

APPENDIX GA36a**LEVEL 4 ASSESSMENT SPECIFICATION**

Student name:	Sam Cartwright	Student ID number:	CAR19846934
Programme:	HNC /HND In Electrical and Electronic Engineering Level 4		
Module:	Dynamics Control and Instrumentation		
Module code:	GOEE4002	Contribution to Overall Module Assessment (%):	50%
Lecturer:	Emily Nyathi	Internal Verifier:	Dr Rabya Kadhim
Assignment Title:	Design and Control of a Mass-Spring-Damper System Using a PID Controller	Word count (or equivalent):	2176
Submission deadline:	29 April 2025	Return date of provisional marks & written feedback:	Within 20 working days
Submission method:	All written assessments, where practical and possible, must be submitted via Turnitin unless otherwise instructed by the Lecturer. (Please DO NOT put this assessment specification into Turnitin or it will match many similarities with other students' submissions.) Alternative submission method (if applicable): Late submission of the assessment will result in a late penalty mark. Penalties for late submission: Up to one week late, maximum mark of 40%. Over one week late, 0%. Only the Extenuating Circumstances Panel may approve a change to submission dates.		
Academic honesty / referencing:	Academic honesty is required. In the main body of your submission, you must give credit to authors on whose research and ideas your work is based. Append to your submission a reference list that indicates the books, articles, etc. that you have used, cited or quoted to complete this assessment.		

Design and Control of Mass-Spring-Damper System Using PID Controller

**By Sam Cartwright
Student Number: CAR19846934**

Contents Page

Introduction.....	4
Objectives.....	4
Research.....	5
Analysis.....	9
Results and Discussion.....	15
Conclusion.....	15
References.....	16
Appendix.....	17

Introduction

The study of mechanical dynamics is huge throughout the engineering space, playing a significant role in the design, analysis, and optimisation of systems exposed to forces and motion. Mechanical systems such as robotics and aerospace all the way to civil engineering, these mechanical systems have a great understanding of how certain components interact under different conditions. One example of this is the mass spring damper system. This model consists of a mass, a spring, and a damper. A real-world example of the mass spring damper in action is in building designs, this is used as the system helps with reducing the effects of vibrations that could come from earthquakes or strong winds. In aerospace, these systems help analyse how aircraft components respond to various forces during flight.

In this experiment, I will have my attention on designing a control system for a mass-spring-damper system using a PID (Proportional-Integral-Derivative) controller. By implementing and analysing a PID controller, this will help to enhance my knowledge and understanding of mechanical dynamics and control theory.

Objectives

The objective of this experiment is to provide a deeper understanding of the dynamics within a mass-spring-damper system through both theoretical and practical analysis. When researching about the spring damper system I will use relevant sources from textbooks and websites. Then for the practical side of the assignment I will be using a tool called MATLAB, this is where I will upload and run my code allowing me to analyse each figure.

Following this, I will investigate the system's dynamic behaviour, understanding how mass, spring, and damper components interact and influence overall system performance. The experiment will also focus on developing an accurate mathematical model of the system and simulating its behaviour using MATLAB, which will allow for a detailed analysis of its response to various inputs such as different possible PID values. A critical component of the experiment involves designing and implementing a PID controller to effectively manage the mass's position, trying to get the best performance for the system.

Lastly, in this experiment I will be comparing the open-loop and closed-loop responses, evaluating key performance metrics such as overshoot, settling time, and steady-state error, to assess the effectiveness of the control system.

Research

Mass Spring Damper

The Mass Spring Damper system consists of three components, hence the name. The mass in this particular system is the inertia, which will determine the force needed to change its velocity. The spring will connect to the mass which will provide a restoring force when the mass moves, this force will be proportional to the displacement from its equilibrium position. Lastly the damper, which resists the motion allowing proportional resistance to the velocity meaning the energy dissipates and reduces the oscillations. A simple example would be a shock absorber which will help smooth out ride when the vehicle is on a bumpy road.

Here is a formula found from research that is used for the spring component

Hooke's Law - ($F = -kx$)

Also found the differential equation of motion which can be derived from Newton's second law.

$$m \frac{d^2x}{dt^2} + c \frac{dx}{dt} = kx = F$$

PID Controller

What is a PID controller? "A PID (Proportional Integral Derivative) controller works by controlling an output to bring a process value to a desired set point."

