# 

# CSIP5202 – Lab 6 Portfolio

Victoria Hektor

P2629898

# Abstract

This report presents the implementation and tuning of a PID controller using the iRobot Create Toolbox in MATLAB, guided by Tim Wescott's "PID Control: PID without a PhD." [1]. It outlines the stepwise development and adjustment of the controller, beginning with a proportional component, followed by integral and derivative functionalities. A mechanism to limit the controller's output is incorporated, ensuring operational feasibility. The report also details the comparative analysis of various tuning parameters, alongside additional research methods. Performance evaluation, conducted in diverse navigational scenarios, demonstrates the controller's effectiveness and adaptability, offering valuable insights into the practical application of PID control in robotic navigation.

Contents

[CSIP5202 – Lab 6 Portfolio 1](#_Toc153496464)

[Abstract 1](#_Toc153496465)

[Table of Figures 3](#_Toc153496466)

[Introduction 4](#_Toc153496467)

[Task One: Implement a PID Controller 4](#_Toc153496468)

[Experimentation 4](#_Toc153496469)

[Analysis 4](#_Toc153496470)

[Task Two: Tuning of the PID Controller 4](#_Toc153496471)

[Experimentation 4](#_Toc153496472)

[Analysis 5](#_Toc153496473)

[Optional Task – Adapting the program to a Feedforward Control PID 5](#_Toc153496474)

[Overall Results Analysis 5](#_Toc153496475)

[Learning Curves 9](#_Toc153496476)

[Issue One - The robot's proclivity to revert prematurely to random wandering: 9](#_Toc153496477)

[Issue Two - Computational challenges: 11](#_Toc153496478)

[Issue 3 – Attempt to Tidy up The Code: 11](#_Toc153496479)

[Resolutions: 12](#_Toc153496480)

[Different Control Approaches Research 12](#_Toc153496481)

[Non-linear Control: 12](#_Toc153496482)

[Adaptive PID: 12](#_Toc153496483)

[Feedforward Control: 12](#_Toc153496484)

[Sliding Mode Control: 12](#_Toc153496485)

[Fuzzy Logic Control: 12](#_Toc153496486)

[Model Predictive Control (MPC): 12](#_Toc153496487)

[State Feedback Control: 12](#_Toc153496488)

[Switching Control Strategies: 12](#_Toc153496489)

[Conclusion 12](#_Toc153496490)

[Bibliography 14](#_Toc153496491)

[Appendices 16](#_Toc153496492)

[Test One: 16](#_Toc153496493)

[Test Two: 19](#_Toc153496494)

[Test Three: 22](#_Toc153496495)

[Test Four: 24](#_Toc153496496)

[Test Five 27](#_Toc153496497)

# Table of Figures

[Figure 1 – PID Error Over Time: exaggerated tuning. 4](#_Toc153321621)

[Figure 2 - PID Error Over Time: reduced tuning (kp = 0.4, ki = 0.001, kd = 0). 5](#_Toc153321622)

[Figure 3 - depicting PID control terms over time before fine tuning. 6](#_Toc153321623)

[Figure 4 - depicting PID control terms over time, after fine tuning. 6](#_Toc153321624)

[Figure 5 – Heat Map Analysis – Pre tuning. 7](#_Toc153321625)

[Figure 6 – Heat Map Analysis – post tuning 7](#_Toc153321626)

[Figure 7 – Robot Trajectory before changes 8](#_Toc153321627)

[Figure 8 - Robot Trajectory after some minor changes 9](#_Toc153321628)

[Figure 9 - Robot Trajectory after all changes 9](#_Toc153321629)

# Introduction

This report delves into the practical implementation and tuning of a PID (Proportional-Integral-Derivative) controller, employing the iRobot Create Toolbox within MATLAB, and is informed by the foundational principles outlined in Tim Wescott's "PID Control: PID without a PhD", [1]. The first task involves the step-by-step development of the controller, commencing with the proportional aspect, followed by the integration of integral and derivative components. A crucial aspect of this implementation is the storage and manipulation of past error values, essential for the integral and derivative calculations. The process also includes the incorporation of a limiting mechanism to cap the controller's output, ensuring the robot's movements remain within feasible operational parameters. [2].

