Faculty of Technology, Design & Environment
School of Engineering, Computing & Mathematics

OXFORD BROOKES UNIVERSITY

Assessment cover

STUDENTS, PLEASE COPY THIS PAGE AND USE AS THE COVER FOR YOUR SUBMISSION

Module No:	**COMP6013**	Module title:	**BSc Computing Project**

Assessment number:	**ID: 19314173**	Assessment title:	**Development and Comparative Analysis of MPC and PID Controllers for Autonomous Vehicles**

Banner assignment identifier	*CWSXWEEKX*	Due date and time:	**04/04/2025 13:00**

Estimated total time to be spent on assignment:	90 hours

LEARNING OUTCOMES

On successful completion of this assignment, students will be able to achieve the following learning outcomes (LOs): *LO numbers and text to be copied and pasted from the module handbook*
Create, design, manage, plan, carry out, and evaluate a project involving the solution of a practical problem set in an appropriate social and economic context, taking into account other relevant factors such as risk
Apply practical and analytical skills acquired in the programme to the investigation of a substantial topic
Apply the scientific method and report findings using accepted formalisms
Identify and utilise trustworthy information sources, such as the ACM Digital Library to develop a coherent understanding of issues in the domain
Demonstrate the ability to carry out a substantial piece of work independently and critically evaluate the student's achievements and their own personal development
Use appropriate technologies such as online libraries and databases to find, critically evaluate and utilise both non-specialist and technical information pertinent to the project
Demonstrate an awareness of and work in a manner guided by the legal, professional, ethical, security and social issues relevant to the IT and telecommunications industry

Engineering Council AHEP4 LOs assessed (from S1 2024 Onwards)	
LO number	**LO text**
B3	Select and apply appropriate computational and analytical techniques to model broadly-defined problems, recognising the limitations of the techniques employed
B4	Select and evaluate technical literature and other sources of information to address broadly-defined problems
B5	Design solutions for broadly-defined problems that meet a combination of societal, user, business and customer needs as appropriate. This will involve consideration of applicable health & safety, diversity, inclusion, cultural, societal, environmental and commercial matters, codes of practice and industry standards
B6	Apply an integrated or systems approach to the solution of broadly-defined problems
B7	Evaluate the environmental and societal impact of solutions to broadly-defined problems
B8	Identify and analyse ethical concerns and make reasoned ethical choices informed by professional codes of conduct
B9	Use a risk management process to identify, evaluate and mitigate risks (the effects of uncertainty) associated with a particular project or activity
B10	Adopt a holistic and proportionate approach to the mitigation of security risks
B13	Select and apply appropriate materials, equipment, engineering technologies and processes

B15	Apply knowledge of engineering management principles, commercial context, project management and relevant legal matters
B17	Communicate effectively with technical and non-technical audiences

Statement of Compliance
By submitting this assessment I declare that the work submitted is my own and that the work I submit is fully in accordance with the University regulations regarding assessments.
(www.brookes.ac.uk/uniregulations/current)

Regulations governing the deposit and use of Oxford Brookes University Projects and Dissertations

Copies of projects/dissertations, submitted in fulfilment of Modular Programme requirements and achieving marks of 60% or above, shall normally be kept by the Library.

Statement of Permission *(please tick to sign)*

🟩	**Thank you for agreeing to have your work (i.e. dissertation and associated deliverables) deposited in the Repository's Undergraduate eDissertation collection.** By depositing this work, you agree to the following Terms and Conditions: You confirm that: a. you are the copyright owner and/or have the right to grant us licence to hold your work in our institutional repository (currently RADAR). We agree to: b. add the work to the COMP6013 Collection so that it is available to Oxford Brookes University members for the lifetime of the institutional repository. c. convert the work if necessary for long term preservation. We reserve the right to remove third party copyright material that we may identify in the work. We reserve the right to remove the work for any legal or administrative reason.

Use of AI Tools: You are required to use this form to declare which AI tools you have used and how you have used them. Please complete the form and attach it to your submission as an Appendix, if you have used such tools.

FORMATIVE FEEDBACK OPPORTUNITIES

Your supervisor will give you the following formative feedback:
- Weekly, during project supervision meetings
- Written feedback on Proposal (See Appendix A)
- Written feedback on Progress Report (See Appendix B)
- Feedback on presentation draft

SUMMATIVE FEEDBACK DELIVERABLES

Deliverable content and standard description and criteria	**Weighting out of 100%**
Presentation (see Appendix C) comprising: a) presentation of software, with video URL b) project slides c) summary poster (i.e. the final project slide)	**10%**
Final Report (see Appendix D) comprising: a) written dissertation b) software artefact URL link to source code	**90%**

ASSIGNMENT IN DETAIL

See Handbook Appendices A – D for assignment details and marking grid.

Title of the Project: "Development and Comparative Analysis of MPC and PID Controllers for Autonomous Vehicles"

Name and Student Number: Eduardo de Quiroga Tello / 19314173

Table of Contents:

Glossary

MPC (Model Predictive Control):
An advanced control strategy that uses a model of the system to predict future states and optimize control inputs over a prediction horizon, considering constraints.

PID (Proportional-Integral-Derivative):
A classical feedback control mechanism that calculates error and applies corrections based on proportional, integral, and derivative terms.

OBRA (Oxford Brookes Racing Autonomous):
The autonomous division of Oxford Brookes Racing, focusing on the development of self-driving race cars for academic and competitive purposes.

CVXOPT:
A Python library for convex optimization, used in this project to solve the quadratic programming problems required by MPC.

ROS2 (Robot Operating System 2):
A set of software libraries and tools that help build robot applications, including communication, control, and simulation.

GUI (Graphical User Interface):
A visual interface that allows users to interact with the software system through graphical elements such as windows, buttons, and plots.

Yaw:
The rotation of a vehicle around its vertical axis, representing its heading or direction.

x_dot / y_dot:
Symbols commonly used to denote velocity in the x and y directions, respectively.

psi / psi_dot:
Yaw angle (psi) and yaw rate (psi_dot) represent the vehicle's orientation and how fast it is turning.

.venv (Virtual Environment):
A tool in Python used to create isolated environments for managing dependencies and ensuring reproducibility across development setups.

GitLab:
A web-based DevOps lifecycle tool that provides a Git repository manager, used for version control and collaborative development.

FPS (Frames Per Second):
In animation, refers to how many frames are shown per second, influencing how smooth the animation looks.

Abstract

This project investigates and compares two widely used control strategies the Model Predictive Control (MPC) and the Proportional-Integral-Derivative (PID), for trajectory tracking in autonomous vehicles. The aim is to evaluate their performance, adaptability, and suitability for real-time applications, particularly in scenarios like autonomous racing. MPC offers predictive capabilities and constraint handling, while PID provides a simpler, reactive approach. Both controllers were implemented in Python within a shared simulation framework, using the same vehicle model to ensure a fair comparison.
The PID controller was designed to manage vehicle orientation and speed based on real-time errors, whereas the MPC controller, implemented with CVXOPT, was developed to optimize control actions over a future horizon. Visualization tools and performance metrics such as trajectory accuracy, yaw stability, acceleration smoothness, and computational load were used to compare both systems.
The results demonstrate that MPC significantly outperforms PID, particularly in handling complex curves, minimizing deviation from reference paths, and ensuring smoother, more stable driving behaviour. Despite its higher computational demand, MPC proves more scalable and better suited

for real-world autonomous systems, such as those developed by Oxford Brookes Racing Autonomous (OBRA). This study concludes that MPC represents a more robust and professional-grade solution for advanced autonomous vehicle control and lays the foundation for future integration and development.

1. Introduction

1.1 Overview of the Topic and Its Significance

Model Predictive Control (MPC) is an advanced control strategy widely utilized in autonomous vehicles due to its predictive capabilities and ability to handle complex multi-variable systems with constraints. MPC is especially effective in controlling multi-input multi-output (MIMO) systems, where interactions between variables such as steering and velocity must be managed in real-time to ensure safety and efficiency [2]. MPC optimizes control actions by predicting future states of the vehicle over a set horizon, adjusting inputs such as steering to minimize deviations from the desired trajectory, while respecting constraints like speed limits or safe distances.

In contrast, Proportional-Integral-Derivative (PID) control is a traditional and widely used method due to its simplicity and ease of implementation. PID controllers operate on a Single-Input Single-Output (SISO) basis, meaning they independently control one output variable in response to a single input signal—typically the error between a desired setpoint and the measured process variable. This structure makes PID suitable for basic control tasks, such as maintaining a target speed or steering angle. However, its inability to simultaneously manage multiple interconnected variables limits its effectiveness in more complex, dynamic environments such as autonomous driving. While PID can manage vehicle motion to an extent, it lacks the predictive and coordinated control capabilities needed for managing interactions between multiple system states [18]. In contrast, Model Predictive Control (MPC) is inherently a Multiple-Input Multiple-Output (MIMO) method that accounts for system dynamics, constraints, and future reference trajectories, enabling more precise and adaptive control in real-time. This makes MPC more suitable for autonomous vehicles where simultaneous control of both steering and velocity under varying conditions is required.

1.2 Motivation for the Project

My motivation for this project stems from a deep interest in automotive engineering. From a young age, I have been fascinated by the mechanics of how vehicles operate, which led me to pursue studies and projects related to the automotive industry. This passion has been further powered by my involvement with Oxford Brookes Racing Autonomous (OBRA), the university's autonomous racing team, where I´ll have the opportunity to work on real-world challenges in autonomous vehicle control systems.

Last year, the OBRA team utilized a Proportional-Integral-Derivative (PID) controller for the vehicle's control systems. While PID controllers are widely used due to their simplicity and effectiveness in various applications, I noticed several limitations when applied to more complex scenarios. PID controllers primarily react to current and past errors without the ability to predict future system behaviour. This reactive approach can lead to suboptimal performance, especially in dynamic environments where foresight is critical to maintaining stability and efficiency [4].

Recognizing the limitations of PID, I became interested in exploring alternative control methods, particularly MPC. This forward-thinking approach allows for more accurate and smoother control, especially in systems like autonomous vehicles where variables such as steering, velocity, and external constraints must be carefully balanced in real time [2].

1.3 Aims

The primary aim of this project is to design, implement, and evaluate two widely used control strategies, the Model Predictive Control (MPC) and the Proportional-Integral-Derivative (PID) for autonomous vehicle trajectory tracking, in order to critically assess their performance, limitations, and suitability for real-time applications in autonomous driving. The ultimate goal is to clearly demonstrate the superiority of MPC over PID in terms of adaptability, accuracy, and future applicability, especially in complex environments that autonomous vehicles encounter.

While PID controllers have traditionally been favoured for their simplicity and ease of implementation, their lack of prediction capabilities and difficulty in handling multi-variable constrained systems limit their usefulness in modern autonomous systems. In contrast, MPC offers a more sophisticated approach by predicting future vehicle states and optimizing control actions in real time under constraints, making it particularly suited for dynamic, high-precision scenarios such as autonomous racing or city driving.

This project seeks to quantify and visually demonstrate how MPC significantly outperforms PID in handling non-linearities, achieving smoother and more accurate path tracking, and maintaining stability under rapidly changing conditions. The project is also intended to lay the groundwork for the implementation of MPC in OBRA (Oxford Brookes Racing Autonomous), providing a scalable, robust, and efficient control algorithm that can be integrated into real-world autonomous systems.

1.4 Objectives

To meet the aims of this project, several objectives have been established. Firstly, a PID controller will be designed and implemented to manage the vehicle's yaw orientation and longitudinal velocity, ensuring it follows a predefined trajectory with acceptable stability and responsiveness. In parallel, a fully functional MPC system will be developed using Python and the CVXOPT solver, enabling real-time trajectory optimization while accounting for the vehicle's dynamic constraints. Both control strategies will be deployed within a common simulation environment that uses the same physical vehicle model, ensuring that performance comparisons are fair and consistent. The control behaviour of both systems will be visualized through animated simulations, allowing for direct comparison of critical performance indicators such as tracking accuracy, system stability, responsiveness, etc.
Additionally, the project will quantify and contrast the computational demands of each controller, emphasizing the trade-offs between control performance and real-time processing load. The scalability and robustness of both approaches, particularly that of MPC, will be evaluated in the context of future deployment in real-world autonomous systems such as OBRA. Finally, all findings will be thoroughly documented, including the respective strengths and weaknesses of each approach, along with clear recommendations for their future adoption in both academic and professional applications.

1.5 Product Overview

This project delivers a software-based implementation of two advanced vehicle control strategies, MPC and PID, for managing the steering and acceleration of an autonomous vehicle in a simulated environment. The primary focus is on the MPC controller, which leverages real-time optimization and predictive modelling to anticipate the vehicle's future trajectory based on sensor-like input data and system state estimations. This allows the controller to adjust steering angles and acceleration or braking commands dynamically, with the aim of maintaining lane position, handling curves, and navigating track conditions smoothly and efficiently. The final product simulates the real-world behaviour of an autonomous vehicle under both control strategies, enabling real-time decision-making and performance evaluation. The software not only demonstrates the feasibility of MPC for high-performance autonomous navigation but also provides a comparative benchmark against traditional PID control, with the goal of informing future applications.

1.5.1 Scope

The project will cover a detailed explanation of MPC and PID principles and how they apply to the vehicle, also this project will deliver a simulation system tailored for an autonomous vehicle, specifically designed for the autocross driving scenario. The project will focus on developing a control system that uses MPC to optimize inputs based on the vehicle's predicted trajectory, ensuring the vehicle navigates through the autocross course efficiently, and a PID control system to compare the controllers.

1.5.2 Audience

The primary target users for this project are researchers and engineers working in the field of autonomous vehicles, particularly those interested in advanced control systems such as MPC or PID. Additionally, this project will be highly relevant for the Oxford Brookes Racing Autonomous (OBRA) team, as it aims to develop an improved control system to enhance their vehicle's performance in competitions. The outcomes of the project will support OBRA's efforts to achieve greater precision and effectiveness in real time for the races.

2. Background Review

2.1 Overview of Existing Approaches

In the development of autonomous vehicles, PID controllers have been widely used due to their simplicity and effectiveness in regulating system behaviour. PID controllers adjust system output based on the current error, past accumulated error, and predicted future error rate, making them effective for many control applications. They are commonly employed in tasks like speed regulation and basic trajectory following in autonomous vehicles. However, PID controllers face limitations in handling more complex driving scenarios, especially when dealing with multiple variables or changing system dynamics [4].

In contrast, MPC represents a more advanced and dynamic approach in the field of autonomous vehicle control. Unlike PID, MPC incorporates a mathematical model of the vehicle's dynamics to predict its future states over a defined time horizon. At each control step, it solves an optimization problem that considers constraints on inputs (e.g., steering angle, acceleration) and outputs (e.g., velocity, yaw rate), selecting the optimal control actions to minimize a cost function. This enables MPC to anticipate future system behaviour and adapt more effectively to changes in the environment or vehicle state. MPC excels in complex tasks such as path planning, obstacle avoidance, and maintaining stability in curved trajectories. It offers superior performance in terms of constraint handling and trajectory optimization, making it highly suitable for real-time autonomous driving applications. However, its benefits come with higher computational costs, which can be challenging for real-time implementation in embedded systems without sufficient processing power. [17]

2.2 Related Literature

The development and comparison of PID and MPC controllers for autonomous vehicle control builds upon a solid foundation of existing research. Each of the references used in this project contributes specific insights and technical foundations that guide the implementation and evaluation of the control strategies.

The paper by Meng [1] provides a highly relevant contribution by proposing a fast iterative MPC method tailored for real-time, high-precision vehicle tracking. This source is particularly useful for understanding how to improve the computational efficiency of MPC in dynamic environments, an aspect directly aligned with the objectives of this project, especially in the context of real-time performance.

