

LEVEL 4 ASSESSMENT SPECIFICATION

Student name:	Dylan Jones	Student ID number:	JON19851773
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):	2000
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.		

Module Learning Outcomes (from module syllabus)

1. Solve, analyse and simulate the behaviour of dynamic mechanical systems using traditional analytical techniques and computer-based simulation.
2. Apply instrumentation and control system methodologies to dynamical systems and analyse the behaviour and response of the system and how it can be controlled.

Table of Contents

INTRODUCTION	3
BACKGROUND THEORY	3
OBJECTIVES.....	4
SYSTEM PARAMETERS.....	4
<i>Simulink Spring Damper</i>	5
<i>Comparison of PID Tuning Methods</i>	5
<i>System Modeling</i>	6
<i>Standard Open-Loop System</i>	7
<i>Standard Closed-Loop System (PID: 20, 5, 10)</i>	8
<i>Manually Tuned PID</i>	9
RESULTS AND DISCUSSION.....	12
CONCLUSION.....	13
REFERENCES.....	14
APPENDIX	15
<i>Code Used for System</i>	15

Introduction

Mechanical dynamics is a fundamental area of engineering that is associated with the motion of physical systems that are under any load/force. It plays a vital role in the design, analysis and optimisation of many mechanical and electromechanical systems. A good understanding of mechanical dynamics will allow an engineer to predict system behaviours, allow them to identify and rectify any system weaknesses, ensuring its stability and improving system performance across a wide range of industries. The experiment I am carrying out will focus on designing a control system for a mass-spring-damper system using a PID (Proportional-Integral-Derivative) controller, allowing me to have a system with the correct damping.

Background theory

The mass-spring-damper system consists of three primary components:

- Mass (m): Represents the inertia of the system, determining how much force is needed to change its velocity.
- Spring (k): Provides restoring force proportional to the displacement from its equilibrium position, following Hooke's law ($F = -kx$).
- Damper (c): Opposes the motion, providing resistance proportional to the velocity, which serves to dissipate energy and reduce oscillations.

The governing differential equation of motion for the mass-spring-damper system can be derived using Newton's second law:

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

Where:

- x is the displacement,
- $F(t)$ is the external force applied to the system.

When looking at a mass spring damper, the systems response depends on two things, the damping ratio and natural frequency. These define whether the system is underdamped, critically damped, or overdamped. This analysis helps engineers predict how real-world systems, such as vehicle suspension, behave under dynamic loading conditions. (Ogata and Brewer, 1971). When looking at what the PID controller (Proportional-Integral-Derivative) does when introduced to the system, with it being one of the most widely used feedback control mechanisms in both academic and industrial applications, it is designed to reduce the steady-state error, overshoot, and settling time of dynamic systems. (Åström, Albertos and Quevedo, 2001). To achieve the best results, proper tuning of the PID parameters is essential as this will allow you to gain the best optimal system performance.