This type of controller is constantly checking the error value which then will calculate the proportional, the integral and the derivative values. (PID Explained, 2018)

Set Point

The "set point" is a value entered by the user, for example a heating system which in my line of work I will need to set this for one of our machines (doughnut fryer) as I work for a company that manufactures commercial bakery equipment, having to position the set point is part of my checks when testing and making sure the temperature is set correctly. (PID Explained, 2018)

Process Value

Continuing from my set point example the "process value" is the current temperature value of the heating system. (PID Explained, 2018)

Error

To achieve the error value, you need the set point and process value. The error value has a simple equation which is Set point – Process value = Error. (PID Explained, 2018)

Gain

Gain is how much each part of the PID controller is multiplied. By changing the gain for the proportional, integral, and derivative parts, the user can control how strongly the controller will respond. This affects how the system reacts when the process value changes. (PID Explained, 2018)

Proportional (P)

The Proportional is found by multiplying the error by the P-gain. The purpose of the proportional is a quick response to help bring the process value closer to the setpoint. As the error gets smaller, the effect of the proportional part also reduces. (PID Explained, 2018)

Integral (I)

The Integral is worked out by multiplying the I-gain with the error, then with the controller's cycle time, the cycle time is how often the PID calculation is done. The value is added up over time to form the total integral, which builds up the longer the error continues and will have more of an effect on the output. (PID Explained, 2018)

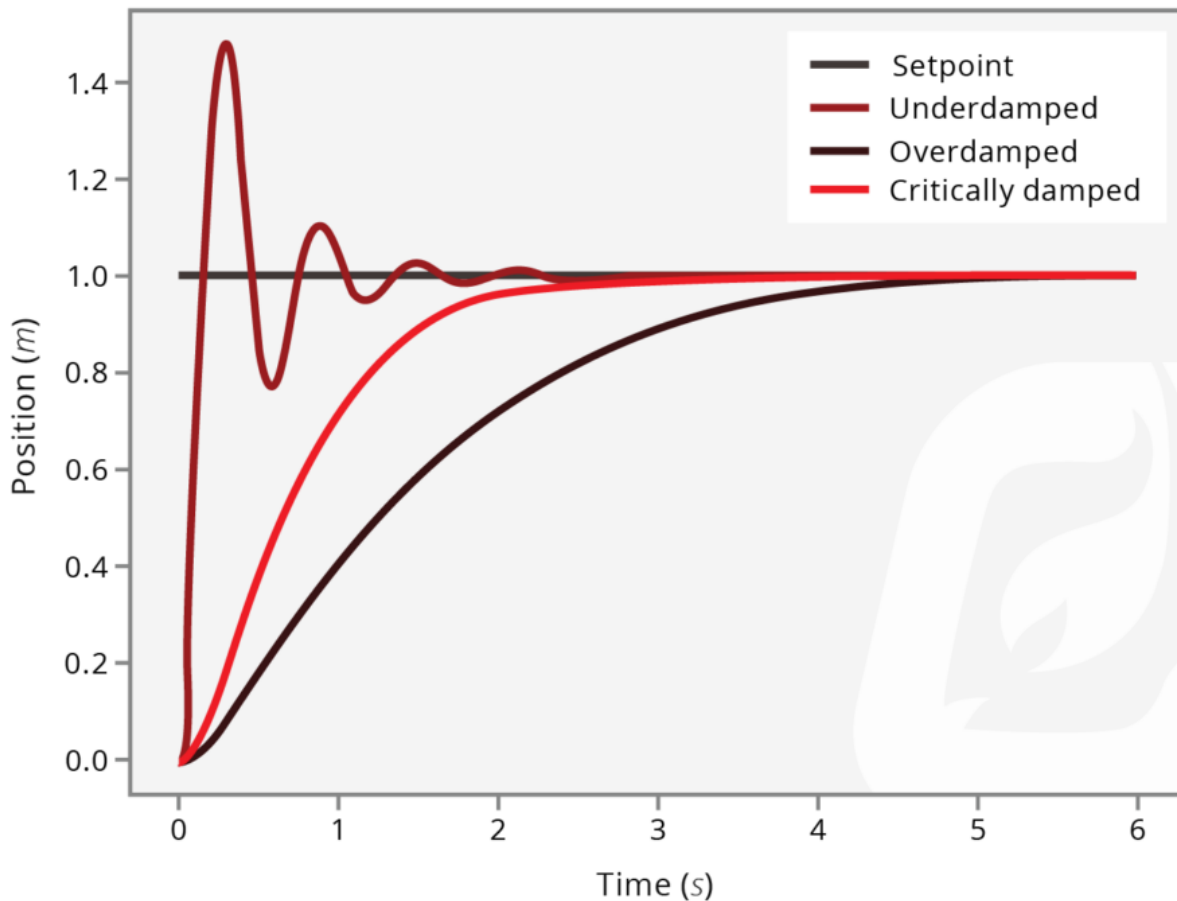
Derivative (D)

The Derivative is found by multiplying the D-gain by how quickly the process value is changing this is also called the ramp rate. The purpose is to predict where the process is heading and adjust the output in the opposite direction of the proportional and integral parts. This helps reduce overshoot if things are changing too quickly. (PID Explained, 2018)

PID Controller Response Types

A closed loop system is where you can find the following response types.

- Underdamped on a graph shows the response time is fast but will overshoot and have oscillations before levelling off and settling.
- Overdamped on a graph will have a very slow response time but at a steady rate.
- Critically damped on a graph will have a response time that is fast as possible but without overshooting.



