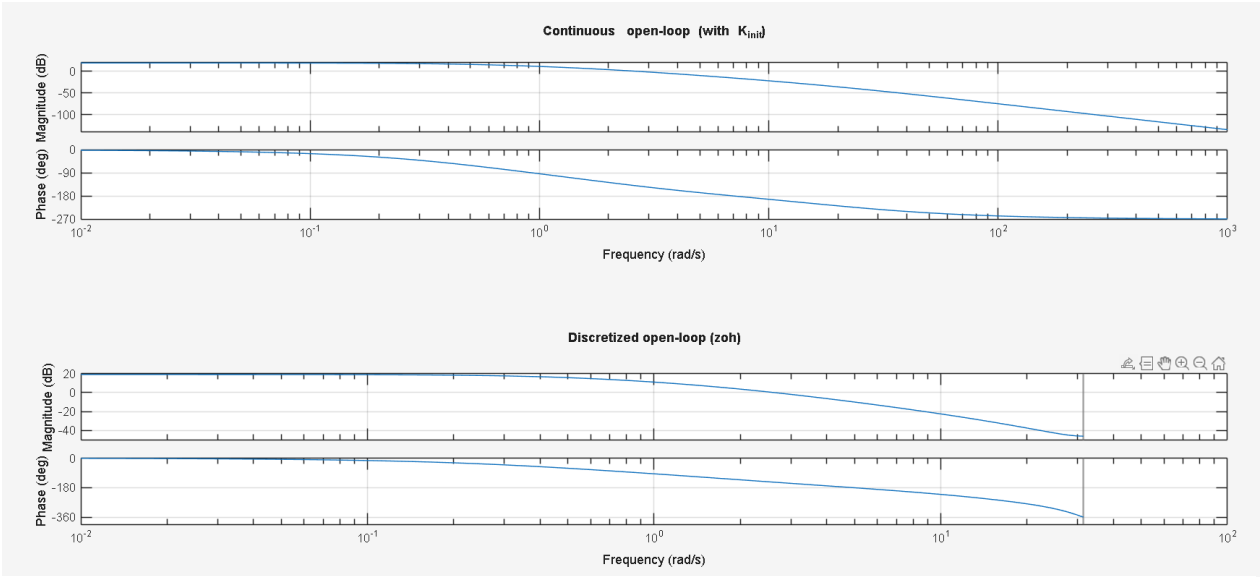


Figure 6 Bode Plot contimuous vs discrete[1]

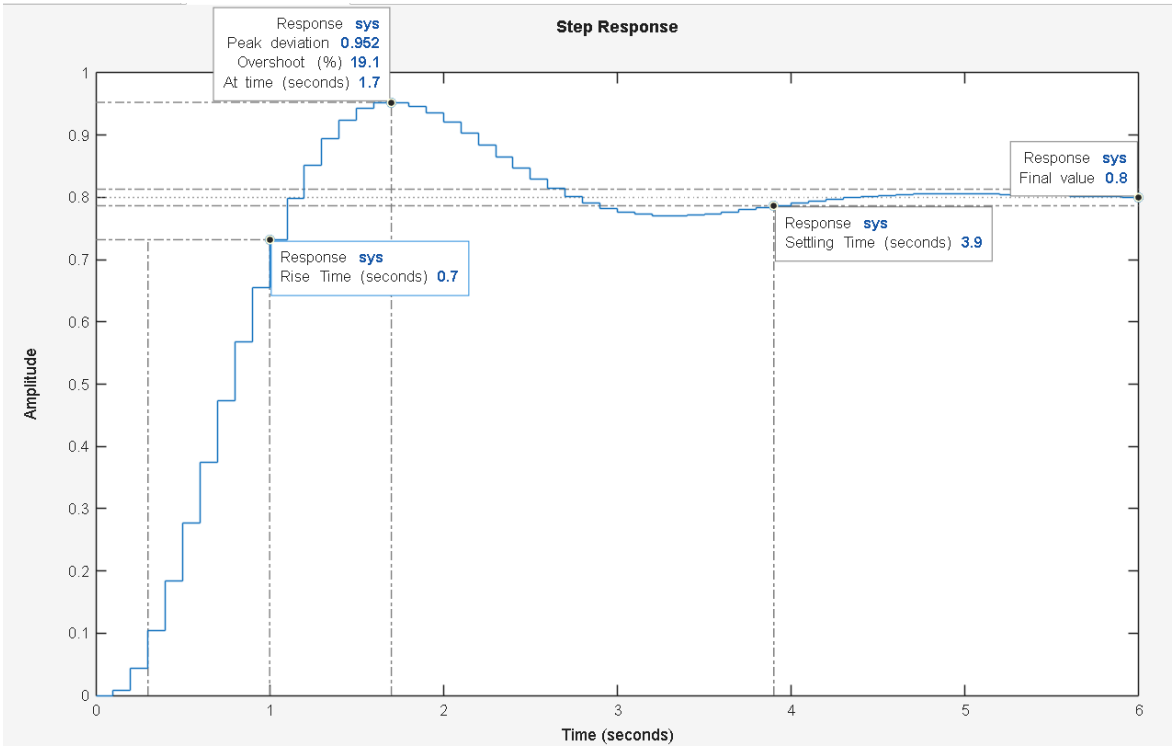


Figure 7 System step response[1]

Comment: As we see the Ess is 19% and is boger than the required Error (10%) so we will add Gain . and this gain is Required gain multiplier: 2.2500

this will give us Ess = 10% .But this lead to reduce the **PM to 32** which is less than required (PM> 50).