Objectives

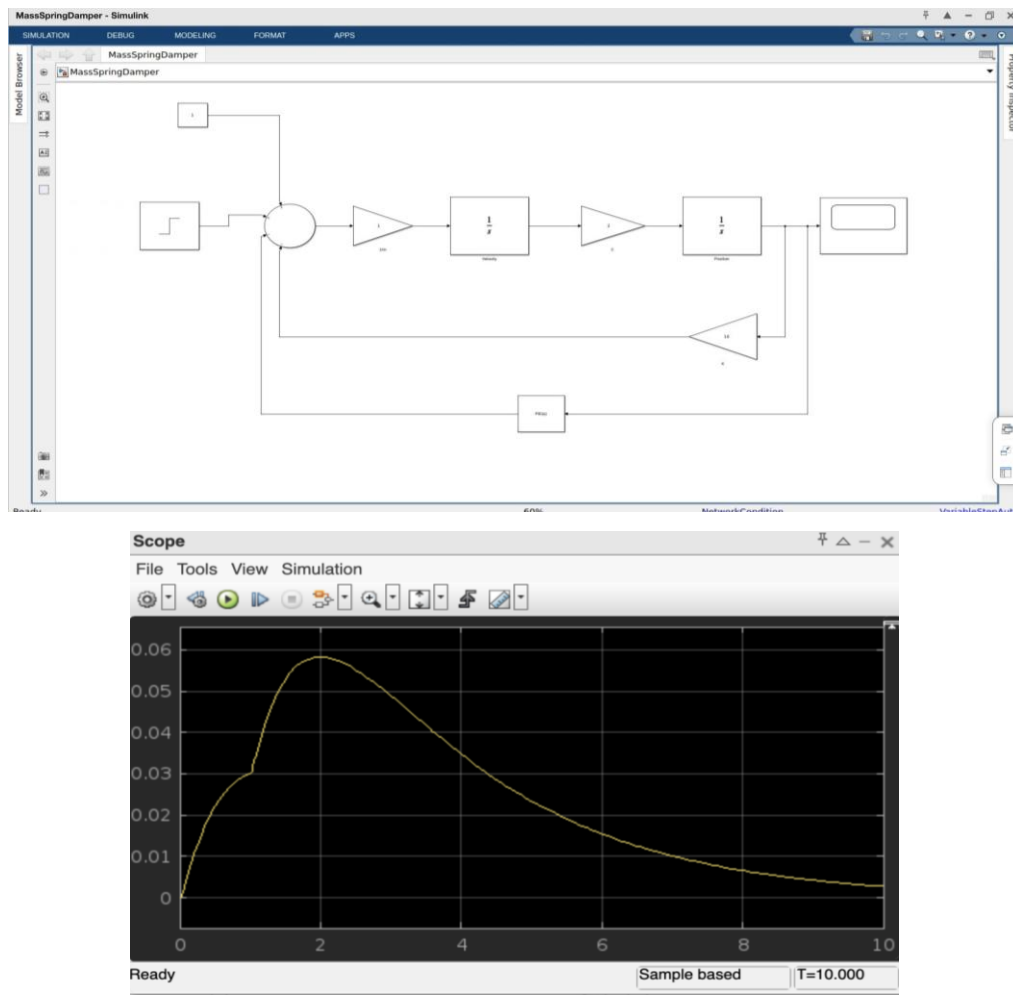
- To explore the dynamic behaviours of a mass-spring-damper system and understand its relevance in real-world engineering applications.
- To develop a mathematical model of the system and simulate its response using MATLAB to gain insight into its characteristics.
- To design and implement a PID controller aimed at accurately regulating the position of the mass, and to evaluate its effectiveness through performance analysis.
- To perform data analysis by comparing the responses of the system under open-loop and closed-loop control, with particular attention to key performance indicators such as overshoot, settling time, and steady-state error.

System Parameters

For this experiment, you will model a mass-spring-damper system with the following parameters:

- Mass (m): 1 kg
- Spring constant (k): 10 M/m
- Damping coefficient (c): 2 Ns/m

Simulink Spring Damper



This is my Simulink design of my manually tuned PID system. I have used the same values as obtained from my MATLAB results ($K_p = 15$, $K_i = 8$, $K_d = 15$), however the result shown by Simulink is different. I am assuming that there has been some system errors that have caused it to display an incorrect result. Due to the Simulink also solving errors compared to the MATLAB code only being me, the Simulink could be causing errors with the result. (MATLAB, 2023)

Comparison of PID Tuning Methods

	Ziegler–Nichols	Cohen–Coon	Lambda Tuning	Trial-and-Error
Best For	Self-regulating systems	First-order dead-time processes	Industrial & chemical plants	Nonlinear processes
Complexity	Medium	Medium	High	High
Agressiveness	High	Moderate	Low	Variables
Ease of Use	Simple	Requires step test	Requires process model	Manual tuning required

(Hulatt and Freitas, 2019)

System Modeling

$$m\ddot{x} + c\dot{x} + kx = F(t)$$

This is a second-order linear differential equation that models the motion of a mechanical system consisting of:

- Mass (m)
- Damping Coefficient (c)
- Spring Constant (k)
- External Force $F(t)$

(Nyathi, 2024)

Term Breakdown

$m\ddot{x}$ – Initial Force

This term represents the mass times the acceleration of the object. $\ddot{x}$ represent the second derivative of displacement with respect to time (acceleration) – Newtons second law: $F = ma$

$c\dot{x}$ – Damping Force

This term is representing the resistance from the damper, which is proportional to velocity. $\dot{x}$ represent the first derivative of displacement (velocity).

