Lab 1 Report

Problem Statement

Engineers utilize **control theory** in systems to implement an automatic controller or process in order to regulate systems autonomously. Systems are able to use feedback from previous outputs to adjust future inputs, minimizing the error between expected and actual outputs. This laboratory aims to design practical PID (proportional-integral-derivative) controllers using dominant pole placement via the **root locus method**, a way to examine how the locations of a system's poles change with varying gain values in order to assess system behavior. This controller is characterized by the equation $u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt}$, where K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively. To implement the derivative term, the approximation filter $\frac{1}{T_d s+1}$ is used, which prevents noise amplification that can occur due to ideal differentiation. This laboratory discusses two problems: the first problem asks to design a practical PID controller using the root locus method for a given plant, ensuring zero steady-state error to a step input while meeting specific settling time and percentage overshoot requirements; the robustness of the design will also be examined. The second problem involves obtaining a linearized model of a nonlinear SISO plant described by the Duffing equation at a specified operating point in order to design another practical PID controller to meet given stability and performance criteria.

Electrical and Electronic Engineering
University College Dublin

EEEN 40010
Control Theory

CT1
Dominant Pole Placement via Root Locus

Declaration of Authorship

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

Signed: Devon James Knox
Date: 20 October, 2024

Devon James Knox

Problem 1: Given Transfer Function

Introduction

The given problem involves analyzing and designing a control system for a plant that has a specific transfer function. This system's behavior will be studied and improved by employing a practical PID controller designed through dominant pole placement via the root locus method.

The transfer function of the plant, $G_p(s)$, is given by:

$$G_p(s) = \frac{6b(s + (7.4)a + 2c)}{(s + (2.1)a)(s + (8.5)a + 3b + 2c)} \tag{1}$$

where $a = 7$, $b = 3$, and $c = 4$. Substituting these values, the transfer function becomes:

$$G_p(s) = \frac{18(s + 59.8)}{(s + 14.7)(s + 76.5)} \tag{2}$$

The goal is to design a practical PID controller for this system using the root locus method, ensuring that the closed-loop system is stable, there is zero steady-state error to a step input, the 2% settling time does not exceed 85% of the settling time of the plant (but should not be less than half the 10%-90% rise time of the plant), and the percentage overshoot (%PO) does not exceed 40% of the percentage overshoot of the plant or 25%, whichever is larger.

Parameter	Symbol	Value
Proportional Gain	K_p	-
Integral Gain	K_i	-
Derivative Gain	K_d	-
Plant Zero	z	59.8
Plant Poles	p_1, p_2	14.7, 76.5

Table 1: Parameters for plant transfer function (values that are blank will be determined in future steps)

The plant model is analyzed to derive the closed-loop response, and a practical PID controller is designed to meet the performance specifications using the root locus method.

A-priori Analysis

From equation (2), the transfer function indicates that the system is second-order, with a zero at $s = -59.8$ and poles at $s = -14.7$ (dominant pole) and $s = -76.5$. The presence of poles and zeros in the left half of the complex plane suggests that the plant is stable in open-loop operation.

Since it is required for there to be no steady-state error to a step input, a pole at the origin must be employed (hence, the integral term is needed in the PID controller). Furthermore, the percentage overshoot is related to the damping ratio ζ as follows:

$$PO = e^{\left(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}\right)} \times 100\% \tag{3}$$

This equation can be solved for ζ. From here, since the settling time for a second-order system is given by

$$t_s(2\%) = \frac{4}{\zeta\omega_n}, \tag{4}$$

solve for the natural frequency ω_n that achieves the desired $t_s(2\%)$. Lastly, using the equation

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}, \tag{5}$$

the desired dominant poles can be found.

Once we have determined the appropriate poles through this method, various root loci will be sketched to determine how the system poles move with varying controller gain K_c, and ultimately place the dominant poles at the desired locations. Since the location of z affects the shape of the root locus, different z values will be sketched to ensure that the final root locus passes through the desired pole locations. After the controller is designed using the root locus method, it will be verified through step response simulation to ensure all specifications are met. Here are the root loci sketched for this system:

Root Locus Sketches

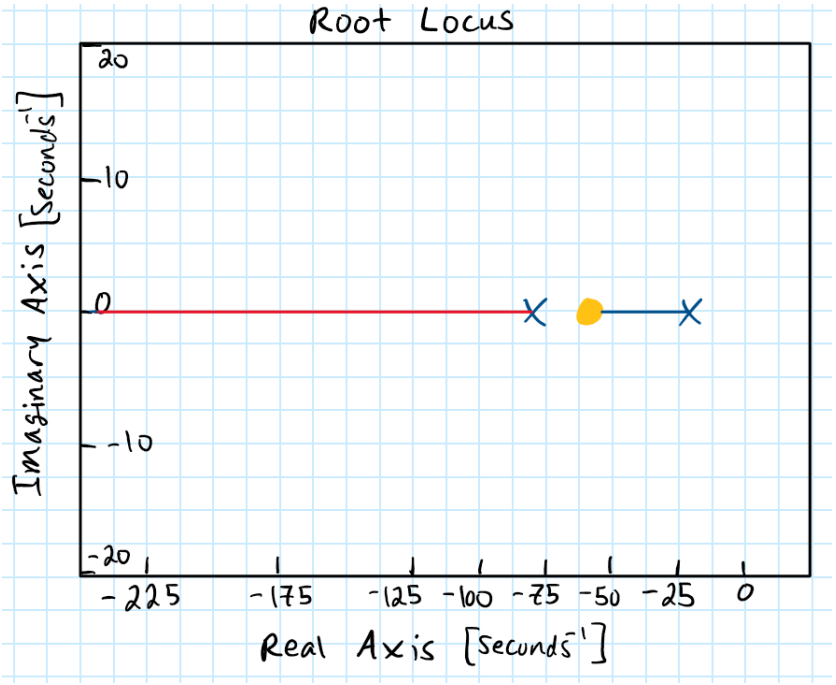

Figure 1: Root locus sketch of plant

This sketch shows the 2 poles and one zero that the plant contains. However, for there to be zero steady-state error, the addition of an I-term by adding a pole at the origin had to be implemented:

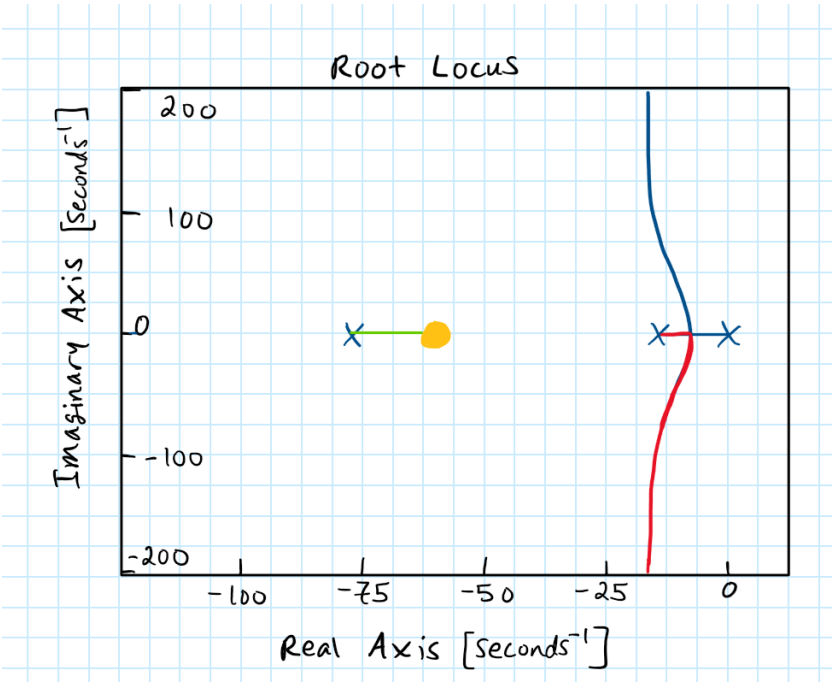

Figure 2: Root locus sketch with pole at origin

Upon testing the closed-loop system with the added controller (see code in numerical solutions section), it did not meet the design requirements; this is because the added pole at the origin introduced slow dynamics into the system, significantly increasing the system's settling time. Additionally, the root locus shows no complex conjugate poles, which limits the system's ability to fix the transient response. Because of this, the controller had to be further altered.