In the second task, the focus shifts to the fine-tuning of the PID controller. This involves comparing different tuning parameters to identify the most effective configuration, guided by techniques presented in the referenced literature and supplemented by additional research methods, [3]. The evaluation of the controller's performance is conducted through a series of tests, placing the robot in various challenging scenarios, including navigating corners, to rigorously assess its adaptability and effectiveness. This report not only documents the technical process of configuring a PID controller but also offers insights into its practical application in robotic navigation, underlining the nuances of achieving optimal performance in dynamic environments. The inclusion of test images provides a visual representation of the controller’s efficacy, enhancing the comprehensiveness of the study. [4].

# Task One: Implement a PID Controller

## Experimentation

The initial stage of the project entailed the development of a PID controller using the iRobot Create Toolbox in MATLAB. The controller's implementation was systematically conducted, starting with the proportional component. This phase focused on achieving primary control by adjusting the robot's response to the environmental stimuli, primarily through sonar sensor readings. The integral and derivative components were incrementally integrated, allowing the controller to account for cumulative past errors and the rate of error change, respectively. [3].

## Analysis

The analysis of the initial PID controller revealed a notable improvement in the robot's navigation, particularly in maintaining a consistent distance from walls and avoiding obstacles. However, it became apparent that the robot occasionally encountered scenarios where it became stuck or its movements became less efficient, indicating the need for further refinement, particularly in handling complex navigational challenges.

# Task Two: Tuning of the PID Controller

## Experimentation

The second phase involved the fine-tuning of the PID controller. Different sets of Kp, Ki, and Kd values were experimented with to find the optimal balance for various navigational scenarios. The tuning process was iterative, involving adjustments based on the robot's performance in real-time simulations. Special attention was given to the robot's behaviour in corner navigation and its ability to switch effectively between wandering and wall-following modes. [5].

## Analysis

Post-tuning, the robot demonstrated enhanced adaptability and efficiency in navigation. The fine-tuned PID controller facilitated smoother transitions between different operational modes and improved the robot's ability to extricate itself from potential stuck conditions. The adjustments made to the PID parameters effectively mitigated previous issues, resulting in more fluid and reliable navigation. [6, 7].

# Optional Task – Adapting the program to a Feedforward Control PID

So here I attempt to implement the Feedforward control method of a PID Controller, it was a path fraught with danger but with some success, here is my analysis. (you can see the outputs in the appendices under *Test Five*):

State Transitions Over Time: This graph should ideally show the robot's state as it changes from "Seeking Wall" to "Wall Following." However, this suggests that it may not be transitioning correctly, or it is transitioning but not maintaining the "Wall Following" state effectively.

Heat Map of Robot Position: This heatmap illustrates where the robot spent most of its time. The heat map shows that the robot is spending time along a line, which could indicate that it is following a wall.

PID Control Terms Over Time: The Proportional, Integral, and Derivative terms of the PID controller are plotted over time. This shows that the robot may not be following the wall correctly and shows in some parts, an aggressive response thus overshooting occurs.

PID Error Over Time: This graph shows the error in the robot's position relative to the wall over time. Ideally, this should converge to zero, indicating that the robot is maintaining a consistent distance from the wall. Fluctuations here suggest the controller needs to be more finely tuned, as it is showing that it’s leading to the robot veering off course.

Path Followed by the Robot: This plot indicates the actual path taken by the robot. A smooth curve without sharp turns is ideal for wall-following behavior. If the path shows too much deviation from the wall or erratic behavior, this is a sign that the PID controller's parameters need refinement.