Paper [2] explore the implementation of MPC for automated vehicle steering, focusing on its constraint-handling capabilities and adaptability. This paper helps clarify the practical advantages of MPC over simpler controllers, particularly in maintaining stability and control accuracy during complex manoeuvres. It also informs part of the simulation and design approach used in the MPC component of the project.

The article by Zhang [3] introduces a data-driven approach to MPC for autonomous steering, highlighting the benefits of integrating predictive models with machine learning to adapt to real-world variability. Although this project does not employ learning-based methods, the paper inspires considerations for future extensions and underlines the importance of accurate vehicle dynamics modelling in MPC design.

Veysi [4] serve as a foundational reference that outlines both PID and MPC control methods within engineering systems. It provides a comparative overview that reinforces the theoretical differences and practical trade-offs between the two controllers, supporting the core aim of this project: to evaluate their relative effectiveness in autonomous vehicle applications.

The paper [17] offer insight into randomized MPC techniques suitable for real-time autonomous driving. Their work emphasizes computational speed and performance under uncertainty, which complements this project's investigation into real-time feasibility and provides a benchmark for the optimization tools used (e.g., CVXOPT).

Finally, the work by Gokaraju [18] provides valuable insights into the practical application of PID controllers for manoeuvring autonomous vehicles. The paper examines how each of the PID components contributes to managing the dynamic behaviour of autonomous systems under various operating conditions. This study is particularly useful for understanding the limitations of PID control, such as overshooting or poor responsiveness in complex driving scenarios.

3. Methodology

This project followed an iterative and incremental development methodology inspired by agile principles. Given the technical nature of control algorithm design and simulation-based testing, a flexible approach was necessary to accommodate experimentation, performance analysis, and continuous improvements.[5]
The overall development process was divided into multiple sprints, each focused on specific objectives such as reference trajectory generation, implementing the MPC controller, integrating the PID controller, debugging the simulation, and evaluating performance. After each sprint, results were reviewed, and improvements were planned for the next cycle.

Key elements of the methodology included:

- **Sprint Planning**: At the start of each stage, clear goals were defined, such as implementing the MPC structure, tuning PID gains, or integrating the animation system.

- **Prototyping & Testing**: Algorithms were developed incrementally. Early prototypes were tested in simulation to evaluate behaviour, allowing adjustments to be made before adding complexity.

- **Continuous Integration & Version Control**: Using GitLab, each development task was committed regularly. This ensured codebase stability and facilitated collaboration and rollback when necessary.

- **Documentation & Task Management**: Notion was used to maintain documentation, record progress, and manage tasks across iterations. It helped structure the project timeline and track deliverables.

- **Feedback Loops**: Frequent testing and analysis of vehicle performance under both MPC and PID control strategies allowed for continuous refinement of the algorithms and system parameters.

This methodology ensured that the project remained adaptable, organized, and continuously progressing toward the final goal of evaluating and comparing two distinct control strategies under a unified simulation framework.

To support a clear and structured development process, a visual methodology diagram was created using Mermaid [22], a lightweight markdown-based syntax for generating diagrams. This tool enables quick and readable representation of workflows directly from text-based input, ideal for technical documentation. The resulting diagram provides a concise, yet comprehensive view of the major phases involved in the implementation and comparison of the PID and MPC controllers.

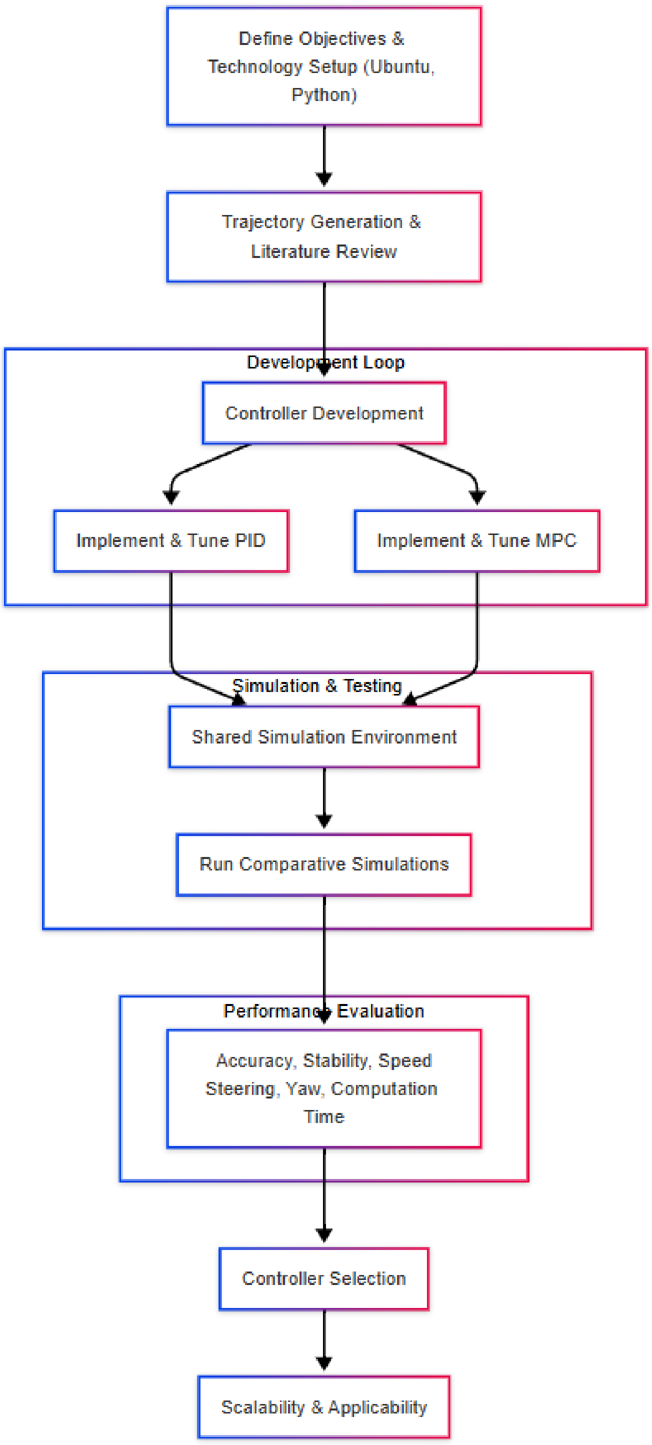

Define Objectives & Technology Setup (Ubuntu, Python)
Trajectory Generation & Literature Review
Development Loop
Controller Development
Implement & Tune PID
Implement & Tune MPC
Simulation & Testing
Shared Simulation Environment
Run Comparative Simulations
Performance Evaluation
Accuracy, Stability, Speed Steering, Yaw, Computation Time
Controller Selection
Scalability & Applicability

Diagram 1. Flow chart of the Methodology

The diagram (Diagram 1) is a visual approach that not only clarifies the sequence of tasks but also underscores the agile and modular design of the project, facilitating iterative development and data-driven decision-making throughout the process.

3.1 Approach

The research approach adopted in this project combines a thorough review of current academic literature with practical system development and testing. A comprehensive survey of scientific journals, conference papers, and technical articles, primarily sourced from databases such as Google Scholar, IEEE Xplore, and arXiv, has been conducted to establish a strong theoretical foundation. This literature review ensures that the design and implementation of the MPC based steering control system are informed by state-of-the-art methodologies in autonomous vehicle control, vehicle dynamics, and real-time optimization. By grounding the project in validated research and proven frameworks, the approach aims to ensure robustness, relevance, and alignment with best practices in the field. Additionally, this theoretical foundation supports a comparative analysis with traditional PID controllers, facilitating a deeper understanding of performance trade-offs in practical scenarios.

3.2 Technology

To develop, test, and analyse the MPC and the PID control systems, a variety of hardware and software tools were employed to support implementation, collaboration, and presentation.

- **Ubuntu 22.04**: The project collaborates with ROS2 (Robot Operating System) on Ubuntu to facilitate communication between control algorithms and the vehicle model within the simulation environment. This setup allows for a realistic and modular simulation of autonomous driving systems.

- **Python**: Both the MPC and PID control algorithms were implemented in Python, a versatile programming language well-suited for scientific computing. Python was used for numerical calculations, control logic, simulation of vehicle dynamics, and data analysis.

- **Visual Studio Code**: As the primary integrated development environment (IDE), VS Code enabled efficient coding, debugging, and version control through Git integration. It provided a reliable and customizable workspace to handle the various components of the control systems.

- **GitLab**: GitLab was employed for version management and collaborative development. Through branches, commits, and issue tracking, it ensured traceable progress and safeguarded the integrity of the codebase during iterative testing and enhancement of the control strategies.

- **Notion**: This tool was used extensively for project planning, research documentation, and task management. Notion allowed the team to maintain a clear overview of milestones, literature findings, and implementation notes, supporting agile project coordination.
- **Google Drive**: Google Drive is being used to store and share the demonstration video of the project, ensuring accessibility for assessment and review purposes. It serves as a reliable cloud platform for managing large multimedia files generated during the development and testing phases.
- **Microsoft PowerPoint**: PowerPoint was used to create the final presentation and the accompanying poster, which will be used to communicate the project findings in a professional and visually structured format.

Together, these tools provided a comprehensive technology stack that enabled robust development, simulation, collaboration, and presentation of the project.

3.3 Version management

To ensure efficient collaboration and maintain a clear development history, Git was used as the version control system throughout the project. All the code was managed using Git commands locally and synced with a remote repository hosted on GitLab. This approach provided a reliable and organized way to track changes, implement new features, and revert to previous versions when needed.
The full source code is publicly accessible at the following GitLab repository:
🔗 https://gitlab.com/mpc_pid_controller/mpc

Relevant code excerpts are included in the report where necessary, and the repository includes all implementation files, such as the PID and MPC controllers, the support files car, the requirements needed for the execution of the code and a readme file with all the information of the controllers and the code.

3.4 Development of the algorithm

This section outlines the implementation of the two control strategies central to this project, the PID controller and the MPC algorithm.

PID Controller

The PID controller is a classical feedback control mechanism that has been widely used due to its simplicity and effectiveness in many engineering applications. In the context of this project, the PID controller is used to regulate two vehicle variables: yaw angle (orientation) and longitudinal velocity, each governed by its own PID formulation.

The general PID control law in continuous time is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

Image 1. PID control law [20]

Where:

- u(t): Control signal (output of the PID controller)
- e(t): Error signal (difference between reference and actual value at time t)
- T: sampling time (our Ts variable)
- Kp: Proportional gain
- Ki: Integral gain
- Kd: Derivative gain

Each term plays a specific role:

Proportional (P):

$$u(t) = Ke(t)$$

Image 2. Proportional Control [19]

Reacts proportionally to the current error. The larger the error, the stronger the control response.

Integral (I):

$$u(t) = \frac{K}{T_i} \int_0^t e(t) dt$$

Image 3. Integral Control [19]

Accumulates past errors over time, helping to eliminate steady-state error.

Derivative (D):

$$u(t) = KT_d e(t)$$

Image 4. Derivative Control [19]

Predicts future error based on its rate of change, improving stability and reducing overshoot.

In digital systems, PID control is implemented using discrete time approximation, this equation was discretized and tuned iteratively to provide real-time corrections to the steering angle and acceleration inputs.

<u>Model Predictive Control (MPC)</u>

MPC is a more advanced, model-based control strategy that uses the current system state and a predictive model to compute a sequence of optimal control actions. It minimizes a cost function over a future time horizon while taking constraints into account.
At each control step, MPC solves an optimization problem of the form:

$$J = \sum_{i=1}^{Hp} e^T Q_e e + \sum_{i=0}^{Hc-1} \Delta u^T Q_u \Delta u$$

Image 5. MPC cost function [21]

Where:

- e: is the predicted state error at time step
- Δu: is the change in control input between steps
- Qe, Qu: are weighting matrices
- Hp: is the prediction horizon
- Hc: is the control horizon

This formulation enables MPC to not only follow the desired trajectory with precision but also ensure smoother control inputs by penalizing sharp changes in acceleration or steering.
The optimization problem was solved using **CVXOPT**, a convex optimization solver that computes the optimal sequence of control actions at each time step. The first control input of the optimized sequence is applied to the vehicle, and the process repeats in a receding horizon fashion.

The PID controller offers a straightforward, low-computation solution but lacks predictive ability and constraint handling. On the other hand, MPC, though computationally heavier, enables optimal and robust control in dynamic environments, making it ideal for autonomous driving applications where accuracy and adaptability are crucial.

4. Results

This section presents a comprehensive evaluation of the performance of two control strategies, the PID and the MPC applied to autonomous vehicle steering and velocity control. The primary objective is to analyse, interpret, and compare the results generated by each controller within a shared simulation environment based on the same physical vehicle model.

The implemented simulation framework allows for real-time tracking of the vehicle's position, velocity, orientation, and control inputs (steering and acceleration). It generates animated visualizations as well as quantitative plots illustrating a lot of parameters. By doing so, the results provide both qualitative and quantitative insights into the behaviour of each controller under identical conditions.

Key performance metrics analysed in this section include tracking accuracy, steering responsiveness and system stability. Moreover, the computational efficiency of each approach is also measured and discussed, highlighting the trade-offs between control performance and processing time. The results are interpreted not only in terms of control system theory but also in the context of practical application, particularly the potential integration of MPC into autonomous platforms like OBRA.

This comparative study ultimately aims to assess which controller is more effective and robust for autonomous driving tasks, especially when dealing with constraints, real-time requirements, and complex trajectories. Limitations encountered during implementation are also discussed, as well as challenges related to parameter tuning, code structure, and execution consistency.

4.1 Visualization and Behaviour Analysis

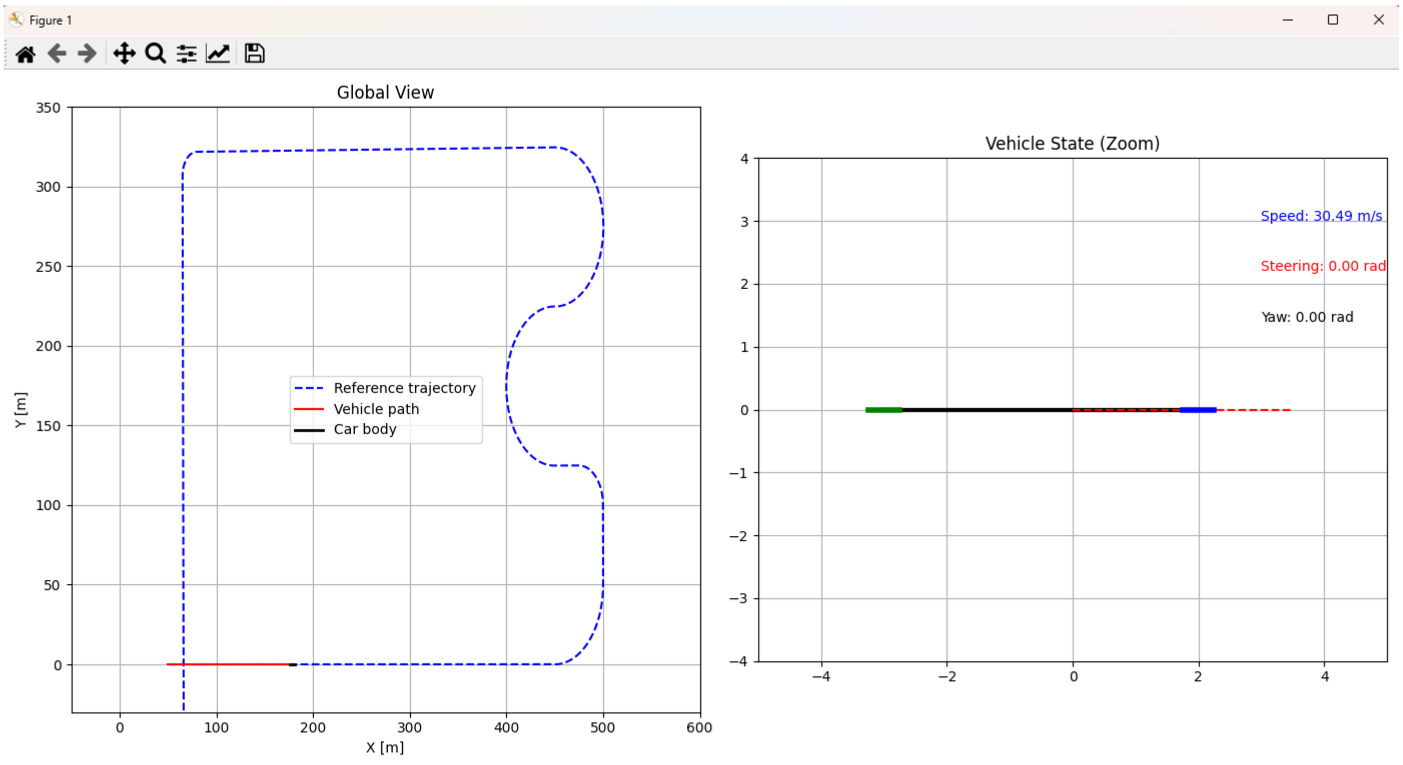


Figure 1. Animate vehicle

To enable a clear and comprehensive comparison between the two control strategies, a shared animated visualization was implemented. This animation setup provides both a global view of the vehicle trajectory within the track layout and a zoomed-in vehicle view for precise observation of vehicle dynamics.