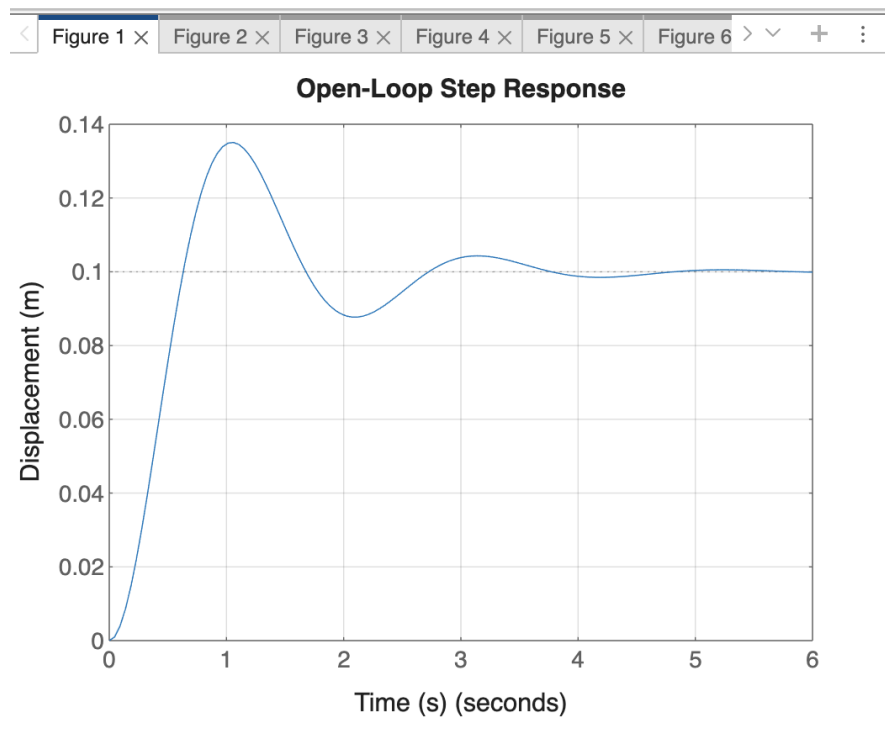
kx – Restoring Force From The Spring

The force exerted by a spring is proportional to its displacement (Hooke's Law). k is the spring constant (stiffness), and x is the displacement.

$F(t)$ – External Force

This is any force outside, acting on the system.