# Overall Results Analysis

*Figure 1* depicts the transient performance of a PID-controlled system, characterised by pronounced oscillations, which imply a persistent deviation from the setpoint. The observed error pattern suggests a propensity for the system to overshoot, followed by compensatory corrections, indicative of an aggressive control strategy potentially attributable to an elevated proportional gain. The absence of a discernible steady state within the presented temporal scope further suggests suboptimal tuning of the PID parameters, corroborating the necessity for a refined adjustment of the Kp, Ki, and Kd values. Moreover, the erratic nature of the error oscillations could be symptomatic of external perturbations or sensor noise infiltrating the feedback loop, necessitating an investigation into the robustness of the control system. In an academic context, this graph is imperative for the iterative process of PID controller optimisation, seeking to attenuate the amplitude of oscillations and achieve a damped system response with a minimal steady-state error.

*Figure 2* demonstrates the temporal progression of the error in a PID-controlled system after adjustments to the tuning parameters, specifically a reduction in the proportional gain (Kp) and the elimination of the derivative term (Kd). The markedly reduced amplitude and frequency of oscillations compared to the prior state indicate a significant diminishment in system responsiveness and a corresponding decrease in overshoot. The integral term (Ki) is minimally set, as evidenced by the gradual convergence towards zero error over a prolonged duration, suggesting a tempered corrective influence and a diminished cumulative error effect. These adjustments have ostensibly led to a more stable system with less aggressive corrective action, albeit at the cost of response speed and potentially increased steady-state error. The plateauing of the error at a non-zero value towards the end of the graph could imply a persistent offset that the controller is unable to correct due to the conservative tuning, which may necessitate a slight increase in Ki to mitigate any residual steady-state error. The alterations in PID parameters thus reflect a trade-off between stability and precision in achieving the setpoint, underscoring the nuanced balance required in PID tuning to cater to specific control objectives.

A graph showing a graph

Description automatically generated with medium confidence

Figure 1 – PID Error Over Time: exaggerated tuning.

A graph of a graph

Description automatically generated

Figure 2 - PID Error Over Time: reduced tuning (kp = 0.4, ki = 0.001, kd = 0).

In the *Figure 3*, depicting PID control terms over time, there is a notable persistence of high frequency and high amplitude oscillations across the proportional and derivative signals, indicating a potentially over-responsive or under-damped system. These characteristics suggest that the controller's parameters may have been set too aggressively, thus failing to effectively minimise error whilst ensuring stability. Conversely, the integral component, represented in green, accumulates over time, reflecting the integral action's objective to eliminate steady-state error, yet it also highlights that the error is not being corrected swiftly, which can be indicative of cumulative drift or a systematic bias unaddressed by the proportional and derivative components.

In *figure 4*, post-tuning of the PID values, illustrates a marked reduction in the amplitude of oscillations within the proportional term and a complete absence of the integral and derivative actions. This significant change implies a substantial decrease in the proportional gain, leading to a less responsive system. However, the absence of the integral and derivative terms in the control signal post-tuning might be due to setting the respective gains (Ki and Kd) to zero. This adjustment results in a control strategy purely based on the proportional response, which can suffice for maintaining stability, but may not effectively counteract steady-state errors or respond adequately to dynamic changes in the system's behaviour.

The comparative analysis between these two states reveals the intricacies and trade-offs in tuning PID controllers. The pre-tuning phase seems to be characterised by a system that is excessively responsive to errors, hence exhibiting oscillatory behaviour. The post-tuning phase seems to err on the side of caution, potentially at the cost of precision and adaptability. The ideal tuning would strike a balance, where the proportional action promptly responds to errors, the integral action corrects for any residual bias, and the derivative action anticipates future errors, thus damping the response for a stable and accurate control system. [2].

A graph showing a line of red and green

Description automatically generated

Figure 3 - depicting PID control terms over time before fine tuning.

A screen shot of a graph

Description automatically generated

Figure 4 - depicting PID control terms over time, after fine tuning.