PID Controller Response Types

[Image 1, PID Controller Response Types (PID Explained, 2018)]

Overshoot

Overshoot is when the output goes past the set point before settling down, this happens because there could be a high proportional gain meaning the system goes past the set point. A good example of this is the temperature being set a specific temperature and it is overshooting by 5,10 or even more in temperature. This would cause the chamber of the heating system to get too hot resulting in possible issues later down the line. (Gowda, 2021)

Settling Time

The definition of settling time is in the name, it is the amount of time taken for the output to settle within a given range, this is typically between 2-5% of the final value. Settling time is important because the system will be more stable faster with a shorter settling time compared to a long settling time which will make it feel slow. (Gowda, 2021)

Open vs Closed Loop System

An open loop system is a control system which has not got a feedback path, this means the output will have no effect on the input. An open loop system from a design aspect is a very simple design model. The control action of the open loop system is completely independent of the output of the overall system. Looking into the response time for this system it has a fast response because of the no feedback path, but the reliability of the open loop system can be less reliable. Now onto the maintenance and cost of the open loop system, I found that the maintenance and cost of the system is less comparing it to the closed loop system. (Ragno, 2023)

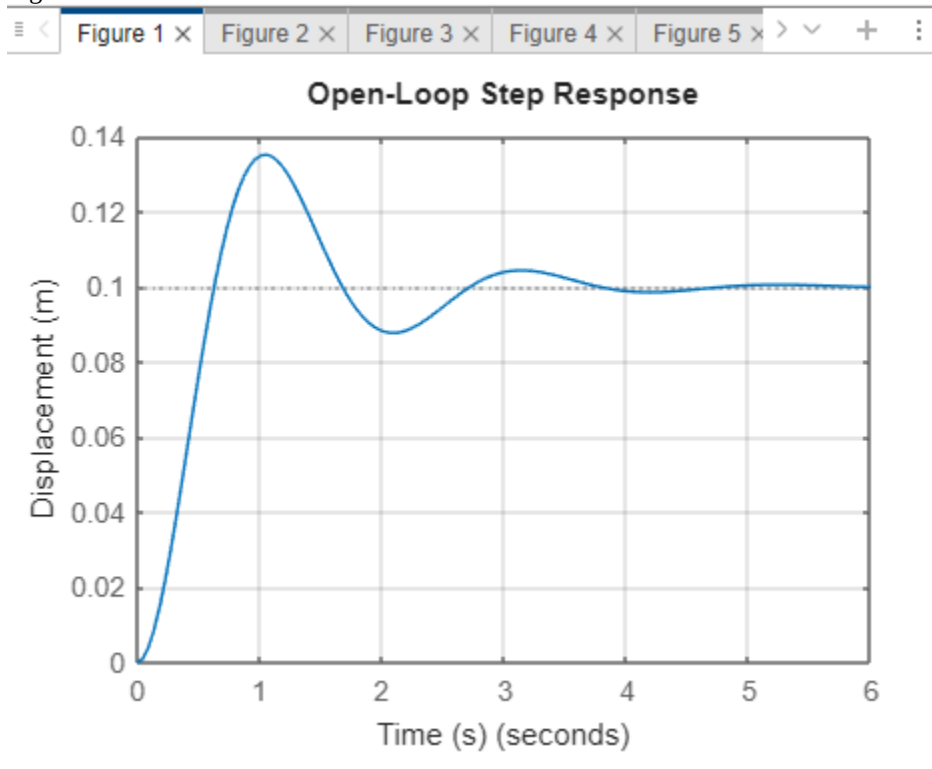
The closed loop system does have a feedback path in the control system and the control action is dependent on the output of the system. The closed loop system design is more complex compared to the open loop system, and has more parts. The response time for the closed loop system is slow and this is because of the system having the feedback path. Reliability for this system is high and the maintenance needed on this type of system is high. (Ragno, 2023)

Analysis

After researching about the mass spring damper system and gaining knowledge about the PID Controller I am ready to look into the design element on MATLAB. In this assignment I will be using AI to help create and check my codes. For my first task I went to ChatGPT to create the code. I then transferred the code into MATLAB Online to run the code and produce graphs that I will later investigate. (OpenAI, 2025). After running the code MATLAB produced 29 graphs in total, I then asked ChatGPT to add into the original code the settling time, overshoot and the steady state error for the closed loop system. (OpenAI, 2025). Having these graphs produced, I am now ready to analyse them and do so I will be choosing a number of different graphs that have different PID values to give me the best spread of results.