On the left side, the Global View displays the predefined reference trajectory (blue dashed line), which the vehicle attempts to follow. The red line represents the actual path taken by the vehicle, while the black segment indicates the car body orientation. These visual enables observers to assess how well each controller tracks the path, especially around sharp curves and transitions.

On the right side, the Vehicle State (Zoom) panel offers a magnified view of the car, allowing detailed inspection of the control variables in real time. This includes the vehicle's:

- Speed (displayed in blue), measured in meters per second,
- Steering angle (in red), given in radians,
- Yaw angle (in black), representing the car's orientation with respect to the global axis.

This split visualization provides critical insights into the performance of each controller. While the global view captures high-level accuracy and lap coverage, the zoomed view allows evaluation of local stability, oscillations, and responsiveness in steering and heading behaviour. The use of consistent visual layouts for both controllers ensures a fair and uniform basis for comparison.

4.2 Trajectory Tracking Performance Analysis: MPC vs PID

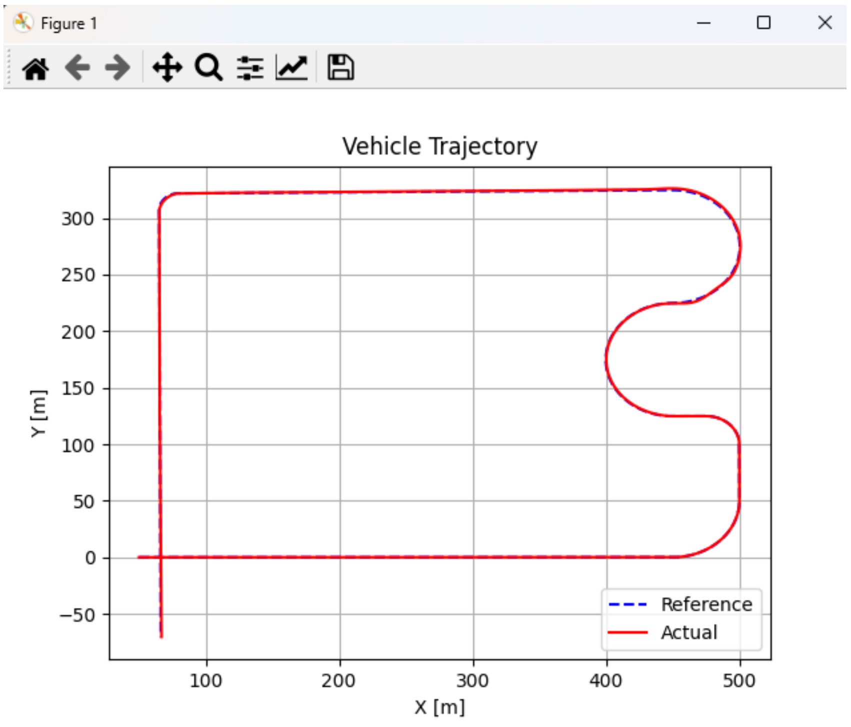


Figure 2. MPC Vehicle trajectory vs Reference

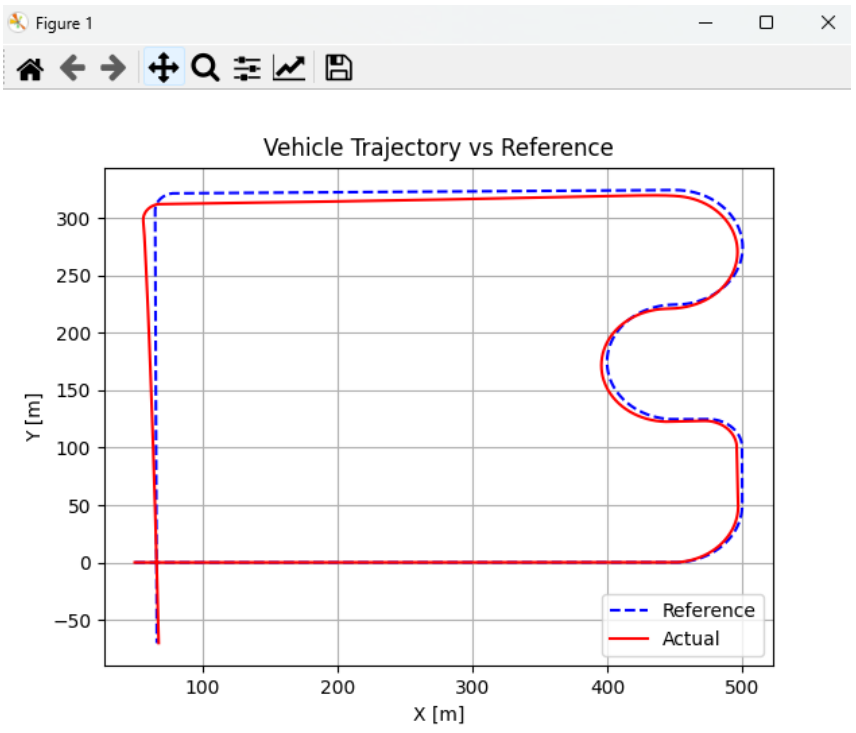


Figure 3. PID Vehicle trajectory vs Reference

School of Engineering, Computing & Mathematics

The figure 2, representing the trajectory tracking using the MPC controller, shows an excellent alignment between the reference trajectory (blue dashed line) and the actual vehicle path (solid red line). The vehicle closely follows the complex curvature of the track, including sharp turns and varying radii, with minimal lateral deviation. This demonstrates the MPC's capability to predict and adjust control actions optimally over a prediction horizon, considering both the current state and future constraints.
In contrast, the figure 3 displays the performance of the PID controller. While the overall shape of the trajectory is followed, it is immediately evident that the tracking error increases significantly during curved segments. The red line consistently lags behind the reference, especially in sharper turns. This behaviour indicates the limited adaptability of the PID controller when confronted with dynamic environments or non-linear system behaviours. Because PID reacts to current and past errors without anticipating future ones, it struggles with rapid or complex transitions.

Additionally, while both controllers seem to perform adequately on straight segments, the MPC excels in maintaining path fidelity throughout the entire circuit, whereas the PID exhibits undershooting and overshooting behaviours due to its reactive nature.

4.3 Analysis of Steering and Acceleration Control Inputs: MPC vs PID

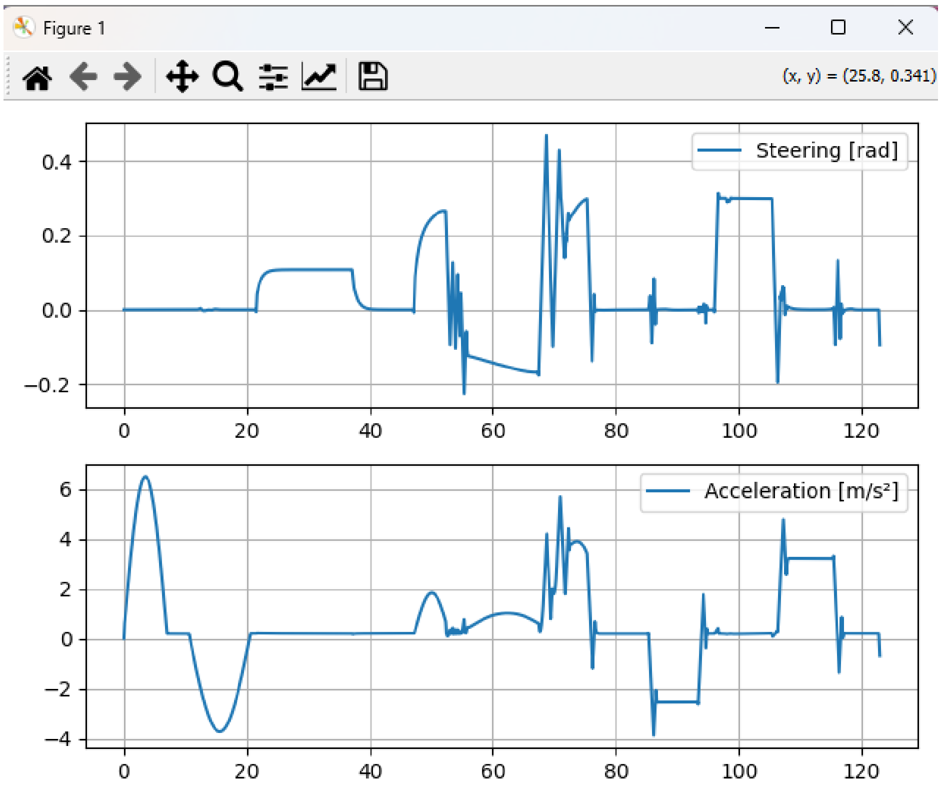


Figure 4. Steering and Acceleration Control Inputs of the MPC

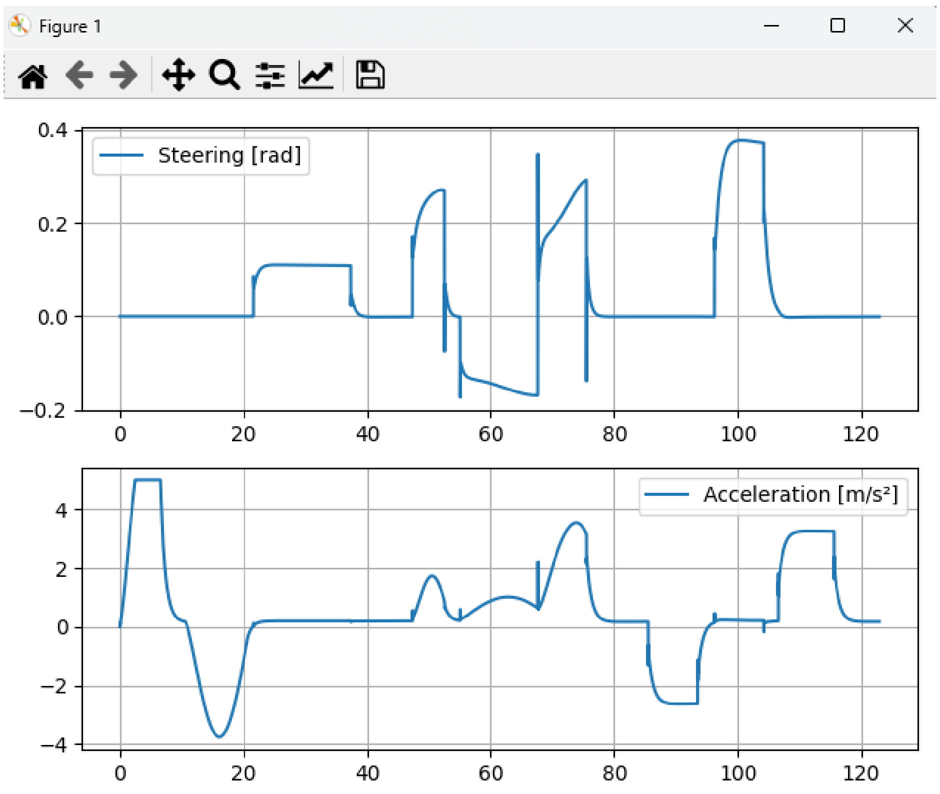


Figure 5. Steering and Acceleration Control Inputs of the PID

The two figures presented illustrate the control inputs, the steering angle (radians) and the acceleration (m/s2), applied over the duration of the simulation for both the MPC and PID controllers.

In the Figure 4 (MPC), we observe a more frequent and abrupt adjustment of both steering and acceleration. Particularly, between time indices 50 and 100, the steering input shows sharp peaks and fluctuations that indicate highly dynamic correction behaviour. This reflects the MPC's inherent nature of anticipating future trajectory deviations and applying corrections proactively, even if it means making more aggressive changes. Similarly, acceleration varies significantly, with noticeable bursts both in positive (acceleration) and negative (braking) directions. This shows MPC’s effort to optimize not just the path but also the vehicle’s velocity profile, reacting precisely to turns and straight segments.

On the other hand, in the Figure 5 (PID), the controller demonstrates a smoother and more predictable pattern in both steering and acceleration. Steering inputs are less frequent and exhibit a more gradual curve, suggesting that the PID controller corrects deviations reactively and slowly. The acceleration pattern follows a similar trend, with changes that are less intense and more uniform, likely contributing to smoother driving but reduced responsiveness to sudden trajectory changes.

This contrast aligns with the core hypothesis of the project, that while PID may be suitable for simpler environments or low-speed tasks, MPC stands out in complex, high-demanding tasks requiring constraint handling, prediction, and optimal control.

