UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Università degli Studi
di Modena e Reggio Emilia

Dottorato di ricerca in
"Computer and Data Science for Technological and Social Innovation"

Ciclo XXXVIII

Vehicle Dynamics and Friction-Adaptive Model Predictive Control in Autonomous Racing: A Comprehensive Modeling Approach

Candidato **Nicola Musiu**

Relatore (Tutor): **Prof. Marko Bertogna**

Correlatore (Co-Tutor): **Prof. Silvio Sorrentino**

Secondo Correlatore (Co-Tutor): **Prof. Alessandro De Felice**

Coordinatore del Corso di Dottorato: **Prof. Andrea Marongiu**

Acknowledgments

I would like to sincerely thank Prof. Marko Bertogna for the opportunity and the trust placed in me throughout this work, as well as Prof. Silvio Sorrentino and Prof. Alessandro De Felice for their valuable support and guidance during the research activity.

I am also deeply grateful to the Unimore Racing research group, and in particular to Ayoub Raji and Alessandro Toschi. Without their guidance, my research would have lacked direction. They continuously motivated me to push beyond the limits of what I believed possible, profoundly shaping both my work and my approach to research. This thesis builds strongly upon the work of Ayoub Raji, without which this contribution would not have been possible.

A special thanks goes to Eugenio Mascaro and Pietro Musso, whose commitment and development efforts were fundamental to achieving the results presented in this work.

Finally, I would like to express my immense gratitude to my family and my loved one, whose constant support accompanied me throughout this incredible journey.

Contents

List of Figures

List of Tables

Abstract

In recent years, the development of autonomous driving systems has increasingly focused on advanced planning and control architectures capable of operating in complex dynamic conditions. Among these, model-based approaches, particularly those relying on Model Predictive Control (MPC), have emerged as some of the most effective solutions. Despite the central role of the dynamic vehicle model in such frameworks, it is often treated as a compromise between physical fidelity and implementation simplicity, with the latter frequently prioritized. This may limit model effectiveness, especially in high-performance scenarios or under variable grip conditions, and often leads to model-based solutions being overlooked due to formulation complexity, parameter identification challenges, and limited vehicle dynamics expertise.

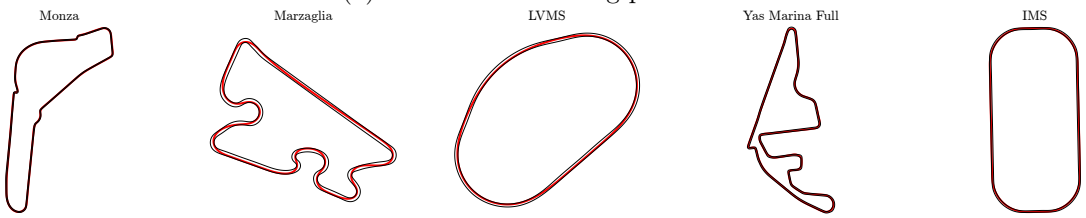
This work builds on the idea that, within a model-based philosophy, the vehicle model must represent the core of the planning and control stack, rather than a secondary or purely instrumental component. From this perspective, the vehicle model is not merely a prediction tool, but the common element governing planning, control, and parameter estimation.

This thesis extends a consolidated line of research on planning and control for autonomous racing, moving toward a systematic, model-centric approach that directly links advanced planning and control algorithms with accurate physical vehicle modeling, bridging the gap between autonomous driving research and vehicle dynamics. A range of vehicle models, from oversimplified to highly complex, is formulated and analyzed to identify a modeling solution capable of operating at, and beyond, the tire-road friction limit, while achieving an optimal trade-off between accuracy, computational efficiency, and calibration effort. Building on previous works, several design gaps in motion planning and control architectures are addressed, including the development of online model adaptation modules to handle varying operating conditions. The resulting framework is extensively validated in a custom software-in-the-loop simulation environment, developed and validated during the route, and subsequently tested in real-world autonomous racing competitions, demonstrating high efficiency and performance compared to the state of the art and most approaches currently employed in this domain.

Chapter 1

Introduction

(a) Autonomous Racing platforms.



(b) Autonomous Racing Track.

Figure 1.1: In Subfig.1.1a, the autonomous vehicles used in this thesis as reference for design and validation. From left to right: the Dallara EAV-25 and the Dallara AV-24. In Subfig.1.1b, the race-track which have hosted races and events. From left to right: Monza F1 circuit, Marzaglia (Modena), Las Vegas Motor Speedway, Yas Marina F1 circuit, and Indianapolis Motor Speedway.

1.1 The context: autonomous racing

Similar to the historical role of traditional motorsport in advancing the automotive industry (where many improvements in vehicle safety and performance originated), autonomous racing has emerged as a research laboratory for developing and testing advanced autonomous driving hardware and algorithms. The unique challenges of racing scenarios, including high-dynamics scenarios and extreme operating conditions, act as a catalyst for designing highly effective autonomous driving systems. Operating at the edge of vehicle dynamics imposes stringent requirements on modeling, planning, and control, promoting the development of algorithms capable of matching (or even outperforming) human drivers in decision-making and vehicle management.

The rapid growth of interest in autonomous racing has been further boosted by the availability of low-cost research platforms and the rise of dedicated competitions. Initiatives such as F1TENTH and Formula Student Driverless (FSD) have played a key role in making research in this field accessible, enabling fast experimentation and large-scale validation of autonomous driving algorithms. A comprehensive overview of research in autonomous racing is provided in [1], which systematically reviews the main scientific contributions related to autonomous driving architectures, vehicle dynamics modeling, simulation techniques, and planning and control strategies. The work highlights the rapid growth of academic interest in the field and the increasing maturity of proposed solutions. Among the most recent surveys, reference [2] explores the latest trends in vehicle modeling specifically for autonomous racing. This survey provides an up-to-date perspective on vehicle model selection, their formulations, and applications.

More recently, full-scale competitions like the Indy Autonomous Challenge (IAC[1]) and the Abu Dhabi Autonomous Racing League (A2RL[2]) have raised the bar significantly, introducing full-size vehicles, extremely high speeds in multi-agent scenarios, and unprecedented safety and performance requirements. These two full-scale competitions provide the scenarios in which the solutions studied in this thesis were both inspired and validated.

Indy Autonomous Challenge. The IAC AV-24 platform (Subfig.1.1a, right) is based on a modified Dallara Indy NXT chassis, fitted with Bridgestone racing slick tires and configurable for both oval and road course circuits. The vehicle is powered by a turbocharged 2.0-liter inline four-cylinder engine derived from the Honda K20C platform and equipped with a locked differential. The vehicle is equipped with an onboard edge-computing unit based on a dSPACE AUTERA AutoBox, featuring a high-performance multi-core CPU (Intel Xeon D-2166NT, 3.00 GHz) and a dedicated GPU (NVIDIA A5000) for real-time autonomous driving workloads. Custom steer-by-wire and brake-by-wire actuation systems enable full drive-by-wire control. For perception and state estimation, the platform integrates a heterogeneous sensor suite, including four LiDARs, two radars, and six cameras, together with a high-precision GNSS/INS system, providing robust environment awareness and vehicle localization at racing speeds. The Indy Autonomous Challenge (IAC) competitions began in 2021 on the historic Indianapolis Motor Speedway oval. Subsequent events were held on oval circuits, including Las Vegas Motor Speedway and Texas Motor Speedway. In 2024, the competition expanded to road course racing, with events hosted on the Formula 1 Monza racetrack. Early competitions focused on single-vehicle lap-time challenges, where the winner was determined by the fastest average speed over two consecutive laps, awarding a one-million-dollar prize. In later editions, starting from the 2022 event at Las Vegas Motor Speedway, the race format evolved toward multi-agent scenarios, introducing head-to-head races and progressively increasing complexity, culminating in four-car wheel-to-wheel racing at LVMS in 2025.

Abu Dhabi Autonomous Racing League. A similar high-performance autonomous racing platform (Subfig.1.1a, left) is adopted in the Abu Dhabi Autonomous Racing League (A2RL), where the Dallara EAV-24 / EAV-25 autonomous race car is derived from the Dallara Super Formula (SF23) chassis. The A2RL platform is designed specifically for road circuits and benefits from significantly higher aerodynamic downforce and racing-spec tyres, delivering overall performance that approaches that of current Formula 1 cars. The powertrain configuration shares similarities with the IAC's, using the same turbocharged

[1] https://www.indyautonomouschallenge.com/
[2] https://a2rl.io/

Honda engine, but it is equipped with a self-locking differential. Sensor's setup resemble those implemented in the IAC AV-24 as well. The first A2RL car races were held in April 2024 at the Yas Marina Formula 1 circuit (Full layout), featuring a competition format that included hot-lap qualifying, head-to-head, and a four-vehicle final race. In 2025, A2RL expanded the event format further, pushing the boundaries of multi-agent autonomous competition by introducing a qualifying session, a 3-vehicle multi-car sprint, and a final race with six autonomous vehicles simultaneously on track at the Yas Marina North layout.

1.2 Background

The author's published works form the experimental and methodological basis of this thesis. The formulations, figures, and results presented are mostly derived from the following publications and co-advised theses:

- [3] Ayoub Raji, Alexander Liniger, Andrea Giove, Alessandro Toschi, Nicola Musiu, Daniele Morra, Micaela Verucchi, Danilo Caporale, and Marko Bertogna. **Motion planning and control for multi vehicle autonomous racing at high speeds**. In 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC). IEEE, October 2022.

- [4] Ayoub Raji, Danilo Caporale, Francesco Gatti, Andrea Giove, Micaela Verucchi, Davide Malatesta, Nicola Musiu, Alessandro Toschi, Silviu Popitanu, Fabio Bagni, Massimiliano Bosi, Alexander Liniger, Marko Bertogna, Daniele Morra, Francesco Amerotti, Luca Bartoli, Federico Martello, and Riccardo Porta. **er.autopilot 1.0: The full autonomous stack for oval racing at high speeds**. Field Robotics, 4:99-137, 01 2024.

- [5] Alessandro Toschi, Nicola Musiu, Francesco Gatti, Ayoub Raji, Francesco Amerotti, Micaela Verucchi, and Marko Bertogna. **Guess the drift with lop-ukf: Lidar odometry and pacejka model for real-time racecar sideslip estimation**. In 2024 IEEE Intelligent Vehicles Symposium (IV), pages 885-891, 2024. 05 2024.

- [6] Ayoub Raji, Nicola Musiu, Alessandro Toschi, Francesco Prignoli, Eugenio Mascaro, Pietro Musso, Francesco Amerotti, Alexander Liniger, Silvio Sorrentino, and Marko Bertogna. **A tricycle model to accurately control an autonomous racecar with locked differential**. In 2023 IEEE 11th International Conference on Systems and Control (ICSC), pages 782-789. IEEE, December 2023.

- [7] Ayoub Raji, Danilo Caporale, Francesco Gatti, Alessandro Toschi, Nicola Musiu, Micaela Verucchi, Francesco Prignoli, Davide Malatesta, AndrÃ© Fialho Jesus, Andrea Finazzi, Francesco Amerotti, Fabio Bagni, Eugenio Mascaro, Pietro Musso, and Marko Bertogna. **er.autopilot 1.1: A software stack for autonomous racing on oval and road course tracks**. IEEE Transactions on Field Robotics, 1:332-359, 2024.

- [8] Adrian Remonda, Nicklas Hansen, Ayoub Raji, Nicola Musiu, Marko Bertogna, Eduardo E. Veas, and Xiaolong Wang. **A simulation benchmark for autonomous racing with large-scale human data**. In The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Neurips), 2024.

- [9] Nicola Musiu, Eugenio Mascaro, Ayoub Raji, Alessandro De Felice, Silvio Sorrentino, Marko Bertogna. **A Comprehensive Benchmark of Vehicle Dynamics Models for Autonomous Racing: a Deep Dive into MPC**, Vehicle System Dynamics, vol. 0, no. 0, pp. 1-33, 2026.
- [10] Pietro Musso, **Development of a thermal tyre model for an autonomous race car**, Master Thesis, 2022/2023.
- [11] Eugenio Mascaro, **Sviluppo del modello multi-body della vettura autonoma Dallara AV21 e ottimizzazione per applicazioni real-time**, Master Thesis, 2022/2023.

The work presented in this thesis builds upon a consolidated line of research on planning and control for autonomous racing, extending previous studies toward a more systematic and model-centric approach. In particular, it focuses on key vehicle dynamics-related tasks, including vehicle modeling, dynamic behavior representation, and online parameter estimation.

Model development was driven by progressively more complex real-world racing scenarios. Initial efforts focused on Indy Autonomous Challenge oval racing, where vehicle asymmetry and strong banking required explicit modeling of road inclination and asymmetric dynamics, resulting in substantially improved trajectory tracking [3, 4]. To further exploit model-based approaches, state estimation techniques were introduced, leading to improved path-tracking performance and increased robustness [5]. The research then expanded to road-course scenarios, where braking and acceleration phases introduce strong longitudinal-lateral coupling and different track layout, motivating enhanced dynamic formulations capable of capturing effects such as locked differentials [6, 7]. More recently, a simplified limited-slip differential (LSD) model was investigated. While promising in simulation, real-world deployment exposed practical limitations, leading to a key insight of this research: model quality is not solely determined by accuracy, but also by numerical stability, robustness, and calibration effort [9].

In [3], a multi-layer motion planning and control architecture for autonomous racing was presented, capable of obstacle avoidance, active overtaking, and high-speed operation. The motion planning relied on a model-free Frenet-frame-based approach, generating collision-free trajectories through geometric spline selection, while path tracking was handled by an MPC controller based on a single-track model with nonlinear tire dynamics. By incorporating vehicle asymmetry and combined-slip effects, the controller achieved reliable performance close to the grip limit, as confirmed by on-track experiments on a highly banked oval circuit.

In [5], the motion framework was extended by introducing a state estimation module to update the vehicle model states and enable fully closed-loop architectures. The proposed approach, based on onboard sensors and an Unscented Kalman Filter (UKF), significantly improved robustness and path-tracking performance in high-dynamics scenarios, and represented a first step toward online tire-road friction estimation, further detailed in Sec. 6.2.

The role of model fidelity was further explored in [6], where the vehicle dynamics were extended to a tricycle formulation capturing the effects of a fully locked differential. Validated against a high-fidelity multibody reference model, this formulation was integrated into the MPC controller and demonstrated a substantial reduction in lateral tracking error in high-curvature sections, without degrading performance elsewhere.

A qualitative shift was introduced in [12], where the vehicle model became the core element in both planning and control, enabling a fully model-based architecture. A physically

accurate vehicle model was directly embedded into an MPC-based planner, replacing the traditional geometric Frenet planner and yielding more realistic, controller-aligned trajectories. Although validated exclusively in simulation, this work demonstrated the potential of fully integrating vehicle dynamics into the autonomous racing stack.

Motivated by these findings, the impact of model fidelity and numerical stability on closed-loop path-following performance was systematically analyzed in [9]. The study highlighted that model quality is not solely determined by accuracy, but also by robustness, numerical stability, and calibration effort, emphasizing the practical limitations of both overly simplified and excessively complex formulations within an MPC framework.

1.3 Research Objectives and Contribution

This thesis is structured around three tightly connected research objectives, all grounded in the premise that, within a model-based framework, the vehicle model must represent the core of the planning and control stack, rather than a secondary or purely instrumental component:

- The formulation and systematic evaluation of vehicle models, ranging from oversimplified to highly complex, to identify solutions capable of operating at, and beyond, the tire-road friction limit in autonomous racing planning and control.
- The development of online model identification and adaptation modules to handle varying operating conditions, enabling continuous updates of the planning and control algorithms and improving overall robustness and performance of the system.
- The design of a custom software-in-the-loop simulation environment, essential for the development, testing, and validation of the autonomous driving stack prior to real-world deployment.

These three topics are analyzed in detail in the subsequent sections.

1.3.1 Simplified Vehicle Models

The first research direction focuses on vehicle dynamics modeling, aiming to answer a key question:

> *What level of model complexity is truly beneficial for autonomous planning and control, and when does additional complexity become unnecessary or even detrimental?*

A wide range of vehicle models was investigated, spanning from point-mass (detailed in Sec.A.1) and kinematic formulations to linear and nonlinear dynamic models. Starting from classical single-track models, new formulations were derived, such as simplified tricycle models, which retain computational efficiency while capturing additional physical effects relevant to high-performance driving, and enhanced kinematic single-track models, which incorporate dynamic effects while preserving a compact state representation. Systematic experimentation across modeling paradigms highlighted that model selection is scenario-dependent: the appropriate model varies with the operating context (urban driving, racing, drifting). For example, in the most recent A2RL event, a simple nonlinear single-track dynamic model was found to provide the optimal trade-off for autonomous racing, effectively balancing accuracy, computational efficiency, and calibration effort.

1.3.2 Model Parameters Estimation

Once the vehicle model is established as the core element governing planning and control, parametrization and adaptation become a fundamental research problem:

> *How can the model be parametrized and adapted effectively to the actual operating conditions?*

Higher model complexity comes at a cost: more parameters require more sophisticated estimation algorithms, increasing complexity both in the model and in the estimation framework. This motivated the development of robust identification techniques for estimating vehicle states and model parameters. While initial efforts focused on offline identification, it soon became clear that online adaptation was essential to maintain performance under varying conditions. Consequently, online parameter estimation strategies were developed, allowing the model to continuously adapt to tire-road interactions. In particular, a grip estimation module enables real-time updates of tire model parameters, enhancing both planning and control performance. Within the proposed framework, all model-based modules benefit from consistent adaptation, ensuring that the vehicle operates near optimal conditions even under changing track conditions or evolving tire characteristics.

1.3.3 High-Fidelity Simulation Environment

The third research direction addresses a practical but critical challenge:

> *How to safely and efficiently validate simplified models and algorithms in contexts where on-track testing is expensive, risky, or limited?*

A high-fidelity vehicle dynamics simulator was developed as a virtual testing environment, exploiting multi-body modeling to provide a reliable vehicle ground truth. Digital twins of the vehicles were created using Dymola, a Modelica-based multi-domain environment widely adopted for vehicle-dynamics-oriented simulations. Building on these models, a custom software-in-the-loop (SiL) simulator was implemented using the Functional Mock-up Interface (FMI) standard for real-time algorithm testing and validation. This setup allows entire test sessions to be reproduced virtually before real-world deployment, enabling early detection of potential planning or control issues and reducing development risk. Beyond this, the simulator is an integral part of the model-centric workflow, serving as the reference against which all simplified models are derived and tested.

1.3.4 Contribution

This thesis extends previous works on model-based planning and control for autonomous racing by consolidating these insights into a unified and fully operational framework, addressing several design gaps that emerged during intensive simulation testing. The proposed framework is highly flexible and portable across different applications, being robust to a wide range of tasks: low- and high-speed driving, high-dynamics maneuvers, multi-agent scenarios, and different vehicles and tracks, all requiring minimal calibration effort. A simple, yet effective state and parameter estimation module is presented to track evolving model parameters over time, enabling the planning and control framework to adapt to varying operating conditions, such as low- and high-grip environment. Moreover, a validated methodology for developing a high-fidelity, real-time capable simulator is presented, which proved essential during the design and development phases.

Finally, the proposed framework is validated in real-world applications during A2RL autonomous racing events, demonstrating superior performance compared to the state of the art and most currently employed approaches.

1.4 Thesis outline

This thesis is organized as follows:

Chapter 2 Reviews the state of the art and current research trends in vehicle dynamics modeling, with a focus on tire modeling and high-performance driving applications.

Chapter 3 Presents commonly adopted vehicle dynamics models, from simplified to more complex formulations. The kinematic single-track, dynamic single-track, and tricycle models are formulated with reference to a rear-wheel-drive formula car architecture. For each model, governing equations, schematics, and tables of constitutive parameters are provided, along with operating ranges and underlying assumptions. Tire modeling aspects are also discussed. Simple maneuver simulations are used to assess numerical stability with common solvers and highlight formulation- and implementation-related challenges. Execution times are reported to evaluate suitability for real-time. Point-mass formulation is not compared against the others, for the reasons explained in Sec.A.1.

Chapter 4 Introduces the simulation environments developed for simplified model validation (Section 4.2.1) and autonomous driving algorithm testing (Section 4.2.2). The chapter first presents the high-fidelity digital twin of the autonomous vehicle, implemented as a complex multibody model optimized for real-time simulation while faithfully reproducing vehicle dynamics. Road modeling aspects are also described, along with results on model accuracy and real-time performance.

Chapter 5 Presents the latest version of the motion planning and control framework. Both planning and control layers are described in detail, with emphasis on the design choices behind the final architecture. Practical implementation aspects, such as the cost-function state machine for selecting the MPC objective and low-level control strategies to handle unmodeled longitudinal effects, are discussed. Testing results highlight challenges related to model linearization and cost-function tuning, providing insights for future improvements.

Chapter 6 Presents a validated methodology for state estimation and online model adaptation under varying operating conditions. The approach integrates multiple onboard sensors (e.g., LiDAR, IMU) with filtering techniques to enable the motion framework to perform at its full potential. A dedicated tire-road friction estimation module continuously updates model parameters, ensuring robust operation across diverse scenarios. Chapter results detail the performance of these modules and discuss potential directions for future improvements.

Chapter 7 Presents extensive simulation analyses to evaluate the physical accuracy and numerical stability of the simplified vehicle models. Open-loop tests compare vehicle motion and handling diagrams against the high-fidelity multibody reference. Closed-loop path-following performance is assessed by integrating each model into an MPC framework

under high-dynamics conditions, examining control effectiveness and robustness to model mismatch. These results identify the most suitable models for autonomous racing applications and provide insights into the trade-offs between accuracy, computational cost, and calibration effort.

Chapter 8 Presents the experimental results of the proposed framework, highlighting planning and control performance. Tests were conducted with the Dallara EAV24 Super Formula car at the Yas Marina Circuit (north layout) during the A2RL autonomous racing event. The planning layer demonstrated flexibility in multi-agent scenarios, generating feasible overtaking trajectories and leveraging vehicle feedback to optimize race progression. The control layer ensured precise path-tracking and robustness. Selected edge cases further illustrate the controller's performance under challenging, high-dynamics conditions.

Chapter 2

Related Works

Vehicle modeling is a key research topic in autonomous driving, with numerous approaches proposed to support planning and control applications [13–16]. In this context, model-based solutions, particularly those relying on Model Predictive Control (MPC), have proven among the most effective [3, 17–22]. Yet, despite the central role of the dynamic vehicle model, it is often treated as a compromise between physical fidelity and implementation simplicity, with simplicity frequently prioritized [23–26] due to problem complexity, parameter identification challenges, and limited vehicle dynamics expertise. Such simplifications can significantly limit performance, especially in high-dynamics scenarios or under varying grip conditions.

2.1 State-of-the-art

This section reviews the state of the art in vehicle modeling for autonomous driving, analyzing the most commonly adopted formulations and discussing their results.

2.1.1 Models survey and benchmark

A comprehensive study presented in [13] analyzes the effect of model fidelity on trajectory optimization for autonomous vehicles, comparing models from simplified point-mass to detailed double-track formulations across varying friction conditions. Results show that while complex models yield more physically accurate trajectories, simpler ones can still generate effective reference paths. The study in [14] benchmarks the impact of various combined-slip tire models on trajectory, yaw rate, wheel forces, and maneuver times in a hairpin turn. Using a single-track vehicle model, the analysis compares friction ellipse and weighting-function tire formulations in isotropic and non-isotropic configurations. Results indicate that a few-state single-track model can replicate experienced driver behavior, suggesting that relatively simple models suffice for optimization-based open-loop trajectory planning. The study in [15] examines how simulation fidelity affects the evaluation of closed-loop autonomous racing controllers. By comparing vehicle models of varying complexity (including tire formulations, suspension, and actuator dynamics) the authors show that accurate tire modeling and proper parametrization are more critical than including every vehicle dynamic detail. Overly simplified models, such as a single-track with linear tires, diverge significantly from real-world behavior, particularly at high accelerations. The study in [16] compares kinematic and dynamic single-track models with linear tire formulations in urban autonomous driving scenarios using an MPC framework. The kinematic model performs well at low speeds, including stop-and-go maneuvers, offering

high computational efficiency. However, at higher speeds, path-tracking errors increase, making the dynamic single-track model safer and more suitable.