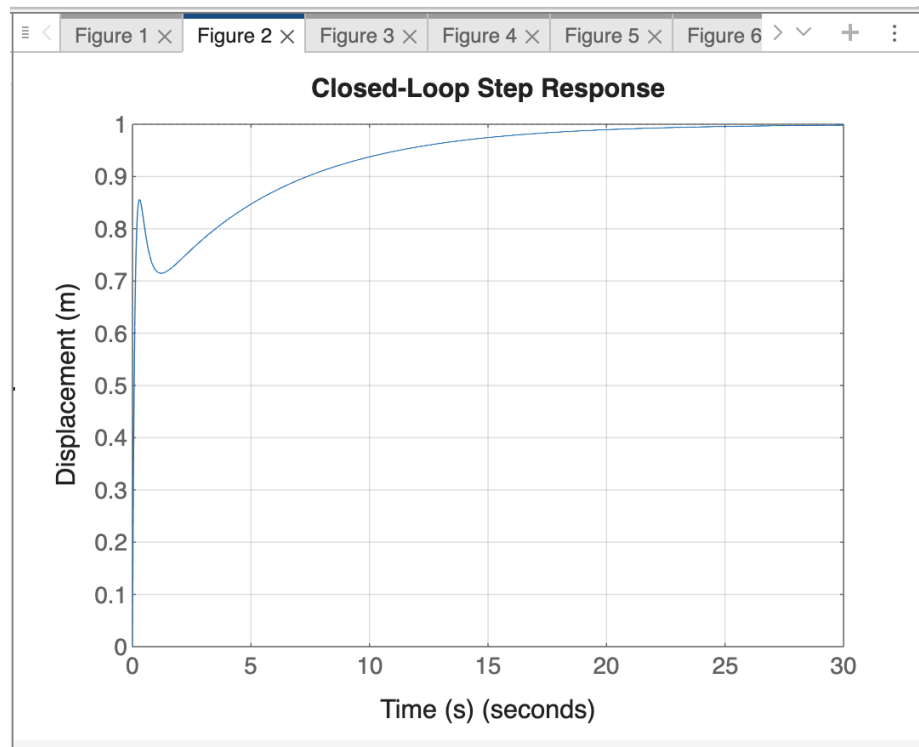
Standard Open-Loop System



Analysis

The graph above shows the behaviour of an open-loop system. The standard open-loop system shows significant instability due to the absence of any feedback control. When analysing the response of the system, the performance metrics taken into account are, overshoot, settling time, and steady-state error. The overshoot for this system is high, because it reaches a much higher point originally compared to the point the system settles at. This system also has lots of oscillations, indicating that the system is underdamped. This leads on to showing the system has a very long settling time, failing the system's stabilisation, meaning it took too long to get back to within 2% of the system's final stabilisation value. This is backed up by Bavafa-Toosi when he states "The settling time, τ_s , is the time taken for the oscillatory response to decay below a percentage of the input amplitude, Δ , often taken as $\pm 2\%$." (Bavafa-Toosi, 2017). Finally, when looking at the steady-state error, which is the difference between the final output and target value, it is clear to see there is a significant error as there is no corrective feedback. My aim setpoint is 1m and mine settles at 0.1m, therefore it has a displacement of 0.9m, meaning this system is underdamped.

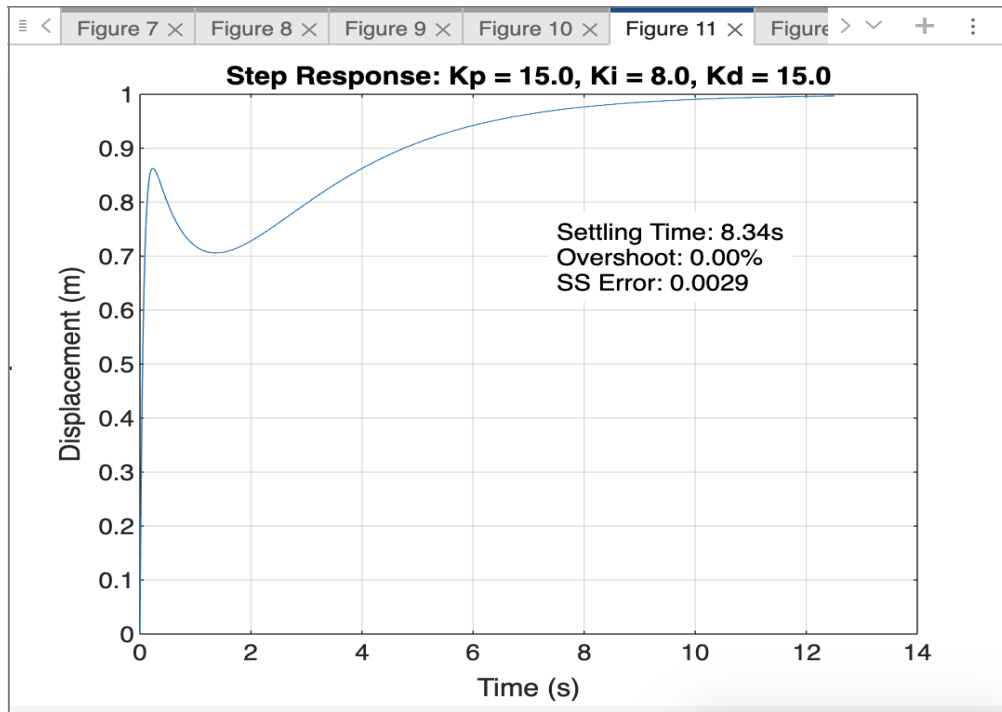
Standard Closed-Loop System (PID: 20, 5, 10)