4.4 Comparison of Positional and Yaw Accuracy: MPC vs PID

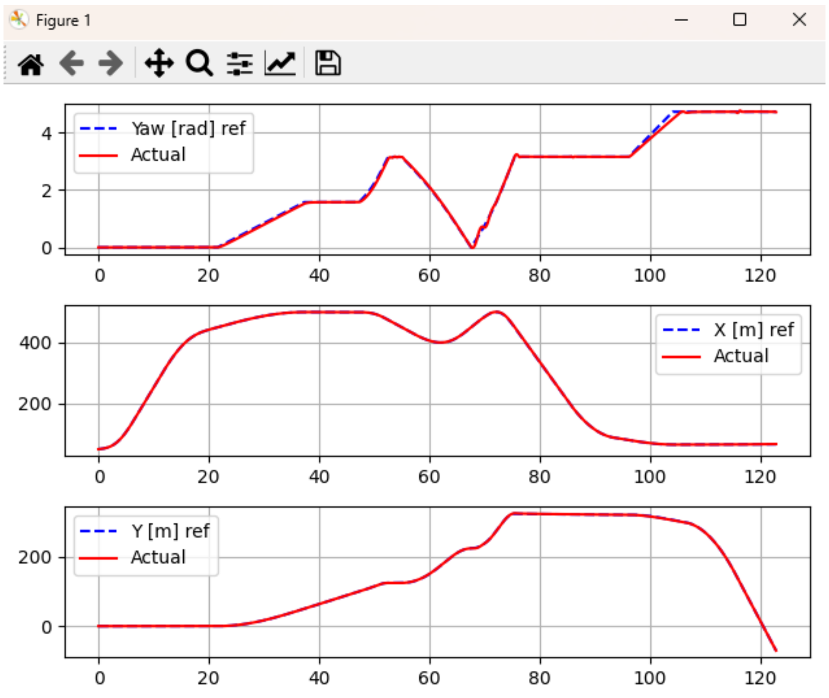


Figure 6. Yaw angle, X position, and Y position of the vehicle. (MPC)

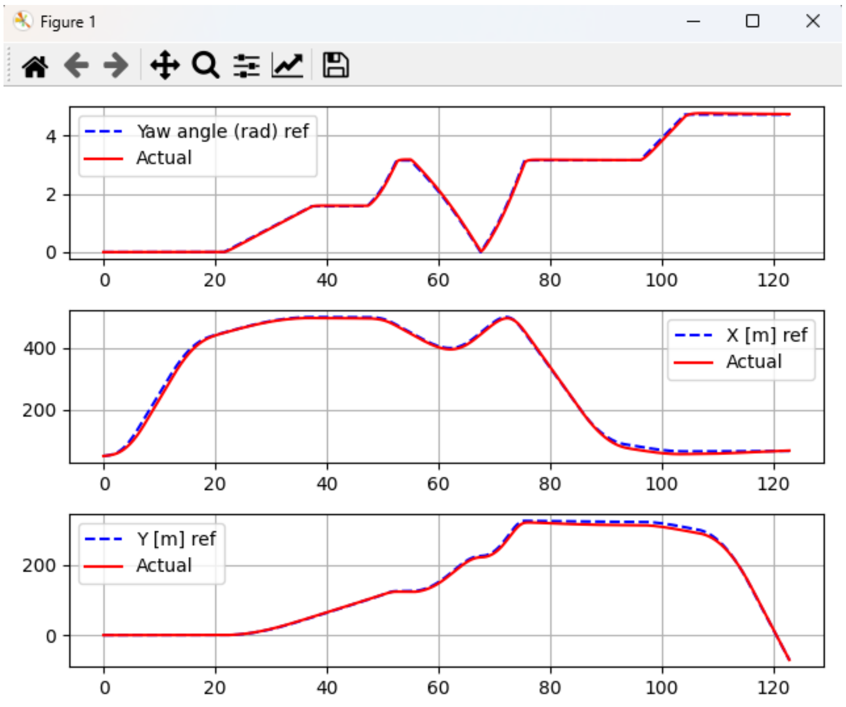


Figure 7. Yaw angle, X position, and Y position of the vehicle. (PID)

These figures display three fundamental metrics used to evaluate the control performance of both algorithms: yaw angle, X position, and Y position of the vehicle. What follows is a comparative analysis.

Yaw Angle (rad) Tracking

Both graphs compare the reference yaw angle (dashed blue line) with the actual yaw angle (solid red line).

- In the MPC, yaw tracking is remarkably precise, showing near-perfect alignment across the entire course, even during sharp turns and directional changes. The controller's ability to anticipate future behaviour allows it to maintain stability without abrupt corrections.
- The PID controller, on the other hand, shows slightly larger deviations, particularly when quick changes in direction occur. While it still follows the reference acceptably, there are visible errors, which can lead to rougher or delayed steering behaviour.

X Position Tracking

- In the MPC, the vehicle's X-position nearly overlaps the reference path throughout the simulation. This showcases MPC's ability to predict and proactively correct its trajectory.
- In contrast, the PID shows a minor lag in response, particularly noticeable around the second peak (around second 60), which is a consequence of its reactive nature, correcting only after an error has occurred.

Y Position Tracking

- The MPC again demonstrates exceptional lateral stability, closely following the reference Y-trajectory even through the circuit's most complex curves.
- The PID struggles slightly more here, showing small oscillations and deviations, especially in the more technical sections. These lateral inconsistencies suggest reduced precision in trajectory handling compared to the MPC.

This graph reinforces the superiority of MPC over PID in both orientation and positional accuracy. The MPC consistently shows better alignment with reference trajectories, both in yaw and spatial coordinates, delivering smoother and more controlled movements. These characteristics are essential for safe and efficient autonomous driving, especially under demanding conditions. While the PID controller can perform adequately in simpler scenarios, it becomes clear that MPC is the more robust and a professional solution.

4.5 Velocity and Angular Rate Analysis: MPC vs PID

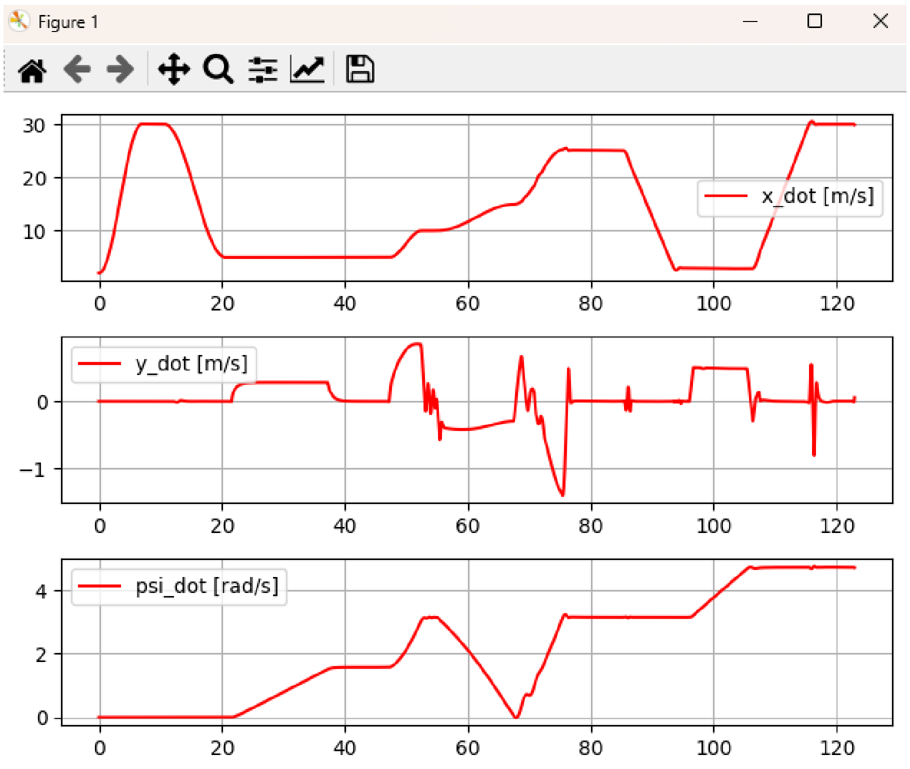


Figure 8. Velocity and Angular Rate (MPC)

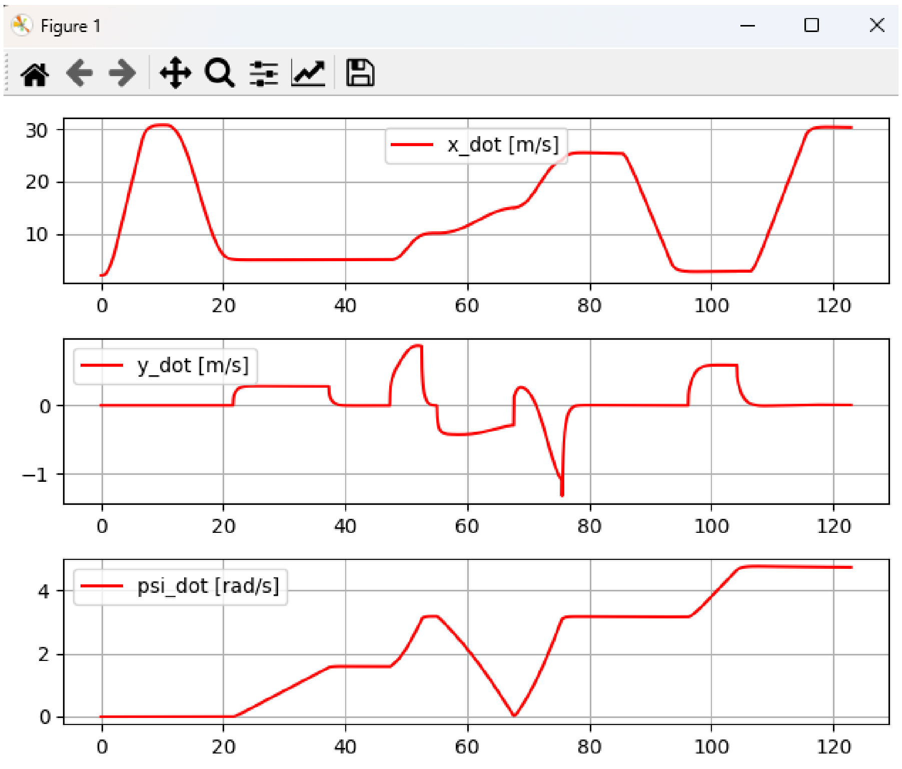


Figure 9. Velocity and Angular Rate (PID)

In both controllers, the longitudinal velocity (x_dot) follows a similar pattern: a rapid increase at the start, followed by periods of steady cruising and several deceleration phases in curved or demanding track sections. However, the MPC controller exhibits a slightly more refined modulation of x_dot, with smoother transitions and slightly better maintenance of higher speed during long straight sections.

The lateral velocity (y_dot) shows key differences. In the MPC controller, there are visible oscillations during sharp turns (e.g., around the mid-simulation mark), but they are more quickly damped compared to the PID controller. In contrast, the PID shows broader oscillations with slower convergence, indicating more lateral instability or delay in correction, especially when exiting curves.

The yaw rate (psi_dot) is very similar between both controllers, reflecting that both adjust the heading at comparable rates. However, again, the MPC exhibits slightly smoother transitions with fewer high-frequency variations, reinforcing its predictive and constraint-aware design that prevents excessive control action.

Overall, while both controllers maintain acceptable vehicle behaviour, the MPC's results are more stable, especially in lateral velocity control. It is better at smoothing transitions in all three velocity-related components, reducing unnecessary oscillations and contributing to a more comfortable and precise drive. This is particularly important for applications like autonomous racing, where every fraction of stability translates to enhanced lap times and safety.

4.6 Acceleration and Stability Analysis: MPC vs PID

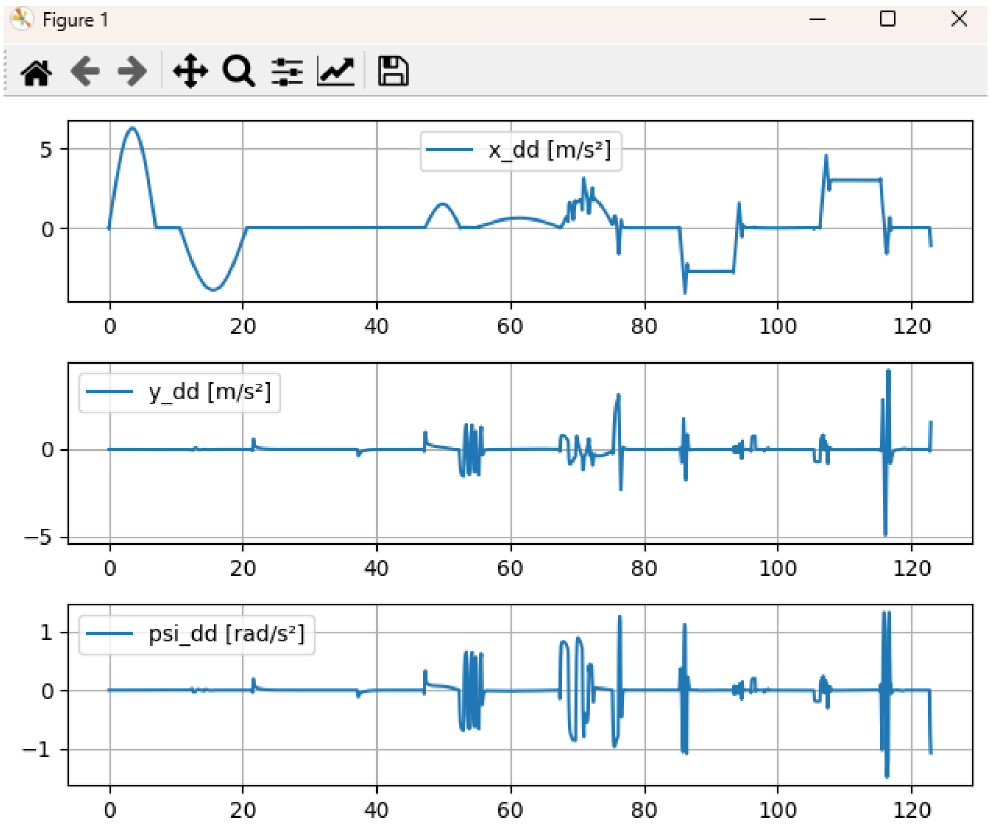


Figure 10. Acceleration and Stability (MPC)

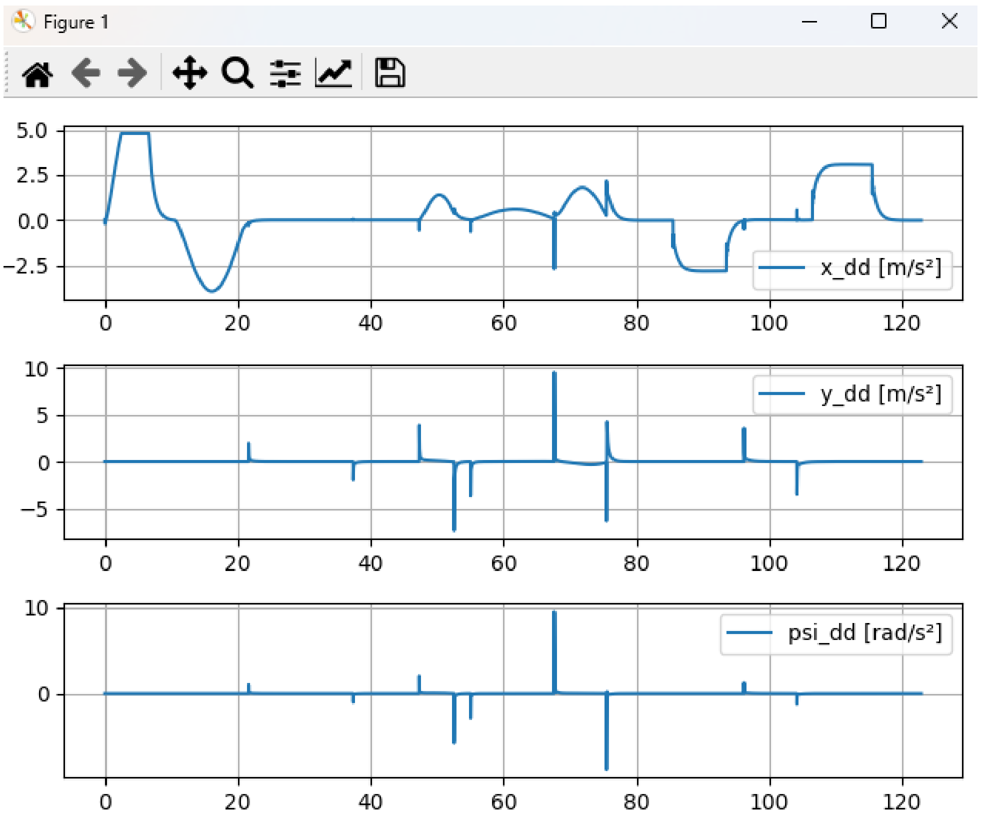


Figure 11. Acceleration and Stability (PID)

The figure 10, presents three subplots that represent the second derivatives of key dynamic states of the vehicle over time:

x_dd [m/s2] - Longitudinal Acceleration

The top plot shows how the vehicle accelerates and decelerates in the direction of travel. The MPC controller manages to maintain a generally smooth profile with a strong acceleration phase at the start (up to ~6 m/s2), followed by fine-tuned adjustments throughout the rest of the trajectory. These small corrections are a result of the controller anticipating upcoming changes in the reference path and adjusting the vehicle's speed accordingly.

y_dd [m/s2] - Lateral Acceleration

In the second subplot, the lateral accelerations remain close to zero for most of the trajectory, only showing activity during curves and steering changes. The graph reveals high-frequency, low-amplitude oscillations when cornering, which are expected in precision control scenarios. These oscillations represent the vehicle's attempts to stay aligned with the reference trajectory, especially in tighter sections of the circuit.

psi_dd [rad/s2] - Yaw Angular Acceleration

The third plot provides insight into how quickly the vehicle adjusts its heading. The MPC controller displays frequent but bounded yaw corrections, with spikes around +/- 1 rad/s2. These are smooth and well-distributed, indicating that the controller is actively minimizing yaw error without inducing instability. The consistency of these corrections shows that MPC anticipates heading requirements early enough to avoid aggressive steering changes.