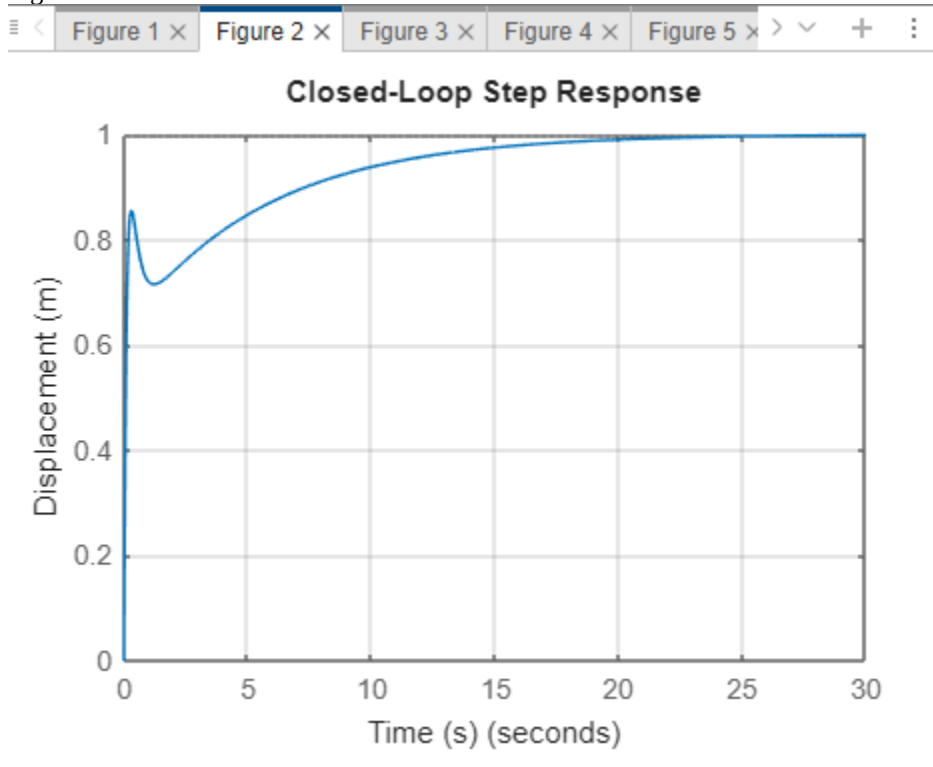
The first two graphs given to me by MATLAB were the Open-loop Step Response and the Closed Loop Step Response, as shown below.

Figure 1

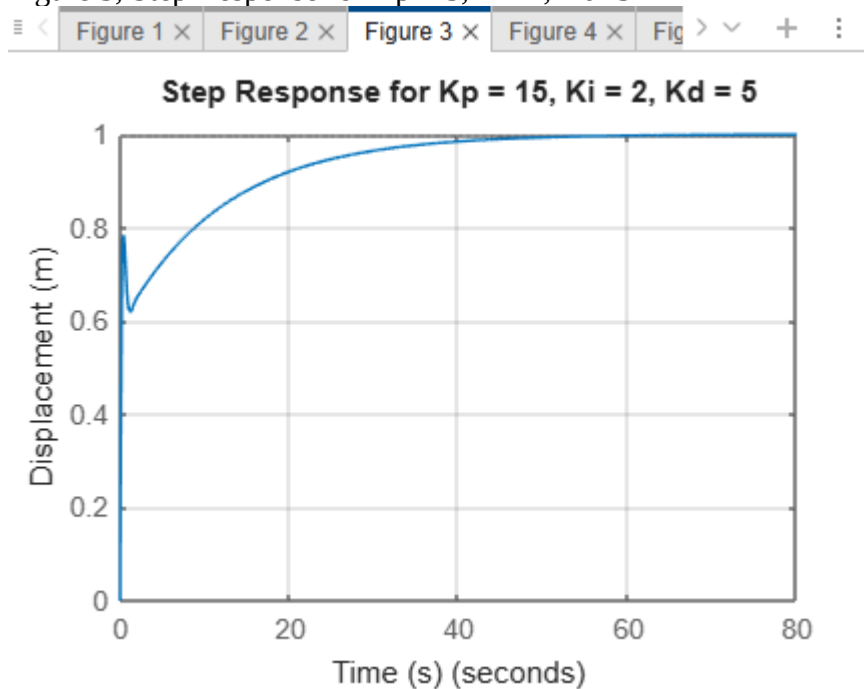


This graph shows an open loop system that is underdamped, as it has a fast response time but overshoots and has oscillations before it settles. Seeing that the graph is titled as an Open Loop System there is no feedback or PID controller.