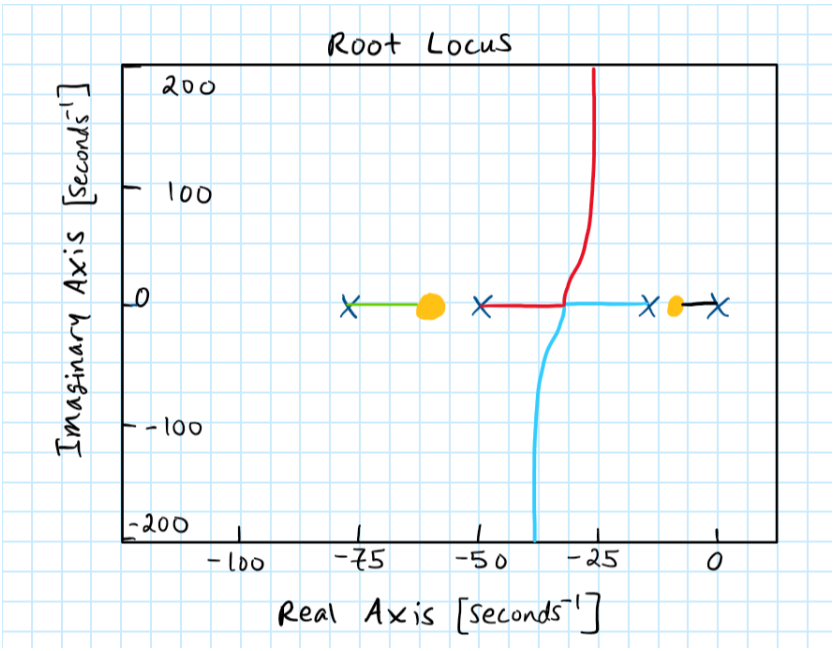

Figure 3: Final root locus sketch (z=10)

This root locus sketch implements a practical PID controller, introducing a zero and a derivative term to counteract the effects of the pole at the origin. This would allow for improved system speed and performance while maintaining stability. The controller structure was adjusted from prior iterations by adding a zero to shift the root locus toward the desired poles, which were determined based on the specified damping ratio, $\zeta = 0.4$, and natural frequency, w_n, necessary to achieve less than 25% overshoot and a fast settling time. $z = 10$ was chosen an initial guess for the zero placement based on the need to counteract the pole at the origin and pull the poles to the left for better transient performance. The desired poles were used as reference points for placing the actual closed-loop poles. The adjusted controller structure successfully met the design specifications, addressing the slow response issues from previous iterations and ensuring that the closed-loop poles were placed in a region that met the required transient performance criteria.

To confirm the quality of the controller design, robustness analysis will later be conducted by varying the plant parameters in order to assess the controller's reliability under non-ideal factors.

Numerical Solution

In this section, MATLAB code will be used to analyze the step response of the system, both with and without the controller. In the following section, the output of the code will be analyzed.

Original Plant

The following program simulates the plant without the implementation of a controller; through the step response and performance characteristics, it will be evident in the analysis that this plant does not meet the required specifications. In addition, this program will provide important information about the plant and it's initial performance measures; with information such as rise time and percent overshoot, in addition to equations 3, 4, and 5, this information will be used to determine the design of the PID controller.

```
1
2  %% This program prints the step response and
       corresponding performance
3  %% information for the original plant
4
5  % Constants
6  a = 7;
7  b = 3;
8  c = 4;
9
10 % Plant transfer function (no controller)
11 numerator = [18, 18*51.8];
12 denominator = conv([1, 14.7], [1, 72.5]);
13 Gp = tf(numerator, denominator);
14
15 % Step response
16 figure;
17 step(sys_cl);
18 title('Step Response of Plant (No Controller)');
19 xlabel('Time');
20 ylabel('Amplitude');
21 grid on;
22 info = stepinfo(sys_cl);
23
24 % Print performance metrics
25 fprintf('Settling Time: %.4f seconds\n', info.
       SettlingTime);
26 fprintf('Rise Time: %.4f seconds\n', info.RiseTime);
27 fprintf('Percentage Overshoot: %.2f%%\n', info.
       Overshoot);
28 fprintf('Peak Time: %.4f seconds\n', info.PeakTime);
29 fprintf('Steady-State Value: %.4f\n', info.
       SettlingMin);
```

Figure 4: MATLAB code for step response and performance characteristics of Original Plant

PID Controller

Math

These calculations were done before code was written to simulate the PID controller.

- Settling Time Calculation Plant's Settling Time:

$$T_{s,\text{plant}} = \frac{4}{|Re(s_{\text{dominant}})|} = \frac{4}{14.7} \approx 0.2721 \text{ s}$$

  Desired Settling Time:

$$T_{s,\text{desired}} = 0.75 \times T_{s,\text{plant}} \approx 0.2041 \text{ s}$$

- Overshoot Specification

  Damping Ratio Calculation:
  From equation (3): solving for ζ when $\%OS = 25\%$: $\zeta = 0.4037$

- Natural Frequency Calculation
  Manipulating equation (4),

$$\omega_n = \frac{4}{\zeta T_s} = \frac{4}{0.4 \times 0.2} = 50 \text{ rad/s}$$

- Desired dominant Poles
  From equation (5),

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -20 \pm j45.825$$

Code

Using the root locus sketches and above calculations, this program is able to simulate the closed-loop step response of the system with the added PID controller; a graph of the step response and performance metrics will be later used for analysis.

```
% Plant transfer function
numerator = 18 * [1, 59.8];
denominator = conv([1, 14.7], [1, 76.5]);
Gp = tf(numerator, denominator);

% Desired damping ratio and natural frequency (
   previously calculated)
zeta = 0.4; % Damping ratio
Ts = 0.2; % Settling time
w_n = 4 / (zeta * Ts); % Natural frequency

% Desired closed-loop poles
sigma = -zeta * w_n;
w_d = w_n * sqrt(1 - zeta^2);
desired_poles = [sigma + j * w_d, sigma - j * w_d];

% PID Controller parameters
z = 10; % Controller zero location
Td = 0.02; % Derivative time constant
numerator_Gc = [1, z];
denominator_Gc = conv([1, 0], [Td, 1]);
Gc = tf(numerator_Gc, denominator_Gc);

% Open-loop transfer function
Gol = series(Gc, Gp);

% Use rlocfind to find Kc that places closed-loop
   poles at desired locations
[Kc_final, ~] = rlocfind(Gol, desired_poles);

% Include the gain in the controller numerator
numerator_Gc_final = Kc_final .* numerator_Gc;

% Controller transfer function with gain
Gc_final = tf(numerator_Gc_final, denominator_Gc);

% Open-loop transfer function with gain
Gol_final = series(Gc_final, Gp);
% Closed-loop transfer function
Gcl = feedback(Gol_final, tf(1));

% Display K_c
fprintf('Calculated controller gain Kc: %.4f\n',
   Kc_final);
% The following code for graphing step reponse and
   checking for meeting system specifications was
   not included in this code snippet
```


Figure 5: MATLAB code for step response and performance characteristics

Robustness Analysis

Once the controller successfully met system requirements, the system could be tested for robustness. The following two programs are used to determine the robustness to variation in the dominant plant poles and zeros.