The figure 11, also displays the second derivatives of the vehicle's motion variables, giving insights into how the PID controller handles acceleration:

x_dd [m/s2] - Longitudinal Acceleration

The longitudinal acceleration pattern in the PID controlled vehicle is much more abrupt and irregular compared to MPC. The spikes are more sudden and less predictable, especially visible at the beginning and near the end of the lap. This irregularity indicates that the PID controller is reacting to immediate errors rather than anticipating future states, which can lead to jerkier motion and potentially more wear on mechanical components.

y_dd [m/s2] - Lateral Acceleration

The lateral acceleration profile shows large, sparse spikes, indicating that the controller is making aggressive lateral corrections. These peaks likely coincide with sharp turns or sudden trajectory deviations, pointing to reduced smoothness in handling. Unlike MPC, which displayed tightly clustered micro-adjustments, the PID controller seems to apply discrete, coarse steering actions.

psi_dd [rad/s2] - Yaw Angular Acceleration

The yaw acceleration in PID is characterized by a series of short-lived, sharp spikes followed by flat zones. This again suggests a lack of predictive smoothing, where the controller allows significant angular deviations to build up before responding strongly to correct them. The result can be more oscillatory or delayed steering adjustments, which negatively affect path stability.

When comparing the acceleration behaviour of MPC vs. PID, the MPC controller clearly exhibits superior smoothness, consistency, and anticipation. Its acceleration and yaw corrections are frequent but minor, indicating a continuous forward-looking adjustment strategy. This contributes to a stable and comfortable vehicle trajectory, especially important in real-world applications like autonomous driving.
In contrast, the PID controller demonstrates a reactive approach. The plots show sudden, high-magnitude accelerations and jerky yaw corrections, which reflect its difficulty in handling complex trajectories and dynamic constraints. These characteristics can cause oscillations, overshooting, and delayed responsiveness.
Overall, the MPC controller outperforms PID in terms of control quality, trajectory precision, and system stability, reinforcing its suitability for advanced autonomous vehicle applications.

4.7 Conclusion of the analysis

Here's a comparative summary table that highlights the key performance differences between the MPC and PID controllers based on the graphs and results.

Criteria	MPC Controller	PID Controller	Observation
Trajectory Accuracy	Closely follows the reference trajectory with minimal deviation	Deviates more noticeably, especially in curves	MPC demonstrates higher precision in path tracking
Steering Response	Smooth and predictive steering adjustments	More abrupt and reactive steering actions	MPC offers better vehicle stability and control smoothness
Acceleration Control (x_dd)	Gradual changes in longitudinal acceleration	Sudden and less consistent acceleration patterns	MPC ensures a more comfortable ride and less mechanical stress
Yaw Stability	Predictive and consistent yaw control across time	Fluctuating yaw corrections with lag	MPC provides superior rotational stability and vehicle orientation control
Computation Complexity	High	Low	PID is simpler to implement
Responsiveness	Proactively adjusts based on future predicted states	Reacts only to current and past errors	MPC better anticipates changes in trajectory and environment
System Smoothness	High smoothness in all control variables (steering, acceleration, yaw)	Noticeable jerk and discontinuities in motion variables	MPC is more suited for real-world autonomous driving
Handling of Constraints	Integrates physical limits during prediction	Cannot inherently handle constraints	MPC allows safer operation by staying within system limitations
Scalability for Real Use	Easily expandable to more complex models and real time implementations	Limited scalability beyond simple control problems	MPC is more robust for future deployment

Table 1. Comparison Table: MPC vs. PID Performance

This table clearly reflects that MPC not only outperforms PID in nearly all performance related aspects but also lays a stronger foundation for scalable and intelligent autonomous vehicle control, especially in applications demanding precision, adaptability, and stability.

5. Deliverables

5.1 Project Management

Effective project management was crucial for structuring the development of the MPC and the PID Control System. A detailed project schedule was designed at the start to allocate time efficiently across the different stages of the project, from research and implementation to testing and documentation.

The timeline for these activities is illustrated in the Gantt chart below:

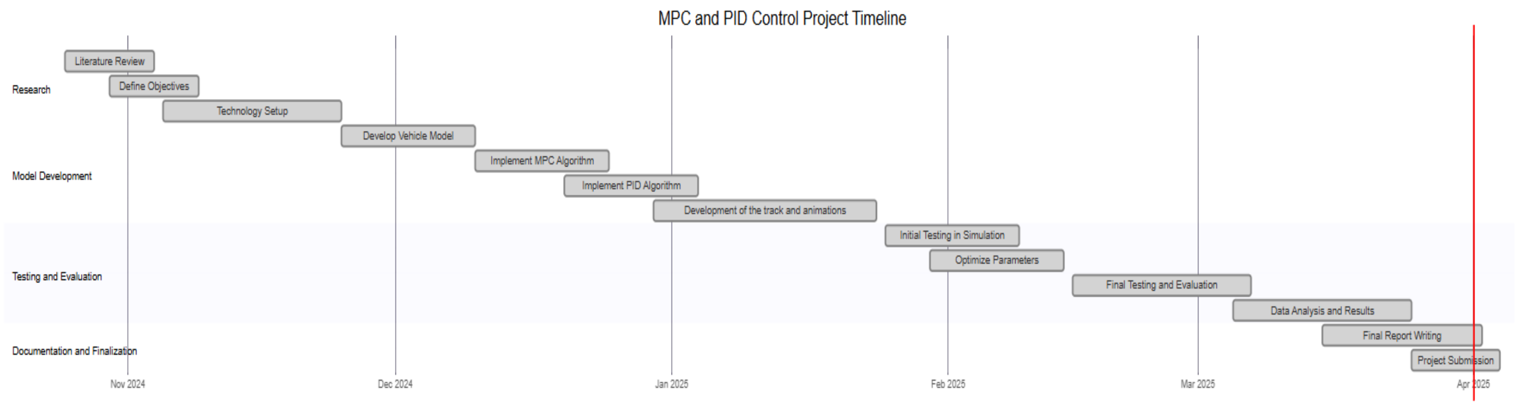


Figure 12. Gantt chart of the Project Timeline

This visual schedule helped manage dependencies between tasks and ensured that testing and optimization were completed ahead of documentation. It also allowed flexibility for unforeseen adjustments, especially during the testing and debugging of the algorithms.
Agile principles were applied to allow iterative development and integration, especially when switching between control strategies and tuning parameters during the simulation stage.

5.2 What Outputs You Will Deliver

This project will deliver a comprehensive set of outputs that demonstrate the successful implementation, evaluation, and comparison of two vehicle control strategies the MPC and the PID within an autonomous driving context. These deliverables are designed to provide both theoretical insight and practical value for future academic and real-world applications.

Dual-Control Simulation System

The core output is a complete software environment developed in Python that simulates the behaviour of an autonomous vehicle under the influence of both MPC and PID control systems. This includes:

- A shared physical vehicle model that is used consistently across both controllers, ensuring fairness in comparative analysis.
- A PID controller capable of managing vehicle yaw (steering angle) and longitudinal speed.
- An MPC controller implemented with CVXOPT for real-time trajectory optimization, including dynamic constraint handling and predictive adjustment.

Real-Time Animated Visualizations

To clearly demonstrate the differences in performance, efficiency, and behaviour between the two control systems, animated simulations have been developed using Matplotlib. These visualizations include:

- A global view of the vehicle navigating a track under each control scheme.
- A zoomed view to observe steering behaviour and body orientation.
- Dynamic feedback of speed, yaw angle, and steering input displayed as overlays in the animation.

Performance Metrics and Evaluation Results

The project produces comparative graphs and performance metrics including:

- Trajectory tracking accuracy (how closely the vehicle follows the reference path).
- Controller responsiveness and stability during lane keeping and turns.
- Execution/computation time, benchmarking the real-time feasibility of each controller.
- Acceleration and velocity profiles, for evaluating smoothness and physical realism of the drive.

Analytical Report and Technical Documentation

A detailed final report is delivered as a structured academic document covering:

- The background, motivation, and problem statement.
- Design rationale for both controllers.
- Implementation methodology.
- Experimental setup and performance evaluation.
- Risks, limitations, and scalability considerations.
- Ethical, legal, and environmental implications.
- Conclusions and recommendations for future work

Source Code Repository

A public GitLab repository has been maintained throughout the project for version control and collaboration. The repository contains:

- Full source code for all control logic and simulation scripts.
- A README.md file detailing setup instructions, dependencies, and usage guidelines.
- A requirements.txt for Python environment setup using venv.

GitLab Repository Link: https://gitlab.com/mpc_pid_controller/mpc

Video Demonstration and Presentation Materials

To support communication and dissemination of the work:

- A screen-recorded video demonstration showcases the running simulations and highlights the key differences between the controllers. Google drive link: https://drive.google.com/file/d/1vSvt96XEjO6DG1ZAwPiL-J2xPlodLKcc/view?usp=drive_link
- A PowerPoint presentation summarizes the methodology, findings, and future applications.
- A poster design is provided for academic exhibitions and assessment panels.

- All multimedia content is stored and shared via Google Drive to ensure accessible distribution during the final evaluation.

5.3 Risk

5.3.1 MPC Risk

Computational Complexity

One of the significant limitations of MPC is its high computational demand. MPC requires solving optimization problems in real-time, which can be computationally intensive, particularly for embedded systems with limited resources. As a result, it may struggle with real-time performance under high-speed conditions or complex environments unless powerful processors or specialized hardware (like FPGA or SoC) are employed [6].

High Power Consumption

Implementing MPC on hardware like Field Programmable Gate Array (FPGA) can lead to high power consumption, particularly in systems that require continuous real-time optimization. Even though MPC provides great accuracy and control performance, this trade-off can limit its use in battery-powered or resource-constrained systems [6].

Parameter calibration

Vehicle dynamics models used in MPC require extensive parameter calibration through experiments, which increases the time and resources required for system setup. This is a major limitation for large-scale or flexible deployments [7].

5.3.2 PID Risk

Single Input Single Output (SISO)

One major limitation is that PID is inherently a Single Input Single Output (SISO) controller, which restricts its ability to handle multivariable systems. This becomes a critical drawback in autonomous driving where control tasks (e.g., speed, yaw, steering, and trajectory) are strongly coupled and interdependent, requiring a coordinated control strategy that is something that PID cannot offer natively.[18]

Limitations in Anticipating Future States and Managing Constraints

PID does not consider future states or constraints. It is purely reactive, it corrects current errors based on historical and instantaneous deviations without anticipating future trajectory deviations or limitations in control inputs (e.g., max acceleration or steering angle). This can lead to instability in dynamic or fast-changing environments such as sharp curves, obstacle avoidance, or high-speed lane changes, especially under real-world uncertainties and sensor noise.[18]

5.3.3 Implementation Challenges

During development, several technical issues were encountered, particularly in integrating the MPC and PID controllers into the same simulation environment. Visualization problems arose with the animation system (such as disappearing vehicles or incorrect rendering), which required debugging of matplotlib and animation timing. There were also mismatches between the actual speed and what was plotted due to reference misalignment, which took considerable time to resolve.
Additionally, inconsistencies with lap time detection for the PID controller revealed the need for more robust logic in position tracking and state resetting.

Future Risk

- **Scalability**: The current simulation is tailored for a single vehicle. Extending it to multi-agent or real-world deployment scenarios would introduce complexities in coordination, sensing, and communication.
- **Code Maintainability**: Without modularization and unit testing, future updates or extensions to the codebase may introduce bugs or compatibility issues.
- **Real-Time Constraints**: Porting the simulation to embedded platforms or real-time systems would require significant optimization and potential algorithm simplification, especially for MPC.
- **Data Integrity**: Incorrect trajectory references or misaligned sampling rates could introduce instability in real applications if not detected early.

5.4 Professionalism

Legal, Social, Ethical and Environmental Considerations

In the development of autonomous systems and control algorithms such as MPC and PID for simulated vehicle dynamics, several professional responsibilities must be considered. These responsibilities align with recognized professional codes of conduct, such as those from the British Computer Society (BCS) and the Association for Computing Machinery (ACM). Both emphasize a duty to act with integrity, prioritize public interest, and consider the broader impact of technological work.[8][9]

Legal Considerations

From a legal standpoint, the project primarily handles simulated data. However, if the system were to be deployed in real vehicles or shared across organizations, it would be subject to various data protection regulations and intellectual property rights. For example, if any real vehicle telemetry or location data were used, GDPR compliance in the EU or Data Protection Act 2018 in the UK would become mandatory.[11][12]

Furthermore, the software uses open-source libraries such as NumPy, Matplotlib, and Python-based solvers. It is essential to respect the license agreements of these tools to avoid legal infringement. Any contribution or public release must include proper attribution and compliance with licenses such as MIT, BSD, or GPL.[13][14][15]

Social and Ethical Considerations

Autonomous vehicle technologies raise various ethical dilemmas, especially in real-world scenarios. While this project remains in a simulation environment, future adaptations or implementations could influence decision-making processes that affect human safety.

- **Transparency and Accountability**: In line with the ACM Code of Ethics, developers must ensure that their algorithms are understandable and explainable, avoiding black-box behaviour where possible.
- **Safety and Reliability**: According to BCS guidelines, a professional must "have due regard for public health, privacy, security and well-being." Ensuring that the control system is robust, predictable, and thoroughly tested is a moral obligation, not just a technical requirement.

- **Bias and Fairness**: Even in control systems, unintended bias may arise if the model is overfitted to a specific environment. Ethical engineering requires evaluating whether the controller would behave equally well across diverse operational contexts.

Environmental Considerations

While the current project is simulated and runs on standard computing hardware, environmental impact becomes relevant when control systems are deployed at scale in vehicles. MPC is computationally intensive, which implies greater power consumption. In embedded systems or real-time vehicle controllers, this could lead to increased battery drain or carbon footprint if not optimized.

Furthermore, by aiming to improve vehicle efficiency (e.g., smoother trajectories, optimized fuel usage through intelligent control), this work indirectly supports sustainable transport systems—a positive environmental contribution.