The first heat map, *Figure 5* - pre-tuning, exhibits a concentrated band of high-frequency visits along a specific path, as denoted by the gradient transitioning from blue to yellow. This suggests that the robot's movement was constrained to a narrow region, possibly indicating a repeated traversal over the same path or a lack of adequate exploration strategies within the algorithm.

In contrast, the second heat map in *figure 6*, post-tuning, shows a more distributed pattern of visits, with a broader range of colours indicating varied frequency across the grid. The presence of cooler colours (blues and greens) suggests a more exploratory behavior, with the robot covering different areas more evenly. The peak values are lower than in the pre-tuning map, which implies that the robot spent less time retracing its steps and more time navigating new areas.

The differences between the two maps reflect the impact of PID tuning on the robot's navigational behaviour. Initially, the robot might have been stuck or oscillating in a certain pattern, resulting in overrepresentation of specific coordinates on the map. After tuning, the improved PID parameters show a more effective wandering and wall-following behaviours, allowing for a broader exploration of the environment. This suggests that the tuning process successfully mitigated issues such as excessive dwelling in certain areas or ineffective state transitions, leading to a more efficient and purposeful exploration pattern. [2].

A graph of a heat map

Description automatically generated

Figure 5 – Heat Map Analysis – Pre tuning.

A chart of heat map of robot position

Description automatically generated

Figure 6 – Heat Map Analysis – post tuning

# Learning Curves

## Issue One - The robot's proclivity to revert prematurely to random wandering:

In the pursuit of refining the autonomous navigation of a mobile robot within an unstructured environment, the code presented herein faced a significant challenge in the seamless transition between random wandering and wall-following behaviours. The robot's proclivity to revert prematurely to random wandering, rather than adhering to a wall for a prescribed minimum duration, necessitated a nuanced approach to the state transition logic. To address this, a new state was introduced, encapsulating a 45-degree turn post wall-following, to ensure a deterministic behavioural sequence. Furthermore, a temporal mechanism was employed, wherein a timer was initiated upon entry into the wall-following state, thus enforcing a minimum adherence period of three seconds, as stipulated by the operational parameters. This adjustment curtailed the erratic state switching and endowed the robot with a more predictable and stable navigation pattern, as demonstrated by the *figures 3, 4 and 5,* which depict the journey from fledgling to finely tuned. The implementation of such a solution not only enhanced the algorithmic robustness but also provided a clearer framework within which the robot's decision-making process could be evaluated and further optimised. [7].

A graph with a line drawn on it

Description automatically generated

Figure 7 – Robot Trajectory before changes

A graph with a red line

Description automatically generated

Figure 8 - Robot Trajectory after some minor changes

A graph of a line drawn on a white background

Description automatically generated

Figure 9 - Robot Trajectory after all changes

## Issue Two - Computational challenges:

Throughout the process of refining a PID-controlled robotic navigation algorithm, a series of computational challenges were encountered and systematically addressed. Initially, the robot exhibited a propensity for random wandering without adequate wall-following behaviour, necessitating an adjustment to the state transition logic. Modifications to the PID constants, resulted in an enhanced balance between the robot's exploratory movements and its interaction with environmental obstacles. Subsequent errors, such as those caused by an incompatibility between sonar sensor readings and logical operator requirements, were rectified by employing the ANY and ALL functions to ensure scalar logical evaluations. The introduction of odometry data significantly improved the trajectory tracking, allowing for the generation of visual outputs that accurately represent the robot's path. Furthermore, a heat map was implemented to visualise spatial coverage, and a state transition diagram was incorporated to elucidate the robot's behavioural patterns over time. These iterative enhancements culminated in a robust simulation of autonomous navigation, which is instrumental in the robot's ability to efficiently explore and interact with its environment. [17].