Then we need to add lead compensator to get PM >50.

```
Original DC Gain (before adding gain): 4.0000
Original Steady-state Error: 0.2000 (19% expected)

Required gain multiplier: 2.2500
New Steady-state Error after gain: 0.1000 (Target = 0.10)

=== BEFORE CONTROLLER ===
Gain Margin: 9.64 dB
Phase Margin: 32.80 deg
Wgc: 4.92 rad/s
Wpc: 2.64 rad/s
```

Figure 8:the calculation to get the gain and the new PM and GM

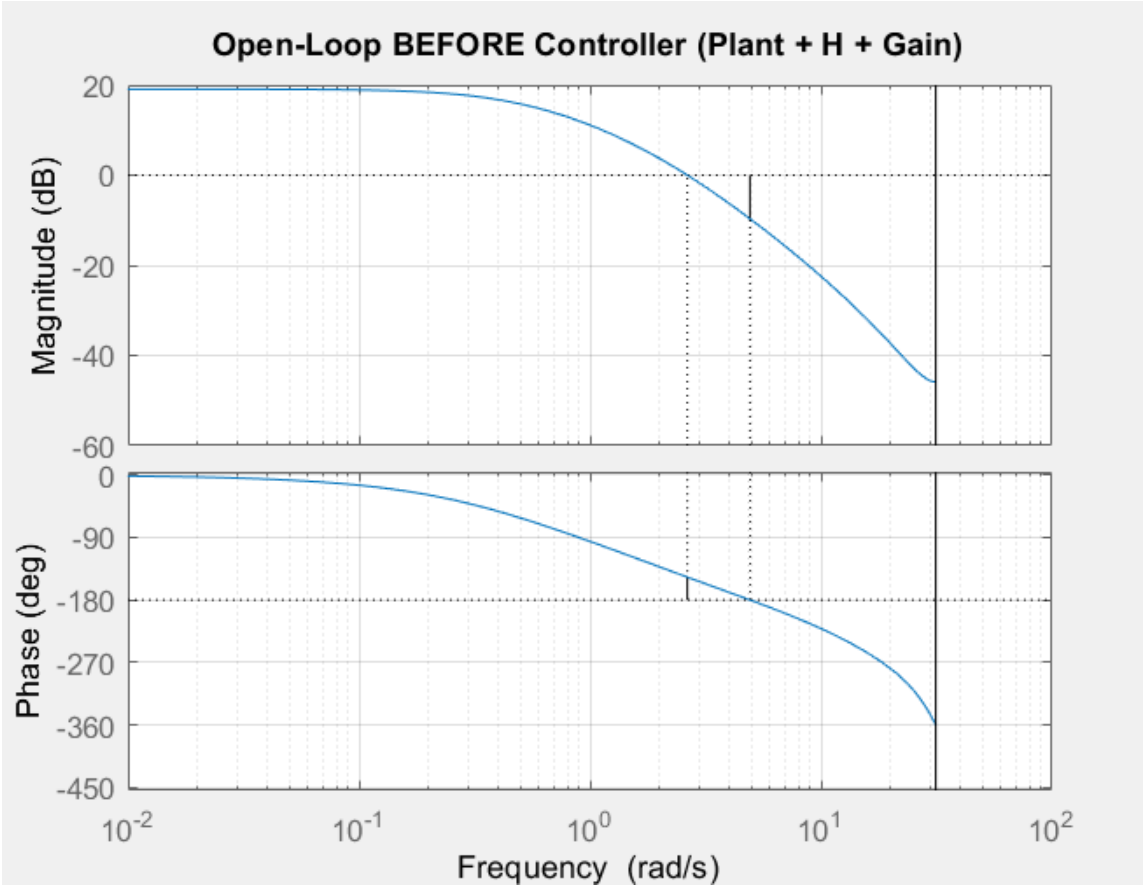


Figure 9:pode plot after we add gain and before compensator

7. Controller Design

Design Objective

As seen in Table 3, The controller must satisfy the following specifications:

Table 1 Project Requirements

Requirement	Target Value
Steady-State Error to Unit Step	≤ 10%
Phase Margin	≥ 50°
Closed-Loop Stability	Must be stable

Since the goal is improving steady-state accuracy and achieving a stable response with higher phase margin, a **compensator must be added** to the system.

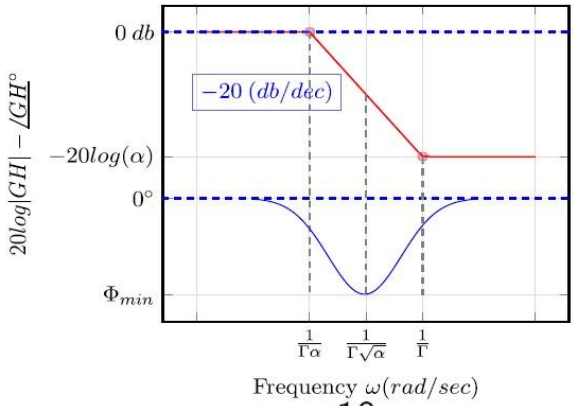
Now we will use Lead compensator .

Lead Compensation Approach

Lag/Lead Compensation

Lag

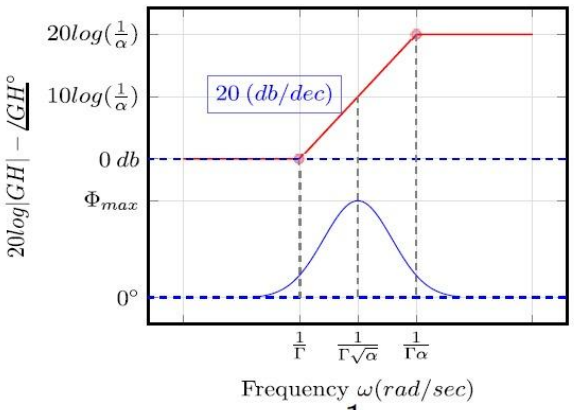
$$G_D(w) = \frac{1+\Gamma w}{1+\Gamma\alpha w},\ \frac{1}{\Gamma\alpha} < \frac{1}{\Gamma}$$
i.e. $\alpha > 1$



$$\omega_{gc_{new}} = \frac{10}{\Gamma}$$

Lead

$$G_D(w) = \frac{1+\Gamma w}{1+\Gamma\alpha w},\ \frac{1}{\Gamma\alpha} > \frac{1}{\Gamma}$$
i.e. $\alpha < 1$



$$\omega_{gc_{new}} = \frac{1}{\Gamma\sqrt{\alpha}}$$

Figure 10 Lag Vs Lead Comparison [2]

1. Get $GH(z)$.
2. Obtain $GH(z)$ using the "Bilinear Transformation".
3. Adjust k from e_{ss} requirements.
4. Draw the gain plot of $kGH(w)$ to get $\omega_{gc_{old}}$ and PM_{old} (exactly).
5. Check $PM_{old} \geq PM_{req}$ & $\omega_{gc_{old}} \leq \omega_{gc_{req}}$, otherwise design either Lag or Lead compensator.
6. If PM_{old} is small, $PM_{old} \leq 0$ or $\omega_{gc_{old}} \geq \omega_{gc_{req}}$, then use Lag compensator. Otherwise, use Lead compensator.