Variation in Plant Zero

```
1
2  plant_zero = -59.8; % Plant zero
3
4  % Define variation range of +-10%
5  percent_variation = 0.10;
6  zero_variations = plant_zero * (1 + [-
       percent_variation, 0, ...
7      percent_variation]);
8
9  % Controller parameters
10 Kc = 3.5870;       % Controller gain
11 z = 10;            % Controller zero
12 Td = 0.02;
13
14 % Controller transfer function
15 numerator_Gc = [1, z];
16 denominator_Gc = conv([1, 0], [Td, 1]);
17
18 % Add gain
19 numerator_Gc_final = Kc * numerator_Gc;
20 Gc = tf(numerator_Gc_final, denominator_Gc);
21
22 % Time vector
23 t = 0:0.001:1;
24
25 % Arrays to store performance metrics
26 settling_times = zeros(length(zero_variations), 1);
27 overshoots = zeros(length(zero_variations), 1);
28 steady_state_errors = zeros(length(zero_variations),
       1);
29 specs = zeros(3, 1)
30
31 % The following code for graphing step reponse and
       checking for meeting system specifications was
       not included in this code snippet
```

Figure 6: MATLAB code for robustness analysis with Varying Plant Zero (-59.8) by 10%

Variation in Dominant Plant Pole

```
% Dominant plant pole
pole_nominal = -14.7;

% Variation range of +-10%
variation_percentage = 0.10;
pole_variations = pole_nominal * (1 + [-
   variation_percentage, 0, variation_percentage]);

% Same parameters (Kc, z, Td) from previous design
% Plant transfer function Gp and controller transfer
    function Gc already in workspace

t = 0:0.001:1; % time vector (for plotting step
   response)

% Performance metrics
settling_times = zeros(length(pole_variations), 1);
overshoots = zeros(length(pole_variations), 1);
steady_state_errors = zeros(length(pole_variations),
    1);
specs = zeros(length(pole_variations), 1);

for i = 1:length(pole_variations)
    spec_met = true;

    % Adjusted dominant pole
    pole_adjusted = pole_variations(i);
    numerator = 18 * [1, 59.8];
    denominator = conv([1, -pole_adjusted], [1,
       76.5]);

    % Adjusted plant transfer function
    Gp_alt = tf(numerator, denominator);

    % Transfer functions
    Gol_alt = series(Gc, Gp_alt);
    Gcl_alt = feedback(Gol_alt, tf(1));
end

% The following code for graphing step reponse and
   checking for meeting system specifications was
   not included in this code snippet
```


Figure 7: MATLAB code for robustness analysis with varying dominant plant pole (-14.7) by 10%

A-posteriori Analysis & Results

Original Plant

The following is the output of the original plant:

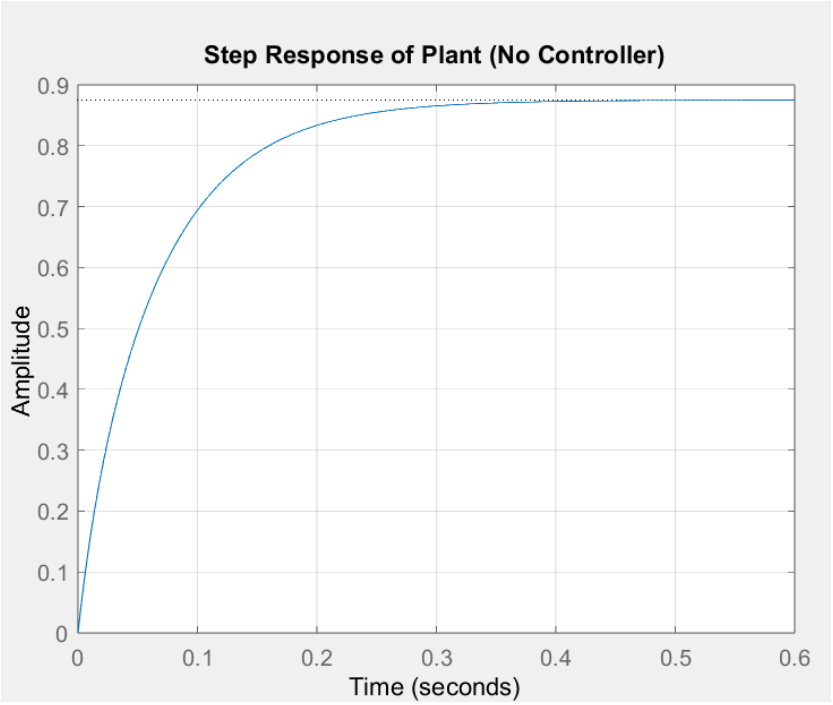

Figure 8: Step response of original plant

```
Settling Time (2% criterion): 0.2588 s
Settling time exceeds maximum specification.
Specifications are not met.
```

Figure 9: Performance results for simulation of plant

This information proves that a controller must be added in order for the system to meet the required specifications.

PID Controller

The following shows the results of the updated PID controller:

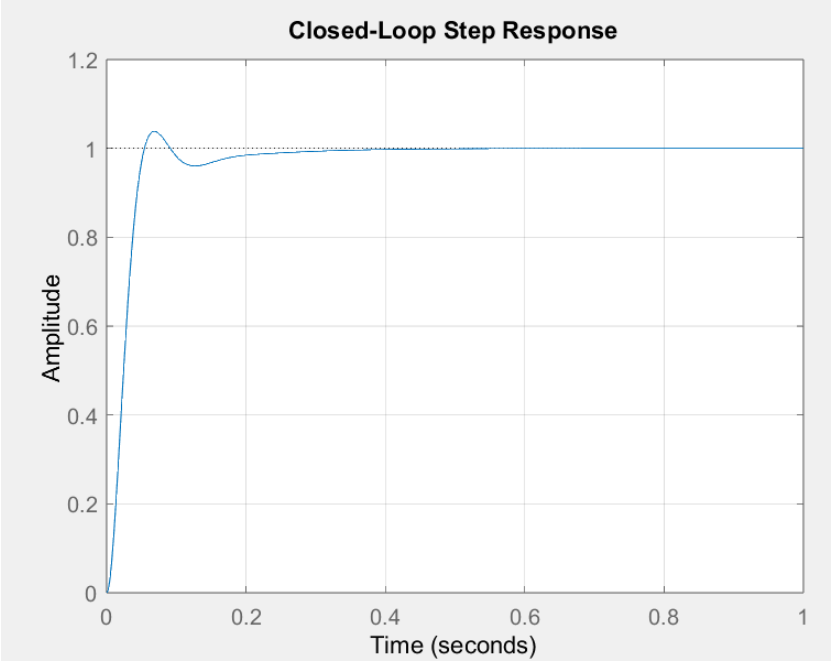

Figure 10: Closed-loop step response of system with added PID Controller

```
Calculated controller gain Kc: 3.5870
Settling Time (2% criterion): 0.1791 s
Percentage Overshoot: 3.82%
Steady-State Error: 0.0000
Settling time specification met.
Percentage overshoot specification met.
Steady-state error specification met.
All specifications are met.
```