## Issue 3 – Attempt to Tidy up The Code:

Corner Detection - The robot initially relied on a simple counter to determine when it had lost track of the wall, potentially indicating a corner. This method proved insufficient, as it led to premature or inappropriate turns, particularly in the scenario where the robot would turn right instead of left upon approaching a wall head-on.

Stuck Robot - Another challenge was ensuring that the robot could detect when it was stuck and not making progress. The original logic was based on movement over a fixed time interval, which wasn't reliably triggering corrective action.

Code Complexity - As the robot's behavior grew more complex, so did the code, necessitating careful integration of new logic into the existing structure without disrupting functional aspects.

### Resolutions:

Enhanced Corner Detection – An attempt to improve corner detection by calculating the rate of change in distance to the wall took place. This approach aimed to provide a more nuanced and timely response to impending corners.

Robust Stuck Detection - Revisiting the stuck detection mechanism and changing it to be more responsive by potentially incorporating more sophisticated sensor data analysis.

Refinement and Debugging - Continuous testing and debugging were crucial as the project progressed and with integration of new logic. The code was refined iteratively, ensuring that each change brought the project a step closer to the desired autonomous navigation.

# Different Control Approaches Research

Non-linear Control: This method implements a non-linear scaling factor for the control signal based on the magnitude of the error. This could involve quadratic scaling or other non-linear functions that might offer a more nuanced control as the robot approaches the desired setpoint. [9]

Adaptive PID: This method is implementing an adaptive PID controller that can adjust its parameters in real-time based on the current performance of the robot. This might involve increasing gains when the error is consistently in one direction or decreasing them if oscillation is detected. [10]

Feedforward Control: Combine PID control with a feedforward component that anticipates the need for adjustment based on the robot’s velocity and the curvature of the path it is following. [11]

Sliding Mode Control: Sliding mode control is a form of variable structure control that changes the gains based on the position of the robot within a boundary layer. This method can be robust to changes and uncertainties in the robot's model and environment. [12]

Fuzzy Logic Control: Fuzzy logic controllers can handle non-linear systems well by using fuzzy sets and rules instead of precise mathematical models, potentially offering smoother transitions in control when dealing with varying distances to the wall. [13]

Model Predictive Control (MPC): Though more complex, MPC uses a model of the robot’s dynamics to predict future states and determines control actions that optimise a cost function over a future time horizon. [14]

State Feedback Control: This method uses the state of the robot (position and velocity) to calculate the control action. You could use a state-space representation of the robot's dynamics and design a state feedback controller that includes integral action (similar to a PID). [15]

Switching Control Strategies: This method could implement a strategy that switches between different sets of PID parameters based on the situation. For example, have one set of parameters for when the robot is far from the wall and another for when it is close. [16]

# Conclusion

The endeavour to programme an autonomous robot with a PID controller not only refined the robot's navigation through intricate environments but also enhanced the project manager’s computational and programming expertise. By carefully tuning the PID parameters, the project achieved a delicate equilibrium between the robot's reactivity and its stability, enabling it to adeptly respond to dynamic environmental factors. This process illuminated the criticality of meticulous parameter adjustments and the value of empirical validation. In the *Test Four* section in the appendices, you will see that the robots behaviour was better at following walls when in a confined space.

The progression of the project facilitated a deeper understanding of PID controllers, to the extent that I now possess a well-founded confidence in articulating the nuances of PID control mechanisms. The insights gained from this project have laid a foundation for the potential exploration of sophisticated sensor integration and the adoption of machine learning paradigms. This exploration signifies a forward leap in my personal development at pushing the boundaries of what can be achieved in the evolving landscape of autonomous robotics.

The project aimed at programming an autonomous robot to navigate using a PID controller and offered a practical case study in robotics control systems. The initial approach to encapsulate the robot's behavior within a class structure met with challenges, underscoring the complexities of object-oriented programming in robotics, leading me to switch to a more comfortable (but messy) style of coding. Despite these initial obstacles, the iterative process of modifying and refining the controller's functions led to tangible improvements in the robot's navigation abilities, as evidenced by changes in the visual outputs.