Figure 11 Design Algorithm [2]

Lag/Lead Compensation

Lag

(I) Get $\omega_{gc_{new}}$ from $\angle GH$ plot:
$\angle GH_{\omega_{gc_{new}}} = PM_{req} - 180 + SF$,
$SF = 5, 10, 15, 20$

Check $\omega_{gc_{new}} \leq \omega_{gc_{req}}$, else repeat (I) with $SF \uparrow$.

(II) Get Γ: $\omega_{gc_{new}} = \frac{10}{\Gamma}$

(III) Get α:
$|GH|_{db_{\omega_{gc_{new}}}} = 20log(\alpha)$

(IV) Check PM_{new}:
$PM_{new} \geq PM_{req}$

Lead

(I) Calculate α using Φ_{max}:
$\alpha = \left(\frac{1 - sin(\Phi_{max})}{1 + sin(\Phi_{max})}\right)$ where
$\Phi_{max} = PM_{req} - PM_{old} + SF$
$= sin^{-1}\left(\frac{1-\alpha}{1+\alpha}\right)$,
$SF = 10, 15, 20$

(II) Get $\omega_{gc_{new}}$:
$|GH|_{db_{\omega_{gc_{new}}}} = 10log(\alpha)$

(III) Get Γ: $\omega_{gc_{new}} = \frac{1}{\Gamma\sqrt{\alpha}}$

(IV) Check PM_{new}:
$PM_{new} \geq PM_{req}$

Figure 12 Lag Vs Lead Steps [2]

Lead Controller Synthesis

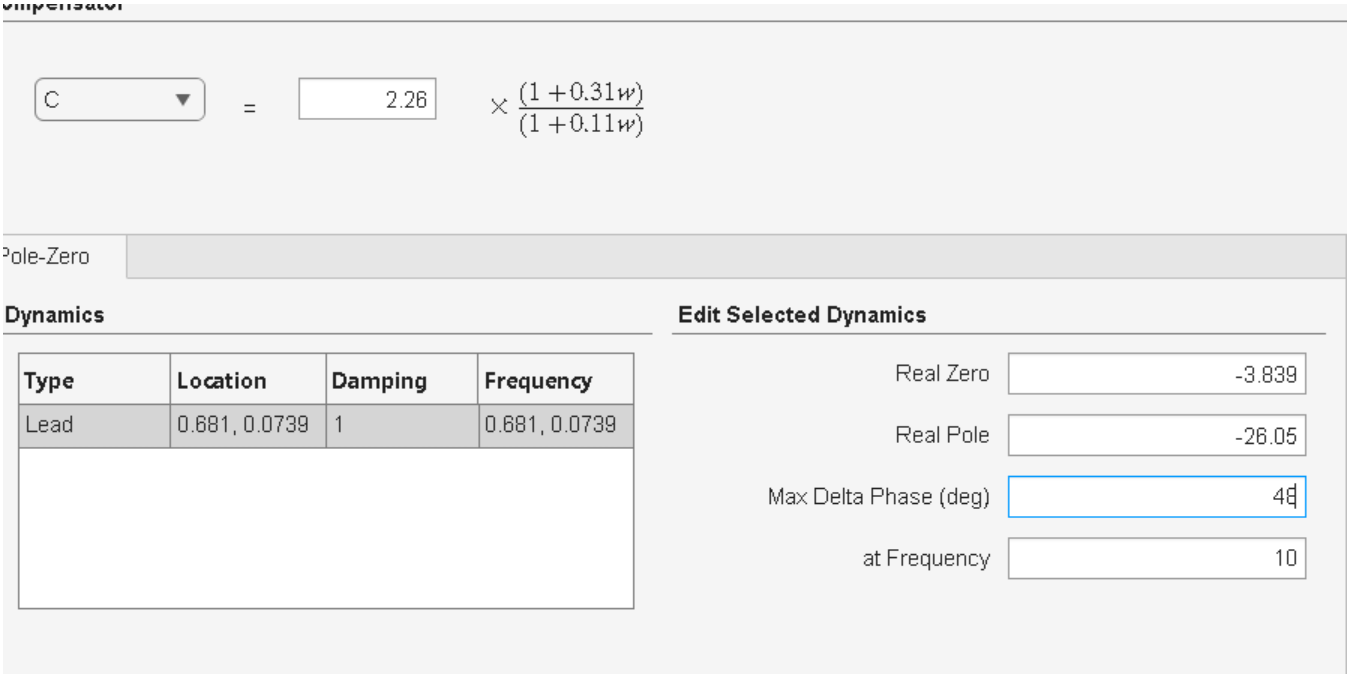


Figure 14 Compansator in S domain [1]

```
Cz_unscaled =

  z - 0.6812
  ----------
  z - 0.0739
```

Figure 13 Compansator in Z domain [1]

From Figure 11 and 12, Using MATLAB **SISO Tool**, a lead compensator structure was selected:

$$C(z) = \frac{z - z_0}{z - p_0} C(z)$$

Where:

- Zero $z_0 = 0.6812$
- Pole $p_0 = 0.0739$
- Gain K=6.525

Final **discrete controller**:

$$\boxed{C(z) = \frac{z - 0.6812}{z - 0.0739}}$$

Open-Loop Bode Characteristics After Compensation

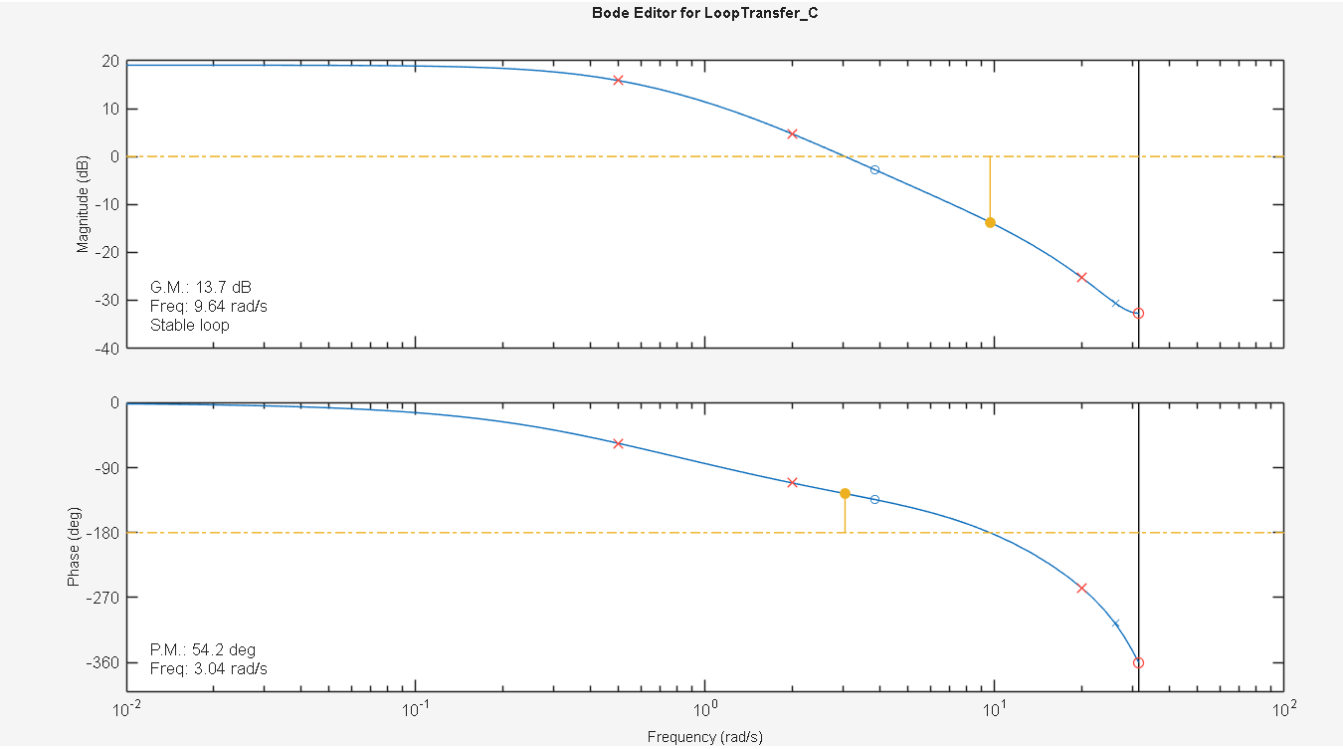


Figure 15 Compansator Bode Plot [1]

Key performance improvements:

Table 2 Compansator Requierments [1]

Metric	Value	Status
Phase Margin	**54.2**	Meets ≥ 50°
Gain Crossover Frequency	1.51 rad/s	Increased stability

Root Locus Verification

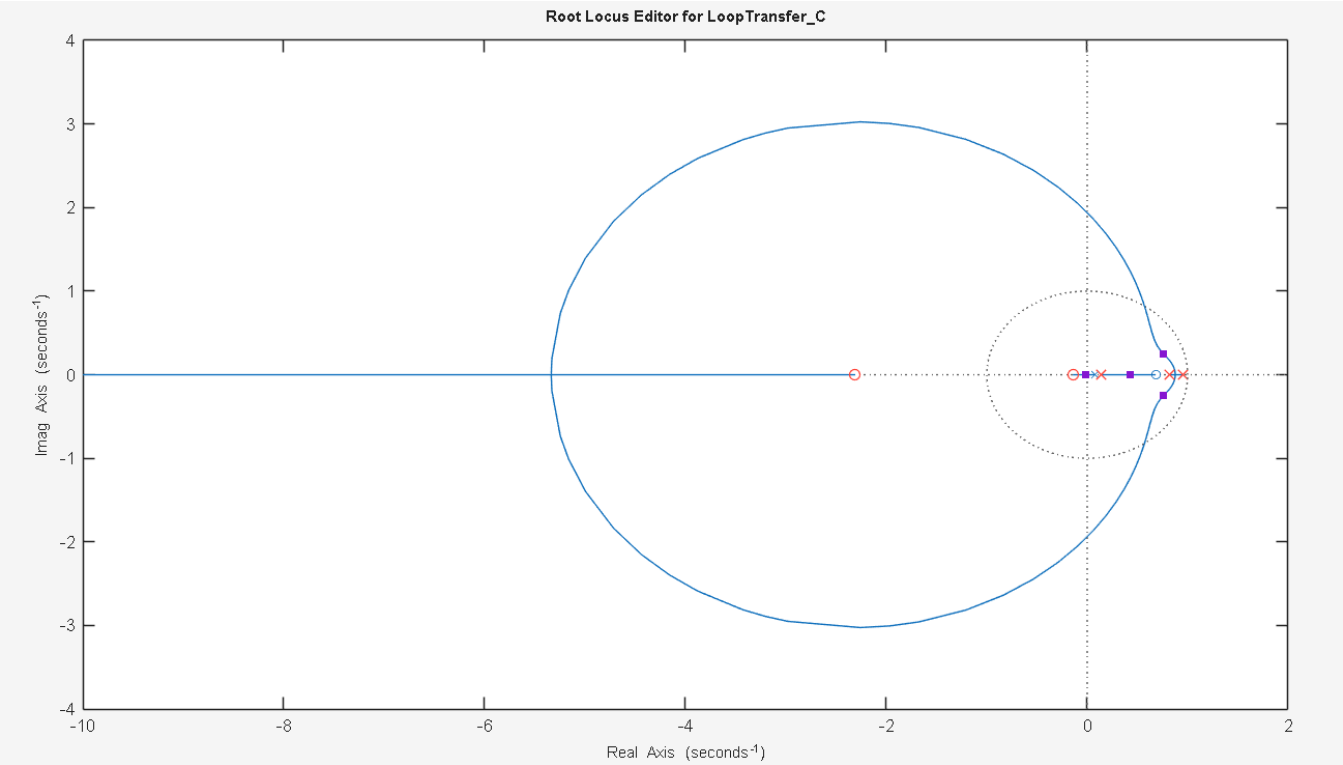


Figure 16 Compansator Root Locus [1]