2.1.2 Modeling techniques for planning and control

Point-mass formulations. In [27] a robust MPC concept for autonomous racing is presented, using a friction-limited point-mass model and constraint-tightening tube-MPC approach for trajectory optimization, achieving high speed performance and dynamic overtaking maneuvers with a focus on smooth control actuation. In [28], a point-mass vehicle model using the GG-v diagram is employed for online path-planning, enabling deviations from the global racing line for obstacle avoidance. The GG-v, derived from a double-track vehicle model with Pacejka tires formulation, defines acceleration limits. Results show that the online trajectory accuracy is comparable to that of an offline trajectory generator. In [29], a hierarchical planning-control framework for autonomous racing is presented. The trajectory planner uses a data-driven GG-v polytopic vehicle model to generate feasible, near-optimal trajectories in real time, while the controller employs a neural network-based feedback-feedforward structure to track the trajectory and correct execution errors. Both layers are fully data-driven, trained on open- and closed-loop data. Results show lap times only few tenths slower than optimal offline minimum-lap-time solutions, highlighting the effectiveness of learned GG-v constraints for real-time trajectory optimization. Note that, despite the very simple point-mass model employed in these works, the results are highly promising (even though many are derived from simulation). This is primarily due to the parametrization methodology used to derive the dynamic constraints (higher-fidelity vehicle models and data-driven approaches).

Kinematic and linear single-track formulations. In [24] is analyzed the limitations of the kinematic bicycle model by comparing it against a high-fidelity 9 Degree of Freedom (DoF) reference model, demonstrating its shortcomings in a path-planning application. Their results showed that, while the kinematic formulation provides satisfactory accuracy for low-speed maneuvers, it becomes unreliable as lateral acceleration exceeds approximately 0.5 g, primarily due to the neglect of tire slips. As another example, a hybrid approach is proposed in [23], where path tracking performance is achieved at both low and high speeds by integrating a kinematic MPC with a feedback controller (PID) acting on the yaw rate. Additionally, a vehicle sideslip angle compensator is employed, derived from a linear single-track model. In [30], a non-optimization-based planner and controller are designed around a shared representation of the vehicle's dynamic limits. A linear single-track model is employed for path-tracking controller, while the planner generates reference trajectories that explicitly respect these limits. This tight coupling prevents the exceeding of nonlinearities, allowing the controller to achieve satisfactory path-tracking performance until operating in linear tires region. In [25], the controller employ a linear single-track model for path-following purposes in autonomous racing scenarios. The authors presented the limitations of using a purely linear representation of the tires, also highlighting the importance of online tire parameter adaption. In [26], a linear MPC is implemented for lateral path-tracking. While this approach proved effective across a range of scenarios, it exhibits inherent limitations that can potentially lead to vehicle instability under certain conditions. Future work is expected to focus on a coupled MPPI controller that explicitly accounts for the nonlinear vehicle dynamics.

Classic single-track formulation. In [17], a hierarchical controller for autonomous racing uses the same vehicle model in a two level optimization framework for motion

planning. The high-level controller, which is used as offline path generator, computes a trajectory that minimizes the lap time, and the low-level controller tracks the computed trajectory online. The framework relies on a nonlinear single-track model with Pacejka magic formula tire's representation. A very similar model is used in [31] within a Randomized Model Predictive Controller (RMPC). The controller drives a Dallara Super Formula in the Yas Marina F1 circuit, achieving a lap-time within 10% of the human driver while maintaining the maximum lateral tracking error around 1 meter. A comparable model is pushed to its limits in a virtual environment by [32], where an Adaptive Learning-based Model Predictive Control (ALMPC) strategy is proposed for drift vehicles. A similar configuration is adopted in [18], where a Nonlinear Model Predictive Control (NMPC) is used to stabilize and maintain a vehicle at high sideslip angles (30 deg) during automated drifting. In a high-fidelity simulation environment, the controller achieved a lateral deviation lower than 1 meter. In [33], a single-track model incorporating the Pacejka tire formulation for pure lateral force estimation is employed to develop a real-time NMPC strategy for direct yaw moment control. The approach enhances vehicle stability while maintaining low computational overhead. In [20], a single-track model incorporating longitudinal weight transfer and lateral brake distribution was used to control a Lexus LC 500 at approximately 95% of the maximum tire force within an NMPC framework. The model was also integrated into an Unscented Kalman Filter (UKF) to estimate tire-road friction online, enhancing overall model accuracy. With this adaptive approach, the system achieved an average lateral error of 0.43 m and a peak error of 0.77 m. Similar models are reported in [34–38].

Enhanced single-track formulation. In several studies on autonomous drifting, the conventional single-track model is often enhanced by adding the rear (driving) wheel angular velocity as an additional DoF. This provides a feedback signal on the slip ratio, improving control performance and vehicle stability. In [22], such dynamics proved essential for safely performing drift maneuvers, achieving lateral errors within 50 cm during transients and under 25 cm in steady state. The model was further refined in [21] by adapting the Fiala tire model to account for thermodynamic effects, improving robustness across varying tire temperatures. The authors note that these enhancements can increase numerical instability at low speeds due to higher system stiffness.

Given the simplifying assumption, the single-track formulation cannot capture more complex axle dynamics that involve interactions between wheels on the same axle, such as limited-slip differentials (LSD) or active torque vectoring management. In [19], a tricycle model is developed, including load transfer effects, to explore torque vectoring in autonomous drifting vehicles. The proposed approach enables tracking a predefined path with a desired sideslip angle and speed using a combination of feedforward action and Linear Quadratic Regulator (LQR) control, achieving steady-state lateral errors within 20 cm (up to 50 cm in the most demanding sections). In [39], a Nonlinear Model Predictive Contouring Controller is developed for obstacle avoidance, leveraging torque vectoring to stabilize and control the vehicle during evasive maneuvers at the handling limit. This work employs a nonlinear double-track model with an extended Fiala tire model to enhance predictive accuracy.

2.1.3 Modeling techniques for state estimation

Accurately estimating the lateral velocity, and consequently the vehicle sideslip angle, is a key aspect in autonomous driving. This information supports tasks such as real-time estimation of the tire-road friction coefficient [40–43], enabling adaptive vehicle models, and

providing feedback for model-based controllers, enhancing their performance, as demonstrated in [5]. The survey presented in [44] offers a comprehensive analysis of sideslip angle estimation techniques, emphasizing both kinematic and dynamic vehicle models. It also explores alternative approaches, such as black-box models (e.g., neural networks), for vehicle dynamics representation. Out of 120 works analyzed, 70 utilize model-based solutions, with algorithm performance closely linked to the accuracy of the underlying vehicle model. In [45] a hybrid sideslip angle estimator combines a model-based UKF with a data-driven Convolutional Neural Network (CNN). The solution employs a single-track model using a nonlinear Pacejka tire model. In [46], a more sophisticated modeling approach is presented, combining kinematic and dynamic representations. The dynamic part employs a UKF estimator alongside a double-track vehicle model and a modified Dugoff tire model. Simpler models are also considered, like in [47], which employs an Extended Kalman Filter (EKF) with a point-mass motion model, extended to account for road inclination and banking, enhancing the estimation of vehicle sideslip.

2.1.4 Data-driven and Neural Network Approaches

When the system is too complex to model analytically or its parameters are uncertain, data-driven approaches can be employed to estimate physical behavior, identify parameters, compensate for inaccuracies in simplified models, or mitigate issues such as numerical instability, including stiff system dynamics. Recent studies also show that these methods can enhance controller prediction in extreme conditions (e.g., icy roads [48] or drifting maneuvers [49]) without fully modeling complex vehicle dynamics.

Several works have explored hybrid and neural network approaches in vehicle modeling. For example, [50] introduces a Physics-Informed Neural Network (PINN) to estimate vehicle dynamics parameters within physical limits, enhancing the reliability of purely data-driven approaches. In [49], neural ordinary differential equation (ODE)-based tire models enable high-performance autonomous drifting with minimal data, demonstrating improved tracking accuracy and smoother control inputs compared to traditional models such as Fiala and Pacejka. Reference [48] presents a neural network integrated into an MPC (NNMPC); while path-tracking performance is comparable in high-friction scenarios, the NNMPC outperforms the physics-based MPC on icy surfaces. The hybrid model in [51] combines an analytic single-track formulation with a Long Short-Term Memory (LSTM) network to compensate for modeling inaccuracies, improving long-term predictions. Similarly, [52] integrates a neural differential equation into a single-track model, showing that a small neural network with limited training data can significantly enhance model accuracy over purely physics-based formulations.

In [8], in collaboration with TU Graz, a racing simulation platform based on Assetto Corsa was developed to test, validate, and benchmark autonomous driving algorithms, including both reinforcement learning (RL) and classical Model Predictive Control (MPC), in realistic and challenging scenarios. This work also provided several benchmarks and a comprehensive human driver dataset. While this thesis focuses on MPC-based solutions due to their practical efficiency and robustness in real-world racing applications, the research group remains actively engaged with neural and RL-based methods, studying their potential and limitations in autonomous racing contexts.

Despite their potential, data-driven and neural network approaches have so far been tested primarily in simulation or limited, controlled real-world demos. Hybrid approaches, combining interpretable physical models with data-driven components, therefore remain one of the most promising research directions for reliable and high-performance autonomous vehicle control.

2.2 Critical Summary

The state-of-the-art analysis confirms that model-based solutions remain the dominant approach for planning and control in autonomous racing and, more generally, for driving at the limits of tire friction. While simpler models can be effective for specific tasks such as trajectory generation or minimum lap-time optimization [13, 15], their validity often relies on strong assumptions, offline calibration, or information derived from more complex models and data-driven techniques [28, 29]. As a result, their applicability is frequently limited when deployed within closed-loop, real-time autonomous driving systems.

In practice, purely simplified formulations (such as kinematic or linear single-track models) exhibit clear limitations in high-dynamics scenarios, where tire saturation dominate the vehicle response [16, 24–26]. The nonlinear single-track model has emerged as the most widely adopted compromise between fidelity and computational tractability, with numerous extensions aimed at capturing additional physical effects [17, 20, 33]. Typical implementations rely on established tire models such as Fiala [19, 22], Pacejka [3, 17], or Dugoff [46, 53], sometimes augmented through tricycle or double-track formulations to improve accuracy under extreme conditions [6, 19, 39].

Although closed-loop path-following performance is frequently reported with lateral errors below one meter, and in some cases between 25 and 50 cm, these results are often achieved within simulation environment, or under carefully tuned conditions and do not necessarily translate to robust system-level performance across varying operating regimes. Moreover, increasing model complexity often comes at the cost of reduced robustness, higher calibration effort, and limited real-time feasibility, particularly when planning and control are treated as loosely coupled problems.

More recently, there has been a growing interest in neural and hybrid approaches. While these methods have shown promising results (sometimes even outperforming human drivers in terms of lap time [54–56]) their performance is frequently demonstrated in arcade-style simulators or task-specific settings. In addition, they often rely on implicit or partially learned physical representations, which can limit interpretability, robustness, and systematic integration into complete autonomous driving architectures. These limitations highlight the ongoing need for motion planning and control frameworks that prioritize architectural coherence, physical consistency, and real-time applicability, while remaining robust across a wide range of driving conditions.

Chapter 3

Simplified Vehicle Models

3.1 Models overview

This section provides an overview of the analyzed simplified models (shown in Fig.3.1), all of which feature three degrees of freedom, neglecting pitch, roll, and vertical dynamics. The equations of motion of each model are outlined, and expressed in state-space form. A list of simplifying assumptions is premised, and a summary table highlighting the minimum set of parameters required for each model's definition is also included.

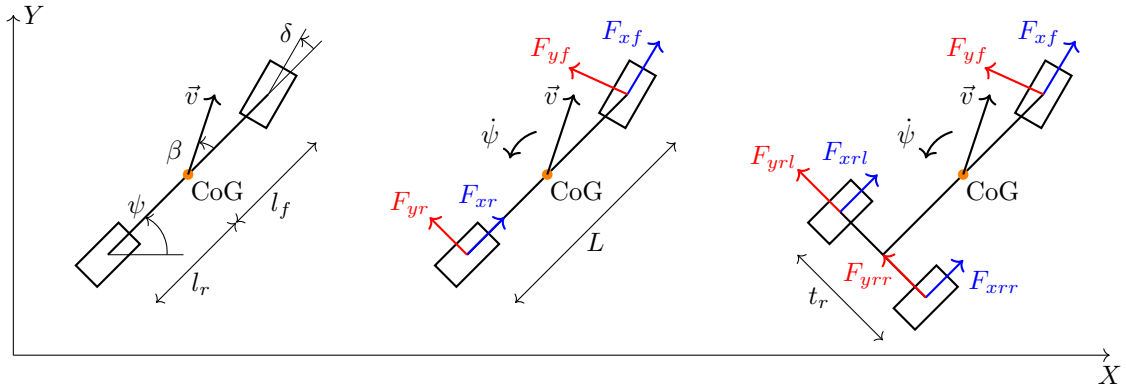


Figure 3.1: Kinematic single-track (left), dynamic single-track (middle), and dynamic tricycle model (right) representations with common reference frame.

3.1.1 Kinematic single-track model

This model relies exclusively on the vehicle geometry to describe the velocity field. In our formulation, the vehicle state is given by $\mathrm{x} = [X; Y; \Psi]$, where X, Y and Ψ represent the longitudinal and lateral position of the center of gravity (CoG), and the heading (yaw) angle, as shown in Fig.3.1 (left). The input is $\mathrm{u} = [\delta; v_x]$, where δ is the steering angle and v_x is the longitudinal velocity (i.e. ahead velocity component). The governing equations read:

$$\dot{X} = v\cos(\psi + \beta)\,, \tag{3.1a}$$

$$\dot{Y} = v\sin(\psi + \beta)\,, \tag{3.1b}$$

$$\dot{\psi} = \frac{v\cos\beta \cdot \tan\delta}{L}\,, \tag{3.1c}$$

where $L = l_f + l_r$ is the wheelbase, $v_x = v\cos\beta$, $v_y = v\sin\beta$, and β is the kinematic slip angle, defined as:

$$\tan\beta = \tan\delta \cdot \frac{l_r}{L}. \tag{3.2}$$

The model can be further enhanced by including the stability factor:

$$K = \frac{m}{L^2}\left(\frac{C_r l_r - C_f l_f}{C_f C_r}\right), \tag{3.3}$$

where m is the mass of vehicle, and C_f and C_r represent the front and rear cornering stiffness of the tires. Its contribution can be forced as a "correction" factor in the yaw angle derivative in Eqs.(3.1), resulting in:

$$\dot{\psi} = \frac{v_x \cdot \tan\delta}{L} \cdot \frac{1}{1 + K \cdot v_x^2}. \tag{3.4}$$

This configuration is hereafter referred to as the "enhanced kinematic model". The model is valid only under kinematic steering conditions, making it suitable for low speed scenarios (negligible slip angles). Its strength is the small number of parameters required for its definition, as highlighted in Tab.3.1.

Parameter	**Symbol**	**Value**
Front wheelbase	l_f	1.7 m
Rear wheelbase	l_r	1.3 m
Mass*	m	750 kg
Front cornering stiffness*	C_f	84647 $\frac{\text{N}}{\text{rad}}$
Rear cornering stiffness*	C_r	210620 $\frac{\text{N}}{\text{rad}}$

Table 3.1: Key parameters for the kinematic single-track model.

3.1.2 Dynamic single-track model

Motion is determined by the interaction of tangential forces between the tires and the ground, where F_{xf}, F_{yf}, F_{xr} and F_{yr} represent the longitudinal and lateral components at the front and rear axles (Fig.3.1, middle). The resulting EOM are:

$$\dot{v}_x = \frac{1}{m}\left(F_{xr} - F_d - F_{yf}\sin\delta + F_{xf}\cos\delta + m v_y r\right), \tag{3.5a}$$

$$\dot{v}_y = \frac{1}{m}\left(F_{yr} + F_{yf}\cos\delta + F_{xf}\sin\delta - m v_x r\right), \tag{3.5b}$$

$$\dot{r} = \frac{1}{J_z}\Big[l_f\left(F_{yf}\cos\delta + F_{xf}\sin\delta\right) - l_r\, F_{yr}\Big], \tag{3.5c}$$

where $r = \dot{\psi}$, $F_d = \frac{1}{2}\cdot\rho\cdot v_x^2\cdot C_x\cdot S$ represent the aerodynamic drag, and J_z is the vehicle moment of inertia respect to Z axis. The vehicle state is given by x = $[v_x;\, v_y;\, r]$, while the input vector adopted in the analysis is u = $[\delta;\, a_x]$, where δ denotes the steering angle, and a_x the longitudinal acceleration. Note that in Eq.(3.7a), m represents the equivalent mass, accounting for the inertial effects of the rotating elements of the vehicle.

The classic single-track model can be extended by including tire rotational dynamics, which is generally used in the literature for drifting applications [57], where the rear axle reaches its friction limit due to engine power. An additional state must be incorporated into the system of Eqs.(3.5), applying a moment equilibrium on the rear axle as follows:

$$\dot{\omega}_r = \frac{1}{J_r}\left(T - R_l \cdot F_{xr}\right), \tag{3.6}$$

where $\dot{\omega}_r$ denotes the angular acceleration of the rear axle, F_{xr} corresponds to the longitudinal force at the axle (computed using the chosen tire model), T represents the axle driving torque, R_l is the loaded tire radius and J_r is the powertrain reduced inertia, including the engine, gearbox, axles, and wheels.

For a more general application (rather than drifting), Eq.(3.6) can also be extended to the front axle by introducing an analogous rotational dynamics equation, where T represents the braking torque and J_f denotes the equivalent rotational inertia of the front rotating components. Given the application, our formulation neglects tire rotational dynamics.

The core of the dynamic single-track model lies in the tire model used to compute the aforementioned forces. Depending on its complexity, the model exhibits varying levels of accuracy and applicability. Tire models and their characteristics are detailed in Section 3.2.

Parameter	**Symbol**	**Value**
Mass	m	750 kg
Inertia	J_z	700 kg m^2
Drag coefficient	C_x	1.2
Downforce coefficient	C_z	3.9
Front projected area	S	1.0 m^2
Powertrain reduced inertia*	J_r	2.5 kg m^2
Wheel loaded radius*	R_l	0.3 m

Table 3.2: Additional parameters for the dynamic single-track model, extending those defined in Tab.3.1.

3.1.3 Tricycle model

A scheme of the model is shown in Fig.3.1 (right), and its main parameters are listed in Tab.3.3. This formulation extends the previous one by accounting for the differential contribution:

$$\dot{v}_x = \frac{1}{m}\left(F_{xr} - F_d - F_{yf}\sin\delta + F_{xf}\cos\delta + mv_y r\right), \tag{3.7a}$$

$$\dot{v}_y = \frac{1}{m}\left(F_{yr} + F_{yf}\cos\delta + F_{xf}\sin\delta - mv_x r\right), \tag{3.7b}$$

$$\dot{r} = \frac{1}{J_z}\Big[M_{\text{diff}} + l_f\left(F_{yf}\cos\delta + F_{xf}\sin\delta\right) - l_r F_{yr}\Big]. \tag{3.7c}$$

The differential yaw moment is calculated as $M_{\text{diff}} = (F_{xrr} - F_{xrl}) \cdot \frac{t_r}{2}$, with t_r denoting the rear track width, and the rear lateral force $F_{yr} = F_{yrl} + F_{yrr}$.

The yaw moment can be estimated using different approaches depending on the vehicle's differential configuration. In this thesis, we present two methodologies, each tailored to a specific mechanical setup: a fully locked differential, which is rarely used in racing or urban vehicles, and a Salisbury-type limited-slip differential (LSD). Both methods provide a simplified estimate of the yaw moment without introducing wheel rotational dynamics. Detailed formulations for each approach are provided in the appendix, Sec.A.2 and Sec.A.3. Alternatively, modeling the rear wheels as independent rotational bodies allows different slip and force at each tire, providing a more detailed rear-axle representation.

This model overcomes the main limitation of the single-track formulation, which assumes an open differential. By accounting for differential effects, the "tricycle" model can distinguish between tight and wide radius turns while using a single set of tire parameters. Its main drawback is the increased number of parameters, which complicates

Parameter	Symbol	Value
Rear Track	t_r	1.6 m
Front roll stiffness	K_{r_f}	1160000 N$\frac{\text{m}}{\text{rad}}$
Rear roll stiffness	K_{r_r}	1015000 N$\frac{\text{m}}{\text{rad}}$
Differential preload	M_0	80 Nm
Locking percentage (driving)	ϵ_d	29 %
Locking percentage (coasting)	ϵ_c	49 %
Longitudinal rear tire grip	D_{xr}	1.4

Table 3.3: Additional parameters for the "tricycle" model, extending those defined in Tab.3.2. Compared to the single-track model, suspensions and differential data are required.

the identification process, especially when suspension geometry or differential data are unavailable.

3.2 Tire Dynamics

Tires are primarily subjected to three force components: vertical load F_z, lateral force F_y, and longitudinal force F_x. Lateral forces are typically expressed as a function of the slip angles α through the approximated kinematic relationships:

$$\begin{aligned} \tan\alpha_f &= -\left(\frac{v_y + r\cdot l_f}{v_x}\right) + \tan\delta\,, \\ \tan\alpha_r &= \left(-\frac{v_y - r\cdot l_r}{v_x}\right). \end{aligned} \tag{3.8}$$

Longitudinal forces are commonly expressed as a function of the slip ratio κ:

$$\kappa = \frac{\omega\cdot R - v_x}{v_x} \tag{3.9}$$

where ω is the angular velocity of the wheel rim and R the wheel rolling radius. Generally, the slip ratio κ is not explicitly considered unless additional degrees of freedom are introduced. The relationship between slips and tire forces varies depending on the selected tire model, which can be either linear or nonlinear.

3.2.1 Linear tire model

The simplest dynamic single-track model assumes lateral tire forces to be a linear function of the slip angles:

$$\begin{aligned} F_{y_f} &= C_f\cdot\alpha_f \\ F_{y_r} &= C_r\cdot\alpha_r \end{aligned} \tag{3.10}$$

The common formulation treats longitudinal velocity v_x as an input and approximates the tangent function linearly (small slip angles assumption). By substituting Eqs.(3.8) into Eqs.(3.10), and subsequently replacing the result in Eqs.(3.5), the full set of equations of motion can be linearized as:

$$\dot{\mathbf{x}} = A\cdot\mathbf{x} + B\cdot\mathbf{u}, \tag{3.11}$$

where

$$A = \begin{bmatrix} -\frac{C_f+C_r}{mv_x} & -\frac{C_f l_f - C_r l_r}{mv_x} - v_x \\ -\frac{C_f l_f - C_r l_r}{J_z v_x} & -\frac{C_f l_f^2 + C_r l_r^2}{J_z v_x} \end{bmatrix} \quad B = \begin{bmatrix} \frac{C_f}{m} \\ \frac{C_f l_f}{J_z} \end{bmatrix} \tag{3.12}$$

This model is simple to implement and easy to parametrize (Eqs.(3.12) and Tab.3.2). Compared to the kinematic single-track, it captures steady-state nose-in and nose-out, improving accuracy, but it cannot reproduce highly nonlinear tire behavior and is inaccurate near the limits of tire performance.