Through successive trials, it was learned that even minor adjustments to the PID parameters—proportional, integral, and derivative gains—could significantly alter the robot's wall-following performance. The visualisations served as a key tool in this process, offering clear feedback on the system's behavior and facilitating a data-driven approach to tuning. The nuanced understanding of how each component of the PID controller influences robot behavior was a critical takeaway. The proportional component's impact on the robot's immediate response, the integral's role in correcting cumulative errors, and the derivative's anticipation of future errors were all explored and balanced against each other.

Moreover, the project illuminated the importance of state management within autonomous systems, particularly when transitioning between different modes of operation, such as seeking a wall and following it. The heat maps generated during the tests provided insight into the robot's interaction with its environment, particularly when starting from different positions. The project offered a comprehensive learning experience in robotic control theory and application, highlighting the importance of system design, parameter tuning, and the interpretative value of visual feedback in developing effective autonomous navigation systems.

*Please note: Large amounts of testing were carried out within this project and I have opted to show a highlight reel of the various stages, these are contained in the appendices for ease of readability.*

# Bibliography

[1]T. Wescott, “PID without a PhD,” Aug. 2018. Accessed: Dec. 07, 2023. [Online]. Available: <https://wescottdesign.com/articles/pid/pidWithoutAPhd.pdf>

[2]K. Agarwal, “A Proportional-Integral-Derivative Control Scheme of Mobile Robotic Platforms Using MATLAB,” *IOSR Journal of Electrical and Electronics Engineering*, vol. 7, no. 6, pp. 32–39, Jan. 2013, doi: <https://doi.org/10.9790/1676-0763239>.

[3]K. Fan and K. C. Dimiduk, “Using the Matlab-based iRobot create simulator to engage introductory computer programming students in program development and observing computational errors,” *Proceedings - Frontiers in Education Conference*, vol. 6, no. 1, Oct. 2011, doi: <https://doi.org/10.1109/fie.2011.6143104>.

[4]R. Carmona, Hong Gun Sung, Young Shik Kim, and H. Alberto Vazquez, “Stable PID Control for Mobile Robots,” *IEEE*, vol. 15, no. 1, Nov. 2018, doi: <https://doi.org/10.1109/icarcv.2018.8581132>.

[5]J. E. Normey-Rico, I. Alcalá, J. Gómez-Ortega, and E. F. Camacho, “Mobile robot path tracking using a robust PID controller,” *Control Engineering Practice*, vol. 9, no. 11, pp. 1209–1214, Nov. 2001, doi: <https://doi.org/10.1016/S0967-0661(01)00066-1>.

[6]A. Bârsan, “Position Control of a Mobile Robot through PID Controller,” *Acta Universitatis Cibiniensis. Technical Series*, vol. 71, no. 1, pp. 14–20, Dec. 2019, doi: <https://doi.org/10.2478/aucts-2019-0004>.

[7]A. N. Albab, E. Rahmawati, M. Yantidewi, I. Sucahyo, Dzulkiflih, and R. R. Firmansyah, “Control Position of Mobile Robot Based on Odometry Method and PID Controller,” *Journal of Physics: Conference Series*, vol. 1491, no. 1, p. 012039, Mar. 2020, doi: <https://doi.org/10.1088/1742-6596/1491/1/012039>.

[8]H. Khan, S. Khatoon, P. Gaur, M. Abbas, C. A. Saleel, and S. A. Khan, “Speed Control of Wheeled Mobile Robot by Nature-Inspired Social Spider Algorithm-Based PID Controller,” *Processes*, vol. 11, no. 4, p. 1202, Apr. 2023, doi: <https://doi.org/10.3390/pr11041202>.

[9]Y.-C. Tian, M. O. Tadé, and J. Tang, “A nonlinear PID controller with applications,” *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 2657–2661, Jul. 1999, doi: <https://doi.org/10.1016/s1474-6670(17)56452-6>.