Professional Codes of Conduct

This project has been developed in line with professional standards outlined in:

- **BCS Code of Conduct**: Emphasizes integrity, competence, and public interest. The use of version control (GitLab), systematic testing, and open documentation supports transparency and collaboration.
- **ACM Code of Ethics**: Encourages the avoidance of harm, being honest and trustworthy, and respecting privacy. The modular and transparent structure of the codebase reflects adherence to these principles.[16]
- **IEEE Code of Ethics**: Stresses ethical design, safety considerations, and the responsible use of technology.[10]

These standards provide guidance not only for software engineering practice but also for responsible innovation and critical self-reflection.

6. Conclusion

This project set out to design, implement, and evaluate two prominent control strategies the MPC and the PID for autonomous vehicle trajectory tracking. By building a shared simulation environment, both controllers were applied to the same vehicle model and assessed under identical conditions, allowing a fair and rigorous comparative analysis across multiple performance dimensions.

The implementation process revealed important distinctions in the operational principles and outcomes of each control approach. The PID controller, while relatively simple and computationally efficient, showed clear limitations in responsiveness, trajectory accuracy, and the ability to handle dynamic environments. Its reactive nature, lack of future-state prediction, and inability to manage system constraints led to larger tracking errors, overshooting in curves, and less smooth acceleration profiles. These results reinforce the notion that PID is suitable primarily for low-speed or linear systems where control objectives are straightforward and constraints minimal.

In contrast, the MPC controller demonstrated superior performance across nearly all tested metrics. It offered significantly better path tracking, proactive and constraint-aware steering and acceleration adjustments, and smoother control behaviour. The ability of MPC to optimize control actions over a future prediction horizon allowed it to maintain stability, minimize deviation, and adapt to rapid trajectory changes effectively. The use of the CVXOPT solver enabled real-time optimization, proving that with appropriate tuning, MPC can be deployed efficiently in simulation environments with high responsiveness.

Animated visualizations further illustrated these differences. The MPC-controlled vehicle consistently followed the reference trajectory with minimal deviation, even through complex curves and transitions, while the PID vehicle displayed visible lag and instability. Metrics such as yaw tracking, lateral velocity, and acceleration derivatives supported the observation that MPC provides a smoother, more professional grade driving experience, aligning better with real-world autonomous vehicle demands.

Additionally, this project emphasized the importance of visualization tools and computation benchmarking as part of control strategy evaluation. The use of Matplotlib animations allowed not only technical validation but also intuitive understanding of each controller's behaviour.

In terms of contribution, this work delivers a scalable, well-documented simulation framework that can serve both as a learning platform and a prototype for more advanced autonomous driving systems. The results strongly support the adoption of MPC in applications like OBRA, where high performance, constraint handling, and precision are essential.

Looking ahead, several avenues for future development are identified:

- Integration with sensor data to test the control strategies in more realistic, noisy environments.
- Extension to full 3D vehicle models to account for pitch, roll, and suspension dynamics.
- Real-time deployment on embedded hardware to test MPC's efficiency and latency outside simulation.
- Multi-agent coordination for autonomous vehicle platooning or traffic management scenarios.

In conclusion, this project not only validates the technical superiority of MPC over PID in complex driving conditions but also establishes a foundational framework for future research, testing, and deployment of advanced control strategies in autonomous vehicle systems.

7. Bibliography

[1] Meng, D., Chu, H., Tian, M., Gao, B. y Chen, H. (2024) *Real-time high-precision nonlinear tracking control of autonomous vehicles using fast iterative model predictive control*. IEEE Xplore. link: https://ieeexplore.ieee.org/document/10387725

[2] Reda, A., Bouzid, A. y Vásárhelyi, J. (2020) *Model predictive control for automated vehicle steering*. Acta Polytechnica Hungarica. link: https://acta.uni-obuda.hu/Reda_Bouzid_Vasarhelyi_104.pdf

[3] Zhang, J., Kong, A., Tang, Y., Lv, Z., Guo, L. y Hang, P. (2024) *Application of data-driven model predictive control for autonomous vehicle steering*. arXiv. link: https://arxiv.org/pdf/2407.08401

[4] Veysi, P., Adeli, M. y Peirov Naziri, N. (2024) *Introduction to PID controller and model predictive control in engineering systems*. ResearchGate. link: https://www.researchgate.net/profile/Parsa-

Veysi/publication/379956442_Introduction_to_PID_Controller_and_Model_Predictive_Control_in_Engineering_Systems/links/66237ffa43f8df018d1ca6be/Introduction-to-PID-Controller-and-Model-Predictive-Control-in-Engineering-Systems.pdf

[5] Laoyan, S. (2025) *Metodología Agile: qué es y cómo implementarla*. Asana. link: https://asana.com/es/resources/agile-methodology

[6] Bøhn, E., Gros, S., Moe, S. y Johansen, T.A. (2021) *Optimization of the Model Predictive Control Meta-Parameters Through Reinforcement Learning*. arXiv. link: https://arxiv.org/pdf/2111.04146

[7] Chatrath, K., Zheng, Y. y Shyrokau, B. (2020) *Vehicle dynamics control using model predictive control allocation combined with an adaptive parameter estimator*. Siemens Digital Industries Software, Magna International Inc. y Delft University of Technology. link: https://www.researchgate.net/publication/343102525_Vehicle_Dynamics_Control_Using_Model_Predictive_Control_Allocation_Combined_with_an_Adaptive_Parameter_Estimator DOI: 10.4271/12-03-02-0009.

[8] British Computer Society (BCS). (2015). *Code of Conduct*. Link: https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/

[9] Association for Computing Machinery (ACM). (2018). *ACM Code of Ethics and Professional Conduct*. Link: https://www.acm.org/code-of-ethics

[10] IEEE. (2020). *IEEE Code of Ethics*. Institute of Electrical and Electronics Engineers. Link: https://www.ieee.org/about/corporate/governance/p7-8.html

[11] European Union. (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation)*. Official Journal of the European Union. Link: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679

[12] UK Government. (2018). *Data Protection Act 2018*. Legislation.gov.uk. link: https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted

[13] The Python Software Foundation. (n.d.). *PSF License Agreement for Python 3*. Link: https://docs.python.org/3/license.html

[14] NumPy Developers. (2024). *NumPy License*. NumPy. Link: https://numpy.org/doc/stable/license.html

[15] Matplotlib Developers. (2024). *Matplotlib License*. Matplotlib. Link: https://matplotlib.org/stable/users/project/license.html

[16] Raji, I. D., & Buolamwini, J. (2019). *Actionable auditing: Investigating the impact of publicly naming biased performance results of commercial AI products*. Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, 429–435. https://doi.org/10.1145/3306618.3314244

[17] Muraleedharan, A., Okuda, H. y Suzuki, T. (2022) *Real-time implementation of randomized model predictive control for autonomous driving*. IEEE Xplore. link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9366366

[18] Gokaraju, B., Steele, J.B., Doss, A.D., Turlapty, A.C. y Agrawal, R. (2019) *Maneuvering of autonomous vehicles through proportional, integral and derivative control variables*. IEEE Xplore. link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9020394

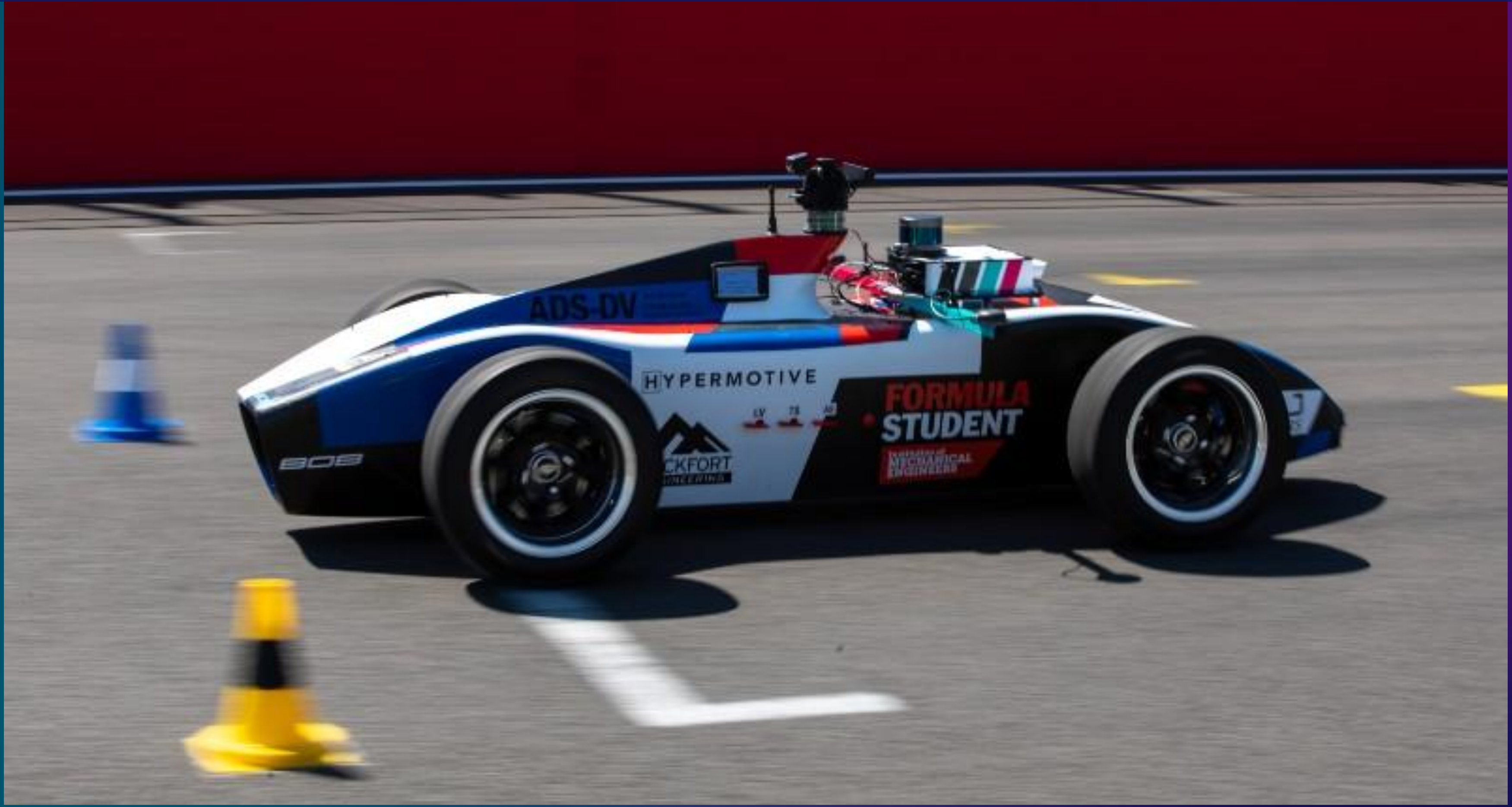
[19] Colorado Arellano, O., Hernández Romero, N., Seck Tuoh Mora, J.C. y Medina Marín, J. (2018) *Algoritmo genético aplicado a la sintonización de un controlador PID para un sistema acoplado de tanques*. Revista ICBI, UAEH. link: https://repository.uaeh.edu.mx/revistas/index.php/icbi/article/view/2935/2957

[20] Brüderlin, M. y Toochinda, D. (2015) *Discrete-time PID controller implementation*. Scilab. link: https://www.scilab.org/discrete-time-pid-controller-implementation?utm_source

[21] Barbero Ruiz, C. (2021) *Comparative analysis of MPC controllers applied to autonomous driving*. Universidad de Alcalá. link: https://ebuah.uah.es/dspace/bitstream/handle/10017/46675/TFG_Barbero_Ruiz_2021.pdf?sequence=1&isAllowed=y

[22] Mermaid Chart. *Mermaid AI*. link: https://www.mermaidchart.com/mermaid-ai?utm_source=mermaid_js&utm_medium=banner_ad&utm_campaign=aibundle

DEVELOPMENT AND COMPARATIVE ANALYSIS OF MPC AND PID CONTROLLERS FOR AUTONOMOUS VEHICLES

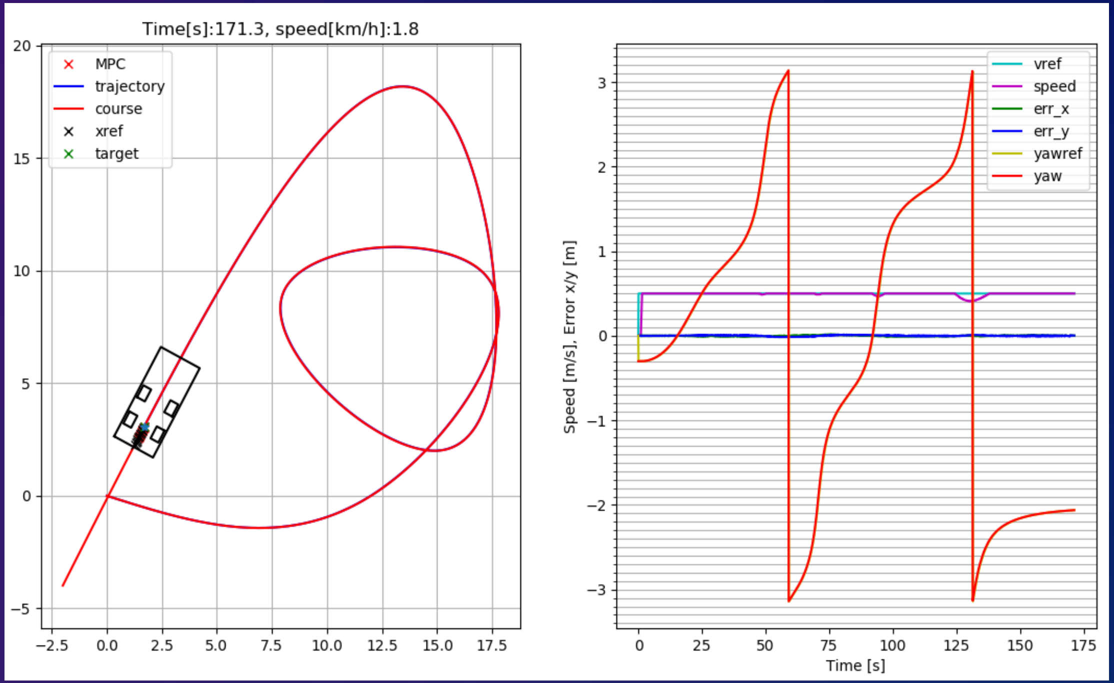


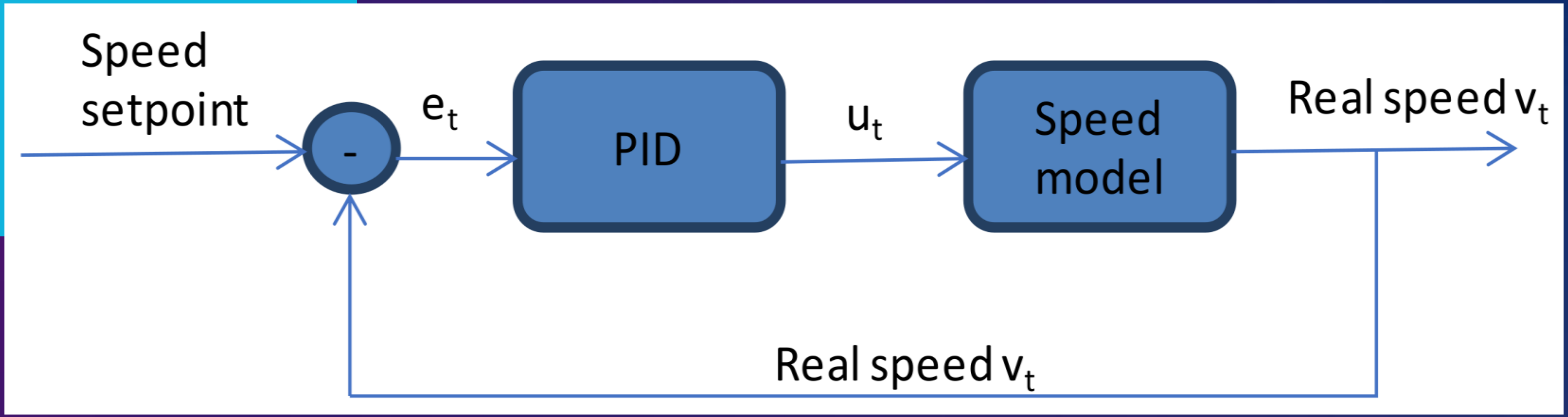
19314173

Eduardo de Quiroga

INTRODUCTION

- MPC Properties
- MPC Performance
- PID Properties
- PID Performance
- MPC VS PID