Analysis

By looking at the graph, it is clear to see that by introducing a PID controller with the parameters of (20, 5, 10), it significantly improves the system response. The systems overshoot is zero due to the system not surpassing the desired setpoint. When looking at the settling time of the closed-loop system compared to the open-loop system, it is clear that the closed-loop has a much longer settling time. As explained by Hellerstein et al. (2004) "There is a fundamental trade-off between achieving zero steady-state error and having fast transient response." The reason it is slower is because the system is understanding what is happening, and manually tuning the PID as it goes, which takes time leading to a slower settling time. The closed-loop system is a much more controlled system as it does not show any harsh oscillations like the open-loop system does. There is also a reduced steady-state error compared to the open-loop system, indicating it has better tracker now a PID has been introduced.

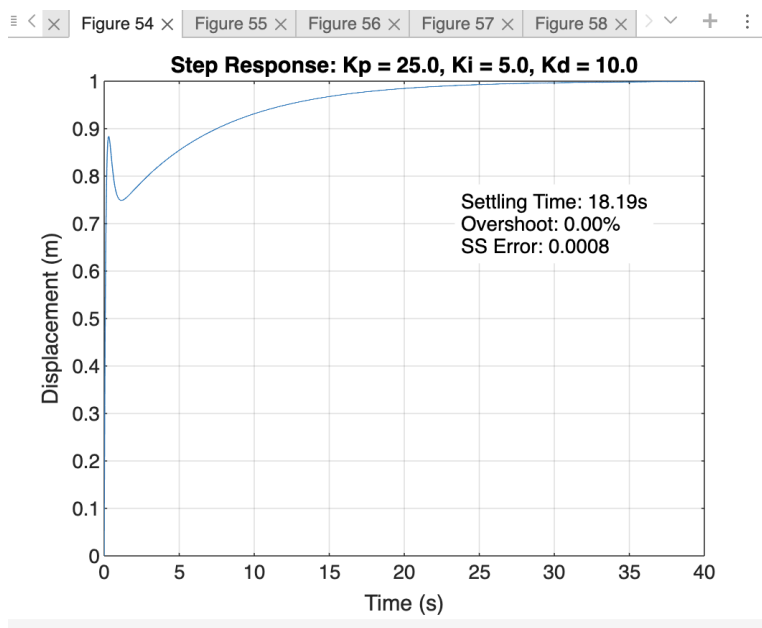
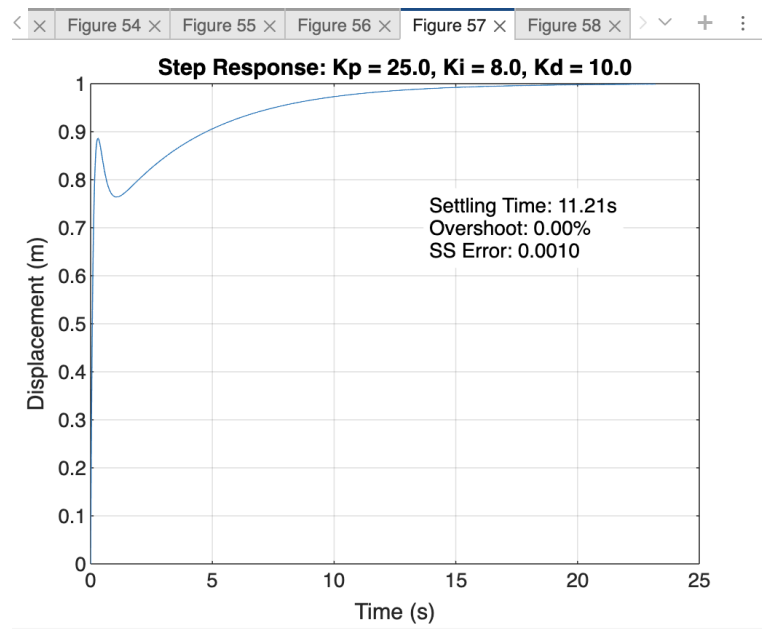
Manually Tuned PID