Figure 11: Performance results for simulation with added PID Controller

From the graph of the step response in addition to the performance metrics shown, it is clear that the implemented PID controller allows the system to meet the stated specifications. With this information, we can use the controller gain K_c as well as the added zero at $z = 10$ to find K_p, K_i, and K_d using partial fraction expansion. Therefore, **the system implemented uses a PID controller such that** $K_p = 2.870$**,** $K_i = 35.870$**, and** $K_d = 0.0717$

Robustness Analysis

The following graphs and tables shows the output of the code for robustness analysis:

Variation in Plant Zero

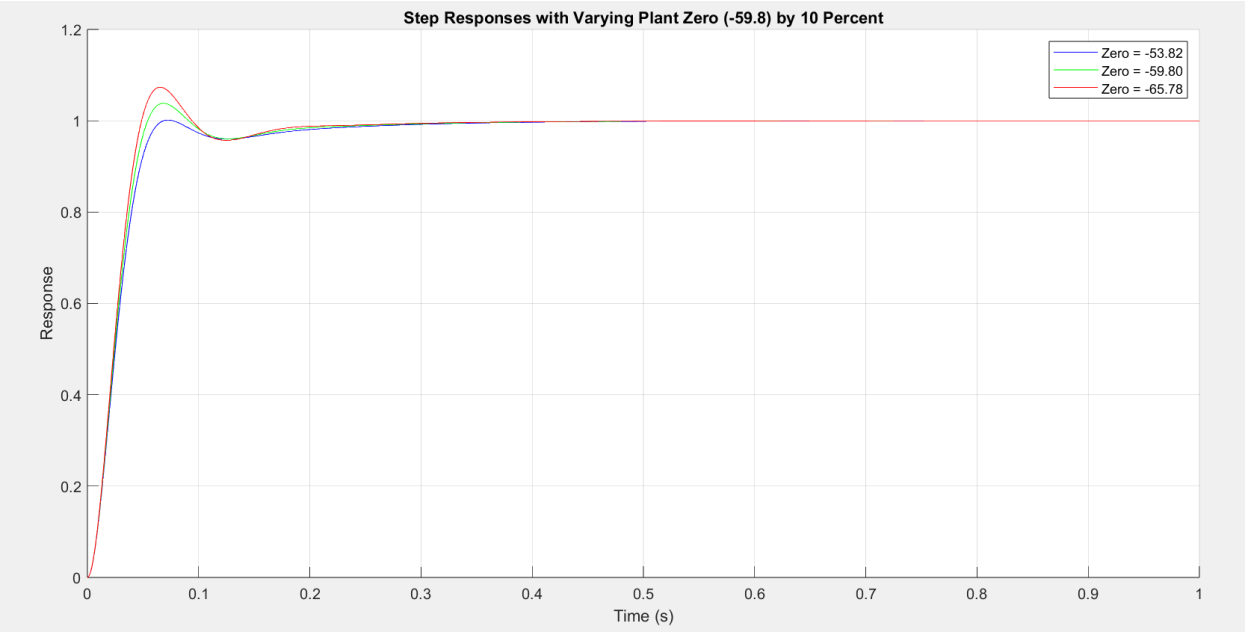

Figure 12: Step response with variation in plant zero

Zero Value	Settling Time (s)	Overshoot (%)	Steady-State Error	Met Specifications (1 if true, 0 if false)
-53.82	0.1961	0.16	0.0000	1.0000
-59.80	0.1791	3.82	0.0000	1.0000
-65.78	0.1684	7.34	0.0000	1.0000

Figure 13: Performance results for variation in plant zero

Variation in Plant Pole

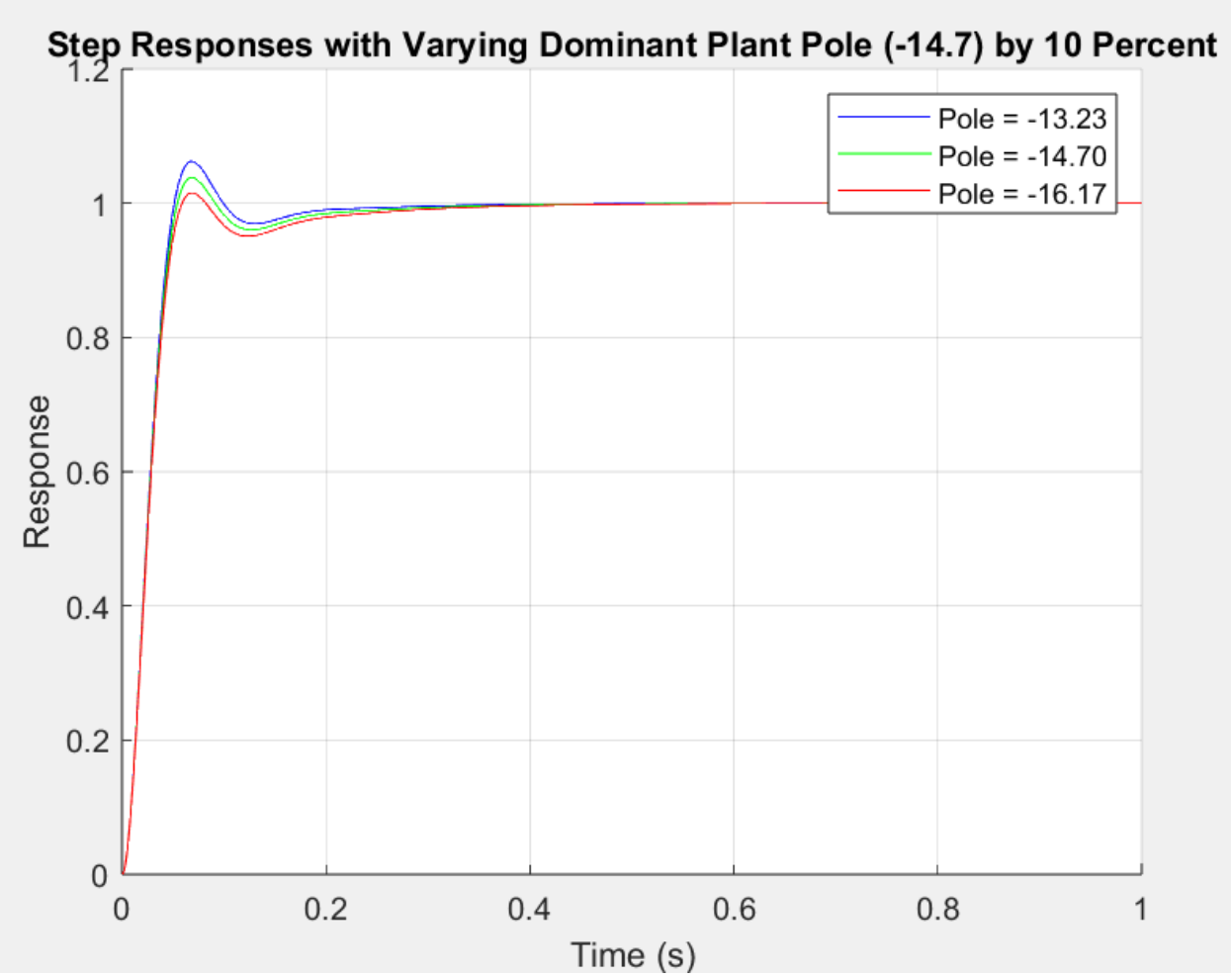

Figure 14: Step response with variation in dominant plant pole

```
Robustness Analysis: Variation in Dominant Plant Pole by 10 Percent of Nominal Value
-----------------------------------------------------------
Pole Value  Settling Time (s)    Overshoot (%)    Steady-State Error  Met Specifications (1 if true, 0 if false)
-13.23      0.1616                  6.22             0.0000          1
-14.70      0.1791                  3.82             0.0000          1
-16.17      0.2055                  1.52             0.0000          1
```