Figure 2



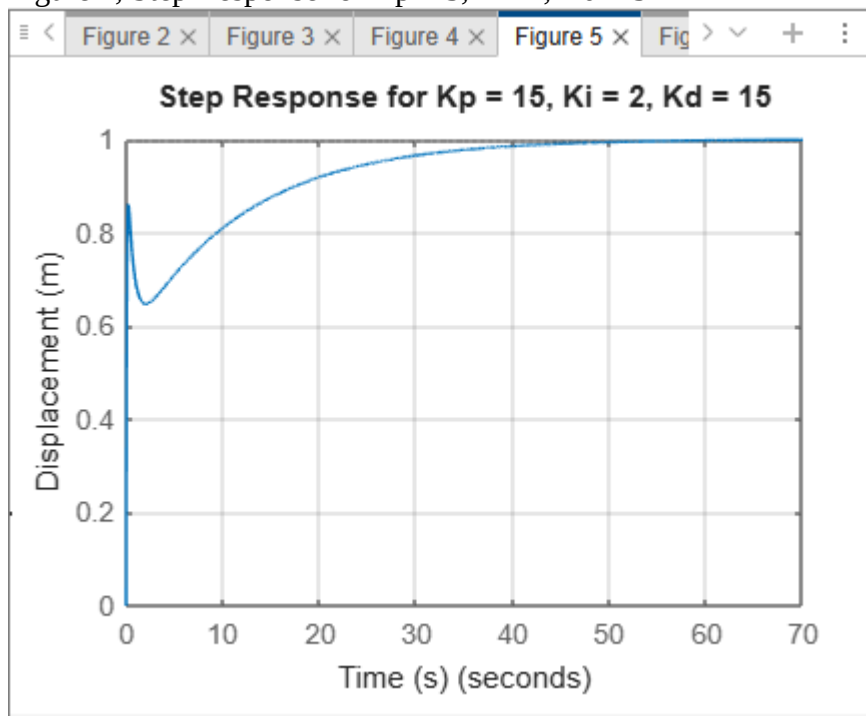
This graph shows a closed loop system, that is underdamped. There is no overshoot and the oscillation has a small drop but then heads up to the setpoint smoothly and slowly.

Figure 3, Step Response for $K_p=15$, $K_i=2$, $K_d=5$ 

This graph shows a stable but slightly underdamped closed loop system as there no overshoot and little oscillation. Looking into the settling time for this graph you can see the settling time is very slow around

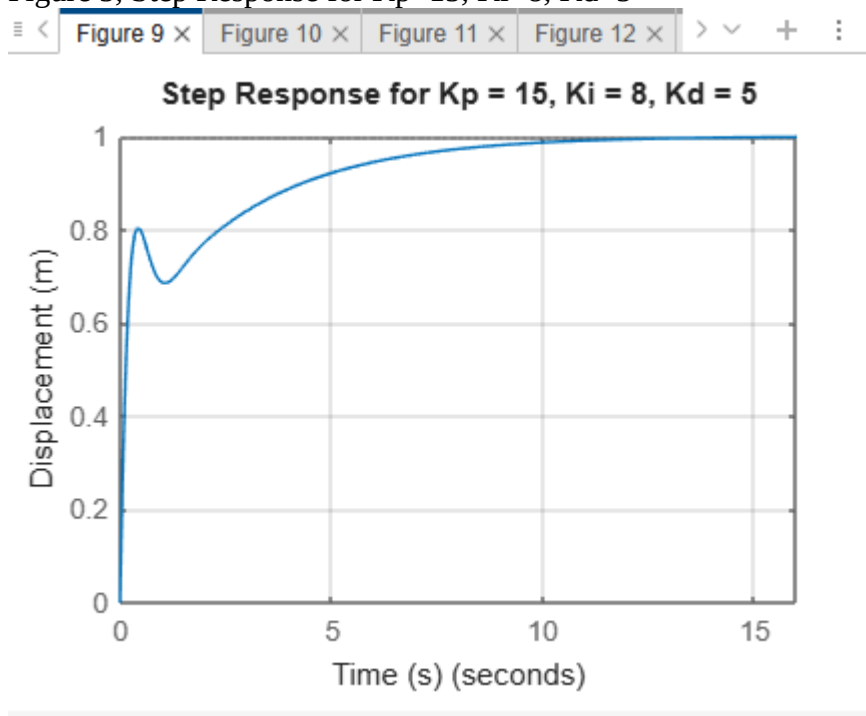
70 or 80 seconds to fully settle. A reason for this graph having a slow settling time could be because the proportional value could be higher, increasing this could reduce the settling time.

Figure 4, Step Response for $K_p=15$, $K_i=2$, $K_d=15$



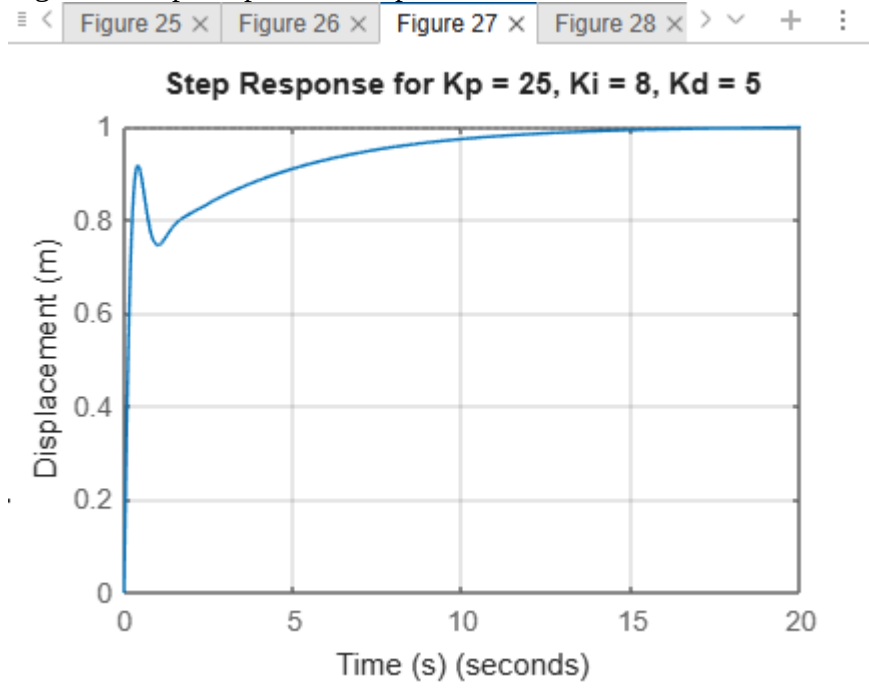
From this graph I can see straight away that the settling time is very slow (around 55 seconds), this is from a low K_i value. The rise at the start was very fast with no overshoot. Having the K_d value high allowed for the damping to be strong and smooth out the oscillations.