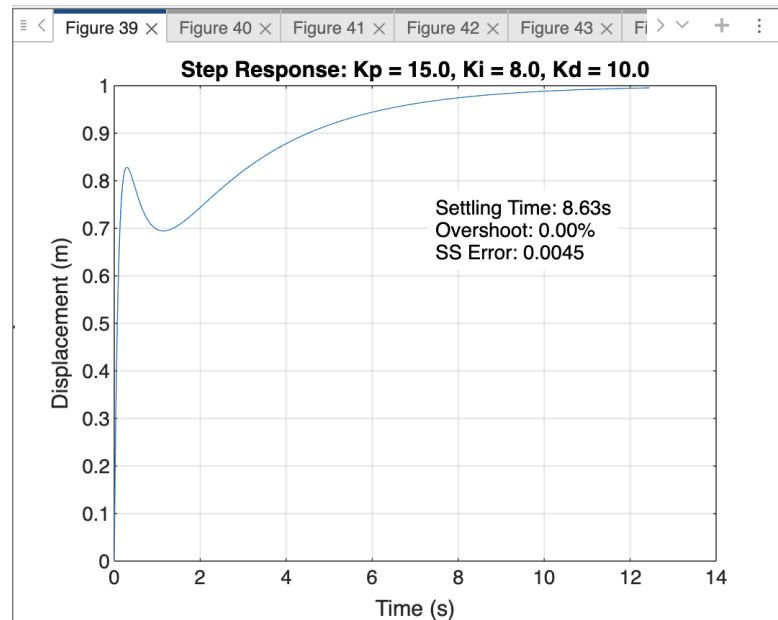
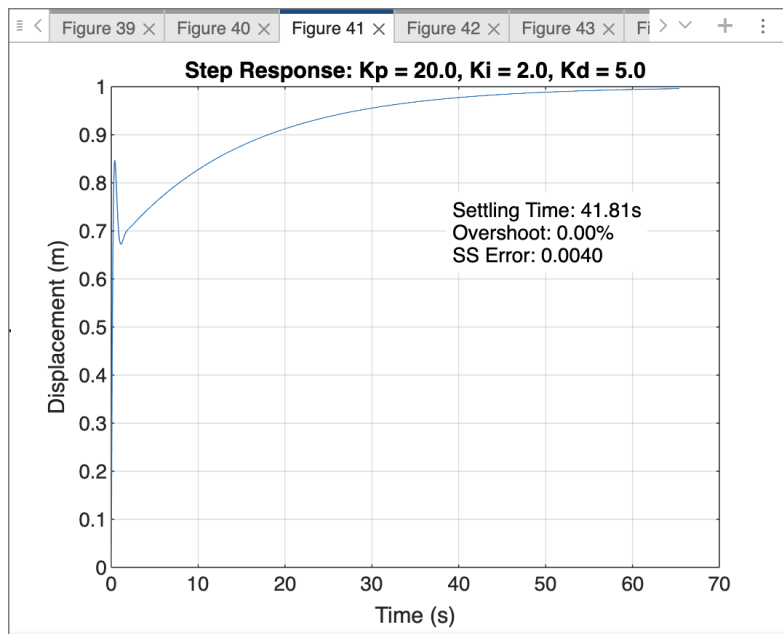
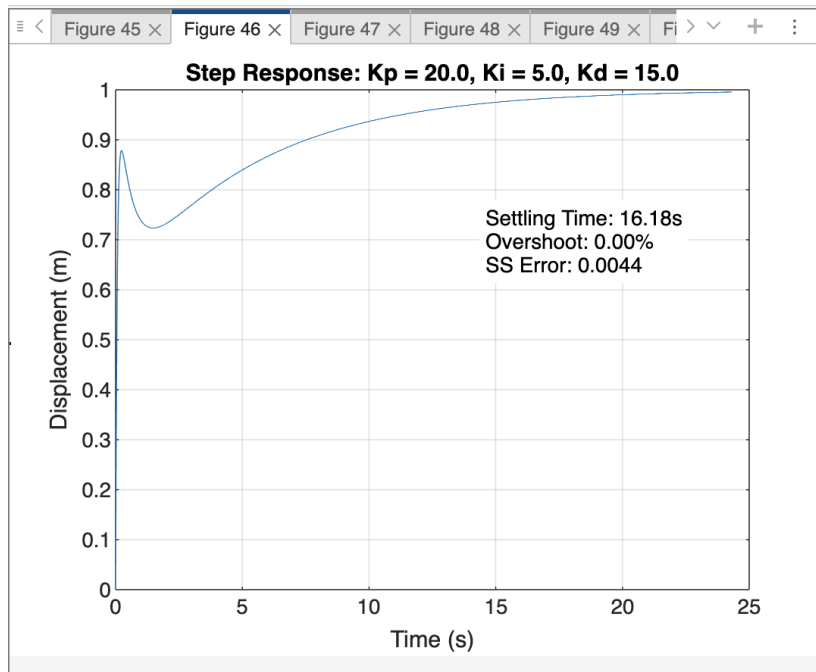