3.2.2 Nonlinear tire model

Very often a nonlinear tire model is necessary for an acceptable description of vehicle dynamics. Unfortunately, analytical physical tire models lack accuracy, hence empirical or semi-empirical formulations are widely adopted. The existing approaches rely on the friction law, which describes the relationship between normal and tangential forces as:

$$F_{\max} = \mu_t \cdot F_z \,, \tag{3.13}$$

where μ_t is the equivalent friction coefficient, which depends on both the static and dynamic friction coefficients, and defines the total grip available at the contact patch.

Among the widely adopted tire models is the Pacejka model [58], which accurately fits experimental tire data across a wide range of operating conditions using simple yet effective mathematical functions. Its simplified form is:

$$\begin{aligned} F_x &= F_z \cdot D_x \sin\Big(C_x \tan^{-1}(B_x\,\kappa) - E_x(B_x\,\kappa - \tan^{-1}(B_x\,\kappa))\Big) \\ F_y &= F_z \cdot D_y \sin\Big(C_y \tan^{-1}(B_y\,\alpha) - E_y(B_y\,\alpha - \tan^{-1}(B_y\,\alpha))\Big) \end{aligned} \tag{3.14}$$

The coefficients (non-physical parameters) define the curve's stiffness, normalized peak force, and shape. Lateral and longitudinal forces from Eqs.(3.14) can be combined to ensure that the total tire force remains within the grip limit (Eq.(3.13)), effectively capturing the balance between lateral and longitudinal contribution under combined-slip conditions [14].

In this thesis, the combined-slip effect is considered through a weighting factor [3], defined as:

$$G_y = \cos\Big(\arcsin\big(\frac{F_x}{F_{\max}}\big)\Big),, \tag{3.15}$$

where $F_{\max} = F_z \cdot D_y$. To prevent singularities, the force fraction is clipped such that $\frac{F_x}{F_{\max}} < 1$. The resulting combined lateral force is then given by $F_y = G_y F_{y,lat}$, where $F_{y,lat}$ is the pure lateral force computed using the lateral tire model (Eqs. (3.14)).

The model in Eqs. (3.14) can be extended to include a force offset and a more realistic parabolic decay of the force peak with vertical load. This dependency is essential to accurately capture vehicle behavior across different speed ranges, particularly when aerodynamic effects and load transfer are significant. The extended formulation is:

$$\begin{cases} F_y &= F_z \cdot \Big(S_v + D_y \sin\big(C_y \tan^{-1}(B_y\,\alpha_{sh}) - E_y(B_y\,\alpha_{sh} - \tan^{-1}(B_y\,\alpha_{sh}))\big)\Big), \\ D_y &= D_{y0} + D_{y1} \cdot \frac{F_z}{F_N} + D_{y2} \cdot \frac{F_z^2}{F_N}, \\ \alpha_{sh} &= \alpha + S_h, \end{cases} \tag{3.16}$$

where F_N is the nominal axle load. The coefficients D_{y1} and D_{y2} capture the linear and quadratic trends of grip, while S_v and S_h represent vertical and horizontal shifts.

3.2.3 Vertical load

The axle vertical loads can be computed as follows:

$$F_{\mathrm{z}_f} = F_{\mathrm{z}_{0_f}} + F_{\mathrm{z}_{\mathrm{d}_f}} + F_{\mathrm{z}_{\theta_f}} - \Delta F_{\mathrm{zx}}, \tag{3.17a}$$

$$F_{\mathrm{z}_r} = F_{\mathrm{z}_{0_r}} + F_{\mathrm{z}_{\mathrm{d}_r}} + F_{\mathrm{z}_{\theta_r}} + \Delta F_{\mathrm{zx}}, \tag{3.17b}$$

accounting for static weight distribution, aerodynamic downforce, and longitudinal load transfer effects, as detailed in [59]. The contribution of road banking to the vertical load is computed as $F_{\mathrm{z}_\theta} = m\, a_y \sin(\theta)$, where θ denotes the road banking angle. This additional load is distributed between the front and rear axles according to the longitudinal position of the vehicle center of gravity. Lateral load transfer, essential for defining the "tricycle" model rear axle, requires a non-planar vehicle representation. At low lateral accelerations, a simplified double-track model with constant track width, fixed roll axis, and linear load transfer proportional to tire forces is sufficient:

$$\Delta F_{\mathrm{zy}_f} = F_{y_f} \frac{h_{\mathrm{frc}}}{t_f} + K_{f_{\mathrm{eq}}} \frac{M_{\mathrm{roll}}}{t_f}\,, \tag{3.18a}$$

$$\Delta F_{\mathrm{zy}_r} = F_{y_r} \frac{h_{\mathrm{rrc}}}{t_r} + K_{r_{\mathrm{eq}}} \frac{M_{\mathrm{roll}}}{t_r}\,, \tag{3.18b}$$

where $K_{f_{\mathrm{eq}}} = \frac{K_{r_f}}{K_{r_f}+K_{r_r}}$ and $K_{r_{\mathrm{eq}}} = \frac{K_{r_r}}{K_{r_f}+K_{r_r}}$, while K_{r_f} and K_{r_r} represent the front and rear rolling stiffness, and h_{frc} and h_{rrc} are the heights of front and rear roll center, respectively. Note that, for high lateral acceleration, a different method should be used [60].

3.3 Simulation and Numerical Stability

This section presents simulation results for basic standard maneuvers, such as steering-pad and straight-line driving, aimed at assessing the numerical stability of the integration schemes. This analysis serves as a fundamental preliminary step before deploying the model within any planning or control algorithm. The results are obtained using the following numerical scheme:

$$\mathbf{x}_{t+1} = \Phi_t(\mathbf{x}_t, \mathbf{u}_t, h), \tag{3.19}$$

where Φ_t denotes the selected numerical integrator. In particular, the explicit Euler method and the fourth-order Runge-Kutta scheme are considered. The integration timestep is varied in the range $h \in [0.001, 0.04]$ s. Fixed-step explicit integration methods are adopted, as they are generally preferred in real-time applications due to their predictable computational cost. However, an inappropriate choice of the timestep may introduce significant numerical errors or even lead to unstable behavior, as discussed in the following sections.

3.3.1 Lateral dynamics

Stability can be evaluated by verifying whether the eigenvalues of the system matrix lie within the absolute stability region of the chosen integrator and timestep [61]. Analyzing the matrix A in Eqs. (3.12) shows that lower longitudinal velocity and higher cornering stiffness produce eigenvalues with larger absolute real parts, making the system stiffer.
Fig.3.2 illustrates the pole locations of the dynamic models during a steering pad maneuver (constant steering angle with increasing velocity). Poles cannot be determined for the kinematic model, as it does not represent a dynamic system. The left-side plots highlight

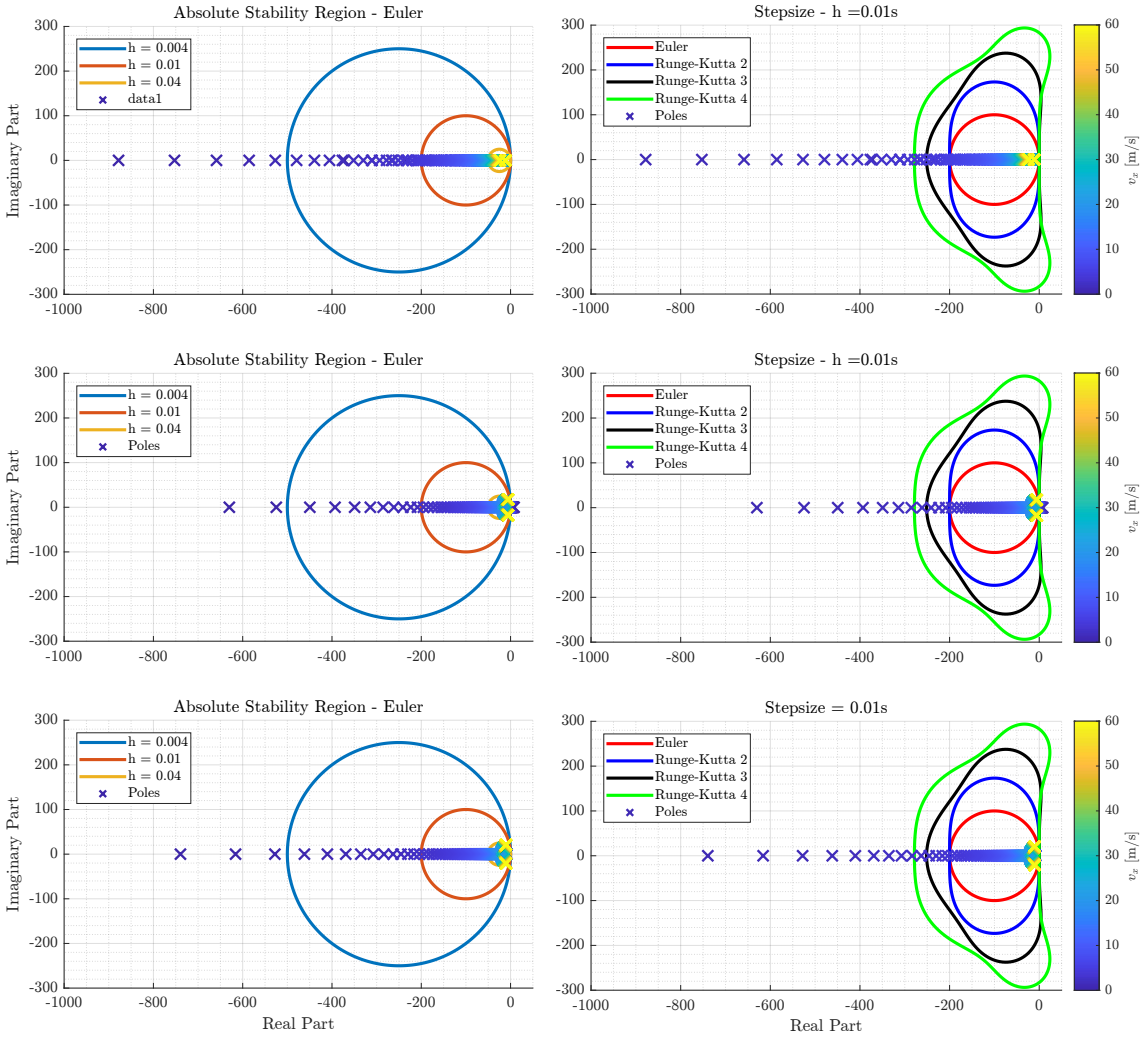


Figure 3.2: Poles in the Gauss plane, from top to bottom: linear, nonlinear single-track, and tricycle model. Left column: Euler stability for different stepsizes; right column: absolute stability regions of four common integrators ($h = 0.01\,\mathrm{s}$).

the absolute stability region of Euler integration for different stepsizes: smaller timestep enlarge the stability region, allowing integration of stiff dynamics. In particular, racing tires produce a highly stiff system, requiring a timestep of approximately $h = 0.001\,\mathrm{s}$ to maintain stability across all speeds. Higher-order integrators maintain stability and accuracy across a wide range of operating conditions, from stiff dynamics at low speeds to oscillatory behavior at higher speeds, as shown in the right-side plots. Compared to the nonlinear model, the linear single-track formulation is stiffer but more stable at higher speeds. Compared to the nonlinear model, the linear single-track formulation is stiffer but more stable at higher speeds. Including an LSD model (tricycle) further increases system stiffness, though less than the linear formulation, while enhancing stability at high velocities [6]. Incorporating longitudinal slip, characterized by fast dynamics, requires extremely small timesteps for numerical stability [13]. Simulations using rotational wheel dynamics in the state are proposed in Sec.3.3.2.

Reducing the timestep results in a rapidly growing (exponential) computational cost, as shown in Tab.3.4 and Fig.3.3. This is especially critical in MPC, where the optimization

Stepsize	Kinematic	Enhanced	Linear	Nonlinear	Tricycle
$h = 0.01$	0.0185	0.0249	0.0253	0.0283	0.0406
$h = 0.001$	0.149	0.2376	0.5664	0.8446	1.0157

Table 3.4: Summary table of the time performance (s) achieved by all the models.

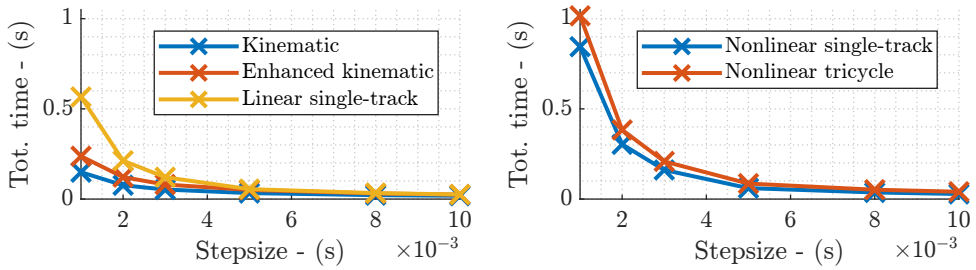


Figure 3.3: Computation time required for a 10 s steering-pad simulation using the fourth-order Runge-Kutta integration scheme.

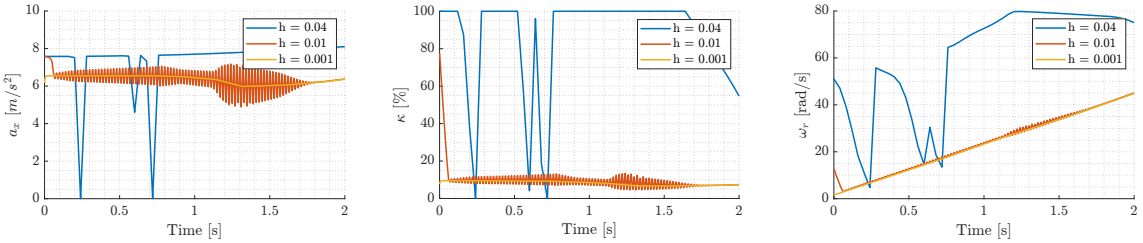
process involves many iterations, further amplifying the computational effort.
In conclusion, the choice of integration scheme and timestep represents a trade-off between computational resources, minimum speeds expected in the maneuver, and vehicle parameters, with particular emphasis on cornering stiffness.

3.3.2 Longitudinal dynamics

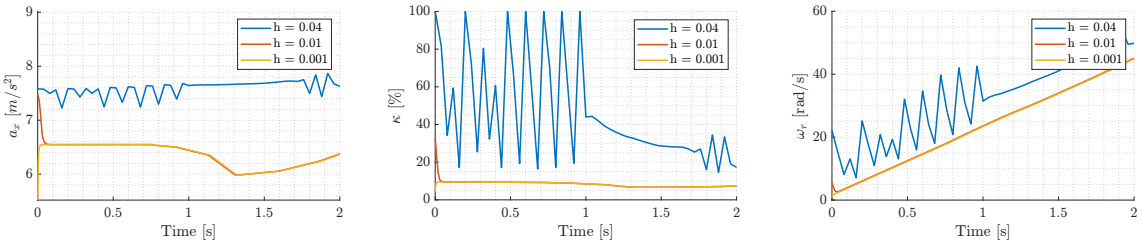
Accurate simulation of longitudinal vehicle dynamics fundamentally requires the inclusion of tire rotational dynamics. Without this contribution, it is not possible to compute wheel slip ratios, and consequently to obtain meaningful feedback on the longitudinal friction limits of the tires. In some applications, where slip ratios remain small and the vehicle operates far from traction or braking limits, neglecting tire rotational dynamics may be acceptable. However, this assumption breaks down in high-performance driving scenarios, where accurate representation of longitudinal tire behavior becomes critical.

Modeling these dynamics introduces several practical challenges. First, additional parameters are required, many of which are difficult to obtain or reliably estimate, such as powertrain inertias (e.g., engine, gearbox, and driveshafts), effective tire radius (which is function of the velocity) and detailed longitudinal tire characteristics. Second, tire rotational dynamics evolve on significantly faster time scales than the vehicle body dynamics. As a result, their numerical integration typically requires very small timesteps to ensure stability and accuracy. This substantially increases the computational burden of the simulation (Tab.3.4), limiting applicability in real-time planning and control frameworks.
We performed pure longitudinal dynamics simulations to evaluate the stability of the single-track model including the tire rotational degrees of freedom. The results are shown in Fig. 3.4. Starting from zero speed, a constant throttle input is applied to the model. This command generates positive engine torque, which produces rear wheel rotation, slip ratio, force and consequently longitudinal motion.

The upper subplot (Fig. 3.4a) shows the simulation using the explicit Euler scheme. In this case, a very small timestep is required to ensure numerical stability. Using the fourth-order Runge-Kutta (Fig.3.4b) scheme improves the results significantly, allowing stable integration even with a timestep of $h = 0.01$ s. However, at very low longitudinal velocities, the solution accuracy is limited, as convergence toward the correct value is noticeable. Larger timesteps, such as $h = 0.04$ s, which generally represent a good compromise for real-



(a) Explicit Euler simulation results.



(b) Fourth-order Runge-Kutta simulation results.

Figure 3.4: Comparison of longitudinal dynamics for different integration schemes and timesteps. The simulation with $h = 0.001\,\mathrm{s}$ is considered the reference. Deviations or oscillations from these values are interpreted as numerical errors.

time planning and control applications, are completely unsuitable for producing accurate and stable results.

Chapter 4

Multi-body Simulation Model

In contexts where on-track testing is limited and expensive, a high-fidelity simulator allows to virtually evaluate scenarios and control strategies, significantly accelerating the development process. In autonomous racing, a precise vehicle-dynamics simulator is crucial, as it provides reliable ground truth for validating algorithms prior to real-world deployment.

Accordingly, digital twins of the autonomous vehicles analyzed in this thesis were developed using Dymola, a modelica-based modeling and simulation environment widely used for multi-domain physical systems. This tool enables the creation of complex, high-fidelity multi-body models primarily through equation-based techniques. A key advantage is the availability of highly specialized, customizable libraries for race-car dynamics modeling, such as VeSyMA and Motorsport by Claytex [62–64], which provide template vehicle architectures that can be adapted to specific needs. These libraries greatly accelerate model development and enable the assembly of accurate, fully parameterizable vehicle models with minimal low-level implementation effort.

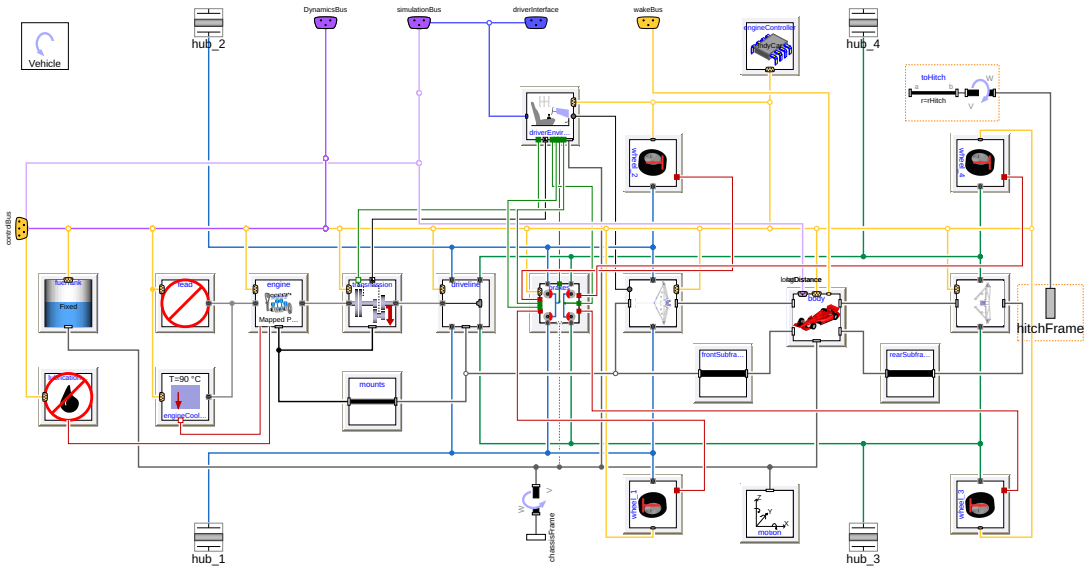


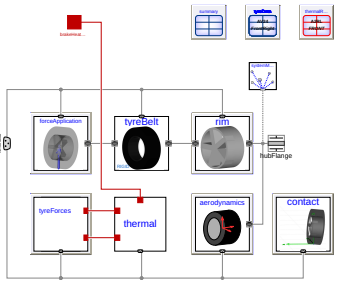
Figure 4.1: Top-level block diagram of the Dymola race car model, showing all main components and their interconnections. Customization includes: engine, driveline, suspensions, tires, brakes, suspensions, body. Detailed submodules view is proposed in Fig.4.2.

4.1 Model development

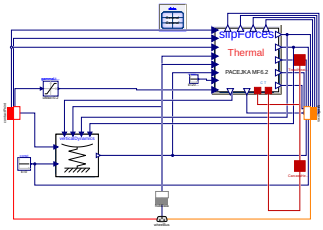
Dymola simulations were initially used offline to generate maneuvers that are difficult to reproduce on-track, particularly when tests were unsafe or impractical. This approach provided rich data for calibrating simplified models, including single-track representations, where accurate axle characteristic identification is critical. Subsequently, the simulations were used to derive simplified model sub-components (for example, the force distribution of a locked differential, the relationship between suspension travel and tire camber, or the banking contribution on tires vertical load). Finally, the high-fidelity model was exported as a Functional Mock-up Unit (FMU) and integrated into a custom software-in-the-loop simulator for real-time testing of the full autonomous stack, as detailed in Sec. 4.3.1.