Figure 5, Step Response for $K_p=15$, $K_i=8$, $K_d=5$



This step response tells me that there is no overshoot for the intended target which is 1 meter, and has a smooth approach to that value with the settling time sitting around 12 seconds.

Figure 6, Step Response for $K_p=25$, $K_i=8$, $K_d=5$



This step response shows me that the initial rise is quick and so is the settling time but there is a big dip and oscillation.

See appendix for my explanation for figure 7.

Figure 7, Step Response for $K_p=15$, $K_i=8$, $K_d=15$

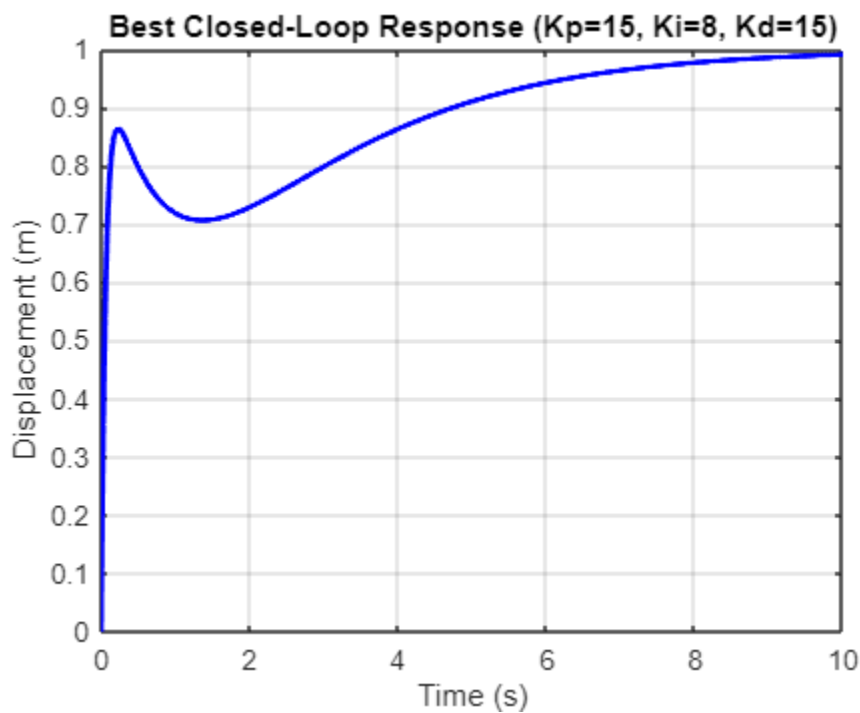
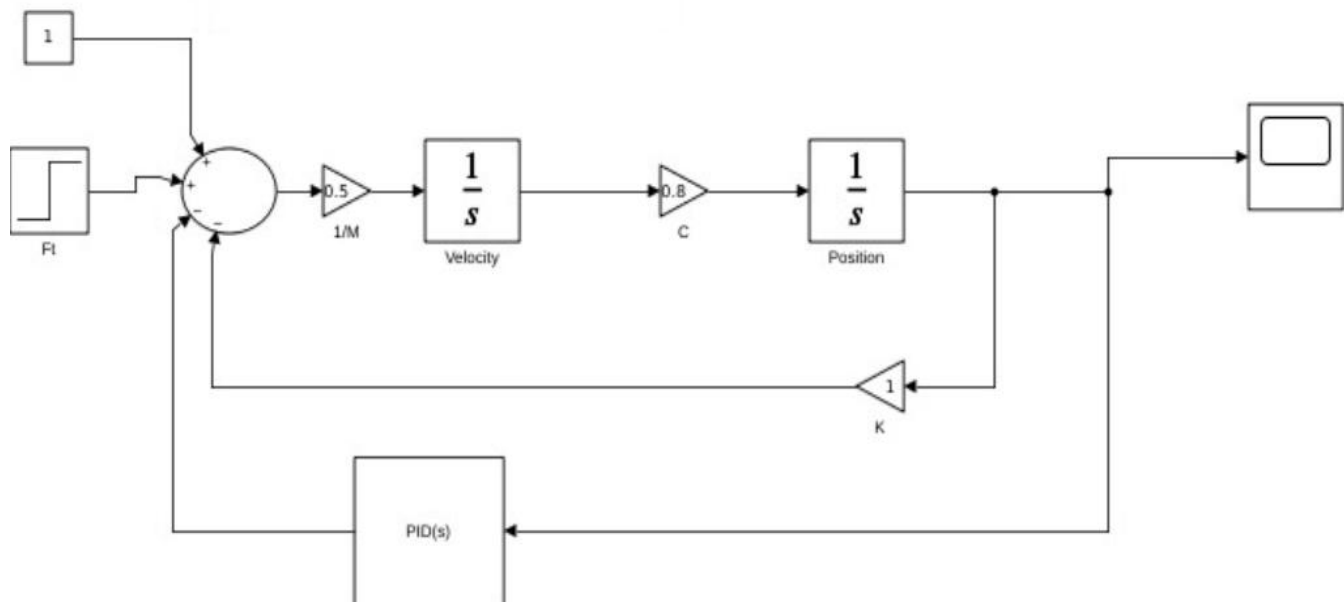