Figure 15: Performance results for variation in dominant plant pole

Since the closed-loop system still meet performance specifications despite varying plant and pole parameters, **the PID controller design is robust.**

Problem 2: Duffing Equation

Introduction

This problem involves analyzing and designing a control system for a Single Input Single Output (SISO) plant described by a nonlinear second-order ordinary differential equation, which is known as the Duffing equation. A linearized model around a specific operating point will be obtained and analyzed, and then a practical PID controller will be implemented using dominant pole placement via the root locus method to achieve additional performance criteria.

The system's equation is as follows:

$$\frac{d^2y}{dt^2} + \delta\frac{dy}{dt} + \alpha y + \beta y^3 = f(t) \tag{6}$$

where $\delta = \frac{a}{10}$, $\alpha = \frac{b}{2}$, and $\beta = \frac{c}{20}$, with a, b, and c being the same personal digits as prior (7, 3, and 4, respectively). Additionally, the system's robustness and failure response will be investigated.

Parameter	Symbol	Value
Damping Coefficient	δ	$\frac{7}{10}$
Linear Coefficient	α	$\frac{3}{2}$
Cubic Coefficient	β	$\frac{4}{20}$
Desired Output	y	0
Desired Input	$f(t)$	0

Table 2: Parameters for the Duffing equation (values determined based on personal digits)

A-priori Analysis

Linearized Model & Performance of Plant

The first part of the question involves obtaining a linearised model of the system in the locality of the operating point; the steady-state error to a step input, 2% settling time and percentage overshoot for the system will be determined in later sections.

To obtain the linearised model of the system, the equation is expanded about the operating point $y = 0$ and $\dot{y} = 0$, and the term βy^3 can be approximated to 0 (for small perturbations around the operating point). Adding in constants, this simplifies equation (6) to:

$$\frac{d^2y}{dt^2} + \frac{7}{10}\frac{dy}{dt} + \frac{3}{2}y = f(t) \tag{7}$$

This is the linearised model of the system in the locality of the selected operating point.
From here, taking the Laplace transform obtains:

$$s^2Y(s) + \frac{a}{10}sY(s) + \frac{b}{2}Y(s) = F(s) \tag{8}$$

This is rearranged to solve for the transfer function $H(s) = \frac{Y(s)}{F(s)}$:

$$H(s) = \frac{1}{s^2 + \frac{7}{10}s + \frac{3}{2}} \tag{9}$$

Thus, this is the transfer function of the model.

Before analyzing the step response computationally, the steady-state error, 2% settling time, and percentage overshoot can be determined numerically via substitution.

- Steady-State Error
  For a step input $f(t) = u(t)$, the steady-state error e_{ss} is:

  $$e_{ss} = \frac{1}{1 + K_p} \tag{10}$$

  where K_p is the position error constant.

  Since $K_p = \lim_{s\to 0} G(s) = \frac{1}{\alpha} = \frac{1}{1.5}$, $e_{ss} = \frac{1}{1+\frac{1}{1.5}}$. Thus, **the steady-state error is approximately 0.6.**

- 2% Settling Time

  For a second-order system, the 2% settling time t_s is given by equation (4). Comparing the denominator of the transfer function with the standard second-order form:

  $$s^2 + 2\zeta\omega_n s + \omega_n^2$$

  We find that:

$$2\zeta\omega_n = \frac{a}{10}, \quad \omega_n^2 = \frac{b}{2}$$

Solving for ζ and ω_n:

$$\omega_n = \sqrt{\frac{b}{2}}, \quad \zeta = \frac{a}{10 \cdot 2\omega_n}$$

Plugging in these values, **the 2% settling time is approximately 11.43 seconds.**

- Percentage Overshoot

  Using equation (3) and the values of ζ and ω calculated earlier, **we can approximate the percentage overshoot as 39.19%.**

Designing a PID Controller for SISO Plant

Here, dominant pole placement via the root locus method will be used to design a PID controller to fit particular specifications, namely:

- The closed-loop system being stable
- Zero steady state error to a step input
- 2% settling time not exceeding 75% of the settling time of the plant while also not being less than half of the 10%-90% rise time
- The PO% not exceeding 35% of the overshoot of the plant itself (or 40% if larger)

Using equation (3), the percent overshoot formula, ζ can be found with some algebraic manipulation, yielding $\zeta = 0.4$. From here, since the settling time for a second-order system is given by equation (4), the natural frequency can be solved for; since the 2% settling time cannot exceed 75% of the plant's settling time ($t_{s,desired} \leq 0.75 \times 11.43 = 8.57$), w_n cannot exceed $1.667\frac{rad}{s}$. **Using equation (5), the desired dominant poles are at** $s = -0.56 \pm 1.92j$**. Furthermore, since there must be zero steady state error to a step input, an integral term must be incorporated in the PID controller through adding a pole at the origin.**

Root Locus Sketches

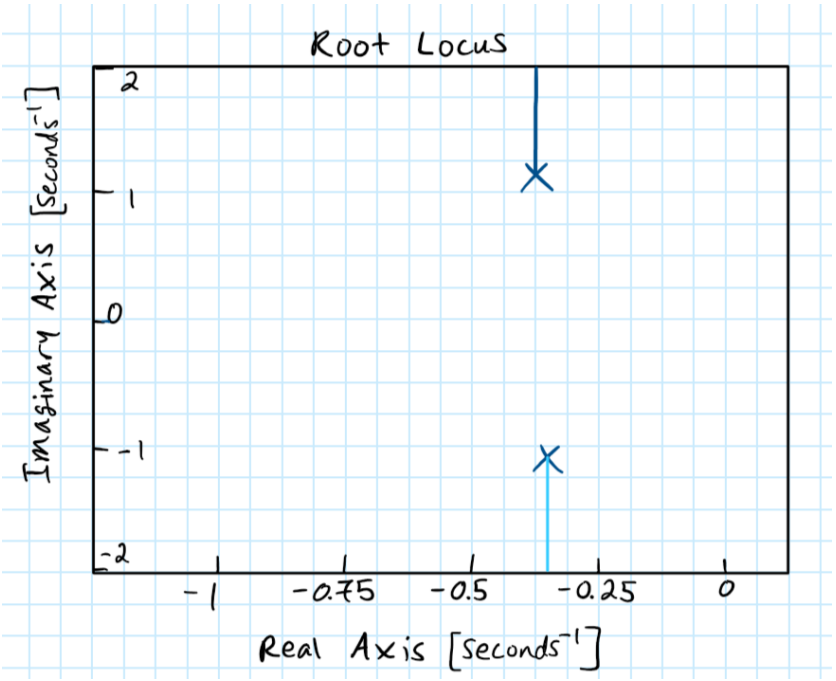

Figure 16: Root locus sketch of SISO plant

This sketch shows the original plant. The poles in the left-hand side but close to the real axis also show that the system would settle slowly, and without a controller, the system will be sluggish, unstable, and will not meet the desired performance specifications.