As shown in Figure 14, The designed lead compensator:

- moves dominant poles toward left half-plane
- increases damping → reduces overshoot

Thus → **closed-loop stability guaranteed**

Closed-Loop Step Response

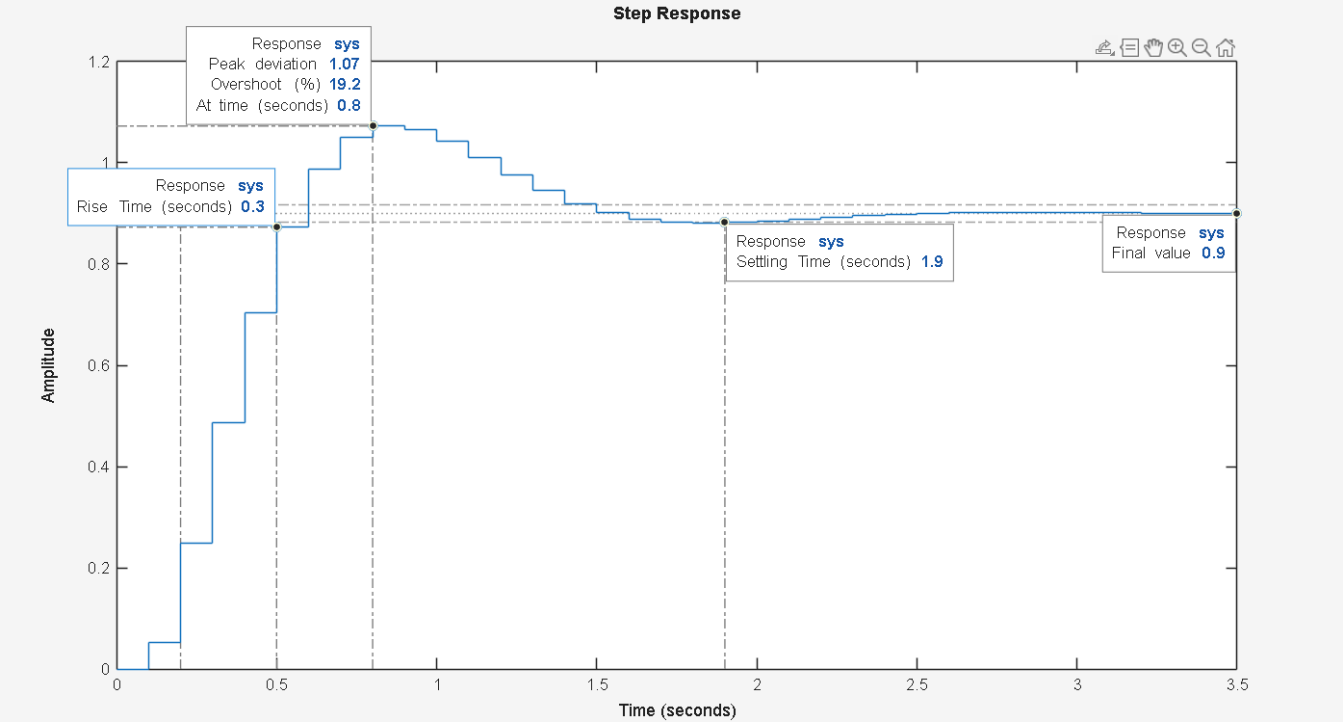


Figure 17 Compansator Step Response

As seen in Figure 15, Measured results from MATLAB:

Table 3 Compansator Time Domain Summary [1]

Performance Metric	Value	Requirement	Status
Final Value	~0.90	≥0.90	OK
ESS	0. 099844	< 0.10	OK
Overshoot	~19%	Acceptable	
Settling Time	~3.9 s	Improved	
Rise Time	~0.7 s	Improved	

All required performance specs **Successfully Achieved**

7.1. Controller Design Conclusion

```
===== PERFORMANCE AFTER CONTROLLER =====
Final Value: 0.900156
Steady-State Error AFTER controller: 0.099844 (Target = 0.10)
Open-loop DC gain L(1) = Cz(1)*dcgain(GHd) = 9.000000
Gain Margin: 13.73 dB
Phase Margin: 54.17 deg (Target >= 50 deg)
Wgc: 9.6388 rad/s
Wpc: 3.0415 rad/s
```

Figure 18 Compansator Performance [1]