Figure 8 , The Command Window

Command Window

```
Best PID Settings:  
Kp = 15, Ki = 8, Kd = 15  
Settling Time: 8.3376 s  
Overshoot: 0.00%  
Steady-State Error: 0.0093  
>> |
```

Figure 9, Block Diagram



This is a closed loop control system for a mass spring damper system with a PID controller and I have presented it as a block diagram using MATLAB/Simulink. This is where I manually tuned the PID, using different values to produce different graphs.

Results and discussion

Analysing this small sample of graphs and using different values throughout, I found that increasing the proportional gain K_p speeds up the initial response but also allows for possible overshoot or undershoot. Also, while boosting the integral gain K_i eliminates steady-state error quickly, but at the cost of having more oscillations. Although increasing the derivative K_d will help have less oscillations and have shorter settling time, though it will slightly slow the initial rise.

From my analysis I thought that the graph to give the best results was $K_p=15$, $K_i=8$, $K_d=5$ because the graph had shown a quick initial response and a fast settling time, but then as explained in Appendix B the combination $K_p=15$, $K_i=8$, $K_d=15$ delivered the best graph out of the other figures, reaching the setpoint in under 12 seconds with only modest bumping. Whereas the more aggressive $K_p=25$, $K_i=8$, $K_d=5$ gave the fastest climb, but produced a big dip and a longer settling time.

Conclusion

In conclusion, this experiment has helped me get a better understanding and importance of a PID controller. I now know what, how and why they are used I believe this experiment has helped me not only theoretically, but also practically as I have gained experience in using MATLAB which was a new software to me. Also, being able to model the mass spring damper system on MATLAB, which allowed me to run codes producing different results and graphs that I later then was able to analyse the different response types that the given graphs had shown me. When reading the graphs I was looking for factors such as overshoot, settling time and steady state. The reason for focusing on these factors was because having different PID values could have a massive impact on the outcome of the graphs.

References

Gowda, Kushal . “Time Response Specifications 2.4 - Electronics Tutorials.” *CircuitBread*, 2021, www.circuitbread.com/tutorials/time-response-specifications-2.4.

Omega, Dwyer. “What Is a PID Controller and How Does It Work?”

<https://www.dwyeromega.com/En-US/>, 2025, www.dwyeromega.com/en-us/resources/pid-controllers.

OpenAI. “ChatGPT.” *Openai.com*, 2025, openai.com/chatgpt/overview/.

PID Explained. “PID Controller Explained • PID Explained.” *PID Explained*, 24 Nov. 2018, pidexplained.com/pid-controller-explained/.

Ragno, Maddie. “Open-Loop vs Closed-Loop Control Systems: Features, Examples, and Applications.” *RT Engineering*, 9 Aug. 2023, www.rteng.com/blog/open-loop-vs-closed-loop-control-systems.

Saini , Manish Kumar . “Difference between Open Loop and Closed Loop Control System.”

Www.tutorialspoint.com, 2023, www.tutorialspoint.com/difference-between-open-loop-and-closed-loop-control-system.

Gowda, Kushal . “Time Response Specifications 2.4 - Electronics Tutorials.” *CircuitBread*, 2021, www.circuitbread.com/tutorials/time-response-specifications-2.4.

Omega, Dwyer. “What Is a PID Controller and How Does It Work?”

<https://www.dwyeromega.com/En-US/>, 2025, www.dwyeromega.com/en-us/resources/pid-controllers.

OpenAI. “ChatGPT.” *Openai.com*, 2025, openai.com/chatgpt/overview/.

PID Explained. “PID Controller Explained • PID Explained.” *PID Explained*, 24 Nov. 2018, pidexplained.com/pid-controller-explained/.