[10]T. Hägglund and K. J. Åström, “An Industrial Adaptive PID Controller,” *IFAC Proceedings Volumes*, vol. 23, no. 1, pp. 251–256, Apr. 1990, doi: <https://doi.org/10.1016/s1474-6670(17)52729-9>.

[11]J. L. Guzmán and T. Hägglund, “Tuning rules for feedforward control from measurable disturbances combined with PID control: a review,” *International Journal of Control*, pp. 1–14, Sep. 2021, doi: <https://doi.org/10.1080/00207179.2021.1978537>.

[12]Rahmi Guclu, “Sliding mode and PID control of a structural system against earthquake,” *Mathematical and Computer Modelling*, vol. 44, no. 1–2, pp. 210–217, Jul. 2006, doi: <https://doi.org/10.1016/j.mcm.2006.01.014>.

[13]J. Lu, G. Chen, and H. Ying, “Predictive fuzzy PID control: theory, design and simulation,” *Information Sciences*, vol. 137, no. 1–4, pp. 157–187, Sep. 2001, doi: <https://doi.org/10.1016/s0020-0255(01)00119-0>.

[14]A. Aboelhassan, M. Abdelgeliel, E. E. Zakzouk, and M. Galea, “Design and Implementation of Model Predictive Control Based PID Controller for Industrial Applications,” *Energies*, vol. 13, no. 24, p. 6594, Dec. 2020, doi: <https://doi.org/10.3390/en13246594>.

[15]A. Uthman and S. Sudin, “Antenna Azimuth Position Control System using PID Controller & State-Feedback Controller Approach,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 3, p. 1539, Jun. 2018, doi: <https://doi.org/10.11591/ijece.v8i3.pp1539-1550>.