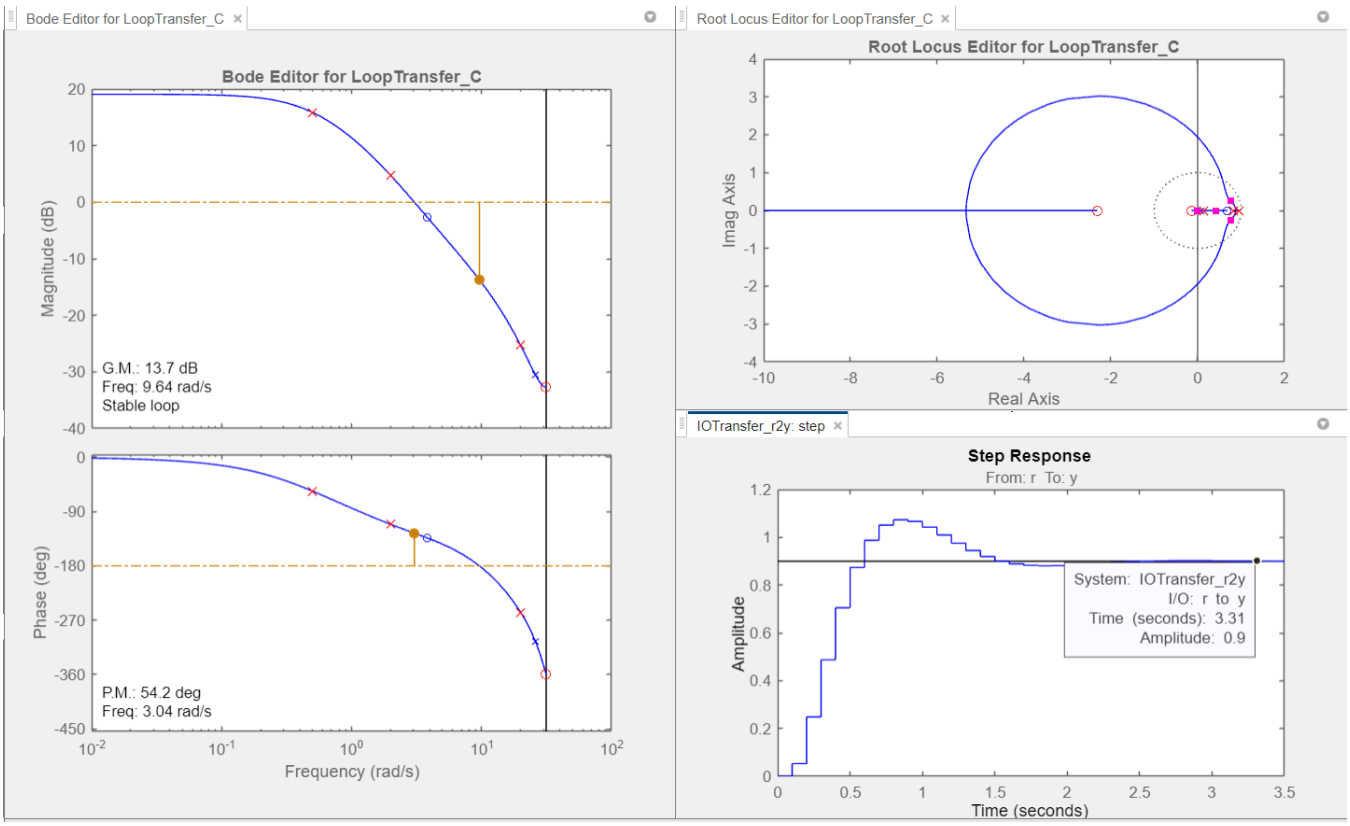


Figure 19:performance after we add compensator

Comment: As we see the result .all required has been achieved . PM = 54.2 > 50 .and the Ess is 0.099844 < 0.1 .

8. References

[1] Project GitHub — Source Code & Design Artifacts 🔗 https://github.com/youefkh05/Digital-Control-System-Design-Using-MATLAB-SISO-Tool

[2] **Dr. Mina's Lecture Slides** Digital Control Systems — Lead/Lag Compensation Design Methodology

[3] MATLAB Documentation MathWorks — SISO Design Tool & ZOH Discretization Theory

9. Code:

```
%% CONTROL PROJECT - FIXED: ENSURE CONTROLLER DOES NOT CHANGE DC GAIN
clear; clc; close all;

%% -------------------------
% 1) Continuous Plant Model
% -------------------------
s = tf('s');
T = 0.1; % Sampling time (sec)

Gp = 4 / ((2*s + 1)*(0.5*s + 1)); % Gp(s)
H  = 1 / (0.05*s + 1);            % H(s)
GHc = Gp * H;                     % Continuous open-loop (plant + sensor)

% Compute original steady-state error (Type-0 step input)
Kp_original = dcgain(GHc);
Ess_original = 1/(1 + Kp_original);

fprintf('Original DC Gain (before adding gain): %.6f\n', Kp_original);
fprintf('Original Steady-state Error: %.6f\n', Ess_original);

%% ----------------------------------------------------
% 2) Add Gain to Make Ess = 10% Before Controller
% ----------------------------------------------------
% Target Ess = 0.10 --> Kp_target = (1/Ess) - 1 = 9
Kp_target = 9;
K_required = Kp_target / Kp_original;

fprintf('\nRequired gain multiplier for Ess=10%%: %.6f\n', K_required);

% Apply gain to plant (ONLY HERE!)
GHc = K_required * GHc;

% New steady-state error (should be ≈ 0.10)
Ess_new = 1/(1 + dcgain(GHc));
fprintf('New Steady-state Error after applying plant gain: %.6f (Target = 0.10)\n',
Ess_new);

%% -------------------------
% 3) Discretize Plant with ZOH
% -------------------------
GHd = c2d(GHc, T, 'zoh'); % Discrete plant with gain

%% ----------------------------------------------
% 4) Bode Plot & Margins BEFORE Controller
% ----------------------------------------------
figure('Name','Bode BEFORE Controller');
margin(GHd); grid on;
title('Open-Loop BEFORE Controller (Plant + H + Gain)');

[GM0, PM0, wgc0, wpc0] = margin(GHd);

fprintf('\n=== BEFORE CONTROLLER ===\n');
fprintf('Gain Margin: %.2f dB\n', 20*log10(GM0));
fprintf('Phase Margin: %.2f deg\n', PM0);
fprintf('Wgc: %.4f rad/s\n', wgc0);
fprintf('Wpc: %.4f rad/s\n', wpc0);
```

```
%% -------------------------
% 5) Final Controller (from SISO design)
%    --- SCALE IT TO HAVE DC GAIN = 1 ---
% -------------------------
z = tf('z', T);

% Controller shape (zero & pole from SISO design)
z0 = 0.6812;
p0 = 0.0739;
Cz_unscaled = (z - z0) / (z - p0);

% Compute scaling so Cz(1) == 1 (unit DC gain)
Kc = 1 / dcgain(Cz_unscaled);
Cz = Kc * Cz_unscaled;

% display controller info
disp('Controller shape (unscaled):');
Cz_unscaled
fprintf('Scaling Kc to enforce Cz(1)=1: %.6f\n', Kc);
disp('Final controller Cz (after scaling to unit DC):');
Cz

fprintf('Cz DC gain check: Cz(1) = %.6f\n', dcgain(Cz));

%% -------------------------
% 6) Closed-loop AFTER Controller
% -------------------------
Ld = Cz * GHd;   % Open-loop with controller
CLd = feedback(Ld, 1);

%% Step Response
figure('Name','Closed-loop Step Response After Compensation');
step(CLd); grid on;
title('Closed-loop Step Response (Final Design)');

[y, ~] = step(CLd);
final_val = y(end);
ess_after = 1 - final_val;

%% Stability Margins After Controller
[GM, PM, wgc, wpc] = margin(Ld);

fprintf('\n===== PERFORMANCE AFTER CONTROLLER =====\n');
fprintf('Final Value: %.6f\n', final_val);
fprintf('Steady-State Error AFTER controller: %.6f (Target = 0.10)\n', ess_after);
fprintf('Open-loop DC gain L(1) = Cz(1)*dcgain(GHd) = %.6f\n', dcgain(Ld));
fprintf('Gain Margin: %.2f dB\n', 20*log10(GM));
fprintf('Phase Margin: %.2f deg (Target >= 50 deg)\n', PM);
fprintf('Wgc: %.4f rad/s\n', wgc);
fprintf('Wpc: %.4f rad/s\n', wpc);

%% -------------------------
% 7) Bode Plot After Controller
% -------------------------
figure('Name','Open-loop With Controller');
margin(Ld); grid on;
title('Open-Loop with Final Discrete Controller C(z)');

%% -------------------------
% 8) SISO Tool Validation (optional)
```

```
% -------------------------
disp('Opening SISO Tool for verification…');
sisotool(Cz, GHd);
```