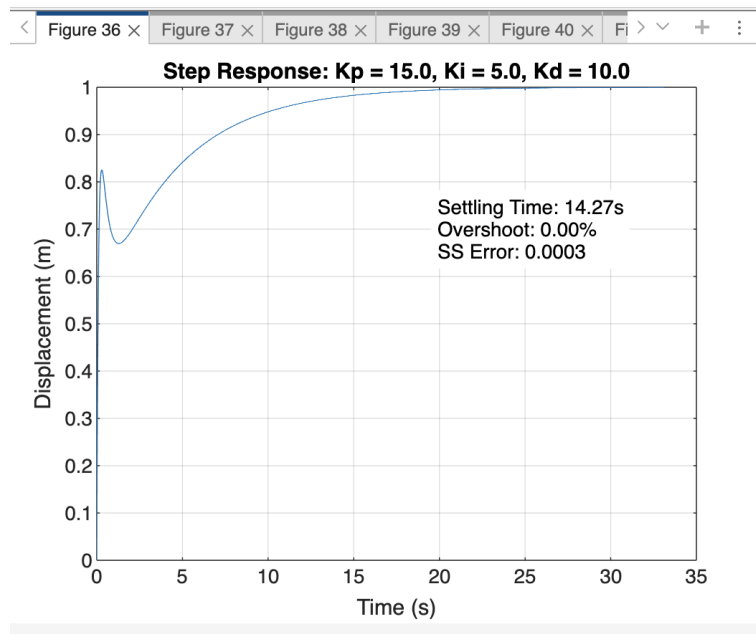
Analysis

This graph above is the damping system I have selected out of the 27 choices I had. The reason I have selected this system is because it has the lowest settling time, no overshoot, and one of the lowest steady-state errors. I could have chosen a system with a lower steady-state error, however that system had a much longer settling time, which led me to choosing this system. This systems overshoot has been eliminated, because as seen, the system reached the setpoint smoothly. The systems settling time is very low, with it having one of the shortest times amongst many different configurations. Lastly, the systems steady-state error is very minimal, with the output of 1 being achieved. This goes to show how important it is to manually tune the PID as this allows you to achieve the best results. As stated by Turan, "determining the other parameters of the PID controller by adjusting the optimum ratio gain (k_p) in a stable loop that minimises the settling time (t_s) and the error rate of the overshoot (M_p) constitutes the basis of the method." (Turan, 2021).

Other Results







All 6 of the above graphs were a few of the other choices I had when choosing from tuned PID controllers. As can be seen, the system I have chosen is the best overall system.

Results and Discussion

The three simulation graphs above have clearly illustrated the differences in system behaviour under open-loop, closed-loop, and a manually tuned PID control. These visually show the difference in performance metrics, confirming how important a correctly tuned PID control is to achieving a correctly damped system.