4.1.1 Model architecture

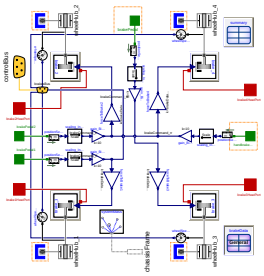
The Dymola vehicle model is structured as a set of interconnected blocks, as shown in Fig. 4.1, where each block (detailed in Fig. 4.2) represents a specific physical subsys-



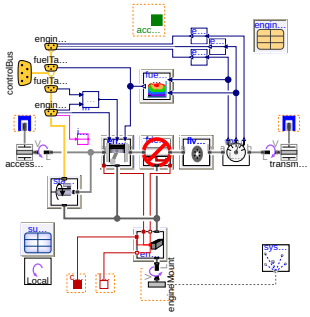
(a) *Pacejka MF 62*



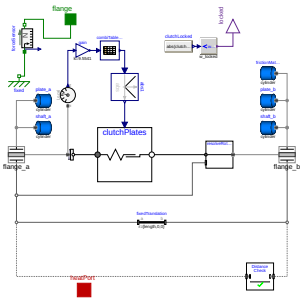
(b) *Tire Forces Thermal*



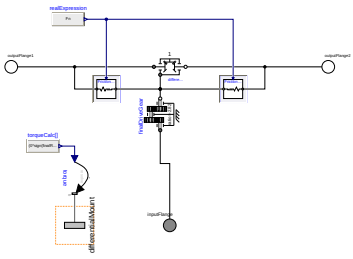
(c) *Four Brakes Thermal*



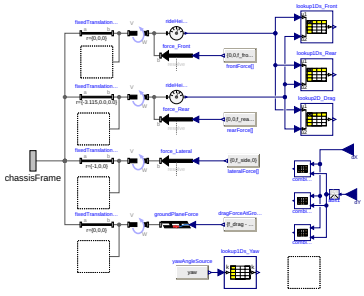
(d) *3D Engine Map*



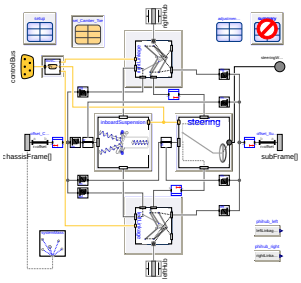
(e) *Centrifugal Clutch*



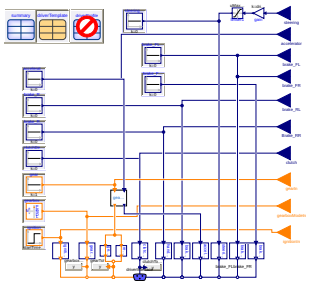
(f) *Differential (LSD)*



(g) *3D Aerodynamic*



(h) *Front Suspension*



(i) *Driver interface*

Figure 4.2: Overview of the main subsystems composing the vehicle model shown in Fig. 4.1.

tem of the modeled vehicle and contributes to reproducing its overall dynamic behavior. Particular attention has been given on the modeling of the following components:

- **Tires**: Tire forces are modeled using the Pacejka Magic Formula 6.2, with vertical load represented via a Kelvin-Voigt spring-damper system. Inflation pressure and camber angles can differ between left and right sides to account for oval-track setups. A thermal model captures variations in grip and stiffness due to temperature (Fig. 4.2a, Fig. 4.2b), as detailed in the following paragraphs.
- **Brake**: The original Motorsport brake block has been extended to support four independent brake commands (Fig. 4.2c). The standard brake model already includes a thermal formulation, and it has been further enhanced with a temperature-dependent pad–disc friction coefficient to account for varying operating conditions.
- **Engine**: A 3D map relating engine torque, speed, and throttle position was derived from engine test bench data and refined with experimental measurements (Fig. 4.2d). Rotational friction of components has been deactivated to reduce low-speed numerical instability.
- **Driveline**: Gear ratios, final drive, and shift times are based on telemetry and manufacturer data. A centrifugal clutch (Fig. 4.2e) engages progressively from 1400 rpm to full engagement at 1800 rpm, generating pedal forces comparable to a human driver.
- **Differential**: A custom model implements both a fully locked and a friction-based limited-slip differential (LSD). The fully locked configuration enforces a rigid connection between axles. The LSD, detailed in Fig. 4.2f, uses two clutches between the axles and the differential housing. The normal force acting on the clutches is computed from differential design parameters (e.g., ramp angles, diameters) to simulate locking behavior.
- **Aerodynamics**: Downforce is defined via maps that account for both magnitude and center-of-pressure position as a function of front and rear ride heights. Drag is modeled using a simplified C_x coefficient. Slipstream effects from nearby vehicles reduce drag and downforce accordingly (Fig. 4.2g).
- **Body**: Sprung and unsprung masses are modeled considering the center of gravity and cross-weight to match experimental static load balance data. The inertia matrix is defined as well.
- **Suspensions**: Double-wishbone geometry with a vertical anti-roll bar is modeled using the Claytex template. Right and left quarter-car assemblies are combined into a half-car model, including springs, rockers, shock absorbers, and the front steering system (Fig. 4.2h). Stiffness and nonlinear damping curves are defined for all components.
- **Actuator dynamics**: Steering and brake actuation include delays and friction effects, modeled via transfer functions identified from on-track data. These models are implemented outside the Dymola environment.
- **Driver**: A custom driver interface (Fig. 4.2i) implements gear control logic, including cut-off and throttle blip strategies, and converts actuator signals for autonomous operation. The driver can use external inputs via connectors or standard user-defined signals (ramp, sinusoid, time-table).

A more detailed overview is given in the following paragraphs.

Tire Thermal model [10]. To capture the temperature-dependent behavior of the tires, the main Pacejka parameters [58], particularly the peak force ($D_y(T)$) and the cornering stiffness ($K_y(T)$), are modeled as functions of the tread temperature.

For the lateral characteristics we define:

$$\begin{aligned} D_y(T) &= D_y\Big(1 + \frac{\delta \mu_y}{\delta T}(T - T_{ref})\Big), \\ K_y(T) &= K_y\Big(1 + \frac{\delta C_y}{\delta T}(T - T_{ref})\Big). \end{aligned} \tag{4.1}$$

where T is the tread temperature and T_{ref} is the reference ambient temperature. The variations of the peak friction coefficient μ_y and of the cornering stiffness C_y with respect to temperature ($\frac{\delta \mu_y}{\delta T}$ and $\frac{\delta C_y}{\delta T}$, respectively) are experimentally identified.

In this work, a simplified piecewise approximation is adopted: as temperature rises, grip increases (warm-up phase); within a given temperature range the grip is assumed constant (thermal window); beyond this range, grip degradation occurs due to overheating. Cornering stiffness decreases monotonically with increasing temperature. The same reasoning is applied to the longitudinal tire characteristics.

Eqs. (4.1) have been implemented as a custom block (named "*thermal*" in Fig. 4.2a) within the existing Dymola Motorsport tire model. The block integrates thermal components from the Modelica Standard Library: heat ports connect lumped masses to their thermal environment (air, ground, internal layers), and dedicated elements model the corresponding heat transfer. Heat generation is computed internally based on tire forces and slip levels. The original "*tireForces*" and "*slipForces*" blocks have been extended to:

1. include thermal inertia through additional lumped masses;
2. exchange forces and heat flow with the new "*thermal*" block;
3. use the temperature-dependent Pacejka parameters defined in Eqs. (4.1).

This integration results in a fully coupled thermo-mechanical tire model, capable of capturing the change in performance induced by temperature variations during high-dynamics racing simulations. The thermal model can be easily disabled, providing a less robust yet more deterministic simulation when required.

Brake Thermal Model The implemented braking system features a custom 2-node lumped parameter thermal model coupled with a data-driven friction map. To simulate the thermal gradient across the disc thickness without the computational cost of a Finite Element (FE) model, the disc mass is discretized into two distinct thermal nodes:

- **Surface Node:** Represents the outer friction ring. It is characterized by a lower thermal inertia, allowing it to react rapidly to the heat flux generated during braking events.
- **Core Node:** Represents the internal bulk material and ventilation vanes. It acts as a thermal capacity, exchanging heat via conduction with the surface node and dissipating energy via convection through the internal ventilation vanes.

The temperature evolution is determined by the energy balance of heat entering, stored, and leaving the system.

The governing equation for the surface temperature evolution is:

$$C_{\text{surf}} \cdot \dot{T}_{\text{surf}} = Q_{\text{in}} - Q_{\text{conv}} - Q_{\text{rad}} - Q_{\text{cond}}, \tag{4.2}$$

where C_{surf} is the heat capacity of the surface node, which is temperature-dependent, and T_{surf} is the disk surface temperature. The model accounts for three specific heat transfer mechanisms:

- **Conduction (Q_{cond}):** Heat flows between the Surface and Core nodes, driven by the thermal conductance of the material.
- **Convection (Q_{conv}):** Heat is dissipated to the surrounding air from both the external faces and the internal ventilation channels. The Heat Transfer Coefficients (HTC) are dynamically calculated based on the disc's angular velocity (ω) and vehicle speed, utilizing Nusselt number correlations for rotating disks.
- **Radiation (Q_{rad}):** Radiative heat loss to the environment is modeled using the Stefan-Boltzmann law, which becomes the dominant cooling factor at high temperatures typical of carbon discs.

The braking torque and the heat flux entering the system (Q_{in}) are governed by the pad-disk friction coefficient (μ_d). Unlike standard models that assume a constant coefficient, this implementation calculates μ_d dynamically as a function of the pad-disk contact pressure and the surface temperature, utilizing experimental data look-up tables.

Aerodynamic map. While aerodynamic effects are often simplified using constant drag and downforce coefficients, in reality they strongly depend on the vehicle's ride heights. Capturing this dependency is crucial to reproduce critical phenomena such as diffuser stall (particularly dangerous in high-speed corners or during hard braking) and variations in aero balance due to changes in rake angle. To model this accurately, a custom Dymola class was implemented, where front and rear aerodynamic coefficients are defined via lookup tables as functions of the vehicle's front and rear ride heights.

Slipstream effects, which reduce drag but can also decrease downforce, are particularly important in multi-agent simulations. The original Dymola block (Fig. 4.2g) was extended to account for these effects, using two additional tables that define drag and downforce reductions based on the relative position of an opponent vehicle. These inputs are provided via Real-type input connectors.

This approach enables the simulator to reproduce complex aero-mechanical interactions without relying on oversimplified constant-coefficient models.

Limited-Slip Differential (LSD) model. The vehicle's Salisbury-type LSD is implemented in Dymola using differential gears and controlled elasto-plastic friction elements. Although the Motorsport library includes a block for modeling friction-based differentials, its full parametrization requires detailed internal geometry and friction data, which are typically unavailable for prototype vehicles. Therefore, a simplified formulation is used, allowing partial locking under normal torque and full locking when the friction limit is exceeded.

Key parameters include the differential radius R_{diff}, ramp angles for acceleration (θ_{d}) and deceleration (θ_{c}), torque preload τ_{preload}, and locking percentages for driving (ϵ_{d}) and

coasting (ϵ_c) conditions. The torque applied to the differential flange is converted to an equivalent force on the ramps:

$$F_0 = \frac{\tau_{\text{diff}}}{2R_{\text{diff}}} \tag{4.3}$$

The normal force on the friction elements is computed as:

$$F_n = \max\left(\frac{|F_0|\cos\theta}{\sin\theta},\ \frac{\tau_{\text{preload}}}{\mu\, c_{\text{geo}}}\right), \tag{4.4}$$

where the ramp angle depends on the torque direction:

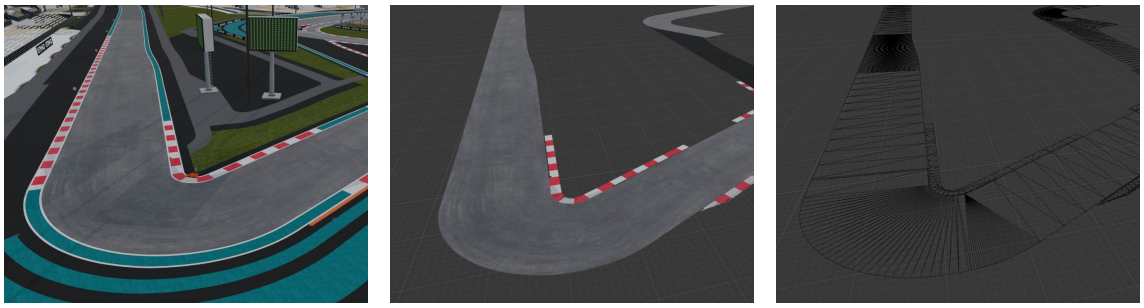
$$\theta = \begin{cases} \theta_{\text{d}}, & \tau_{\text{diff}} \geq 0, \\ \theta_{\text{c}}, & \tau_{\text{diff}} < 0. \end{cases} \tag{4.5}$$

In Eq.(4.4), μ is the friction coefficient and c_{geo} is the friction geometry factor, and represent the tunable parameters.

This approach captures the main yaw-moment effects of the LSD without needing proprietary design parameters or excessive computational effort.

4.1.2 Road Model

The vehicle model was initially run on a flat surface, assuming that road irregularities had minimal impact on vehicle dynamics. This approximation proved too coarse, leading first to the inclusion of a variable banking angle along the track. More recently, the track has been modeled using high-resolution 3D surfaces via CRG (Curved Regular Grid) files, leveraging the OpenCRG framework for accurate representation of complex road geometries.



(a) Assetto corsa view. (b) Track and kurbs. (c) Mesh visualization.

Figure 4.3: AC race-track exportation process, in a detailed visualization of turn 6 of the yas marina circuit, south configuration. In the first subplot the track visualization, including all decorative elements. In the second subplot only road surface and required kurbs are kept. The third subplot shows the grid visualization.

The selected circuit is first exported from a commercial racing simulator (e.g., Assetto Corsa, Fig. 4.3a) and converted into a standard 3D mesh format. The overall workflow can be summarized as follows:

1. **Mesh preprocessing.** The raw track mesh is edited using common 3D modeling software (i.e., Blender) to remove non-road elements such as trees, grandstands, and decorative objects. This step isolates a clean road surface suitable for numerical processing (Fig. 4.3b). Kerbs and local geometric features may be retained; however, particular care is required, as they strongly affect tire-road contact modeling.

2. **STL export and CRG conversion.** The processed road geometry is exported in STL format, i.e., a triangular-facet mesh suitable for numerical processing (Fig. 4.3c). A dedicated tool was developed to convert the STL geometry into the ASAM OpenCRG format [65], which represents the road surface through:
   - a reference line (track centerline), derived from the track borders;
   - a regular elevation grid, defined over longitudinal (u) and lateral (v) coordinates (height-map).
3. **Grid generation and elevation sampling.** The elevation grid is generated using user-defined discretization steps along u and v. For each longitudinal position u, lateral offsets v are generated orthogonally to the reference line until surface is detected. The elevation of each grid node is then computed by querying the STL surface, progressively populating the CRG height matrix.
4. **Local refinement and smoothing (optional).** Local mesh refinement can be enabled to increase resolution in selected regions, such as kerbs or localized road irregularities. Additionally, a moving-average smoothing filter may be applied along the height direction to reduce sharp elevation discontinuities. This step is particularly important when using simplified tire models (e.g., single-point contact), as it ensures smoother vertical tire dynamics response.
5. **Coordinate alignment.** Final roto-translational transformations are applied to ensure consistency between the simulated environment and real-world global coordinates.

The resulting CRG file is fully compatible with Dymola via the "*OpenCRGRoad*" block from the Vesyma and Suspensions libraries. This component reconstructs the 3D road surface from the CRG data and allows the assignment of either a constant friction coefficient or a spatially varying friction map loaded from an external file.

During track test, a Lidar map is also generated but was not used for simulation purposes, as its accuracy is sufficient for localization (e.g., SLAM) and state estimation algorithms [5] but not for high-fidelity simulator development, which requires highly precise tests and suitably positioned sensors. Refinement of a laser-scanned Assetto Corsa map using vehicle feedback, e.g., to validate elevations and banking angles, represents an optimal trade-off between accuracy and efficiency.

4.1.3 Tuning

It is well known that increasing model fidelity typically introduces a larger number of uncertain parameters and potential sources of error. In these complex models, even a small inaccuracy can trigger a cascade of inconsistencies throughout the simulation, turning a potential strength into a weakness. In our case, the tuning effort focuses on three main modules that generally represent the dominant sources of uncertainty: the tire model, the aerodynamic and engine maps.

Tire model. The fitting results are shown in Fig. 4.4, representing approximately two laps of the autonomous Dallara EAV24 Super Formula exploiting full tire performance on the Yas Marina Circuit (North layout). Offline tools were developed to reproduce, either partially or fully in open-loop, the exact maneuvers performed by the real vehicle (see Sec. 4.2.1). The multi-body model is run with identical setup and boundary conditions,

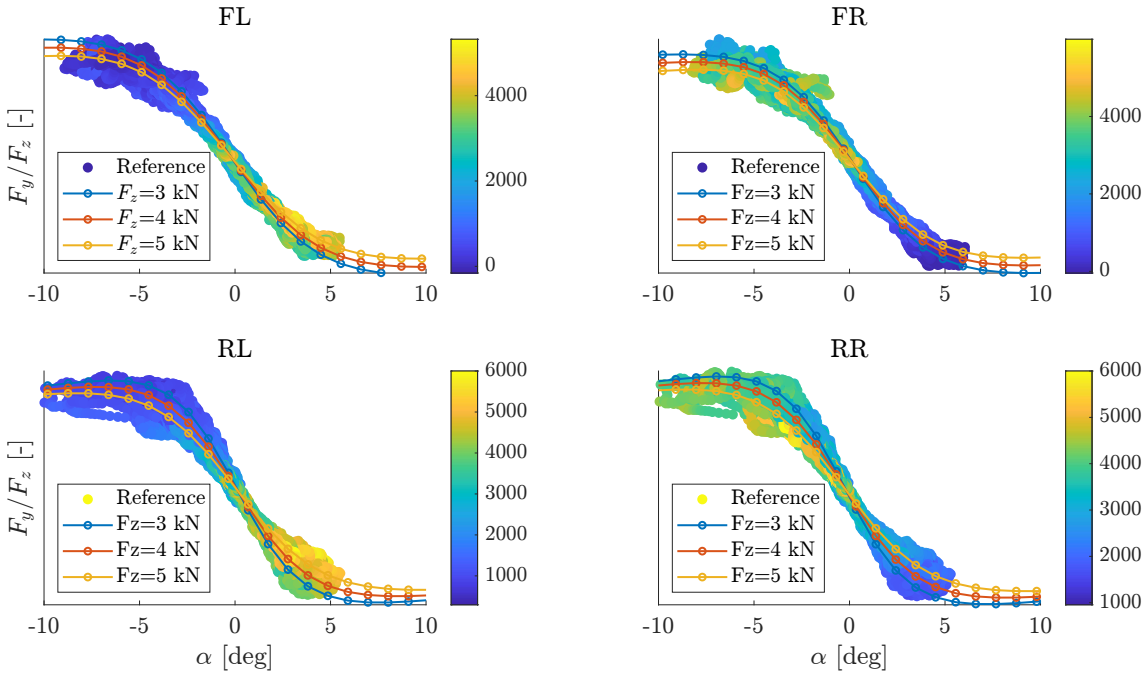


Figure 4.4: Experimental tire characteristics. The colorbars show the wheel load, $F_{z\,ij}$ $[N]$.

Table 4.1: Main tire parameters considered for tuning.

Parameter	Description	Effect on simulation
LMUY	Scale factor of F_y peak friction coefficient.	Change the overall tire lateral grip.
LKY	Scale factor of cornering stiffness.	Change the tire stiffness, modifying axle reactivity.
PDY2	Variation of friction with load.	Modify the tire's grip between low speed and high speed corners.
PKY2	Load at which Kfy reaches maximum value.	Modify the tire's stiffness between low speed and high speed corners.
RCY1	Shape factor for combined F_y reduction.	Modify the tire's performance under combined slip.

and the resulting motion field data are retrieved and compared with experimental measurements. Tire parameters are then fine-tuned, typically by adjusting Pacejka scale factors and micro-parameters (see Tab. 4.1) in the `.tir` files provided by the tire manufacturer.

For reconstructing tire characteristics, each tire's vertical load, slip angles, slip ratio, and lateral forces are required. Vertical loads are obtained from wheel load sensors and filtered to reduce noise induced by road surface irregularities. Slip angles are computed considering the vehicle's toe setup and rely on a combination of sensors and filtering approaches, as detailed in Sec. 6.1.2. The slip ratio are used to estimate the tire's longitudinal force, and compute the yaw moment generated by the differential, which in turn is leveraged to compute the effective lateral tire forces. Lateral forces are derived from vehicle translation (Y) and rotation (Z) equilibrium, while longitudinal forces are included using slip ratios. An overview of the sensors used for identification, denoted as $\mathbf{D}$, is provided below:

$$\mathbf{D} = [\, a_x,\ a_y,\ v_x,\ v_y,\ r,\ F_{z,\mathrm{FL}},\ F_{z,\mathrm{FR}},\ F_{z,\mathrm{RL}},\ F_{z,\mathrm{RR}},\ \omega_{\mathrm{FL}},\ \omega_{\mathrm{FR}},\ \omega_{\mathrm{RL}},\ \omega_{\mathrm{RR}},\ \delta]$$

The identification process began in collaboration with MegaRide - Applied Vehicle Research Group[1]. The methodology is largely inspired by their work. For more details, the reader is referred to [66, 67].

[1]Spin-off from University of Naples, translating vehicle dynamics research into software, testing tools, and industrial applications for motorsport and mobility. Website: `https://www.megaride.eu/`

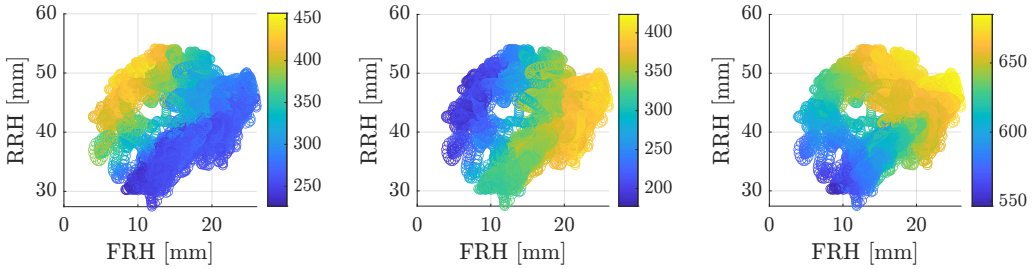


Figure 4.5: Experimental aerodynamic maps. From left to right, the colorbars show the front, rear, and total downforce, respectively.

Aerodynamic maps. Experimental aerodynamic maps are reported in Fig. 4.5. The values are obtained from wheel load sensors and processed to remove non-aerodynamic contributions, such as static vehicle weight and load transfers. The resulting maps represent downforce as a function of the vehicle's ride height. The trends observed are consistent with expected vehicle behavior: increasing the rake angle (lowering the front and raising the rear) shifts downforce toward the front. For example, subplot 1 indicates that the optimal front downforce is achieved at approximately 10 mm front ride height and 50 mm rear ride height (near the maximum). Based on these data, a constant-rake angle grid is created to extract downforce values for a lookup table.

Engine map. Accurately modeling the longitudinal response of an internal combustion engine is challenging, but crucial. Engine torque directly determines the vehicle's longitudinal behavior, influencing slip ratio and, consequently, vehicle stability. Moreover, it also affects lateral dynamics through its interaction with the limited-slip differential (LSD), whose behavior depends on the applied drivetrain torque.

To reconstruct the engine map from experimental data, a longitudinal (X) dynamic equilibrium is solved, accounting for inertial forces, aerodynamic drag, and tire rolling resistance. The resulting rear longitudinal tire force F_{xr} is then mapped back to engine torque (T_{eng}) using gear ratios, final drive, and powertrain rotational inertia. This modeling approach closely follows the one described in Sec. 5.3.2. The experimental map shown in Fig. 4.6 was obtained through stationary maneuvers. The red signals indicate the mod-

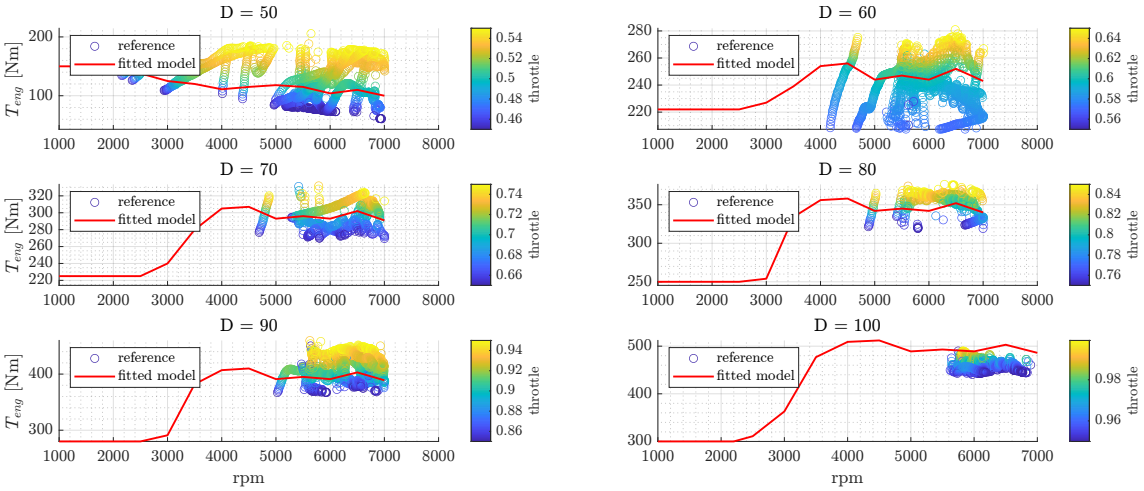


Figure 4.6: Experimental engine torque (T_{eng}) maps as a function of engine rotational speed (rpm). Colorbars indicate the throttle level (normalized command).

eled torque for the throttle (D) as labeled in the figure. Good agreement is observed when the modeled map lies within the experimental cloud, which represents the average torque at each throttle position with a ±5% deviation.