10. ReadMe File

This project implements a complete control system design starting from a continuous-time plant, adjusting its steady-state error, discretizing it, designing a digital controller, and validating the final closed-loop performance.
The code is written in MATLAB and follows a clear, step-by-step structure.

1. Continuous Plant Model

The plant is made of two parts:

- **Gp(s)** → main plant
- **H(s)** → sensor dynamics

Both are multiplied to form the open-loop transfer function:

G(s)H(s)G(s)H(s)G(s)H(s)

The script calculates the **original DC gain** and the **steady-state error** for a step input:

Ess=11+KpEss = \frac{1}{1 + K_p}Ess=1+Kp1

This helps us understand how the system behaves before any controller is applied.

2. Adjusting the Steady-State Error Before the Controller

We want the **steady-state error before the controller to be exactly 10%**.

So we compute the required gain:

$$K(reqired) = \frac{Kp(target)}{Kp(original)}$$

This gain multiplier is applied only to the plant.
This ensures that the discrete controller does **not** change the DC gain later.

3.Discretization (ZOH)

The plant with the new gain is converted into its discrete-time model using Zero-Order Hold (ZOH):

```
GHd = c2d(GHc, T, 'zoh')
```

4. Bode Plot Before Controller

Before adding the controller, the Bode plot and stability margins (**Gain Margin**, **Phase Margin**, **crossover frequencies**) are displayed.
This shows how stable (or unstable) the system is before compensation.

5. Controller Design (Digital Controller)

A discrete controller structure is taken from SISO Tool:

$$\boxed{C(z) = \frac{z - 0.6812}{z - 0.0739}}$$

6. Closed-Loop System After Controller

The closed-loop system is built: $CL(z) = \frac{C(Z)G(Z)}{1+C(Z)G(Z)}$

We calculate:

- Step response
- Final value
- Steady-state error
- Stability margins with the controller

This part confirms if the design meets the project requirements:

- ESS = 10% , Phase margin ≥ 50°,Good stability and Smooth time response.

7. Bode Plot After Controller

The Bode plot with the final controller is shown to verify:

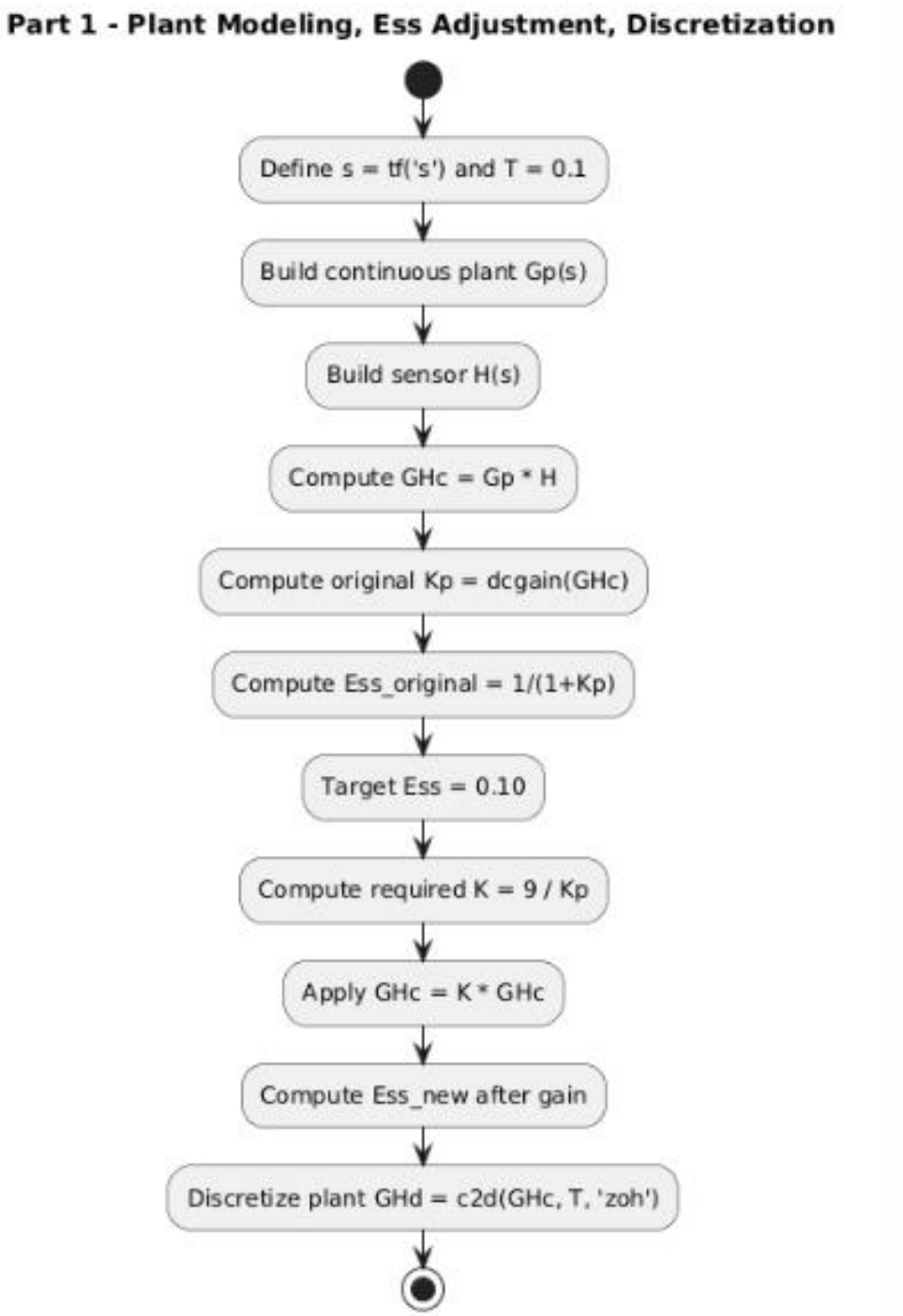
- Better phase margin ,Desired crossover frequency , Closed-loop robustness