1	System Type	Overshoot	Settling Time	Steady-State Error
2	Open-Loop	High	Short	High
3	Closed-Loop (20, 5, 10)	None	Very Long	Low
4	Manual PID	None	Low (Best)	Very Low

The manually tuned PID system effectively reduced/got rid of the systems overshoot, minimised the steady-state error whilst providing one of the fastest settling times out of all available systems to choose from. This shows how important it is to fine tune PID parameters for optimal system control. As discussed by Lily Halatt, “**Manual Tuning:** Adjustment via trial and error, typically used in simple systems. Optimal tuning significantly affects stability and can reduce both overshoot and settling time.” (Halatt and Freitas, 2019). Therefore, by manually tuning my PID I have created a system that has a lower steady-state error and got rid of the systems overshoot.

Conclusion

This experiment has provided me with a practical understanding of control systems, emphasising the importance of PID controllers when managing dynamic mechanical systems. By creating a model of the mass-spring-damper system and implementing a PID controller, I was able to gain important knowledge on the design and tuning of a control system. As Hellerstein et al. explains, "feedback control is critical for achieving desired system behavior in the presence of uncertainty and disturbances," (Hellerstein et al., 2004) highlighting why feedback was essential in this experiment. The results highlighted the poor performance of open-loop systems due to the lack of feedback, which led to high overshoot and long settling times, as open-loop systems "are vulnerable to variations and cannot correct deviations from the desired output" (Hellerstein et al., 2004). By introducing a standard PID controller, the system's response was significantly improved, reducing the overshoot and steady-state error. However, it was through manual PID tuning that the best performance was achieved, proving that "the effectiveness of a control system often hinges on the correct tuning of its parameters" (Hellerstein et al., 2004). This demonstrates the critical role of tuning the PID controller for achieving precise and stable control in dynamic systems.

References

Ogata, K. and Brewer, J.W. (1971). Modern Control Engineering. *Journal of Dynamic Systems, Measurement, and Control*, [online] 93(1). doi:<https://doi.org/10.1115/1.3426465>.

Åström, K.J., Albertos, P. and Quevedo, J. (2001). PID Control. *Control engineering practice*, 9(11). doi:[https://doi.org/10.1016/s0967-0661\(01\)00061-2](https://doi.org/10.1016/s0967-0661(01)00061-2).

Nyathi, E. (2024). *Unsupported Browser | Microsoft Teams*. [online] teams.microsoft.com. Available at: <https://teams.microsoft.com/v2/> [Accessed 24 Apr. 2025].

Hellerstein, J.L., Diao, Y., Parekh, S. and Tilbury, D.M. (2004). *Feedback Control of Computing Systems*. [online] Google Books. Available at: https://books.google.co.uk/books?id=J9aMWtoELWYC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false [Accessed 28 Apr. 2025].

MATLAB (2023). *Ordinary Differential Equations - MATLAB & Simulink - MathWorks United Kingdom*. [online] uk.mathworks.com. Available at: <https://uk.mathworks.com/help/matlab/ordinary-differential-equations.html> [Accessed 28 Apr. 2025].

Turan, A. (2021). *Improved optimum PID controller tuning by minimizing settling time and overshoot*. [online] ResearchGate. doi:<https://doi.org/10.21203/rs.3.rs-844641/v1>.

Hulatt, L. and Freitas, G. (2019). *PID Control Tuning: Methods & Techniques | Vaia*. [online] Vaia. Available at: <https://www.vaia.com/en-us/explanations/engineering/robotics-engineering/pid-control-tuning/> [Accessed 28 Apr. 2025].

Bavafa-Toosi, Y. (2017). Stability analysis. In: *Introduction to Linear Control Systems*. [online] Academic Press, pp.265–331. doi:<https://doi.org/10.1016/B978-0-12-812748-3.00003-3>.

Appendix

Code Used for System

% 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); % 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)]);
```

```
            xlabel('Time (s)');
```

```
            ylabel('Displacement (m)');
```

```
            grid on;
```

```
            pause(2); % Pause for 2 seconds before the next plot
```

```
        end
```

```
    end
```

```
end
```