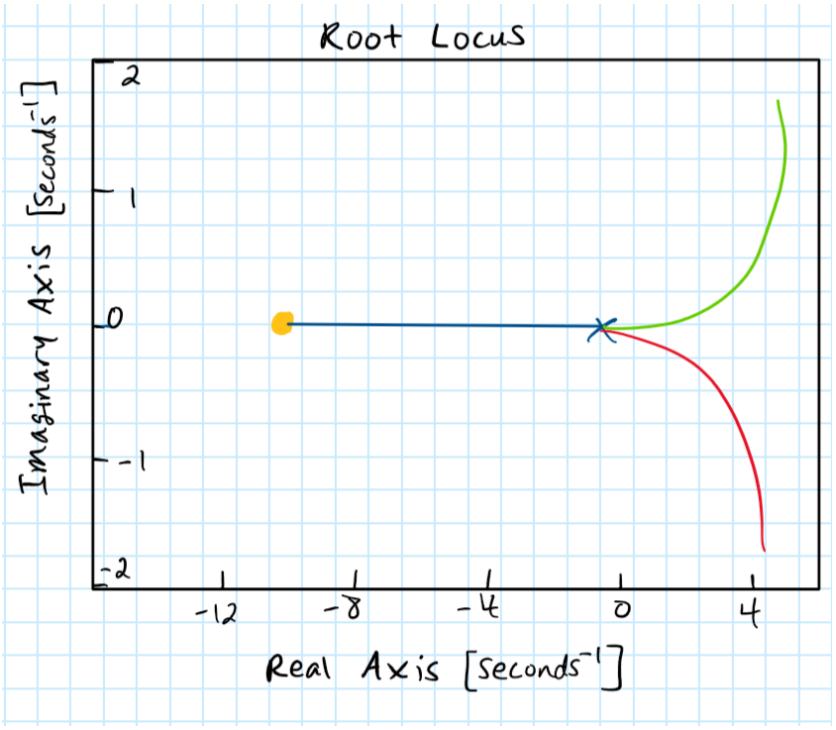

Figure 17: Root locus sketch with zero at s = 10

In this sketch, a zero at -10 the locus is slightly pushed to the left, potentially reducing overshoot. However, this causes two branches to diverge to $\pm\infty$ as it goes right along the real axis, meaning that the step response will be unstable. Thus, another root locus had to be sketched.

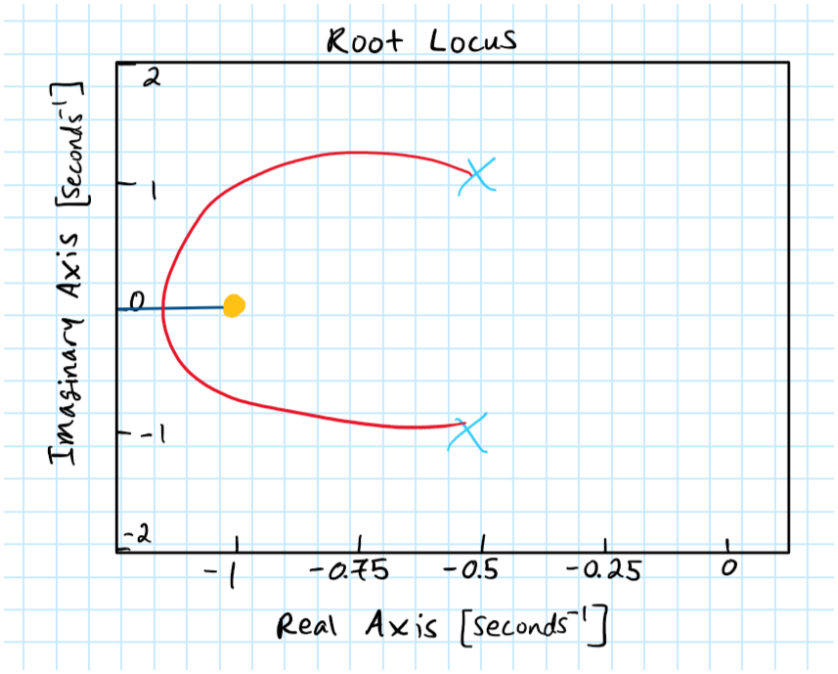

Figure 18: Final root locus sketch

In this final root locus sketch, there will still be oscillations due to the complex poles, but since they have negative real parts the oscillations will decay after some time. Furthermore, the poles and zeros are placed so that the PID controller can take into account both speed and damping while also remaining stable.

Numerical Solution

Local Linear Constant Coefficient Model

The following code was written to computationally determine the performance metrics of the output of the transfer function representing the linearised model of the system.

```
1
2  % Define Constants
3  delta = 7 / 10;
4  alpha = 3 / 2;
5  beta = 4 / 20;
6
7  % Define Plant Transfer Function G(s)
8  den_G = [1, delta, alpha];
9  Gp = tf(1, den_G);          % Plant Transfer Function
10
11 % Step Response Data
12 info_plant = stepinfo(Gp);
13 steady_state_value_plant = dcgain(Gp);
14 fprintf('Plant Steady-State Value: %.4f\n',
     steady_state_value_plant);
15 fprintf('Plant Settling Time (2%% criterion): %.4f s
     \n', info_plant.SettlingTime);
16 fprintf('Plant Percentage Overshoot: %.2f%%\n',
     info_plant.Overshoot);
17 fprintf('Plant Rise Time (10%%-90%%): %.4f s\n',
     info_plant.RiseTime);
18 fprintf('Plant Steady-State Error: %.4f\n', 1 -
     steady_state_value_plant);
```

Figure 19: MATLAB code for simulating SISO plant

PID Controller

Using the root locus method, various root loci were sketched from the A-priori analysis in order to determine the best controller for the system. The following code simulates this closed-loop system, testing to see whether it meets the pre-defined performance requirements.

Using the same methods from problem 1, the controller gain K_c was determined from the desired poles, and from there, K_p, K_i and K_d could be determined. The following code simulates this closed-loop step response, checking if the added PID controller allows it to meet desired system specifications.

```
1
2  % System parameters
3  delta = 7 / 10;
4  alpha = 3 / 2;
5  beta = 4 / 20;
6
7
8  % Linearized plant transfer function
9  den_G = [1, delta, alpha];
10 Gp = tf(1, den_G);
11
12 % PID Controller Parameters (already determined)
13 Kp = 7;
14 Ki = 25;
15 Kd = 10;
16 Td = 0.05;
17
18 % PID Controller
19 C = pid(Kp, Ki, Kd, Td);
20
21 Gol = C * Gp;
22 Gcl = feedback(Gcl, tf(1));
23
24 %% The following code was not included, as it only
      sets up the graphs and
25 %% checks to ensure that all system specifications
      were met.
```

Figure 20: MATLAB code for simulating step response and performance characteristics of linear closed-loop system

Failsafe Investigation

With the previous code already in the MATLAB environment/workspace, the failsafe investigation only involved repeating the simulations but changing the gain to 1% and 500% of its nominal value, and then graphing the step responses and reporting their respective performance characteristics side by side.