The point cloud is obtained using a longitudinal controller that tracks a sequence of constant target speeds, allowing steady-state engine operating points to be sampled and used to populate the map. The initial engine map is derived from engine test bench data and subsequently refined using on-track measurements following the described procedure.

The resulting map provides high accuracy in steady-state conditions. However, significant mismatches may arise in transient phases due to turbo-lag effects. In particular, the engine exhibits substantial delays both in boost pressure build-up during throttle application and in torque decay during throttle release. These dynamics cannot be captured by a static engine map and therefore remain unmodeled in the current formulation. This limitation highlights the need for a more refined powertrain model that explicitly includes turbocharger dynamics to improve transient response fidelity.

4.2 Simulation framework

The vehicle model presented in Sec. 4.1 has been exported from Dymola as a Functional Mock-up Unit (FMU), following the FMI (Functional Mock-up Interface) standard [68,69]. This standard provides a unified interface for exchanging dynamic models across heterogeneous tools, ensuring modularity, portability, and ease of integration within external simulation frameworks.

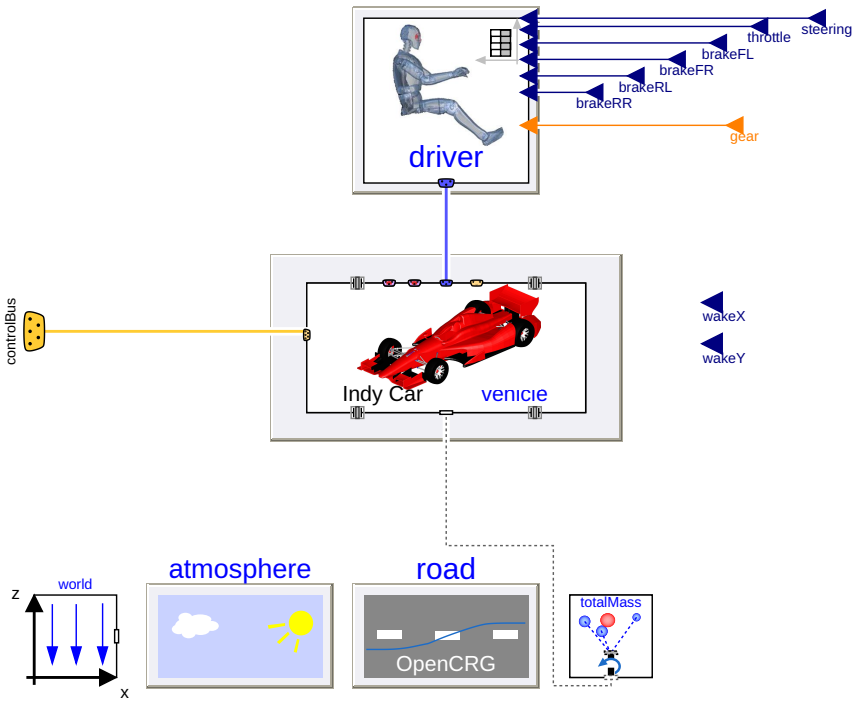


Figure 4.7: Dymola view of the FMU exported experiment. This layer groups the vehicle model block (which include the blocks as in Fig.4.1), the driver (defined as in Fig.4.2i) and the CRG road block (as described in Sec.4.1.2).

A dedicated Dymola experiment was created specifically for FMU export and simulator integration, as schematically shown in Fig. 4.7. Within this experiment, the vehicle model

is coupled with an open-loop driver block. Dymola natively allows external inputs to be provided through Modelica connectors, which are explicitly exposed in the FMU interface. In particular, the following input signals are defined:

- N independent brake commands (real-type), where N depends on the vehicle hardware configuration;
- throttle command (real-type), expressed as a normalized value $D \in [0, 1]$;
- steering command (real-type), expressed as a normalized value $\delta \in [-1, 1]$, where the steering ratio is defined at the steering pinion level;
- gear selection command (integer-type), representing the engaged gear $G \in [1, 6]$;
- slipstream (drafting) information, provided through the opponent vehicle relative position (x, y) (real-type).

Accordingly, the FMU exposes both real-valued and integer-valued input connectors, allowing full control of the vehicle actuators and external aerodynamic effects from the simulation environment. The road block has been modified to load a CRG file from a predefined directory. The track filename is fixed (e.g., Track.crg), and different circuits are selected by placing the corresponding CRG file (renamed accordingly) into the specified path defined within the Dymola block. This enables rapid testing on different tracks without requiring model re-exporting.

Similarly, vehicle setup parameters are managed through an external configuration file: a dedicated setup folder contains a text file defining all tunable vehicle parameters, such as camber and toe angles, tire files, suspension characteristics, and differential configuration. This design choice proved fundamental to ensure flexibility during tuning and validation activities. Vehicle setup and tire parameters can be modified without re-exporting the FMU, significantly reducing iteration time when performing setup adjustments or tire parameter updates.

Once exported, the FMU appears as a pre-compiled black-box model: its internal modelica equations are not accessible, and interaction is limited to the defined input and output variables, which are taken from the Control-Bus connector. Overall, this design brings several advantages:

1. It allows the simulator to incorporate complex vehicle models developed in environments such as Dymola while remaining fully decoupled from the modeling tool.
2. The same FMU can be embedded into different simulation environments depending on the specific objectives of the study (as further detailed in the next sections).
3. Using an FMU abstraction enables seamless substitution of different models without modifying the simulator core, effectively supporting a general-purpose simulation framework.

In all the developed simulators, the FMU is exported in version 2.0 and executed in *Co-Simulation* mode. In this setup, it autonomously advances its internal state using its own solver, ensuring stable and efficient integration with the host environment.

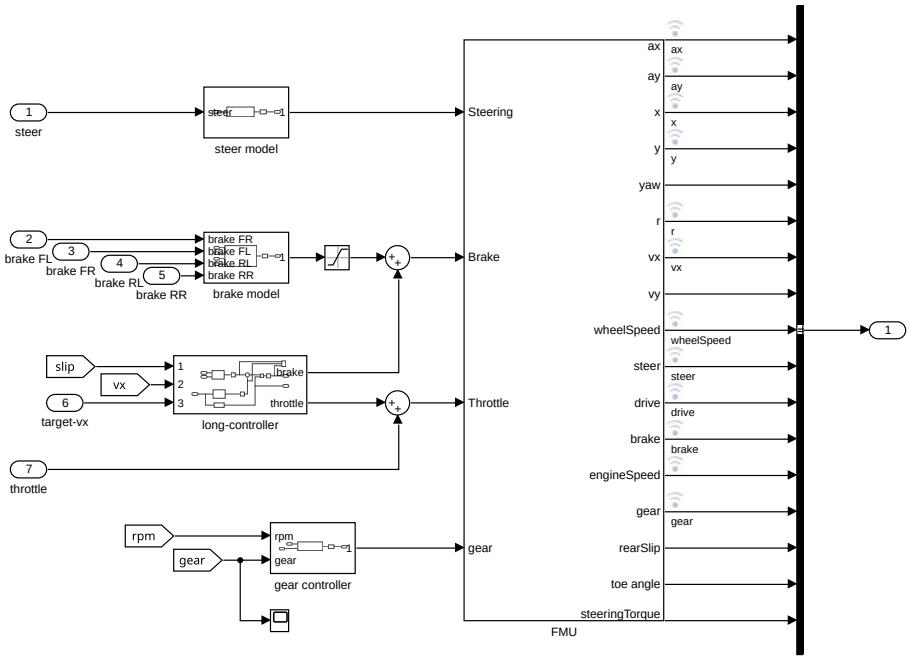


Figure 4.8: Simulink integration of the vehicle FMU using FMIKit, running a partially open-loop simulation with throttle and brake compensation. The gearbox controller replicates the one used in the autonomous vehicle.

4.2.1 Matlab/Simulink environment

The vehicle model has been integrated into MATLAB/Simulink using *FMIKit-Simulink*, an open-source library for importing and exporting FMUs within Simulink [70]. This integration allows seamless execution of FMUs and facilitates the reproduction of standard driving maneuvers for parameter tuning and model validation. A representative simulation setup is shown in Fig. 4.8. The simulator can be operated in three distinct modes, depending on the objective of the analysis.

Full open-loop simulation. In this configuration, the actuator inputs measured on the real vehicle (throttle, brake, steering, and gear commands) are replayed directly into the FMU. This approach allows faithful replication of the real driving scenario and is particularly effective for identifying modeling inaccuracies related to longitudinal dynamics and yaw behavior induced by the limited-slip differential. These tests are primarily used to tune the engine map and the driveline, improving the longitudinal response of the model.

This configuration is also suitable for identifying lateral dynamics in simplified vehicle models, as it enables the execution of quasi-steady maneuvers over a wide range of speeds and curvature radii (such as steering-pad and ramp-steer maneuvers). It is therefore used to extract axle-level characteristics for simplified models (such as single-track formulations) or to empirically map tire-road friction for point-mass models.

Open-loop simulation with speed correction. In this mode, the steering input is directly replayed from real vehicle data, while pedal commands are processed through a velocity-tracking compensation loop. A feedback controller (PID) acts on throttle and brake commands to reduce longitudinal speed tracking errors caused by mismatches between the FMU model and the real vehicle. This configuration ensures that the simulated

vehicle follows the same cornering speed as in the real-world experiment, which is particularly useful for isolating and tuning lateral tire characteristics.

Closed-loop simulation. A simple lateral pure pursuit controller, together with a longitudinal controller (detailed in Sec. 8.2.2), is implemented to track a reference path and speed profile. This closed-loop setup is essential for evaluating vehicle behavior in specific track zones. However, the primary closed-loop experiments are conducted using the more advanced simulator described in Sec. 4.2.2.

4.2.2 Closed-loop C++ simulation

The FMU emulates the autonomous vehicle during simulation, enabling the testing of all vehicle-dynamics-related modules, such as localization, planning, and control, within a closed-loop framework. Since the entire autonomous driving stack is implemented in C++, the integration with the vehicle dynamics model is performed through the FMI4cpp library [71]. FMI4cpp is an object-oriented library that provides high-level abstractions for FMU-based simulation, including model instantiation, input and output management, and time integration. Its clean API minimizes boilerplate code and enables efficient data exchange, making it suitable for real-time closed-loop execution.

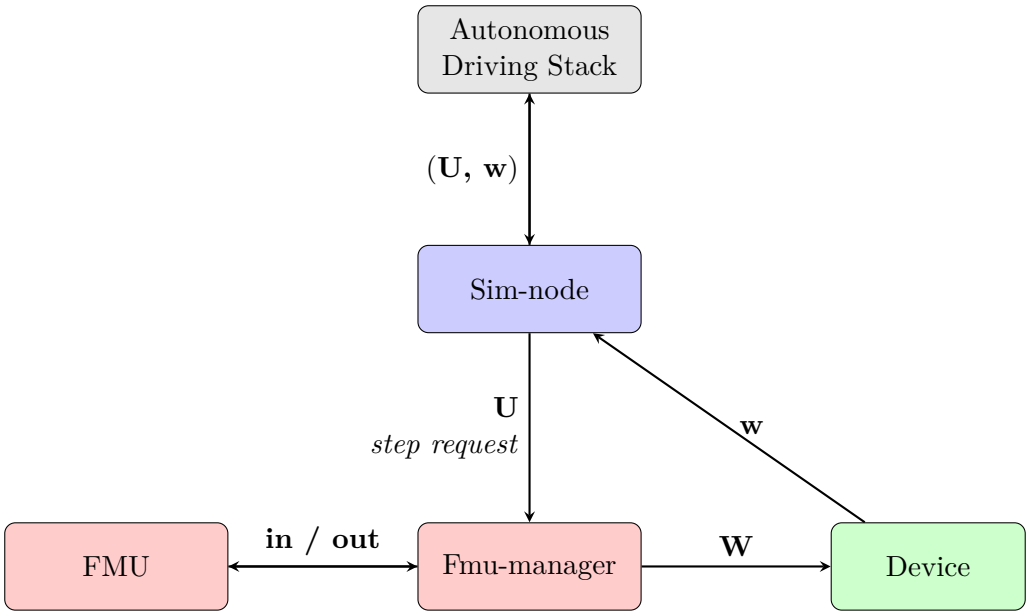


Figure 4.9: Block diagram of the closed-loop C++ simulator. FMU interacts only with Fmu-manager, which mediates communication with Sim-node and Device. In the scheme, **U** denotes control commands from the autonomous driving stack, **in-/out** are FMU inputs/outputs, **W** is the vehicle state computed by the FMU, and **w** represents simulated sensor measurements fed back to the software stack.

The simulator architecture is organized into three main modules: *Sim-node*, *Fmu-manager*, and *Device*, as shown in Fig. 4.9. Interaction with the FMU is entirely encapsulated within the *Fmu-manager*, which wraps the FMI4cpp library and represents the sole interface between the simulator and the FMU, ensuring a clear separation of concerns and improved maintainability. Runtime operation follows a fixed-frequency loop (1 ms), coordinated by *Sim-node*. At each iteration, it receives control commands from the autonomous driving stack and delegates the simulation step to the *Fmu-manager*, which handles all FMU

operations, including initialization, input updates, time stepping, output retrieval, and termination:

1. **Input update:** Vehicle control commands are processed (normalization, steering and brake model application) and written to the FMU.

2. **Time integration:** The FMU advances its internal state using its own solver. The simulator can optionally apply a speed-up factor for faster-than-real-time execution, particularly usefull during automatic simulations. Inputs remain constant during each step.

3. **Output retrieval:** Vehicle state variables (wheel speeds, steering angles, brake pressures, tire forces, accelerations, poses, etc.) are read from the FMU, with unit conversions applied to match simulator conventions.

The *Device* module converts the vehicle state from the FMU outputs into simulated sensor measurements. Each device applies sensor-specific processing, including signal normalization, noise injection, and enforcement of sensor-specific update rates. Sensor characteristics and parameters are defined through configuration files, enabling easy adaptation when switching between different FMUs.

Finally, the processed sensor data are published in the same format as real vehicle sensors, providing coherent and realistic vehicle feedback to the autonomous driving stack and enabling closed-loop simulation. Profiling and debug facilities track execution times for the main loop, integration step, and device updates.

4.3 Main Results and Limitations

The simulation framework is continuously updated and validated using experimental data. This ensures a platform capable of reproducing entire test sessions, a fundamental approach to identify inaccuracies or potential planning and control issues in advance.

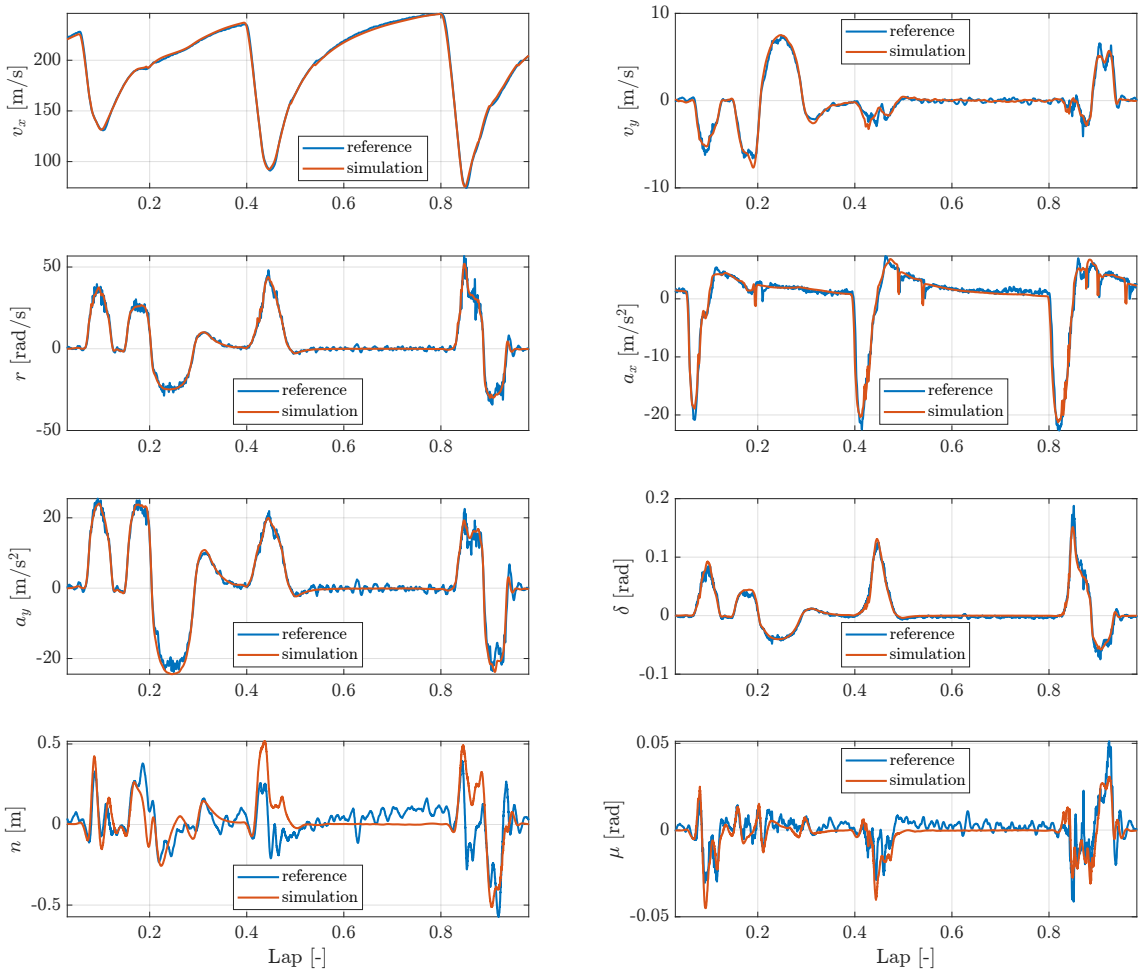
4.3.1 Model accuracy

In this section, we present the results obtained using the simulator described in Sec. 4.2.2.

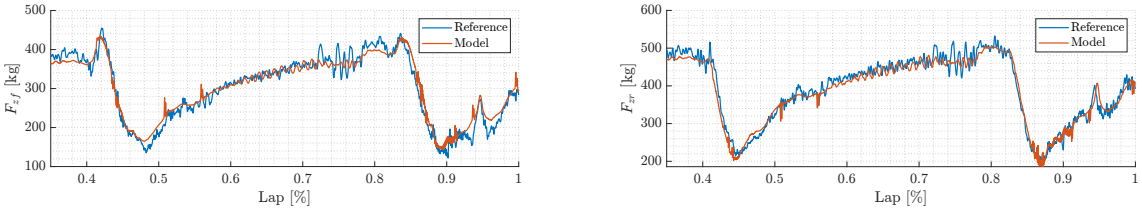
Model dynamics. Fig. 4.10a shows several vehicle dynamics-related quantities (e.g., velocities and accelerations) and control-related signals (e.g., steering command and tracking errors). In these simulations, the software is tested under the same boundary conditions used in the real vehicle (i.e., identical performance targets).

Both longitudinal (v_x) and lateral velocity (v_y) are accurately replicated, with minor discrepancies observed in Turn 5 ($Lap \approx 0.45$). For autonomous racing applications, the ability to reproduce controller tracking errors in simulation is crucial and particularly challenging, as centimeter-level accuracy is required. Despite the high fidelity of the vehicle model, small discrepancies in the motion field propagate into slight mismatches in the simulated lateral error (n), especially in Turn 5 and Turn 6 ($Lap \approx 0.85$). Overall, the tracking errors show good agreement between the simulation and experiments.

The simulator was useful to study in advance some planning and control behavior. For example, positive turn-in peak errors are observed in Turn 1 ($Lap \approx 0.1$), Turn 5, and Turn 6 during the simulation phase. These originate from an overestimation of the tire combined-slip effect within the controller, leading to an anticipatory steering action



(a) Vehicle-dynamics-related quantities (longitudinal and lateral velocities, accelerations) and control-related signals (steering angle, lateral error, n, and heading error, μ).



(b) Vertical load profiles on the front axle (left) and rear axle (right).

Figure 4.10: Comparison between real-world measurements (Reference) and simulation results (Model) under identical operating conditions.

to compensate for the predicted reduction in lateral force. The same peak errors are then observed in the real car. Similarly, an oscillatory behavior during the turn-out phase can be observed, especially in Turn 1 and Turn 8 ($Lap \approx 0.9$). It is worth noting that real-world experiments exhibit larger oscillations due to sensing delays and localization noise.

Road surface. Figure 4.10b validates the 3D road modeling approach described in Sec. 4.1.2. In the derived CRG mesh, a relatively high smoothing factor is applied; as a result, some high-frequency road roughness is filtered out, for instance around Lap ≈ 0.75

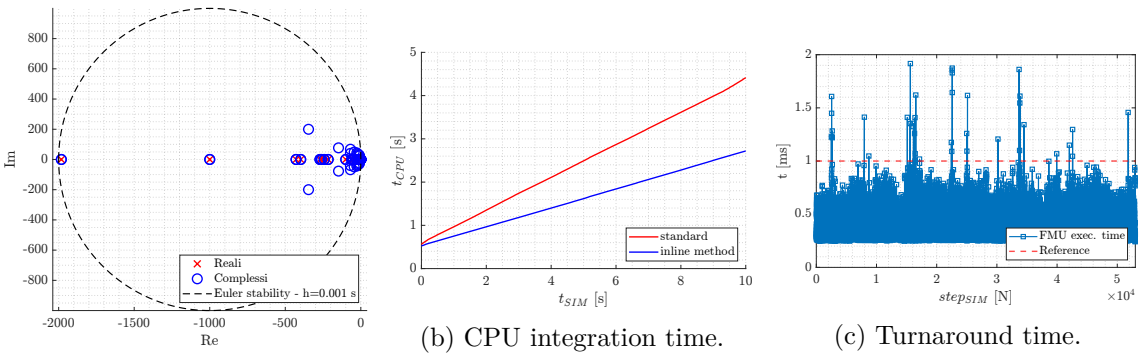
of the lap length. Nevertheless, the most relevant low-frequency road features are accurately captured, such as the bump located at approximately Lap ≈ 0.95. This modeling choice allows the simulator to reproduce key vehicle behaviors that are critical for assessing the stability of the control algorithms under road-induced disturbances. Moreover, it helps preserve physically consistent tire behavior despite the use of a simplified single-point tire-road contact model. In fact, accurately capturing curbs and higher-frequency road irregularities would require a more detailed tire-road interaction model, such as a multi-point contact formulation, which is under evaluation for future works.

4.3.2 Execution time and numerical stability [11]

Achieving real-time performance while ensuring a stable and consistent simulation requires careful selection of the integrator and its time step. Vehicle dynamics, whether in a multi-body or simplified model, follow the same fundamental rules; therefore, the considerations discussed in Secs. 3.3.1 and 3.3.2 also apply here.

In our framework, the model is integrated using a fixed-step explicit Euler scheme with a timestep of $h = 0.001\,\text{s}$, which is required to accurately capture fast tire dynamics. Pole analysis confirms that all system poles have negative real parts, while also revealing the presence of fast dynamics. This indicates that the chosen timestep represents the minimum value ensuring numerical stability. Additional tests were performed with a larger timestep ($h = 0.0012\,\text{s}$). While high-speed simulations remain stable, low-speed scenarios exhibit oscillatory behavior. These instabilities are mainly attributed to the centrifugal clutch engagement and tire transient dynamics, as discussed in [11]. Despite the improved execution time, this configuration was ultimately discarded. A visualization of the system poles, obtained using Dymola's integrated "*Linear Analysis*" tool, is shown in Fig. 4.11a.

To further accelerate simulations, Dymola implements in-line integration [72], a symbolic numerical approach for solving differential-algebraic systems (DAEs). This method embeds discretization expressions directly into the model using symbolic manipulation, allowing numerical integration to be performed 'in-line' with the system equations. By reducing computational overhead, in-line integration improves real-time performance, particularly for explicit fixed-step methods, and significantly reduces CPU time for integra-



(a) Model poles.