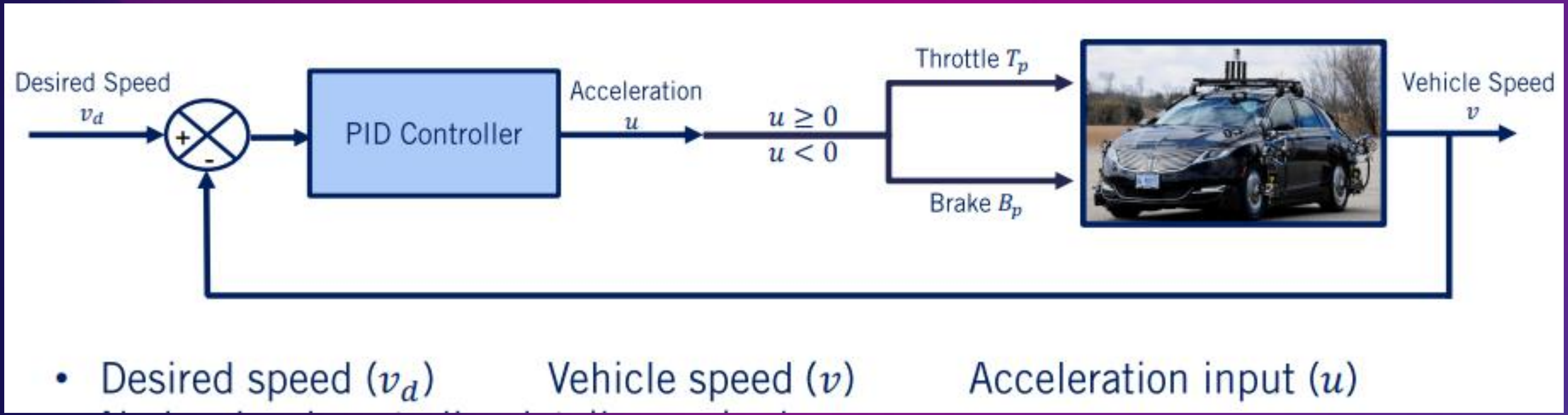


AIM & OBJECTIVES

- FUNCTIONAL CONTROL SYSTEM
- COMPARE THE EFFECTIVENESS
- PERFORMANCE AND FEASIBILITY OF CONTROLLER´S IN REAL-TIME
- FUTURE USE

LITERATURE REVIEW - PID

- EFFECTIVENESS
- EASY TO MAKE
- COMPARISON TO MPC
- SINGLE INPUT SINGLE OUTPUT(SISO)



- Desired speed (v_d) Vehicle speed (v) Acceleration input (u)

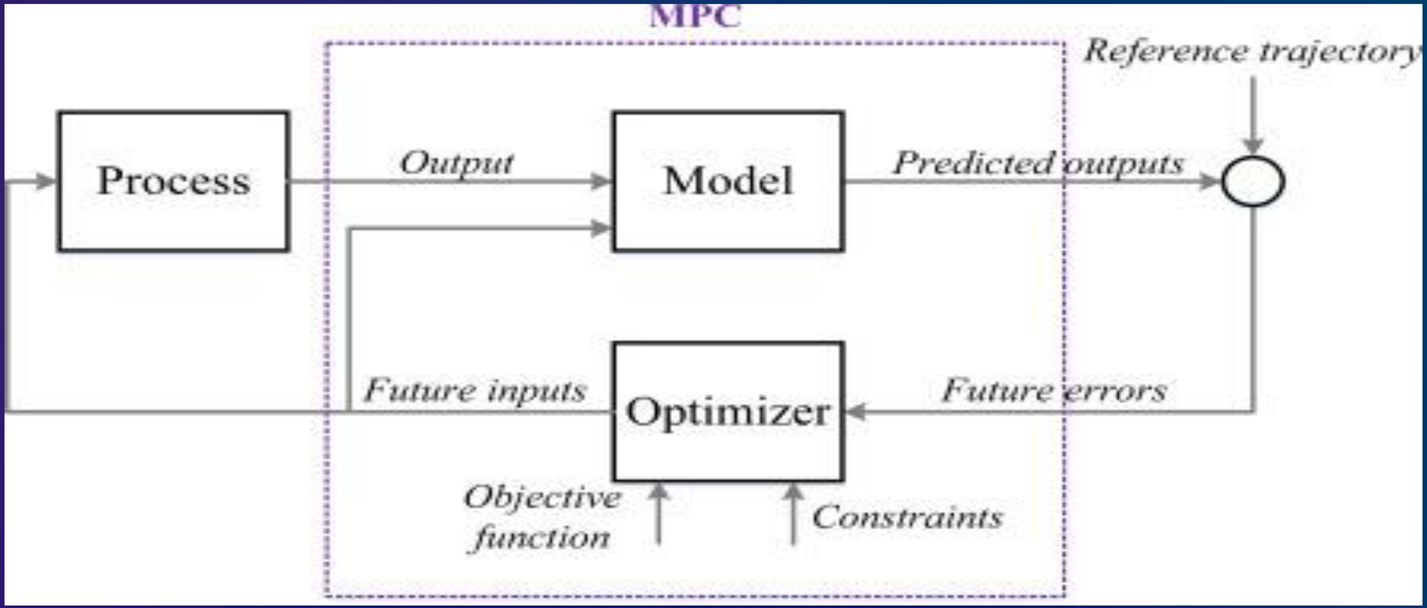
LITERATURE REVIEW - MPC

DATA-DRIVEN

APPLICATION

REAL-TIME CONTROL

HIGH PRECISION TRACKING



TESTING

Visual Studio Code

OBRA(Oxford Brookes Racing Autonomous)

Video Presentation: https://drive.google.com/file/d/1vSvt96XEjO6DG1ZAwPiL-J2xPlodLKcc/view?usp=drive_link

RISK

COMPUTATIONAL COMPLEXITY

HIGH POWER CONSUMPTION

PARAMETER CALIBRATION

SINGLE INPUT SINGLE OUTPUT

ANTICIPATING FUTUREN STATES

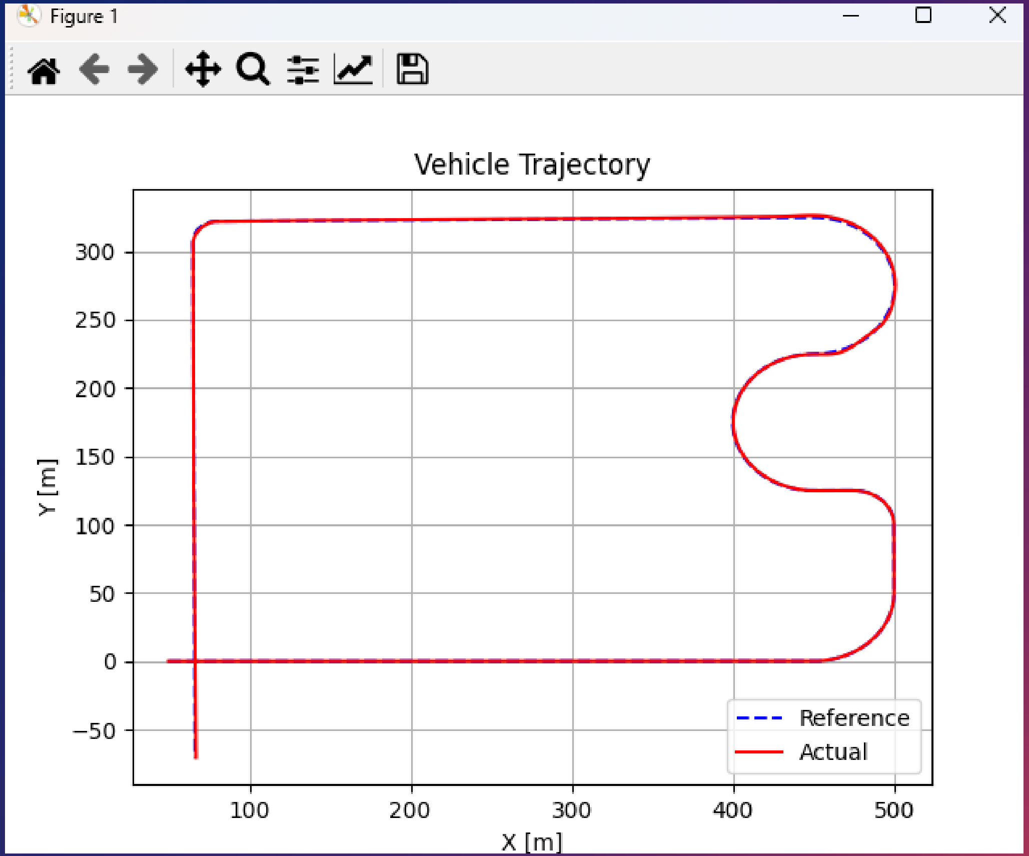


Figure 1
Vehicle Trajectory
300
250
200
150
100
50
0
−50
Y [m]
100
200
300
400
500
X [m]
Reference
Actual

MPC RESULTS

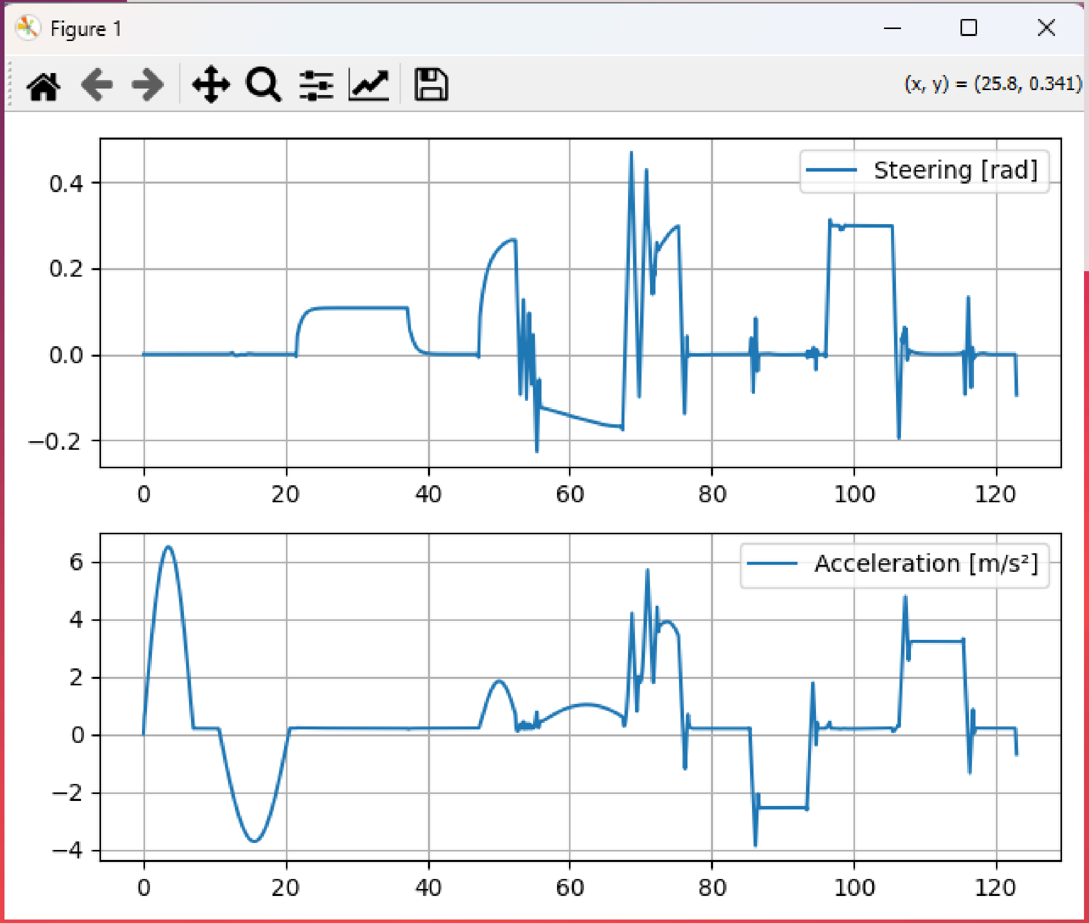


Figure 1
(x, y) = (25.8, 0.341)
Steering [rad]
0.4
0.2
0.0
−0.2
0
20
40
60
80
100
120
Acceleration [m/s²]
6
4
2
0
−2
−4
0
20
40
60
80
100
120

PID RESULTS

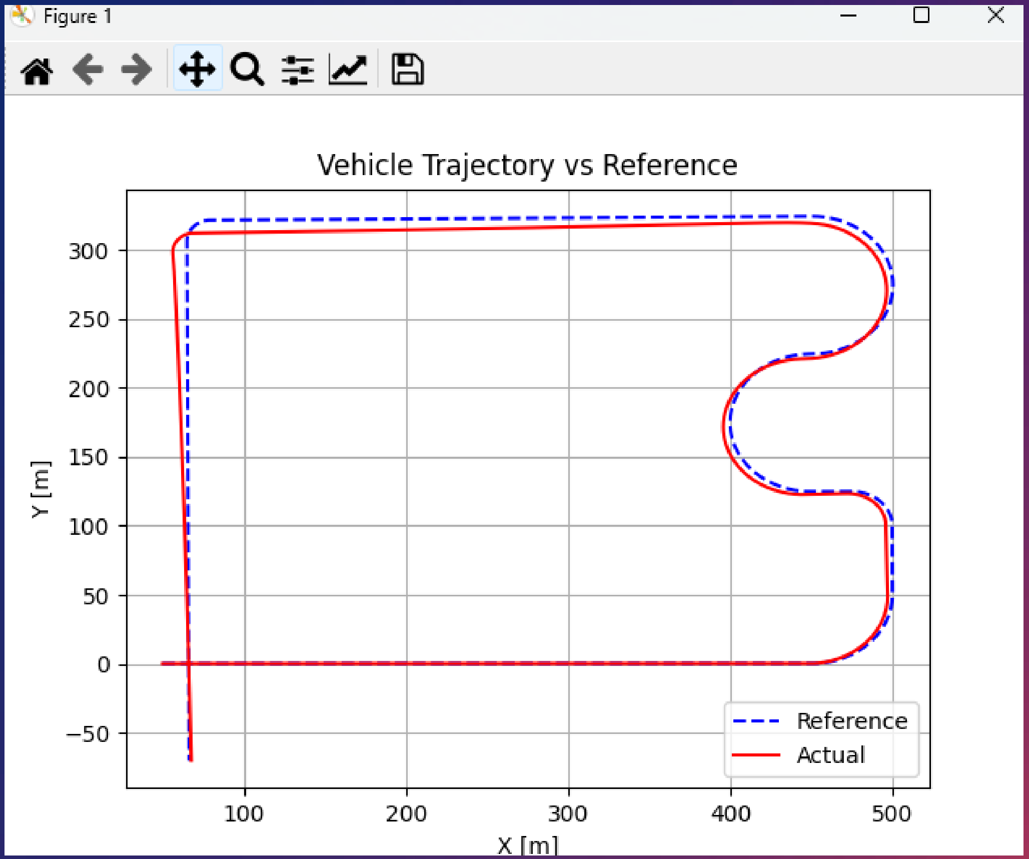


Figure 1
Vehicle Trajectory vs Reference
Reference
Actual
X [m]
Y [m]
100
200
300
400
500
300
250
200
150
100
50
0
−50