Global Model of Nonlinear Closed-loop System

The equations of motion for the actual closed-loop system, which incorporates a PID controller and the nonlinear global model of the plant, will be determined below.

With K_p, K_i and K_d being 7, 25, and 10, and T_d being 0.05, the transfer function for the SISO LTI system could be found via the equation

$$\frac{U(s)}{E(s)} = \frac{(k_P T_D + k_D)s^2 + (k_P + k_i T_D)s + k_I)}{T_D s^2 + s} = \frac{10.35s^2 + 8.25s + 25}{0.05s^2 + s} \quad (11)$$

From here, solving a system of differential equations will obtain the global model of nonlinear closed-loop system in state-space form:

```
1 from scipy.signal import tf2ss
2 Num = [10.35,8.25,25]
3 Den = [0.05,1,0]
4 A,B,C,D = tf2ss(Num, Den)
5 print(A,B,C,D)
```

Figure 21: Python code for solving for closed-loop system using the state-space model

Now that this global model has been obtained, it can be compared with the local linear model. The first script in MATLAB will simulate these two models side-by-side with a small step input, and the following script will be used to determine how performance characteristics including steady-state error, settling time and overshoot change with increasing step input. Lastly, the third script will be used to investigate the robustness of the nonlinear global closed-loop system by varying the damping coefficient so that the true value varies by up to 5% of the nominal value.

```
% Initial conditions for Duffing system
y0 = 0;
y_dot0 = 0;

t_span = [0, 10]; % Time interval for simulation

step_magnitude = 1.7; % Step magnitude for small
   step response
f = @(t) step_magnitude * (t >= 0); % Step function

% System of first-order ODEs
duffing_ode = @(t, Y) [Y(2);
                       f(t) - delta * Y(2) - alpha *
                           Y(1) - beta * (Y(1)^3)];

% Solve the system
[t_nl, Y_nl] = ode45(duffing_ode, t_span, [y0,
   y_dot0]);

y_nl = Y_nl(:, 1);

% Define plant transfer function G(s)
den_G = [1, 0.7, 1.5];
Gp = tf(step_magnitude, den_G);

% PID Controller
C_final = pid(7, 25, 10, 0.05);

% Open-loop transfer function with PID controller
Gol_final = C_final * Gp;

% Add unity feedback
Gcl_final = feedback(Gol_final, 1);

%% The rest of the code is not included, since it is
    only to plot and format the output
```

Figure 22: MATLAB code for simulating step response and performance characteristics of linear and nonlinear closed-loop systems

A-posteriori Analysis & Results

Local Linear Constant Coefficient Model

The following shows the step response and performance characteristics of the linear original SISO plant:

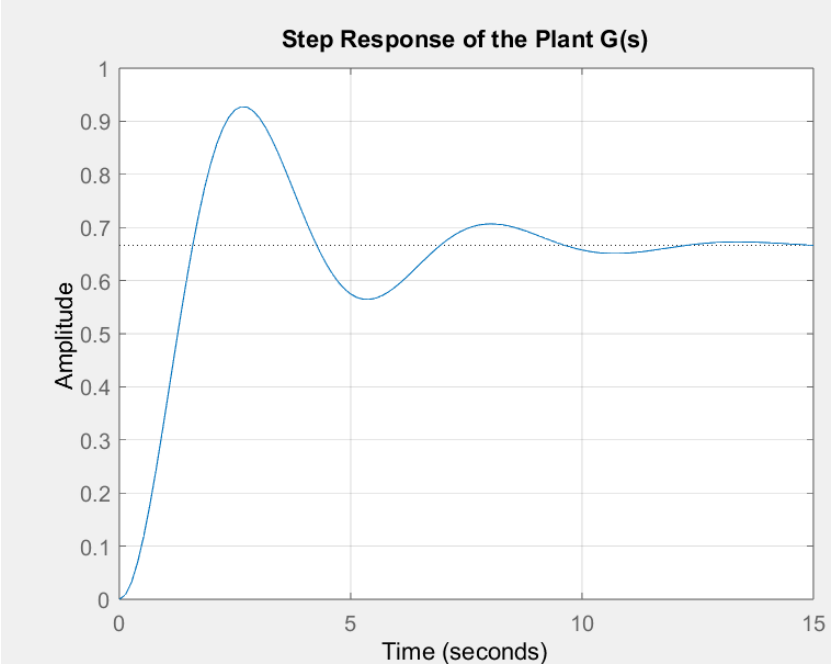

Figure 23: Step response of SISO plant

```
Plant Steady-State Value: 0.6667
Plant 2% Settling Time: 11.1884 s
Plant Percentage Overshoot: 39.13%
Plant Rise Time (10%-90%): 1.0656 s
Plant Steady-State Error: 0.3333
```

Figure 24: Step response information for SISO plant

Since the performance criteria aligns almost perfectly with the expected output according to the math done in the A-posteriori section (with small error due to approximations such as 4 being equal to ln(50)), **the step response characteristics for this SISO plant can be verified.**

Analysis of PID Controller

The following is the output of the linear closed-loop system with the added PID controller:

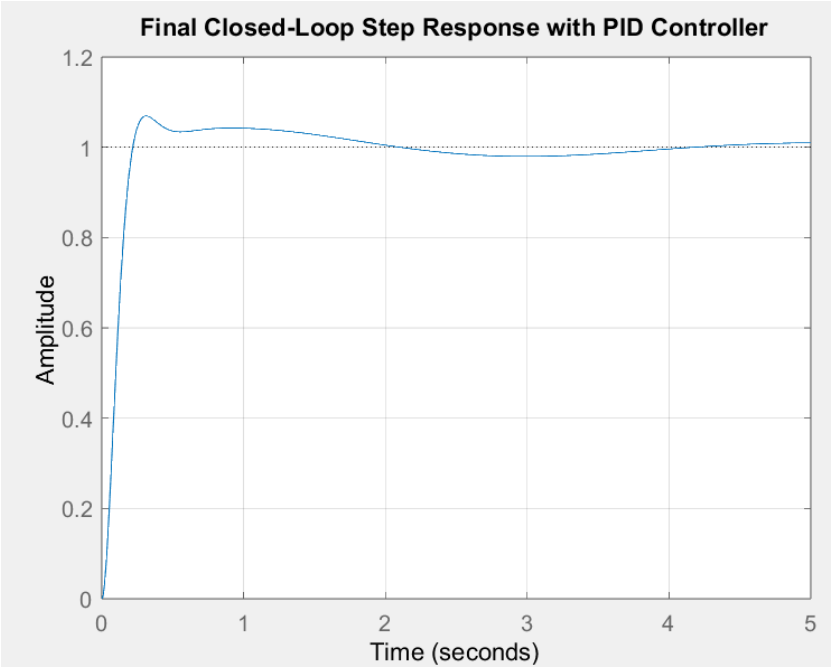

Figure 25: Step response of closed-loop system

```
Final Closed-Loop Steady-State Value: 1.0000
Final Closed-Loop Settling Time (2% criterion): 3.1052 s
Final Closed-Loop Percentage Overshoot: 6.92%
Final Closed-Loop Rise Time (10%-90%): 0.1441 s
Final Closed-Loop Steady-State Error: 0.0000
Settling time specification met.
Percentage overshoot specification met.
Steady-state error specification met.
All specifications are met.
```

Figure 26: Step response information for closed-loop system

From this information, it can be determined that **the PID controller implemented allows the stated design specifications to be achieved.**

Failsafe Investigation

The following is the output of the code written for the failsafe investigation, which is identical to the code written prior to simulate the step response of the closed-loop system, but with varied gain values as stated in the question.