(b) CPU integration time.

(c) Turnaround time.

Figure 4.11: On the left: poles of the multi-body model in the Gauss plane. The dashed circle represents the explicit Euler stability region for a timestep of 1 ms. In the middle: CPU integration time for a 10 s simulation. Inline integration (blue line) significantly improves performance, reducing integration time by over 1 s. On the right: turnaround time (TAT) for an FMU simulation (see Sec. 4.2). The red line indicates the real-time execution target of 1 ms.

tion, as illustrated in Fig. 4.11b.

Figure 4.11c shows the measured turnaround time over a full Yas Marina Circuit lap simulation of the complete autonomous driving stack, including planning, control, and localization, using the simulation architecture described in Sec. 4.2.2. The FMU is capable of real-time performance, with all observed runtimes consistently below the execution target of 1 ms. Only a very small number of peaks slightly exceed this target, and these have no significant impact on the model's response. Simulations were performed on an Ubuntu 24 system equipped with an Intel Core i9-14900HX CPU (32 threads). A summary of the execution times is provided in Tab. 4.2.

	Minimum	**Maximum**	**Average**
Runtime (ms)	0.234	1.916	0.357

Table 4.2: FMU execution times for a full lap simulation. Values are reported in milliseconds.

Chapter 5

Motion Framework

The motion framework, represented in Fig.5.1, follows a standard planning-control scheme: the planner draws the route, while the controller drives along it, correcting for perturbations or dynamic changes.

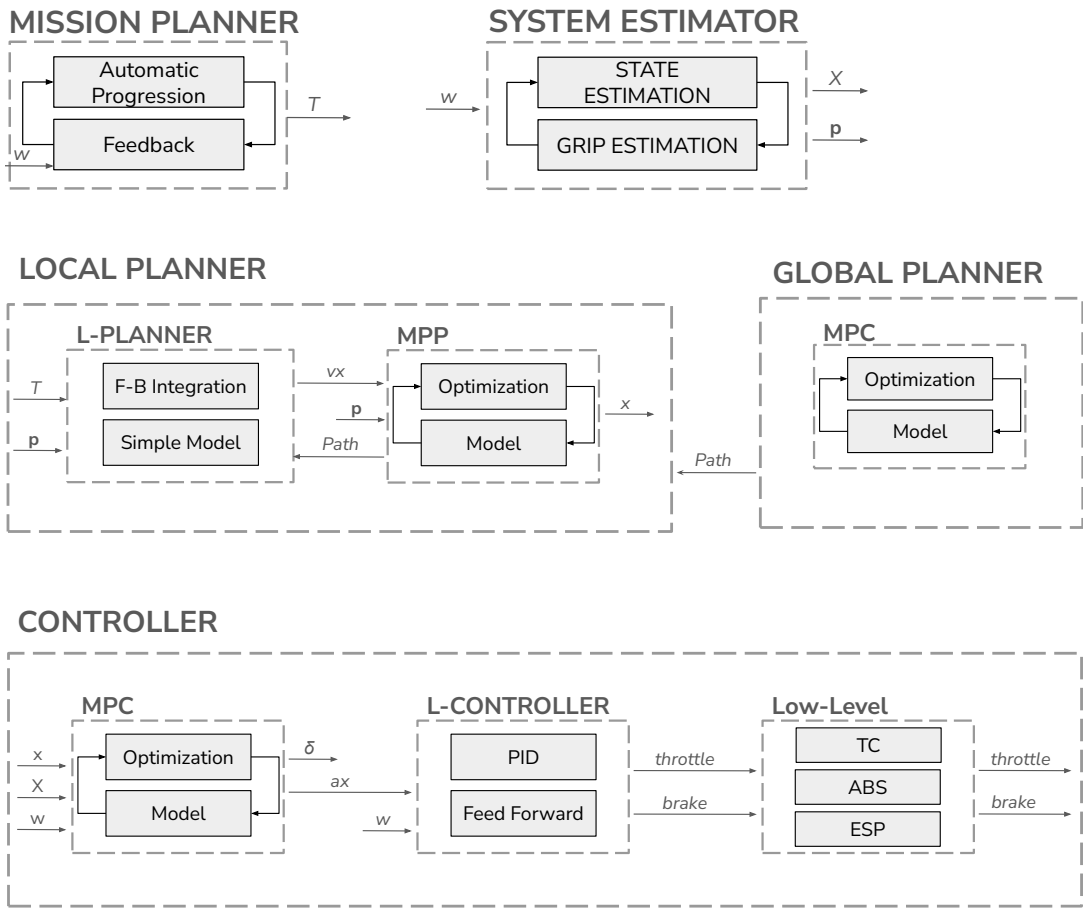


Figure 5.1: Full planning, control and system estimation diagrams. In the scheme, **w** represent the vehicle feedback signals, **T** the target performance, **p** the tire parameters, including tire-road friction coefficient, **X** the vehicle state feedback (filter output), and **x** the planning optimized reference.

The framework presented in this thesis implements a fully autonomous architecture, extending the work presented in [12], composed of the following components:

1. A high-level **mission planner** (Sec. 5.2.2) that, relying on vehicle feedback (**w**), defines the optimal lap-time progression along the track to maximize overall vehicle performance, and sends the performance targets (**T**) to the lower-level local planner.

2. A **global planner** (Sec. 5.2.1) that computes, offline, the optimal path within track limits and dynamic constraints, providing an accurate initial solution to the local planner.

3. A **local planner** (Sec. 5.2.3 and Sec. 5.2.4) that adapts the trajectory whenever deviations from the global path are required, for instance for obstacle avoidance, overtaking maneuvers, or the merging of different precomputed lines (such as pit-lane and racing-line trajectories), and send the updated reference (**x**) to the controller.

4. A **controller** (Sec. 5.3) that ensures tracking of the local planner reference trajectory, while compensating for unexpected vehicle dynamics such as understeer or oversteer. The controller send driving input (δ**, throttle, brake**) to the vehicle.

5. A **system identification** module (Sec. 6) which ensures that both the planner and the controller are aware of the current operating conditions (**p**), such as the tire–road friction level, and provides all relevant dynamic quantities in feedback (**X**) to enable safe and optimal operation.

Most layers of the framework are model-based, with both the main planner (Sec.5.2.4) and the main controller (Sec.5.3.1) relying on optimization-based algorithms (i.e., Model Predictive Control, detailed in Sec. 5.1), and sharing the same vehicle model.

Some modules, such as the local longitudinal planner, employ a simplified vehicle representation. Nevertheless, consistency across the entire architecture is preserved by sharing the same vehicle parameters with the more detailed models, ensuring coherent behavior among planning and control layers.

5.1 Model Predictive Control architecture

The problem is formulated as a nonlinear Model Predictive Control (MPC) scheme solved through a Sequential Quadratic Programming (SQP) framework. At each step, the vehicle model and constraint functions are locally linearized around a nominal trajectory. The SQP scheme is warm-started using the optimal solution from the previous control step (Sec. 5.4.3). Depending on the configuration, one or more SQP iterations can be performed at each sampling step, improving the approximation of the system nonlinearities.

This results in a structured optimal control quadratic program (OCP-QP), solved using HPIPM [73], a high-performance quadratic programming solver designed for MPC applications. The required Jacobians are computed through automatic differentiation using CppADCodeGen [74]. The same formulation is used for both trajectory generation (*Model Predictive Planning*, MPP) and trajectory tracking (*Model Predictive Control*, MPC), and is referred to in this work as *Model Predictive Planning and Control (MPPC)*.

The nonlinear optimal control problem can then be written as

$$\begin{aligned}
\min_{\mathbf{X},\mathbf{U},\mathbf{S}} \quad & \sum_{k=0}^{N} \left(J_{\text{MPC}}(\mathbf{x}_k, \mathbf{u}_k) + J_{\text{sc}}(\mathbf{s}_k)\right) \\
\text{s.t.} \quad & \mathbf{x}_0 = \hat{\mathbf{x}}, \\
& \mathbf{x}_{k+1} = f_t^d(\mathbf{x}_k, \mathbf{u}_k), \qquad k = 0, \ldots, N-1, \\
& \mathbf{x}_k \in \mathcal{X}_{\text{track}}, \quad \mathbf{x}_k \in \mathcal{X}_{\text{ellipse}}, \qquad k = 0, \ldots, N, \\
& \mathbf{x}_k \in \mathcal{A}, \qquad k = 0, \ldots, N, \\
& \mathbf{u}_k \in \mathcal{U}, \qquad k = 0, \ldots, N-1.
\end{aligned} \tag{5.1}$$

where

$$\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N], \qquad \mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}]$$

denote the predicted state and input sequences over the horizon. Since selected constraints are treated as soft, the optimization problem introduces slack variables

$$\mathbf{S} = [\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_N],$$

which allow controlled violations of selected inequality constraints. J_{MPC} and J_{SC} denote the quadratic cost function, while $f_t^d(\mathbf{x}_k, \mathbf{u}_k)$ represents the plant model. Soft constraints are included as $\mathcal{X}_{\text{track}}$, which limits the lateral deviation, ensuring that the predicted trajectory remains within the track, and $\mathcal{X}_{\text{ellipse}}$, which represents a friction-ellipse-like constraints to account for combined slip effect. The box constraints $\mathcal{A}$ and $\mathcal{U}$ are included as hard bounds to limit the MPC state and input vector values, respectively.

5.1.1 Vehicle Model

The plant model $f_t^d(\mathbf{x}_k, \mathbf{u}_k)$ is a nonlinear dynamic single-track model formulated in curvilinear coordinates (Fig.5.2) and discretized by a fourth-order Runge-Kutta integrator. The differential yaw moment (M_{diff}) can be incorporated within the state space formulation, effectively extending the formulation to a tricycle model. The state vector is

$$\mathbf{x} = \begin{bmatrix} s & n & \mu & v_x & v_y & r & a_x & \delta \end{bmatrix}^T, \tag{5.2}$$

where s is the progress along the reference path, n is the lateral deviation from the path centerline, μ is the heading error with respect to the local path tangent, v_x and v_y are the

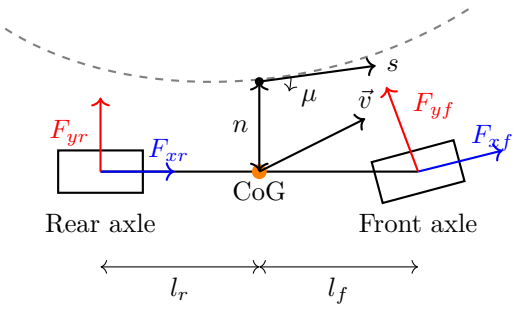


Figure 5.2: Schematic representation of the dynamic single-track model with Frenet-frame coordinates.

longitudinal and lateral velocities, r is the yaw rate, a_x is the longitudinal acceleration, and δ is the steering angle. The control input vector is

$$\mathbf{u} = \begin{bmatrix} J_x & \dot{\delta} \end{bmatrix}^T, \tag{5.3}$$

that is, the optimization variables are the rates of change of the longitudinal acceleration (Jerk) and steering angle. This structure allows the controller to regularize both the physical quantities and their rate variations in a natural way. The continuous-time dynamics are

$$\dot{s} = \frac{v_x \cos\mu - v_y \sin\mu}{1 - n\kappa(s)}, \tag{5.4a}$$

$$\dot{n} = v_x \sin\mu + v_y \cos\mu, \tag{5.4b}$$

$$\dot{\mu} = r - \kappa(s)\dot{s}, \tag{5.4c}$$

$$\dot{v}_x = \frac{1}{m}\left(F_{xr} - F_d - F_{yf}\sin\delta + F_{xf}\cos\delta + m v_y r\right), \tag{5.4d}$$

$$\dot{v}_y = \frac{1}{m}\left(F_{yr} + F_{yf}\cos\delta + F_{xf}\sin\delta - m v_x r\right), \tag{5.4e}$$

$$\dot{r} = \frac{1}{J_z}\left[l_f\left(F_{yf}\cos\delta + F_{xf}\sin\delta\right) - l_r F_{yr}\right]. \tag{5.4f}$$

The tire forces are modeled through a nonlinear Pacejka formulation. To ensure stability at low velocities, the dynamic single-track model is replaced with the enhanced kinematic formulation while retaining the same state-space structure

$$\dot{s} = \frac{v_x \cos(\mu + \beta)}{1 - n\kappa(s)}, \tag{5.5a}$$

$$\dot{n} = v_x \sin(\mu + \beta), \tag{5.5b}$$

$$\dot{\mu} = \frac{v_x}{L}\tan(\delta)\cdot \mathrm{SF} - \kappa(s)\frac{v_x\cos(\mu+\beta)}{1 - n\kappa(s)}, \tag{5.5c}$$

$$\dot{v}_x = \frac{1}{m}\big(F_{xf} + F_{xr}\big), \tag{5.5d}$$

$$\dot{v}_y = \frac{l_r}{L}\big(\dot{\delta}\, v_x + \delta\, \dot{v}_x\big), \tag{5.5e}$$

$$\dot{r} = \frac{\dot{\delta}\, v_x + \delta\, \dot{v}_x}{L}. \tag{5.5f}$$

This approach avoids numerical instabilities thanks to the absence of dynamic terms, while still providing a reliable approximation of the vehicle behavior, as demonstrated in Sec. 7.1.2. The definition of SF is consistent with the enhanced kinematic model described in Eq. (3.4), while the lateral velocity and yaw rate derivatives ($\dot{v}_y$ and $\dot{r}$) are obtained from Eq. (3.2) and Eq. (3.1), respectively, under the small-angle assumption, which linearizes the tangent functions.

The dynamics of both models follow those presented in Sec. 3; the only difference is a change of reference frame introduced to better suit planning and control requirements. In this formulation, the vehicle position (s, n) and heading (μ) are expressed as functions of a reference trajectory, whose curvature $\kappa(s)$ is taken into account, replacing the standard Cartesian coordinates (X, Y, ψ).

As the speed increases, an exponential blending function smoothly transitions the model from kinematic to dynamic, ensuring that all relevant vehicle dynamics are accurately captured. The final state-space model used within the solver takes the form:

$$\dot{\mathbf{x}} = \lambda\, \dot{\mathbf{x}}_{\mathrm{dyn}} + (1 - \lambda)\, \dot{\mathbf{x}}_{\mathrm{kin}}, \tag{5.6}$$

where

$$\lambda = \begin{cases} 0, & v_x \leq v_{x,\mathrm{bl}}, \\ 1 - \exp\big((v_x - v_{x0}) \cdot k_0\big), & v_x > v_{x,\mathrm{bl}}, \end{cases} \tag{5.7}$$

Depending on tire stiffness, as discussed in Sec. 3.3.1, the velocity threshold may vary significantly. In our configuration, $v_{x0} = 8\,\mathrm{m/s}$.

5.1.2 Cost Function

The objective function is formulated as:

$$J_{\mathrm{MPC}}(\mathbf{x}_k, \mathbf{u}_k) = \left(\frac{1}{2} \mathbf{x}_k^T Q_k \mathbf{x}_k + q_k^T \mathbf{x}_k + \frac{1}{2} \mathbf{u}_k^T R_k \mathbf{u}_k + r_k^T \mathbf{u}_k + B(\mathbf{x}_k) \right) \tag{5.8}$$

where Q and R denote the state and input weighting matrices, respectively, with q and r representing the corresponding linear penalty. The state weighting matrix includes path-following terms q_n and q_μ, a velocity tracking weight q_{vx}, and a yaw rate weight q_r to enhance stability and dampen oscillations. The regularization term $\mathrm{u}^T R \mathrm{u}$ penalizes input variations and promotes smoother control actions. Finally, the term $B(\mathrm{x}_k) = q_\beta \beta^2$ penalizes the sideslip angle, further contributing to vehicle stability when operating near the limits of tire friction.

In the cost function, the soft constraints are relaxed through slack variables ($\mathbf{s}_k$) by

$$J_{\mathrm{sc}}(\mathbf{s}_k) = \frac{1}{2} \mathbf{s}_k^T Z \mathbf{s}_k + z^T \mathbf{s}_k. \tag{5.9}$$

where Z and z denote the quadratic and linear slacks weighting matrices, respectively.

In the current implementation, the same quadratic and linear penalties are assigned to both lower and upper slacks. Hence, violating the lower or upper side of a soft constraint is penalized symmetrically. This applies, for example, to the track bound, where leaving the admissible corridor on either side is weighted in the same way.

A single tuning of the cost function may fail to capture the variety of scenarios encountered in autonomous racing, as different maneuvers can have distinct objectives. Using a single cost set can either provide a compromise across all scenarios or be overfitted to a specific one. To address this limitation, we define multiple cost sets targeting specific planning and control goals. A cost-blending state machine dynamically selects the appropriate MPPC cost set based on the current driving scenario, as summarized in Tab. 5.1.
The state machine provides smooth, context-aware updates of the MPPC cost. Cost blending is performed using sigmoid-based interpolation, ensuring gradual transitions between different cost sets without discontinuities.

Table 5.1: MPPC Cost-Blending States and Activation Criteria. State transitions depend on several conditions: high-level requests (e.g., line switching), the understeer angle level, the lateral track position, opponent distance and relative speed, and Adaptive Cruise Control (ACC) activation.

State	Description	Activation Condition
Nominal	Baseline behaviour.	Default state
Line switch	Transition between reference lines.	High-Level request
Overtaking	Attack/defence logic triggered by opponent.	$d_{\mathrm{opponent}} < d_{\mathrm{thresh}}$
Oversteer	Stability-recovery mode.	$\lvert\text{understeer/oversteer}\rvert > \beta_{\mathrm{th}}$
Out of track	Vehicle leaves track limits.	$\lvert n \rvert > n_{\mathrm{track,max}}$

5.1.3 Soft Constraints

The optimization includes soft constraints on selected quantities, such as track limits ($\mathcal{X}_{\text{track}}$) and tire dynamics ($\mathcal{X}_{\text{ellipse}}$). These constraints are enforced through slack variables, which appear both:

- in the inequality constraints, and
- in the cost function via linear and quadratic penalty terms (Eq. (5.9)).

Thus, the slack variables are included in the decision vector of the OCP-QP and represent the amount of constraint violation (Eq. (5.10)). These constraints are assembled in the form

$$\mathbf{l}_{g,k} - \mathbf{s}_k \leq C_k \mathbf{x}_k + D_k \mathbf{u}_k \leq \mathbf{u}_{g,k} + \mathbf{s}_k, \qquad \mathbf{s}_k \geq 0. \tag{5.10}$$

where $\mathbf{l}_{g,k}$ and $\mathbf{u}_{g,k}$ represent the lower and upper bounds, respectively. The matrices C_k and D_k are constructed stage-wise within the prediction horizon.

As a result, larger weights increase the cost of constraint violation, encouraging the optimizer to keep the slack as small as possible unless relaxing the constraint significantly improves the overall objective.

Track Constraint. The track constraint is enforced directly on the lateral deviation with respect to the reference path. The track constraint limits the lateral position of the vehicle within the admissible corridor, expressed using the lateral deviation coordinate n. At each stage, the affine constraint matrix is constructed such that

$$C_{\text{track}} \mathbf{x}_k = n_k. \tag{5.11}$$

The admissible bounds are computed from the track tunnel limits considering both track margin and the opponents position, as detailed in Sec.5.2.4. In particular, the bounds are defined as

$$\begin{aligned} l_{\text{track},k} &= n_{\text{right},k} - n_k, \\ u_{\text{track},k} &= n_{\text{left},k} - n_k, \end{aligned} \tag{5.12}$$

where $n_{\text{left},k}$ and $n_{\text{right},k}$ denote the left and right track limits.

Tire Constraints. The MPPC constrains the solution using a quadratic norm of the normalized tire forces, inspired by the classical friction ellipse representation. This formulation is not intended to model the tire behavior directly, but rather to provide a compact indicator of combined force utilization.

$$S_{\text{tire}}(\mathbf{x}) = \frac{(\varepsilon_x F_{xi})^2}{(F_{xi,\max})^2} + \frac{(\varepsilon_y F_{yi})^2}{(F_{yi,\max})^2} - 1 \tag{5.13}$$

where ε_x, ε_y are tunable scaling coefficients that adjust the relative contribution of longitudinal and lateral force utilization. The maximum admissible forces are obtained from the Pacejka peak coefficients according to

$$F_{xi,\max} = F_{zi} D_{xi}, \qquad F_{yi,\max} = F_{zi} D_{yi},$$

where F_{zi} denotes the normal load acting on tire i, and D_{xi} and D_{yi} correspond to the longitudinal and lateral Pacejka peak parameters. The scalar quantity $S_{\text{tire}}(\mathbf{x})$ is used as a combined-force utilization indicator.

When $S_{\text{tire}}(\mathbf{x}) \leq 0$, the requested longitudinal and lateral forces lie inside the admissible combined-force envelope, with $S_{\text{tire}}(\mathbf{x}) = 0$ representing the friction-ellipse bounds. When $S_{\text{tire}}(\mathbf{x}) > 0$, the requested force combination exceeds the admissible friction limits, thus progressively penalizing the solution.

5.1.4 Hard Bounds

In addition to the general soft constraints, the controller includes hard box bounds on the states $\mathcal{A}$ and inputs $\mathcal{U}$.

This configuration is a common practice in autonomous racing MPPC. Hard bounds enforce the physical limits of the system, which can only be violated in rare cases (e.g., sensor failures), ensuring that the MPPC always computes physically feasible solutions. Other limits, such as track margins, are treated as soft constraints with assigned weights. This is crucial, as the vehicle may occasionally exceed these margins (unless physically blocked by walls). In such cases, the solver must still be able to find a viable solution.

5.2 Planning layer

The planning layer is organized into a global and a local module. The global planner operates offline and generates reference trajectories, such as the racing line and the pitlane line, which serve as inputs for the local planner. The local planner runs online and consists of three main modules:

1. **Longitudinal planner (L-planner)**. Generates a speed profile along a reference trajectory, taking into account dynamic constraints. The desired performance level is set by the mission planner.

2. **Model Predictive Planner (MPP)**. Optimizes the global path and the longitudinal speed profile generated by the longitudinal planner, managing deviations from the nominal path. The MPP ensures physical feasibility and provides a highly-accurate reference for the control layer.

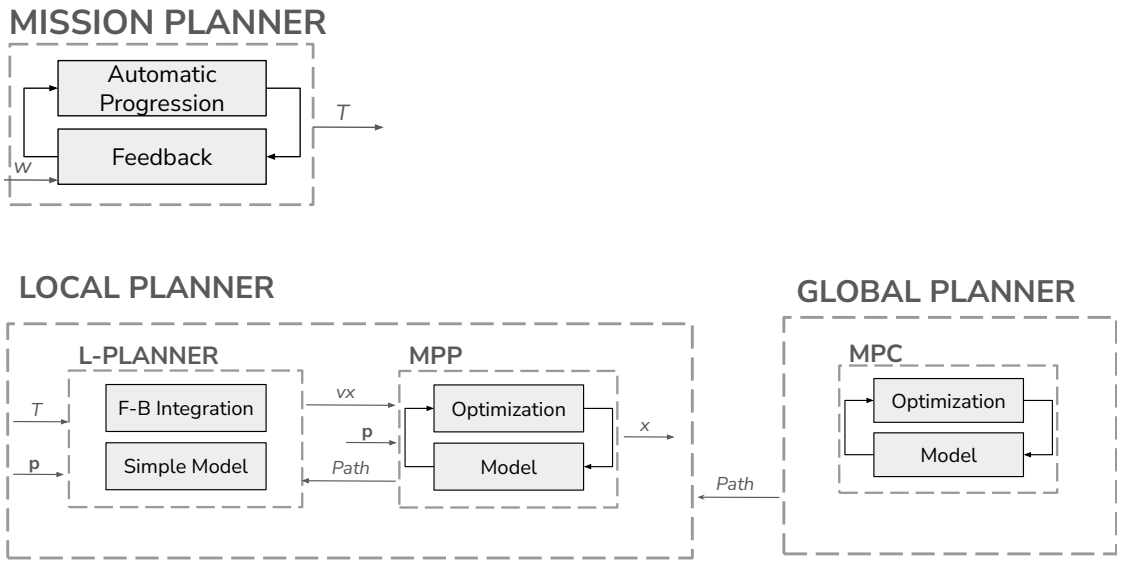


Figure 5.3: Planning diagrams. The mission planner sends performance target (**T**) to the longitudinal local planner. Accordingly, the longitudinal planner create a speed profile on the path. The MPP optimize path and speed, and generate an highly-accurate reference for the control layer.