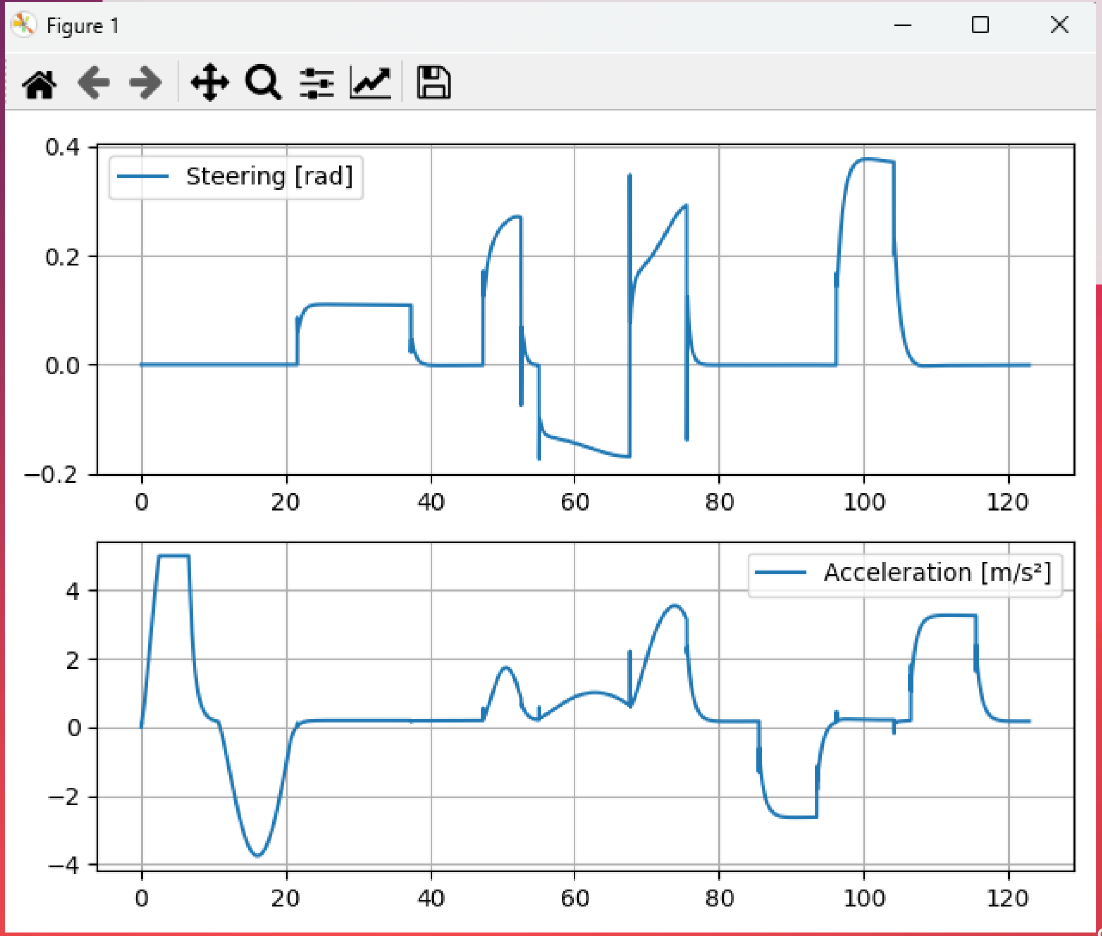
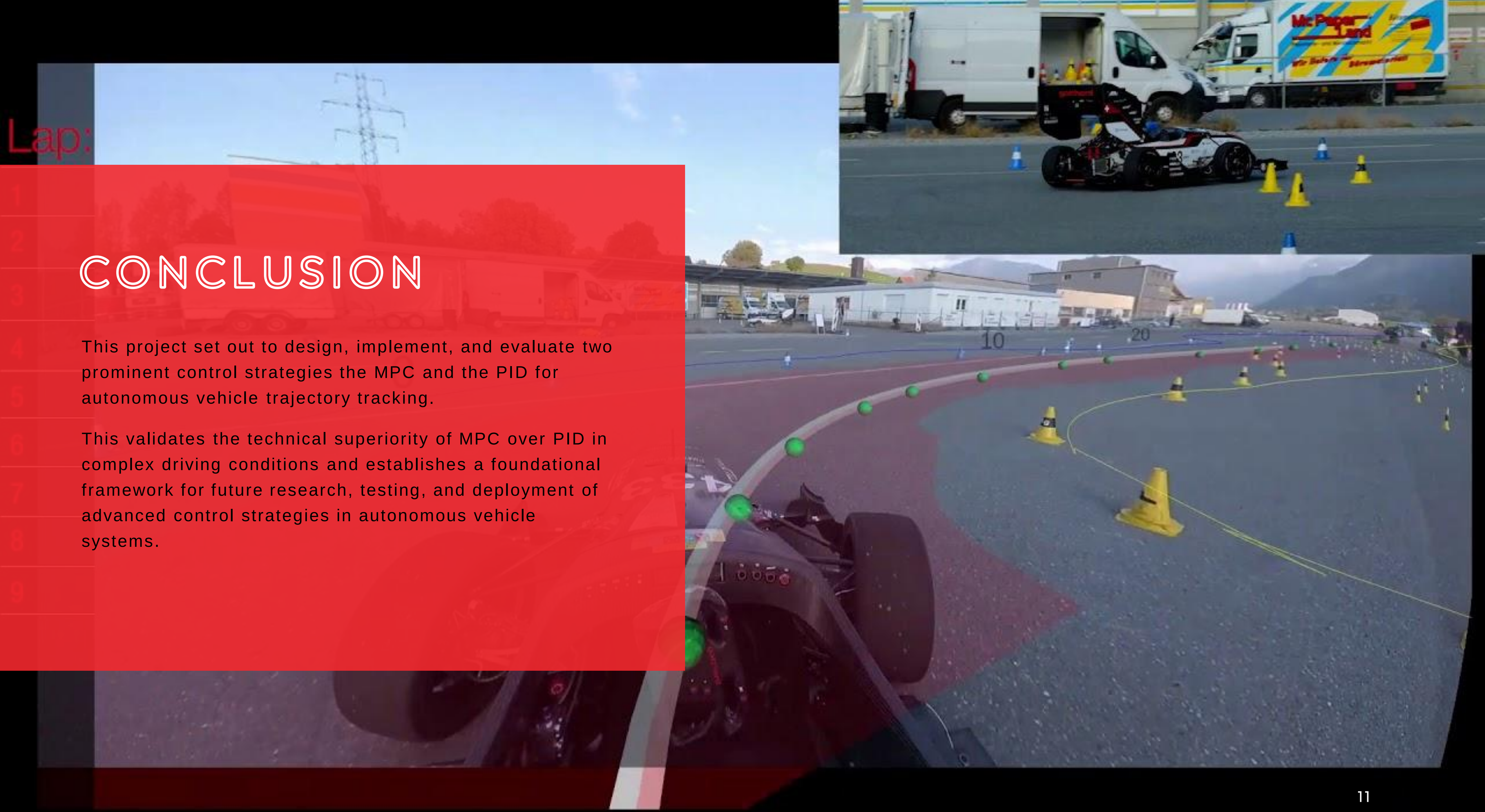


Figure 1
Steering [rad]
Acceleration [m/s²]
0.4
0.2
0.0
−0.2
4
2
0
−2
−4
0
20
40
60
80
100
120

DISCUSSION

THE RESULTS CLEARLY REFLECTS THAT MPC NOT ONLY OUTPERFORMS PID IN NEARLY ALL PERFORMANCE RELATED ASPECTS BUT ALSO LAYS A STRONGER FOUNDATION FOR SCALABLE AND INTELLIGENT AUTONOMOUS VEHICLE CONTROL.

Criteria	MPC Controller	PID Controller	Observation
Trajectory Accuracy	Closely follows the reference trajectory with minimal deviation	Deviates more noticeably, especially in curves	MPC demonstrates higher precision in path tracking
Steering Response	Smooth and predictive steering adjustments	More abrupt and reactive steering actions	MPC offers better vehicle stability and control smoothness
Yaw Stability	Predictive and consistent yaw control across time	Fluctuating yaw corrections with lag	MPC provides superior rotational stability and vehicle orientation control



CONCLUSION

This project set out to design, implement, and evaluate two prominent control strategies the MPC and the PID for autonomous vehicle trajectory tracking.

This validates the technical superiority of MPC over PID in complex driving conditions and establishes a foundational framework for future research, testing, and deployment of advanced control strategies in autonomous vehicle systems.

THANK YOU

Eduardo de Quiroga

19314173

19314173@brookes.ac.uk

Leader of OBRA Control team

REFERENCES

Meng, D., Chu, H., Tian, M., Gao, B. y Chen, H. (2024) Real-Time High-Precision Nonlinear Tracking Control of Autonomous Vehicles Using Fast Iterative Model Predictive Control. IEEE Xplore. Link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10387725

Reda, A., Bouzid, A. y Vásárhelyi, J. (2020) Model Predictive Control for Automated Vehicle Steering. Acta Polytechnica Hungarica. Link: https://acta.uniobuda.hu/Reda_Bouzid_Vasarhelyi_104.pdf

Zhang, J., Kong, A., Tang, Y., Lv, Z., Guo, L. y Hang, P. (2024) Application of Datadriven Model PredictiveControl for Autonomous Vehicle Steering. arXiv. link: https://arxiv.org/pdf/2407.08401

Veysi, P., Adeli, M. y Peirov Naziri, N. (2024) *Introduction to PID Controller and Model Predictive Control in Engineering Systems*. ResearchGate. Link: https://www.researchgate.net/profile/Parsa-Veysi/publication/379956442_Introduction_to_PID_Controller_and_Model_Predictive_Control_in_Engineering_Systems/links/66237ffa43f8df018d1ca6be/Introduction-to-PIDController-and-Model-Predictive-Control-in-Engineering-Systems.pdf

Muraleedharan, A., Okuda, H. y Suzuki, T. (2022) *Real-time implementation of randomized model predictive control for autonomous driving*. IEEE Xplore. link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9366366

Gokaraju, B., Steele, J.B., Doss, A.D., Turlapty, A.C. y Agrawal, R. (2019) *Maneuvering of autonomous vehicles through proportional, integral and derivative control variables*. IEEE Xplore. link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9020394

Colorado Arellano, O., Hernández Romero, N., Seck Tuoh Mora, J.C. y Medina Marín, J. (2018) *Algoritmo genético aplicado a la sintonización de un controlador PID para un sistema acoplado de tanques*. Revista ICBI, UAEH. link: https://repository.uaeh.edu.mx/revistas/index.php/icbi/article/view/2935/2957

Brüderlin, M. y Toochinda, D. (2015) *Discrete-time PID controller implementation*. Scilab. link: https://www.scilab.org/discrete-time-pid-controller-implementation?utm_source

Barbero Ruiz, C. (2021) *Comparative analysis of MPC controllers applied to autonomous driving*. Universidad de Alcalá. link: https://ebuah.uah.es/dspace/bitstream/handle/10017/46675/TFG_Barbero_Ruiz_2021.pdf?sequence=1&isAllowed=y

Development and Comparative Analysis of MPC and PID Controllers for Autonomous Vehicles

Eduardo de Quiroga Final year project. Supervisor: Kashinath Basu 19314173@brookes.ac.uk

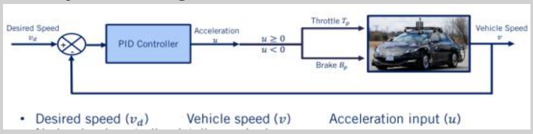
Introduction

This project focuses on developing and comparing PID and MPC—for autonomous vehicle trajectory tracking. It aims to demonstrate the superior performance of MPC in terms of accuracy, adaptability, and constraint handling under dynamic conditions.

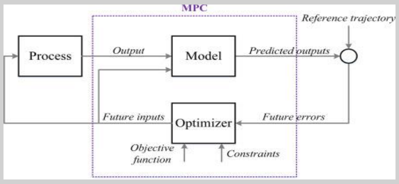
Through simulation, visualization, and performance analysis the project lays the foundation for future MPC implementation in real-world systems like OBRA.

Literature

In autonomous vehicle control, PID controllers are widely used due to their simplicity and effectiveness in basic trajectory tracking. However, They struggle in dynamic or constrained environments.



Model Predictive Control (MPC) uses a predictive model and optimization to handle constraints and anticipate future system behavior, offering superior performance in complex scenarios, though at a higher computational cost.



This project builds on key literature that highlights the strengths and limitations of both controllers. Research demonstrates MPC's precision and adaptability, while others explore the practical roles and constraints of PID in autonomous systems.

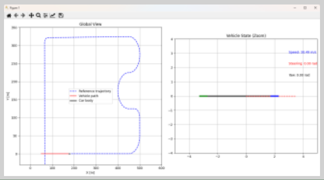
Methodology

This project adopted an iterative, agile-inspired development process, structured around sprints focused on key milestones: reference generation, PID and MPC implementation, debugging, and evaluation. After each sprint, simulations were tested and refined, enabling rapid prototyping and continuous improvement.

Agile methodology
1 Plan
2 Design
3 Develop
4 Test
5 Deploy
6 Review
asana

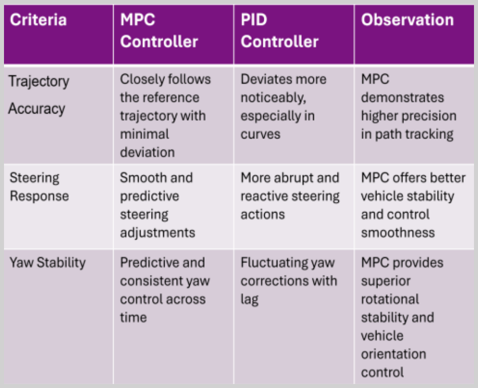
Results

The results highlights MPC's superiority in precision, adaptability, and real-time performance, making it a more robust and scalable solution for autonomous vehicle control compared to traditional PID controllers.



Conclusions

This project successfully achieved its goal of comparing MPC and PID controllers for autonomous vehicle trajectory tracking. Results showed that MPC significantly outperforms PID.
Challenges were overcome through iterative testing, improving system understanding and robustness.
The deliverables are a Functional MPC and PID controllers, shared simulation platform and an animated visualizations.



Criteria	MPC Controller	PID Controller	Observation
Trajectory Accuracy	Closely follows the reference trajectory with minimal deviation	Deviates more noticeably, especially in curves	MPC demonstrates higher precision in path tracking
Steering Response	Smooth and predictive steering adjustments	More abrupt and reactive steering actions	MPC offers better vehicle stability and control smoothness
Yaw Stability	Predictive and consistent yaw control across time	Fluctuating yaw corrections with lag	MPC provides superior rotational stability and vehicle orientation control

Code:https://gitlab.com/mpc_pid_controller/mpc