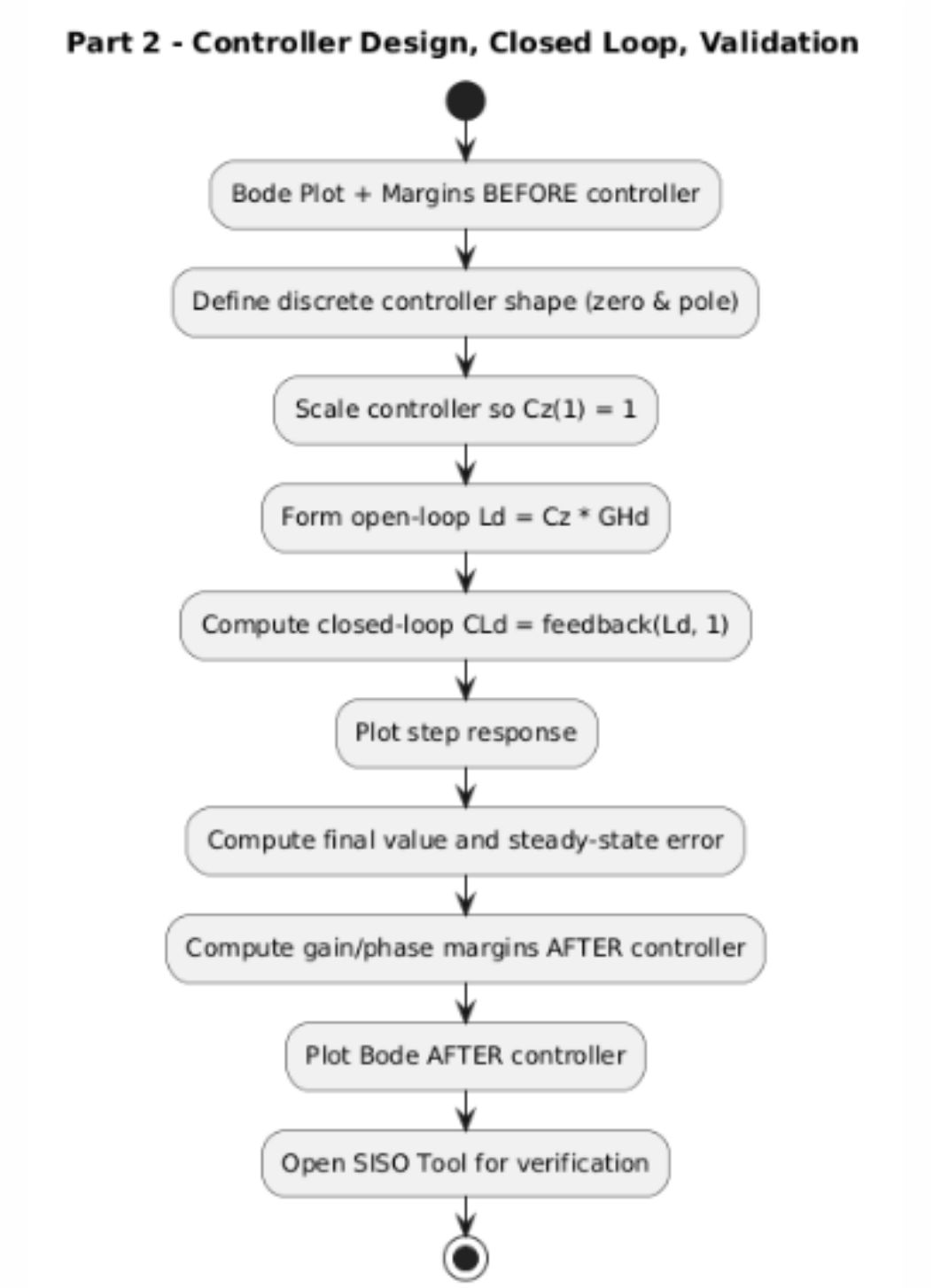
8. SISO Tool Verification

At the end, the SISO Tool opens so the design can be inspected visually:

```
sisotool(Cz, GHd);
```

9.flowchart


Part 1 - Plant Modeling, Ess Adjustment, Discretization
Define s = tf('s') and T = 0.1
Build continuous plant Gp(s)
Build sensor H(s)
Compute GHc = Gp * H
Compute original Kp = dcgain(GHc)
Compute Ess_original = 1/(1+Kp)
Target Ess = 0.10
Compute required K = 9 / Kp
Apply GHc = K * GHc
Compute Ess_new after gain
Discretize plant GHd = c2d(GHc, T, 'zoh')


Part 2 - Controller Design, Closed Loop, Validation
Bode Plot + Margins BEFORE controller
Define discrete controller shape (zero & pole)
Scale controller so Cz(1) = 1
Form open-loop Ld = Cz * GHd
Compute closed-loop CLd = feedback(Ld, 1)
Plot step response
Compute final value and steady-state error
Compute gain/phase margins AFTER controller
Plot Bode AFTER controller
Open SISO Tool for verification

10.Summary of What the Script Does

1. Builds the continuous plant.
2. Calculates original steady-state error.
3. Adds gain to force ESS = 10%.
4. Discretizes the system.
5. Checks Bode plot and stability before controller.
6. Designs a discrete controller with unit DC gain.
7. Computes closed-loop performance.
8. Shows Bode plot after compensation.
9. Opens SISO Tool for final validation.