3. **Overtake Logic (OT).** In multi-agent scenarios, this dedicated module computes the navigable, obstacle-free region of the track based on predictions of the opponents' future motion. This component is outside the scope of this thesis, as it has been extensively detailed by the authors in [75].

The high-level decision-making planner (Mission planner) is responsible for sending performance targets to the longitudinal planner, depending on predefined ideal progression and vehicle feedback.

5.2.1 Global planner

The global path is generated by solving a nonlinear optimal control problem formulated in the spatial domain [3]. The vehicle dynamics are modeled using a single-track formulation that accounts for nonlinear tire behavior under combined slip, while maintaining safety margins from the track boundaries. The cost function aims to maximize the progress rate $\dot{s}$ for optimal lap time, while penalizing rear slip angles α_r and steering rate $\dot{\delta}$ to promote stability and smoothness.

5.2.2 Mission planner

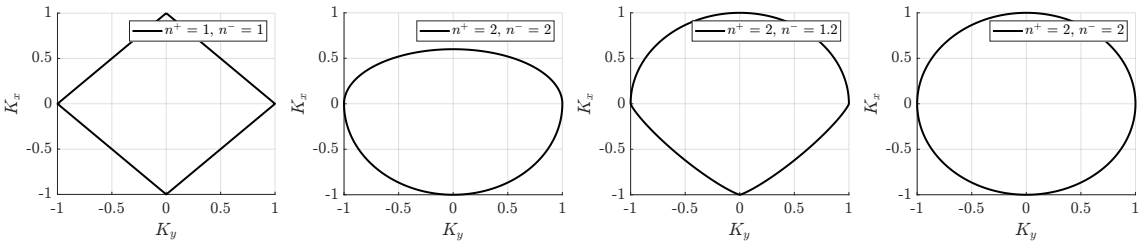


Figure 5.4: Four subplots illustrating different operating conditions of the mission planner. From left to right: full pure performance with no combined slip, reduced acceleration performance with full combined effect, reduced braking combined effects, and full pure and combined performance.

The mission planner provides high-level performance targets to the longitudinal planner (Sec. 5.2.3), with the flexibility to regulate lateral performance (K_y), longitudinal performance (separately for acceleration K_{x+} and braking K_{x-}), and the combined tire utilization effects during acceleration ($n^+ \in [1, 2]$) and braking ($n^- \in [1, 2]$). This module effectively implements a normalized friction ellipse representation of the front and rear axles, as illustrated in Fig. 5.4, allowing the planner to adjust the vehicle's operating point based on the desired combination of lateral and longitudinal performance.

Pure performance targets are expressed as normalized gains K_i with respect to the maximum achievable vehicle performance, which is determined by the estimated tire-road grip, specifically through the peak Pacejka coefficients. The gain can be set by the user based on the specific purpose of a test, for example, low-speed testing ($K_i \approx 60\%$ of full performance) to validate new sensors or hardware updates, or it can be derived from an ideal performance progression. Depending on temperature, the tire may deliver a grip $D_i(T)$, which is naturally lower than the optimal grip for a fully warmed tire (D_i). This translates into a gain defined as:

$$K_i(T) = \frac{D_i(T)}{D_i} \tag{5.14}$$

Where $D_i(T)$ can be identified experimentally offline. This allows the creation of a temperature-performance table, which can then be used to define an ideal warm-up progression target. Any discrepancies on lateral performance can be identified and corrected online using the grip estimation module, which identifies the actual D_y as described in Sec. 6.2. This could further refine the performance target; however, this online adjustment loop has not yet been tested in a fully autonomous setting.

Another possible configuration is to fix the mission planner performance at the full model potential, while adjusting the effective target performance by updating the tire model as a function of measured tire temperatures. However, this approach requires maintaining a comprehensive set of pre-identified tire models ready to be injected directly into the planner and controller, which reduces scalability and increases operational complexity.

In general, the mission planner executes a user-specified ideal performance progression aimed at reaching maximum performance in the shortest possible time. Initially, this progression increases all pure performance gains (K_i), and once the maximum is reached, it explores enhancements in combined effects (n), which have been observed to be the most challenging and critical to manage. The mission planner can decide to increase or decrease performance from lap to lap based on vehicle feedback, adjusting the ideal progression according to specific control and vehicle-dynamics metrics summarized in Tab.5.2.

Table 5.2: Summary of threshold-based performance modulation mechanisms.

Condition	**Detection**	**Action**
Lateral / heading error	Errors exceed nominal limits	Reduce lateral performance (K_y)
Negative slip ratios	Excessive negative slip or prolonged ABS activation	Reduce braking performance (K_{x-})
Positive slip ratios	Excessive positive slip or prolonged TC activation	Reduce acceleration performance (K_{x+})
Understeer angle threshold	Understeer angle exceeds limit	Reduce lateral performance (K_y)
Tire-temperature table	Cold tires or cool-down lap conditions	Clamp lateral (K_y) and acceleration (K_{x+}) performance

If an anomaly in the vehicle feedback is detected, the mission planner can react according to a configurable parameter:

- Retry the lap with the same settings (a confidence parameter, only active in certain stages of the progression), assuming the tires are still in the warm-up phase.

- Reduce the performance target to a value representing the average of the actual and previous performance levels. If this reduction is insufficient, it is further adjusted to the average of the new current and first values. This procedure repeats until convergence to an optimal performance level is achieved.

Each increase in performance is applied starting from a new lap. However, the mission planner operates on user-defined track zones, which generally correspond to specific track sectors. This ensures flexibility: if a particular corner has unusually low grip, it does not negatively affect the progression in higher-grip zones.

This layer is crucial for full autonomy, minimizing the need for human intervention during operation.

5.2.3 Longitudinal Planner

Despite the effectiveness of the global planner (5.2.1) in generating an optimal path, producing different speed profiles along that path is not straightforward or flexible.

To address this, a dedicated longitudinal planner has been designed to compute the velocity profile along the path using a forward-backward integration method. The role of the longitudinal planner, following the methodology described in this section, is to convert the normalized friction-ellipse-based performance requests provided by the mission planner into effective velocity targets for the MPP.

Its primary objective is to determine the most efficient velocity profile that minimizes lap time while ensuring compliance with dynamic constraints on lateral and longitudinal accelerations, according to the following system:

$$\begin{cases} v_x^2 \cdot \rho & < a_y^{\max}(v_x), \\ a_{x+}^{\max}(v_x) & > \frac{\Delta v_x^2}{2\Delta s} > a_{x-}^{\max}(v_x) \end{cases} \tag{5.15}$$

where ρ is the curvature at a given point, and $a_y^{\max}(v_x)$ defines the maximum allowable lateral acceleration at speed v_x. The longitudinal constraints are computed from the change in velocity Δv_x^2 divided by twice the distance increment Δs. Eqs. (5.15) ensures that this value remains within the allowable range, i.e., greater than $a_{x-}^{\max}$ (maximum permissible deceleration) and less than $a_{x+}^{\max}$ (maximum permissible acceleration) for the given speed.

The constraint vector $a^{\max}$ is computed using the friction law already mentioned in Eq.(3.13). For each axle, the maximum force is estimated from the tire model as:

$$\begin{cases} F_{x+,\max} & = D_{xr}\, F_{zr}, \\ F_{x-,\max} & = D_{xf}\, F_{zf} + D_{xr}\, F_{zr}, \\ F_{y,\max} & = D_{yf}\, F_{zf} + D_{yr}\, F_{zr}. \end{cases} \tag{5.16}$$

where D_i are the Pacejka peak coefficients and F_z is the normal load, which accounts for the static weight distribution, aerodynamic downforce, longitudinal load transfer, and 3D road effects (e.g., banking). Note that, for positive longitudinal acceleration, only the rear axle provides longitudinal force (rear-wheel-drive configuration).

Given the total available force, the corresponding maximum pure accelerations follow directly from Newton's law:

$$\begin{cases} a_{x+}^{\max} = \frac{F_{x+,\max}}{m}, \\ a_{x-}^{\max} = \frac{F_{x-,\max}}{m}, \\ a_{y}^{\max} = \frac{F_{y,\max}}{m}. \end{cases} \tag{5.17}$$

To describe combined-slip conditions, the admissible longitudinal and lateral accelerations are constrained through a generalized friction ellipse:

$$\left(\frac{a_x}{a_x^{\max}}\right)^n + \left(\frac{a_y}{a_y^{\max}}\right)^n \leq 1, \tag{5.18}$$

where n is a tunable exponent: $n = 1$ gives a linear combination between extremes, while $n = 2$ corresponds to a pure friction ellipse, according to Fig.5.4.

Finally, the Pacejka peak coefficients in Eqs. (5.16) are scaled using the mission planner gains (K_i), together with the exponent (n) in Eq. (5.18), to shape the desired friction ellipse and thus modulate vehicle performance accordingly. Despite this simplified modeling approach, the resulting speed profile provides a reliable, yet flexible initial guess for the subsequent optimization in the MPP.

5.2.4 Model Predictive Planner (MPP)

The Model Predictive Planner (MPP) operates in open-loop at 20 Hz with a sampling time $\Delta t = 0.04\,\text{s}$, over a horizon of $N = 150$ steps ($T = 6\,\text{s}$). In open-loop operation, the planner computes the path assuming that the previous iteration was executed perfectly, without using vehicle feedback to update its initial state $\hat{\mathbf{x}}$. Since the actual longitudinal position s of the vehicle may differ from the planned one, the planner reprojects the previous solution onto the current vehicle position and uses it as the initial condition for the new MPP execution.

The planner generates smooth and feasible paths within the track margins, which coincide with the physical boundaries under nominal conditions. In multi-agent scenarios, the OT module updates the track margins to account for opponent vehicles, enforcing predefined lateral and longitudinal safety distances. As a result, the planner is required to generate trajectories that lie within the resulting obstacle-free navigable region, while simultaneously satisfying the dynamic constraints imposed by the optimization problem. The main objectives of the MPP are detailed in Tab.5.3.

Compared to previous works [4,7,12], in which the planner operated in closed loop, an open-loop design was adopted for several reasons:

- In a closed-loop formulation, the planner recomputes the trajectory starting from the ego vehicle position at each iteration, artificially resetting the controller tracking error and reducing the effectiveness of feedback. In contrast, the open-loop approach allows the controller to naturally handle tracking errors.
- It provides a consistent and intuitive definition of tracking errors in the vehicle reference frame, such as the lateral deviation from the reference path, for evaluation of controller performance.
- It guarantees a stable and continuous reference trajectory for the controller, simplifying both the planning and control optimization problems, and allowing the controller (operating on a shorter horizon but at a higher frequency) to handle disturbances effectively.
- The open-loop MPP significantly simplifies tuning and validation. Since its behavior is independent of vehicle feedback, it can be evaluated in isolation under controlled simulation conditions, enabling reproducible tests and more systematic calibration using clearly defined scenarios.

The loop can be closed in specific conditions, such as:

- Lateral, heading, or longitudinal errors exceeding a predefined threshold.

Table 5.3: Main objectives of the Model Predictive Planner (MPP).

Objective	**Description**
Optimal trajectory adjustment	Optimize the offline computed global path and longitudinal planner speed profile, based on the current vehicle model or newly identified dynamics.
Optimal trajectory deviation	Compute a new physics-based trajectory for specific maneuvers, such as overtaking, line switch and obstacle avoidance, respecting the collision-free navigable area.

- A change in objective, e.g., line switching.

The planner also implements cost-blending among four main objectives: nominal, line switching, overtaking, and out-of-track scenarios, as detailed in Fig.5.5. These scenarios are closely related to trajectory generation, hence deemed planner tasks, while tracking-error compensation (both in nominal and high-dynamics conditions) is handled by the controller.

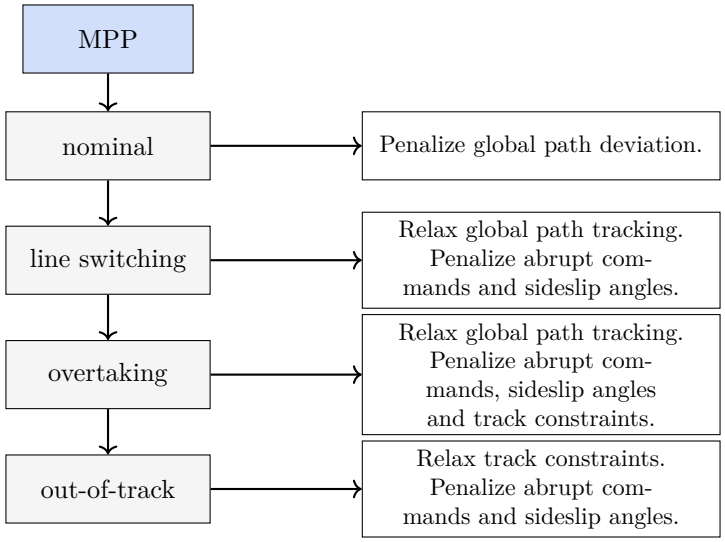


Figure 5.5: Planner process: vertical switch stages with corresponding cost explanations.

5.3 Control layer

The control layer is organized into three hierarchical levels, as shown in Fig.5.6:

- **Model Predictive Controller**. The main controller generates steering and longitudinal acceleration commands. It receives the reference trajectory from the local planner and aims at minimizing deviations from it while compensating for external disturbances.

- **Longitudinal controller**. It converts the MPC acceleration and speed requests into physically realizable throttle and brake commands. This layer handles engine and brake nonlinearities and ensures accurate longitudinal speed tracking.

- **Low-level slip ratio control**. A wheel slip controller supervises the longitudinal slip ratios and prevents them from exceeding unsafe levels. It may override or refine throttle and brake commands to ensure safe operation under varying grip conditions.

This design has been chosen for several benefits:

1. Avoids embedding a complex actuator model in the MPC. Mapping the high-level commands (a_x) to actual actuator inputs (throttle and brake) requires a detailed nonlinear model. Including such a model inside the MPC would significantly increase the problem complexity.

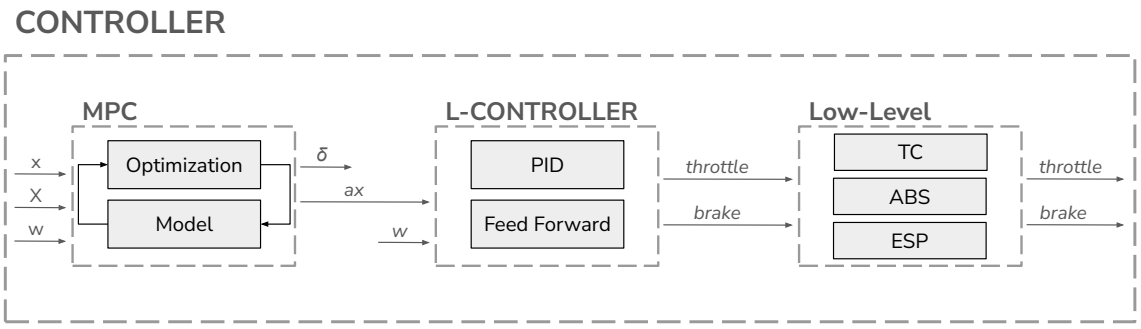


Figure 5.6: Control diagram. The MPC generate steering (δ) and high-level longitudinal command (a_x) for the vehicle. The latter is converted into throttle and brake by a longitudinal controller. Finally, pedals are further manipulated by a low level slip ratios controller.

2. Prevents a blow-up in the state dimension. Regulating longitudinal slip ratios directly within the MPC would require adding up to four additional wheel-dynamics states, drastically increasing computational cost.

3. Avoids numerical instability. The MPC sampling time is selected to capture lateral dynamics in real-time. However, longitudinal wheel dynamics are much stiffer and would require a far smaller time-step to remain numerically stable (making real-time MPC optimization impractical), as detailed in Sec.3.3.2.

4. Improves robustness to longitudinal model mismatch and disturbances. The dedicated longitudinal controller provides closed-loop compensation for model errors and velocity deviations, effects that an MPC cannot regulate effectively alone. This architecture ensures reliable tracking performance without increasing the computational burden of the optimization problem.

It is widely demonstrated that coupling lateral and longitudinal dynamics inside the MPC can theoretically improve performance, especially under combined-slip conditions [22, 36, 38]. However, it also introduces practical complications: a mismatch in longitudinal speed tracking may dominate the cost function, leading the solver to prioritize the speed correction over lateral tracking, potentially causing large lateral deviations from the desired path. For this reason, in our framework we include both lateral and longitudinal behavior in the optimization problem, but in a simplified form. Disturbances and longitudinal model uncertainties are handled more effectively by a dedicated non-optimization module. Furthermore, an additional longitudinal planner is inserted between the MPP and the MPC. This module operates on a zero-curvature reference and is solely responsible for smoothly merging the current vehicle velocity (used to initialize the MPC state) with the velocity profile generated by the MPP.

5.3.1 MPC

The MPC module operates in closed-loop at 100 Hz with the same sampling time ($\Delta t = 0.04\,\text{s}$), over $N = 64$ steps (time horizon $T = 2.6\,\text{s}$). The initial state is updated using vehicle feedback, filtered through localization and state estimation modules. The main objectives of the MPC are summarized in Tab. 5.4.
The closed-loop design follows this rationale:

Table 5.4: Main objectives of the Model Predictive Controller (MPC).

Objective	**Description**
Path-following	Computes the optimal control action to follow the planned path and speed.
Vehicle dynamics stability	Compensate for feedback deviations in high-driving and potentially critical scenarios, such as oversteer.

- Controller weights are primarily focused on tracking and stability, assuming the planner already provides a physically feasible, optimal trajectory for the current vehicle model and track conditions.
- Handling model mismatches, compensating errors, and ensuring path-following are entirely the controller's responsibility, rather than the planner's. Non-nominal scenarios (e.g., overtaking or off-track driving) should be handled by the planner.
- Track constraints are softly enforced, allowing the controller to prioritize stability during critical maneuvers.

Cost-blending is implemented between two main objectives: nominal and oversteer, as detailed in Fig.5.7. These scenarios are tightly linked to trajectory tracking and are therefore considered controller-level tasks.

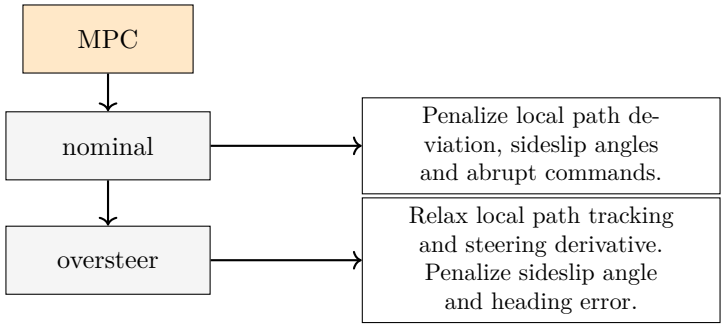


Figure 5.7: Controller process: vertical switch stages with corresponding cost explanations.

5.3.2 Longitudinal Controller

The MPC provides a reference longitudinal acceleration a_x^T. The longitudinal controller converts the longitudinal acceleration into throttle and brake commands using an automatic separation between positive and negative target forces. The target longitudinal force is computed from a simplified longitudinal dynamics model:

$$F_x^{\mathrm{T}} = \left(m + \frac{J_{\mathrm{eq}}}{r_w^2}\right) \cdot a_x^T + F_{\mathrm{roll}} + F_{\mathrm{d}}, \tag{5.19}$$

where J_{eq} is the equivalent rotational inertia of the drivetrain, r_w the wheel radius, F_{roll} the rolling resistance, and F_{d} the aerodynamic drag. The longitudinal force is automatically mapped to either throttle or brake:

- **Throttle (positive F_x^{T})**: converted into an engine torque reference:

$$T_{\mathrm{eng}} = \frac{F_x^{\mathrm{T}}\, r_w\, \tau_i\, \tau_d}{\eta_t}, \tag{5.20}$$

where τ_i is the current gear ratio, τ_d is the final drive ratio, and η_t is the transmission efficiency. The engine torque (T_{eng}), together with the current engine speed (RPM), is used to query a heuristically constructed 3D engine map, providing the throttle feedforward command. Braking (negative) force naturally generate a negative torque target, thus zero throttle command.

- **Brake (negative $F_x^{\mathbf{T}}$)**: converted into a brake pedal command as:

$$B = \frac{F_x^{\mathrm{T}} + F_{\mathrm{eng,brake}}}{(C_{bf} + C_{br}) \cdot \mu_d}, \tag{5.21}$$

where C_{bf} and C_{br} are the front and rear brake coefficients determined by the geometry and characteristics of the braking system, μ_d is the friction coefficient between brake pads and discs, and $F_{\mathrm{eng,brake}}$ is the engine braking contribution.

The engine braking contribution depends on the engine operating conditions and is implicitly captured through the engine torque map used in the feedforward computation. For small deceleration requests, the equilibrium torque required by Eq.(5.20) corresponds to a low engine torque value, which is translated into a reduced throttle command through the engine map. For larger deceleration demands, the requested torque becomes negative and the controller switches to braking actuation while the throttle command naturally approaches zero.

The controller also enforces physical constraints by limiting the target longitudinal force using Eqs. (5.17):

$$\begin{cases} F_x^{\mathrm{T}} \leq D_{xr}\, F_{zr}, & \text{(rear-wheel drive traction)} \\ |F_x^{\mathrm{T}}| \leq D_{xf}\, F_{zf} + D_{xr}\, F_{zr}, & \text{(braking)} \end{cases} \tag{5.22}$$

Any discrepancies in the feedforward actions due to force estimation errors or other modeling inaccuracies (such as track slope) are compensated by a PID feedback controllers, whose contributions are added to the feedforward commands.

Additionally, a look-ahead mechanism is used to select the optimal target along the MPC reference horizon. This allows separate handling of throttle and brake commands, accounting for their inherently different response times.

5.3.3 Low-Level

Low-level controllers are implemented because the higher-levels do not use slip ratio in feedback, even though its effects on tire force are accounted for both in the MPPC and in the feedforward models. Moreover, low-level actuator management becomes much simpler. In general, autonomous systems allow independent control of braking subsystems, enabling advanced functionalities such as ABS and ESP. Implementing such control directly within an MPC framework, while potentially offering high performance, would require solving an optimization problem that demands careful formulation and tuning.

Antilock-Brake-System. The Anti-lock Braking System (ABS) module regulates wheel slip during braking to maintain vehicle stability and maximize braking performance. Inputs include wheel speeds, vehicle longitudinal velocity, yaw rate, and raw brake requests.