Performance Evaluation Table:

Mode	SettlingTime_s	Overshoot_Percent	Stable	Meets_Specs
"Low Gain Failure (1%)"	23.004	0	true	false
"Normal Gain"	3.1052	6.9153	true	true
"High Gain Failure (500%)"	0.3554	36.328	true	false

Figure 27: Step response information for failsafe investigation

Since not all three of these modes met the required specifications, **the linear closed-loop system is not failsafe**.

Global Model of Nonlinear Closed-loop System

The code shown in figure 21 gives the following state-space model:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -20 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} e$$

$$u = \begin{bmatrix} -3975 & 500 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 2e \tag{12}$$

Where u represents the output of the controller and e represents the error signal. **The above equations represent the global model of the nonlinear closed-loop system**

Comparison of Local and Global Closed-Loop Models Over Small Step Input

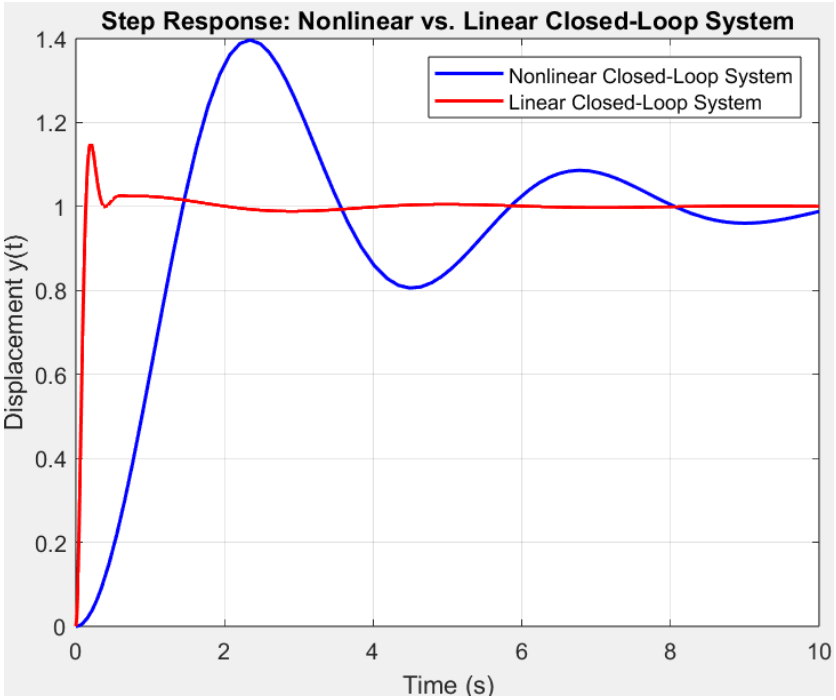

Figure 28: Step response comparison of linear and nonlinear closed-loop models over a small step input

While not quite the same output, the agreement of these two step responses for a small step input proves that the local linear closed-loop model is a good and sufficient approximation of the global closed-loop model for values near 0.

Effect of step Input Size on Performance Characteristics

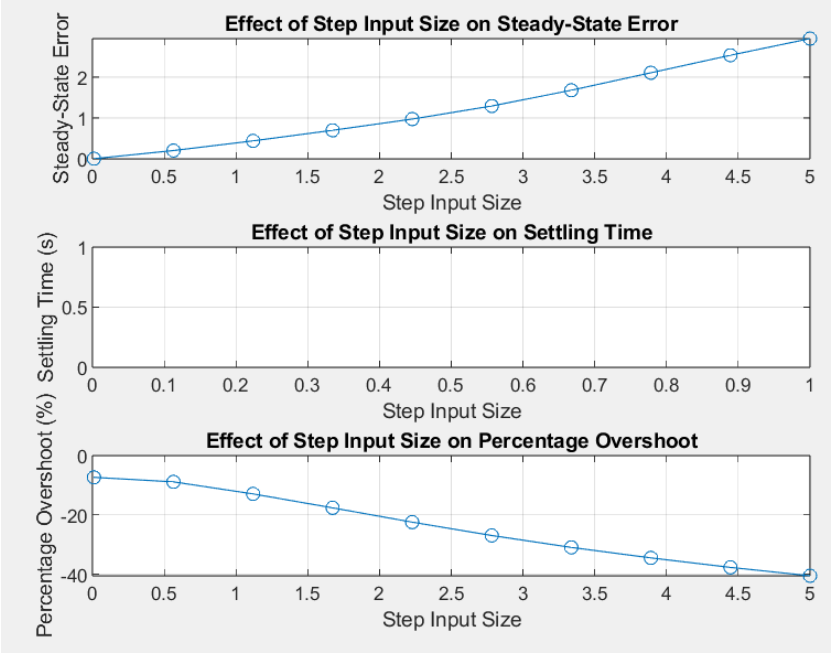

Figure 29: Effect of size of step input on performance characteristics

From these three plots, it can be observed that step size causes steady-state error to rapidly increase, with an increasing rate once the step input size passes 2. Moreover, the the step input has no effect on settling time (since the nonlinear system oscillates and never truly "settles"), but causes the percent overshoot to be more and more negative (creating undershoot, which is less severe).

Robustness Analysis with Damping Coefficient

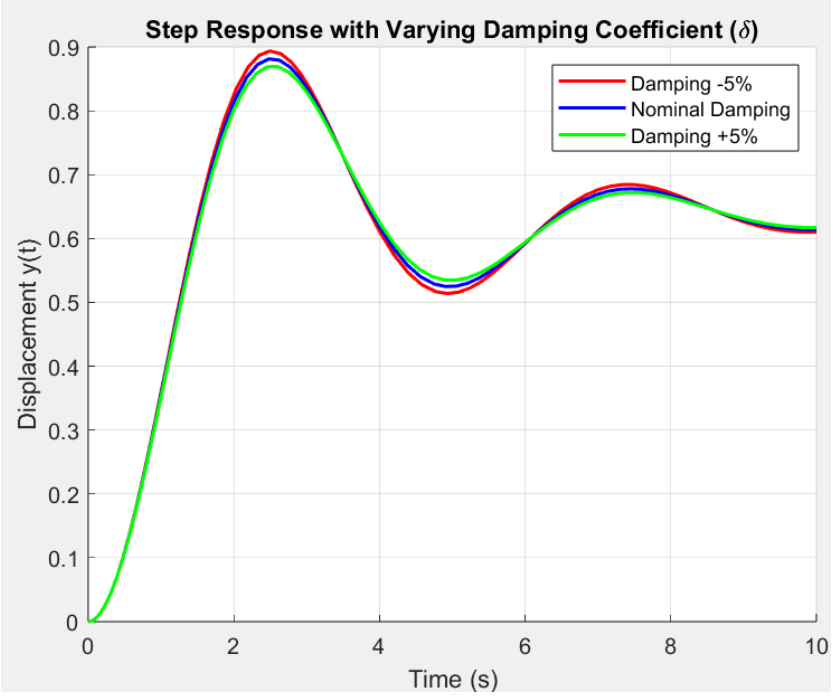

Figure 30: Effect of varying damping coefficient on step response

This graph proves that the global closed-loop system is robust as it pertains to varying the damping coefficient, as the three step responses are very much identical and all meet the performance specifications. To produce this graph, the same code was use from that in figure 22, but changed to plot the two altered damping coefficients.