Ragno, Maddie. “Open-Loop vs Closed-Loop Control Systems: Features, Examples, and Applications.” *RT Engineering*, 9 Aug. 2023, www.rteng.com/blog/open-loop-vs-closed-loop-control-systems.

Saini , Manish Kumar . “Difference between Open Loop and Closed Loop Control System.”

Www.tutorialspoint.com, 2023, www.tutorialspoint.com/difference-between-open-loop-and-closed-loop-control-system.

Appendix

Appendix A

Code 1 – Initial code generated from ChatGPT . (OpenAI, 2025)

```
% MATLAB Code for a Mass-Spring-Damper System
clear
clc;
% Defining System Parameters
m = 1; % mass (kg)
k = 10; % spring constant (N/m)
c = 2; % damping coefficient (Ns/m)
% Transfer Function:  $G(s) = 1/(m*s^2 + c*s + k)$ 
num = 1; % Numerator
den = [m, c, k]; % Denominator
sys = tf(num, den); % Create transfer function
% Open-loop step response
figure;
step(sys);
title('Open-Loop Step Response');
xlabel('Time (s)');
ylabel('Displacement (m)');
grid on;
% PID Controller Parameters
Kp = 20; % Proportional gain
Ki = 5; % Integral gain
Kd = 10; % Derivative gain
% Create the PID controller
C = pid(Kp, Ki, Kd);
% Closed-Loop Transfer Function
sys_cl = feedback(C * sys, 1); % Feedback with unity feedback
% Step response of the closed-loop system
figure;
step(sys_cl);
title('Closed-Loop Step Response');
xlabel('Time (s)');
ylabel('Displacement (m)');
grid on;
% Manual tuning example
Kp_values = [15, 20, 25];
Ki_values = [2, 5, 8];
Kd_values = [5, 10, 15];
for Kp = Kp_values
    for Ki = Ki_values
        for Kd = Kd_values
            % Create PID controller
            C = pid(Kp, Ki, Kd);
            % Closed-loop system
            sys_cl = feedback(C * sys, 1);
            % Step response
            figure;
            step(sys_cl);
            title(['Step Response for Kp = ', num2str(Kp), ' ...',
                ' Ki = ', num2str(Ki), ' Kd = ', num2str(Kd)]);
            label('Time (s)');
            ylabel('Displacement (m)');
            grid on;
            pause(2); % Pause for 2 seconds before the next plot
        end
    end
end
```

Appendix B

Code 2 – This code was generated by AI after my analyses of the graphs. This was done as I wanted to see which figure out of the 29 produced was the best all rounder. (OpenAI, 2025)

Code 2

% Find Best Closed-Loop PID Controller for a Mass-Spring-Damper System

```
clear
clc
% System Parameters
m = 1; % Mass (kg)
k = 10; % Spring constant (N/m)
c = 2; % Damping coefficient (Ns/m)
% Transfer Function  $G(s) = 1 / (ms^2 + cs + k)$ 
num = 1;
den = [m, c, k];
sys = tf(num, den);
% Original PID parameter ranges
Kp_values = [15, 20, 25];
Ki_values = [2, 5, 8];
Kd_values = [5, 10, 15];
% Initialize best values
best_score = inf;
best_Kp = 0;
best_Ki = 0;
best_Kd = 0;
best_info = struct();
best_response = [];
% Target output (unit step)
target = 1;
% Evaluate all PID combinations
for Kp = Kp_values
    for Ki = Ki_values
        for Kd = Kd_values
            % Create PID controller and closed-loop system
            C = pid(Kp, Ki, Kd);
            sys_cl = feedback(C * sys, 1);
            % Get step response and performance metrics
            [y, t] = step(sys_cl, 10); % simulate for 10 seconds
            info = stepinfo(sys_cl);
            % Compute steady-state error (for unit step)
            steady_state_error = abs(target - y(end));
            % Score based on performance metrics
            score = info.SettlingTime + info.Overshoot + 10 * steady_state_error;
            % Update best score
            if score < best_score
                best_score = score;
                best_Kp = Kp;
                best_Ki = Ki;
                best_Kd = Kd;
                best_info = info;
                best_response = [t, y];
            end
        end
    end
end
% Display best PID values and performance
fprintf('\nBest PID Settings:\n');
fprintf('Kp = %d, Ki = %d, Kd = %d\n', best_Kp, best_Ki, best_Kd);
fprintf('Settling Time: %.4f s\n', best_info.SettlingTime);
fprintf('Overshoot: %.2f%%\n', best_info.Overshoot);
fprintf('Steady-State Error: %.4f\n', abs(1 - best_response(end,2)));
% Plot best response
figure;
plot(best_response(:,1), best_response(:,2), 'b', 'LineWidth', 2);
title(sprintf('Best Closed-Loop Response (Kp=%d, Ki=%d, Kd=%d)', best_Kp, best_Ki, best_Kd));
xlabel('Time (s)');
ylabel('Displacement (m)');
```