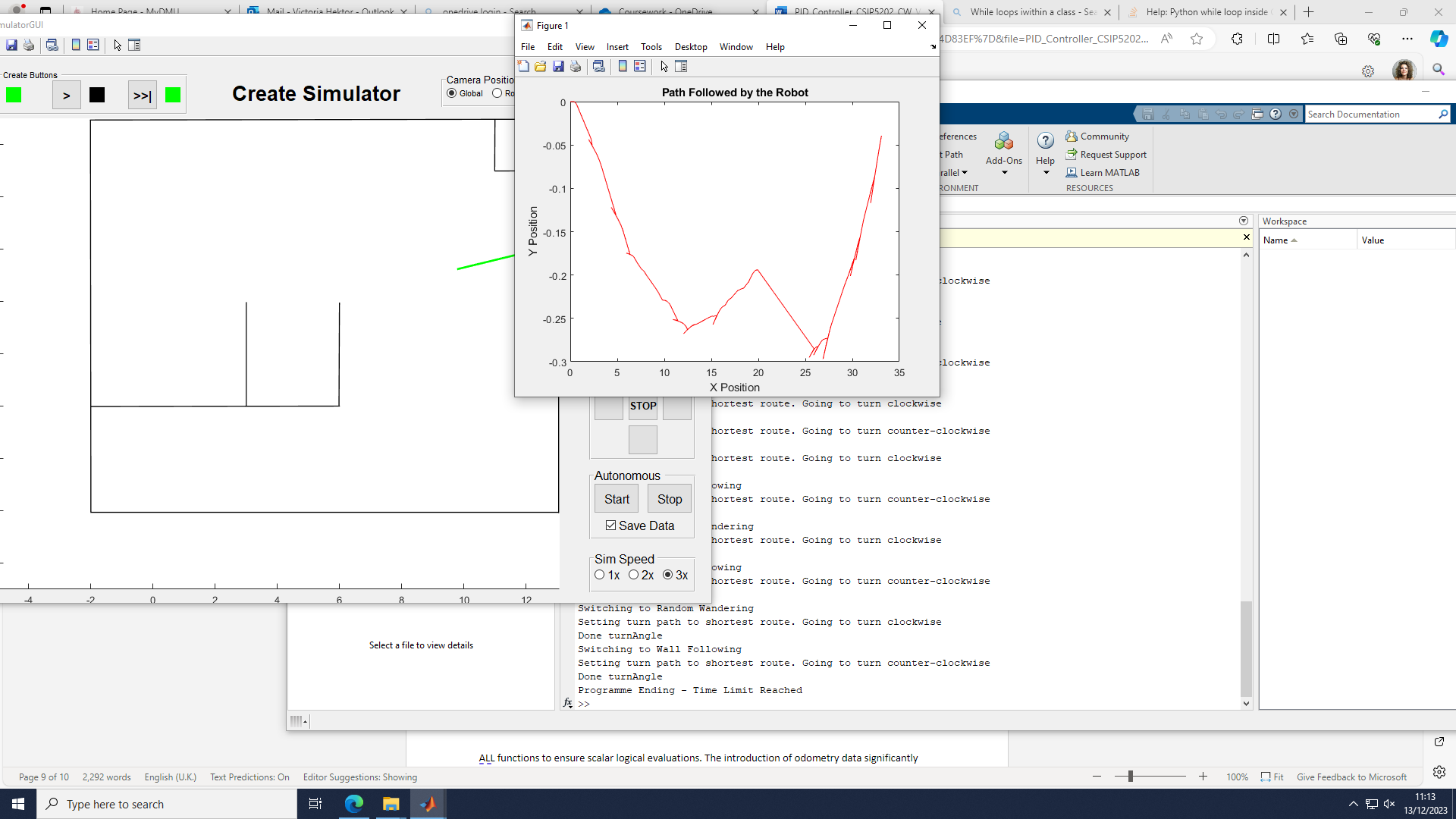
[16]M. Kim and S.-U. Lee, “PID with a Switching Action Controller for Nonlinear Systems of Second-order Controller Canonical Form,” *International Journal of Control, Automation and Systems*, vol. 19, no. 7, pp. 2343–2356, Mar. 2021, doi: <https://doi.org/10.1007/s12555-020-0346-4>.

[17]Mathworks, “Interactively create and explore Visualisations in the Live Editor - MATLAB - MathWorks United Kingdom,” *uk.mathworks.com*, 2021. <https://uk.mathworks.com/help/matlab/ref/createplot.html> (accessed Dec. 15, 2023).

# Appendices

Here are outputs from further testing, these are from further experimentation on adapting the code to better manoeuvrer around the map. Although this wasn’t a success, it is something I will keep trying to refine.

## Test One:



A graph with numbers and lines

Description automatically generated

A diagram of a heat map

Description automatically generated

A graph with red lines

Description automatically generated

A graph with numbers and lines

Description automatically generated

A graph with a red line

Description automatically generated

## Test Two:

Fine tuning the wall following code.

A graph of a state transition

Description automatically generated

A chart of a heat map

Description automatically generated

A graph with red lines

Description automatically generated

A graph with blue lines

Description automatically generated

A graph with a red line

Description automatically generated

## Test Three:

Additional control measures implemented.

A graph with numbers and lines

Description automatically generated

A chart of a heat map

Description automatically generated

A graph with red lines and numbers

Description automatically generated

A graph with blue lines

Description automatically generated

A graph with a red line

Description automatically generated

## 

## Test Four:

Putting the start point within the top right box.

A graph with text overlay

Description automatically generated A chart of a heat map

Description automatically generated A graph with green and red lines

Description automatically generatedA graph of a graph

Description automatically generated with medium confidence A red circle with white text

Description automatically generated

## Test Five

Switching it to a feed forward controller.

A graph with numbers and lines

Description automatically generatedA graph of a heat map

Description automatically generatedA graph with red lines and numbers

Description automatically generatedA graph with numbers and lines

Description automatically generated