For each wheel, the module computes the slip ratio by comparing the measured wheel speed with a reference speed derived from the vehicle longitudinal velocity and yaw rate:

$$\begin{aligned} \kappa_{\mathrm{FL}} &= \left| \frac{v_{xFL}^{\mathrm{ref}} - v_{x\mathrm{FL}}}{v_{xFL}^{\mathrm{ref}}} \right|, \quad \kappa_{\mathrm{FR}} = \left| \frac{v_{xFR}^{\mathrm{ref}} - v_{x\mathrm{FR}}}{v_{xFR}^{\mathrm{ref}}} \right|, \\ \kappa_{\mathrm{RL}} &= \left| \frac{v_{xRL}^{\mathrm{ref}} - v_{x\mathrm{RL}}}{v_{xRL}^{\mathrm{ref}}} \right|, \quad \kappa_{\mathrm{RR}} = \left| \frac{v_{xRR}^{\mathrm{ref}} - v_{x\mathrm{RR}}}{v_{xRR}^{\mathrm{ref}}} \right|, \end{aligned} \tag{5.23}$$

where the reference wheel velocities account for vehicle yaw motion:

$$\begin{aligned} v_{xFL}^{\mathrm{ref}} &= v_x - r \cdot \frac{t_f}{2}, \quad v_{xFR}^{\mathrm{ref}} = v_x + r \cdot \frac{t_f}{2}, \\ v_{xRL}^{\mathrm{ref}} &= v_x - r \cdot \frac{t_r}{2}, \quad v_{xRR}^{\mathrm{ref}} = v_x + r \cdot \frac{t_r}{2}, \end{aligned} \tag{5.24}$$

The slip ratios are then compared with a reference threshold κ_{ref} to determine whether corrective action is required. If the slip exceeds the reference, a proportional controller computes a brake correction, which reduces the commanded brake pedal to prevent wheel lock. In addition, a pulse intervention is applied when the slip exceeds a secondary threshold κ_{bb}, allowing faster reaction in extreme locked-wheels events.

The module includes rate limiters logic to not exceed braking actuator speed. ABS aggressiveness and intervention thresholds can be tuned through a virtual knob parameter. The outputs are the adjusted brake commands for each wheel.

Traction-Control The Traction Control (TC) module manages wheel slip by adjusting throttle and rear brake commands. Inputs include the vehicle's longitudinal and lateral velocities, wheel speeds, throttle request, brake request.

The module first computes the vehicle sideslip angle and wheel slip ratios, which are then compared with reference limits: slip threshold κ_{ref}, brake intervention threshold κ_{bb}, and sideslip reference β_{ref}. For Traction Control, the slip ratio is computed using a single-track approximation, i.e., by averaging the left and right wheel speeds (v_{xR}) instead of treating the wheels independently:

$$\kappa_{TC} = \frac{v_{xR} - v_x}{v_x}. \tag{5.25}$$

If the measured slip exceeds κ_{ref}, a proportional controller computes a corrective action by reducing throttle and, if enabled, applying brake force when the slip exceeds κ_{bb}. A simple proportional action is also implemented for the sideslip angle, reducing throttle without engaging the brakes, thus addressing pure lateral dynamics. The aggressiveness of the TC can be adjusted via a virtual knob, which sets the proportional gain, the maximum brake force, and the slip limits threshold.

5.4 Pratical considerations and Future improvements

In this section, we discuss several practical aspects and framework limitations encountered during the design and validation process. These considerations mainly serve to highlight potential directions for future work aimed at further enhancing the proposed solution. They are also intended as an additional contribution to the literature, which predominantly focuses on theoretical designs or simulation-only studies, by complementing them with insights derived from real-world deployment and system-level integration.

Table 5.5: Nominal weights used in the MPP and MPC formulations. The legend of symbols is presented below.

(a) Planner weights

q_n	q_μ	q_{v_x}	q_r	q_β	$r_{d\delta}$	r_{J_x}	s_{track}	s_{tires}
++	++	++	++	++++	+++	++	+++	+++

(b) Controller weights

q_n	q_μ	q_{v_x}	q_r	q_β	$r_{d\delta}$	r_{J_x}	s_{track}	s_{tires}
++++	++++	+++	++	+++	++	+	+	+

Symbol	**Meaning**
++++	Strong prioritization
+++	Medium
++	Low
+	Weak prioritization

5.4.1 Cost function tuning

Tuning the cost function is a critical aspect when employing a nonlinear tire model within the MPPC, and the selection of the weight values must be performed with great care. An unbalanced weight set can lead to solver instability, producing suboptimal solutions, convergence outside the allowed computation time, or even physically meaningless (divergent) trajectories. The objective function weight used in the MPP and MPC formulations are highlighted in Tabs. 5.5.

A common application consists of an MPC tasked with tracking both a reference path and a reference velocity. In path-following problems, it is common to assign high weights to both lateral deviation and velocity tracking. However, when the reference velocity is not physically feasible (i.e., it exceeds the capabilities of the model or its grip parameters) the optimization problem becomes ill-posed. In such conditions, the solver attempts to enforce an impossible maneuver given the MPC model's physics, and no optimal solution exists for the chosen cost weights. Relaxing either the velocity- or the path-tracking terms mitigates the issue, but this highlights the necessity of extensive testing across a large variety of scenarios. Such a tuning process is often cumbersome and time-consuming, as the stability and reliability of the optimization strongly depend on the specific combination of weights, reference values, and modeled vehicle dynamics. As a result, it is not feasible to thoroughly design and test every possible scenario the planner and controller may encounter.

In our analysis, we identified three particularly critical points that must be addressed:

1. The longitudinal planner relies on oversimplified (and often optimistic) vehicle model compared with the one used in the MPP. This mismatch may lead the former to propose velocities that are physically infeasible for the more accurate MPP model.

2. Similarly, the MPP and MPC not sharing the same model and parameters. In such cases, the MPP could generate trajectories or accelerations that the MPC cannot track. For this reason, in our framework both the MPP and MPC share the same vehicle model.

3. The MPP must deviate from the nominal path, but the reference velocity provided from the longitudinal planner remains excessively high for the new trajectory curvature or local grip conditions.

As a result, the MPC (and MPP as well) attempts to track a trajectory that is no longer compatible with the dynamic constraints, causing instability or solver divergence.

Several planning-control architectures are proposed in the literature: one approach simplifies the planning layer, relying on a robust controller to handle disturbances; another simplifies the control layer, trusting a complex planner to generate physically feasible references. In our framework, we adopt a hybrid solution: the same vehicle model complexity is used for both MPP and MPC by sharing the same model and parameters. This ensures that:

- The MPP generates physically accurate and feasible references for all maneuvers, fully exploiting the model's capabilities, thus filtering unfeasible reference from oversimplified higher-level layer.
- The MPC can focus on aggressive tracking with reduced risk of infeasible commands, since the reference it receives is designed to be executable, still being able to manage high-dynamic scenarios.

This separation allows the planner to be tuned for smoothness and stability, producing high-quality references, while the controller can be tuned for maximal tracking performance, knowing that the input trajectory is already physically consistent.

To mitigate the third critical point, the longitudinal planner uses the curvature of the path computed by the MPP rather than the offline global trajectory (except for the very first iteration), generating a speed profile that is optimal with respect to the actual path, thus accounting for racing-line deviations. While the trajectory is inherently one iteration delayed, since the longitudinal planner runs upstream of the MPP, the short execution interval (20 Hz) and the assumption of minimal deviation between successive predictions make this approach an effective compromise. Moreover, the MPP does not aggressively track the speed profile, naturally filtering small imperfections introduced by the oversimplified models at higher planning layers.

Point one remains an open issue, particularly under highly unbalanced vehicle dynamics (understeer or oversteer vehicle). The point-mass model implemented in the longitudinal planner generates an ideal speed profile that fully exploits each axle's performance. Using a nonlinear single-track model, which captures front-rear axle unbalance, can create a large mismatch between planning layers, potentially destabilizing the solver. To mitigate this, a low weight is applied to speed tracking in the MPP, reducing instability at the cost of suboptimal performance for balanced setups. Additionally, unsafe conditions may arise when online vehicle parameter updates are applied to the MPP: if the updated vehicle balance is incompatible with the chosen weight set, a substantial mismatch between the longitudinal planner and the MPP may occur, risking unsafe solutions.

5.4.2 Simplified longitudinal dynamics

As highlighted by the literature analysis in Sec. 2, relying on accurate models is crucial in autonomous racing applications. Sec. 3 also illustrates the complexity of using more detailed models, which may be theoretically essential for certain tasks but challenging in practice. Although including longitudinal slip might seem unnecessary in nominal conditions, autonomous racing is dominated by edge cases and unpredicted situations.

In the simplified formulation adopted in Sec. 5.1, wheel rotational dynamics are neglected to reduce model complexity, computational burden, and numerical instability. Thus, the longitudinal tire force F_x is estimated directly from the measured vehicle acceleration ($F_x = ma_x + F_d$), rather than being computed from the tire slip ratio through a

tire model. This approximation does not provide information about the tire slip state or the remaining force generation margin. As a result, while the reconstructed longitudinal force is consistent with the measured vehicle acceleration, the MPPC cannot determine whether the tire is operating in the linear or saturated regime, nor whether additional longitudinal force requests are feasible. Furthermore, this approximation may affect the accuracy of the combined-slip constraint formulation (Eq. (5.13)) and potentially reduce controller robustness during aggressive maneuvers or low-grip conditions.

Including low-level controllers, such as TC and ABS, helps mitigate some risks. However, they do not fully address the primary limitation: without the predictive capability of the MPPC, these controllers cannot anticipate future vehicle states or optimize control actions over a horizon, limiting overall performance under poor driving or extreme vehicle states.

To mitigate this, a future improvement could be to incorporate in the MPC state an artificial longitudinal acceleration (or force) to account for the slip ratio effect, providing more accurate combined slip estimation even when longitudinal acceleration feedback is unavailable. This solution may increase model effectiveness while not compromising real-time algorithm capabilities.

5.4.3 Impact of Model Linearization

In automatic control, especially in real-time applications, linear algorithms provide modeling simplicity and computational efficiency. To balance model accuracy with real-time feasibility, the nonlinear vehicle models are linearized around specific operating points, with the Jacobians generated at compile time, thus not influencing MPC execution time. The linearization is performed around the predicted state sequence from the previous MPC iteration, which provides the closest available operating point for an accurate approximation, assuming that the new state evolution does not significantly deviate from the previous prediction. This assumption is justified both by the accuracy of the model and by the algorithm's high execution frequency (100 Hz), which limits the deviation between successive predictions. Any discrepancies caused by suboptimal linearization points are mitigated by the Sequential Quadratic Programming (SQP) scheme. At each SQP iteration, the state sequence is updated, and the model is relinearized around the new solution, progressively improving accuracy and convergence.

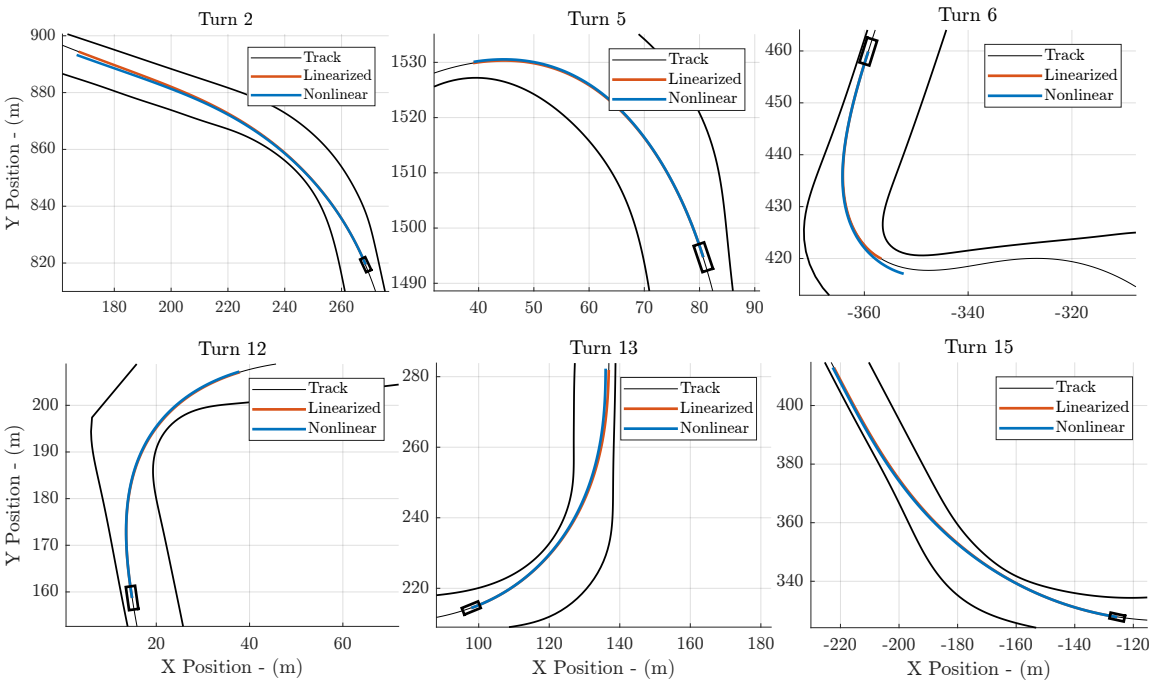
In this section, we compare the nonlinear single-track model with its linearized counterpart, using the same timestep and integration scheme adopted in the MPC formulation described in Subsec. 5.3.1. Results are shown in Fig. 5.8. The linearized model takes the well-known form:

$$\dot{\mathbf{x}} = \mathbf{g} + A \cdot (\mathbf{x} - \mathbf{x}_o) + B \cdot (\mathbf{u} - \mathbf{u}_o), \tag{5.26}$$

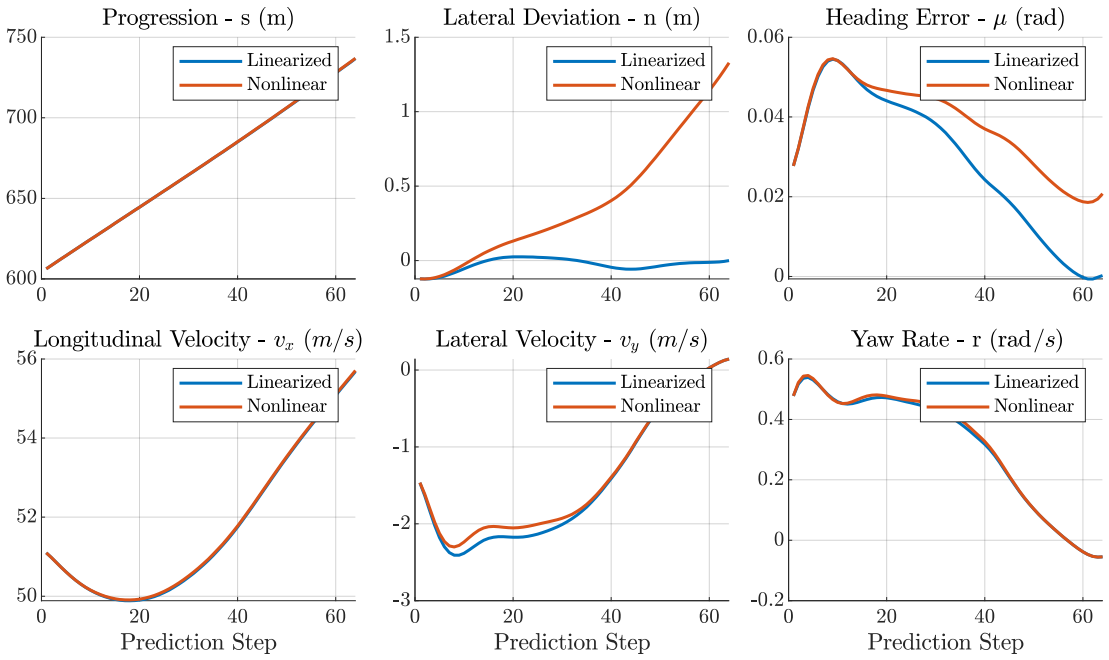
where $\mathbf{g} = f(\mathbf{x}_o, \mathbf{u}_o)$ accounts for the nonlinear dynamics evaluated at the operating point. The Jacobian matrices A and B are computed through automatic differentiation as:

$$A = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\Big|_{(\mathbf{x}_o, \mathbf{u}_o)}, \quad B = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\Big|_{(\mathbf{x}_o, \mathbf{u}_o)}. \tag{5.27}$$

From a given initial state $\mathbf{x}_0$, the MPC computes the optimal input sequence, which is then applied both to the linearized model (used in the optimization) and to the nonlinear model. Since the optimization relies on the linearized dynamics, its predicted trajectory remains aligned with the reference path (Fig. 5.8a). When the same input sequence is applied to the nonlinear model, small discrepancies in the state evolution (Fig. 7.6) accumulate over the $T = 2.6\,\mathrm{s}$ horizon, resulting in a wider path in low-speed turns (T5, T6) and a



(a) Path comparison between the nonlinear single-track model and its linearized counterpart. The black lines indicate track margin and optimal path.



(b) State variable comparison between the nonlinear and linearized single-track models in turn 2, showing the largest deviation among all turns.

Figure 5.8: Comparison between nonlinear and linearized single-track models.

tighter path in high-speed sections (T2), with lateral errors up to 1 m. In closed loop, the high update rate of the MPC mitigates these deviations, bounding the actual lateral error within $n = 40\,\text{cm}$ (Fig. 7.5).

5.4.4 Cascade of jerk command

Using jerk as the control input in MPPC formulations (thus including acceleration and velocity in the state vector) is a common practice, as it provides a direct mechanism to promote smooth control actions. This is particularly beneficial during cornering and traction phases, where smooth longitudinal commands are essential to preserve vehicle stability and maximize performance. However, there exist operating conditions in which aggressive longitudinal actions are fundamental to achieve high performance, most notably during heavy braking. In these scenarios, a high penalty on jerk becomes a double-edged sword. Braking dynamics are required to be extremely fast, especially in formula cars, where the initial phase of high-speed braking must fully exploit the available aerodynamic downforce. Excessive jerk penalization may overly smooth the velocity profile, significantly anticipating the braking point and leading to a tangible loss in lap time. In the proposed framework, this effect is further amplified by the cascade structure of the MPPC. Since both layers regulate jerk with respect to their respective references, the resulting longitudinal behavior may become excessively conservative, causing a substantial degradation in braking performance. Conversely, an excessively weak penalization on jerk is strongly discouraged, as it may induce oscillatory control actions during pure cornering or constant-speed tracking, being also too reactive in case of noisy feedback.

Representative examples are shown in Fig. 5.9, where three braking scenarios are highlighted. The figure reports the predicted velocity profiles over the horizon, expressed as a function of longitudinal progress. In the first column, the braking phase for turn 1 is shown. The MPP anticipates the braking point to promote a smoother maneuver. Subsequently, the MPC further smooths the planner reference. Overall, the braking point is anticipated by approximately $\Delta s \approx 20$ m compared to the ideal reference. A comparable

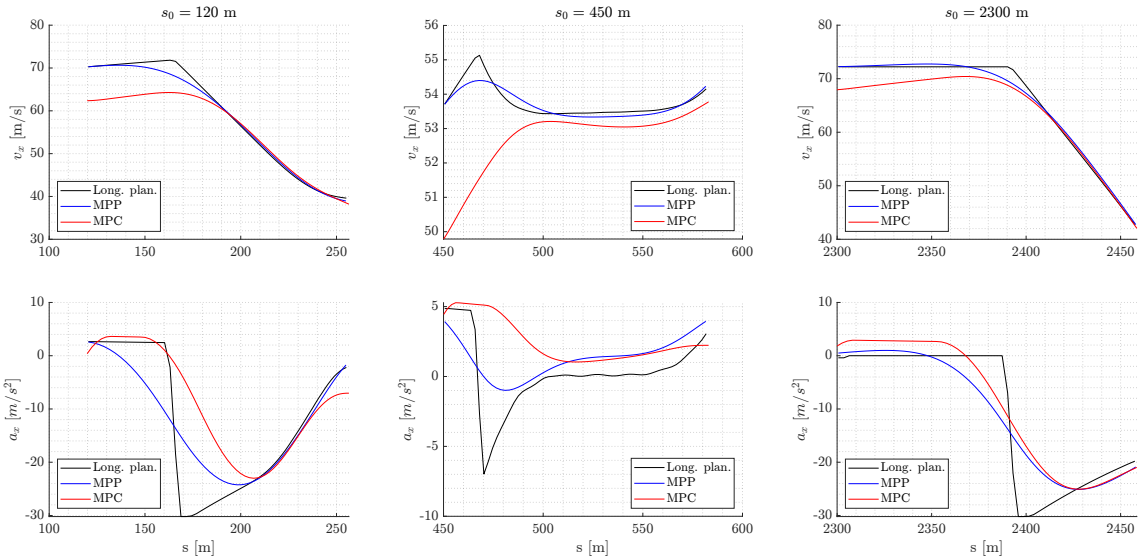


Figure 5.9: Comparison between velocity and acceleration cascade of the motion framework: Speed profile is created by the longitudinal planner (black line), then adjusted by the MPP (blue line), and finally tracked by the MPC (red line).

effect can be observed in the third column, where a noticeable overshoot of the target speed emerges, particularly at the MPP level. This behavior is once again a direct consequence of the high jerk penalization, leading to an anticipation of the braking point of about $\Delta s \approx 27$ m. Conversely, the benefits of jerk penalization clearly emerge in fast cornering scenarios. In the second column (turn 2), the jerk-free longitudinal planner generates a sharp braking action prior to corner entry, which may be infeasible or destabilizing for the given vehicle and track configuration. By leveraging jerk regularization together with a more accurate vehicle model, the MPP effectively filters this aggressive reference, generating a physically consistent velocity profile that can be robustly tracked by the MPC. These results highlight the need for a careful and context-dependent design of jerk penalization when cascaded planning and control layers are employed.

5.4.5 MPPC execution time

The proposed framework runs efficiently in real time. Among all modules, the MPP and the MPC represent the main bottlenecks for further improving overall execution speed. However, thanks to the implementation described in Sec. 5.1, both solvers still leave room for future optimizations.

Fig. 5.10 reports the execution time of the two modules. The MPP consistently operates below 0.05 s (20 Hz), while the MPC runs below 0.01 s (100 Hz). The data were recorded during a non-nominal overtaking maneuver, as indicated by the slight increase in planning time, where peaks of up to 0.02 s were observed. Test were executed in the Dallara EAV25 SF vehicle, equipped with a rugged RGS-8805GC HPC server and powered by the AMD EPYC^{TM} 7003 series "MILAN" processor and 512 GB memory capacity.

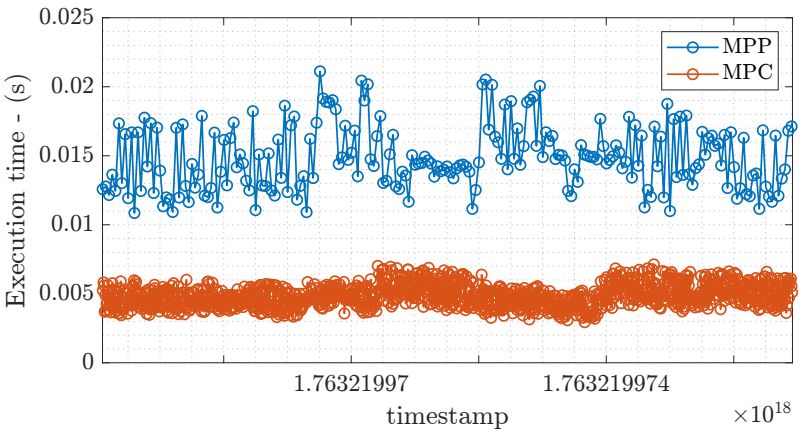


Figure 5.10: MPP and MPC execution time.

Chapter 6

Identification Method

This section presents a validated framework for online system identification, designed to operate in real-time by leveraging the sensors commonly available on autonomous vehicles and a set of model-based algorithms. These modules are crucial for capturing the vehicle's dynamic behavior and for adapting the software's performance to the actual operating conditions.

6.0.1 Modules architecture

An overview of the architecture is shown in Fig. 6.1. The framework is composed of two tightly coupled modules:

- **State Estimation (LOP-UKF)**: An Unscented Kalman Filter that, together with the localization module, provides reliable estimates of the vehicle's lateral dynamic states, in particular the lateral velocity and yaw rate (v_y, r).
- **Grip Estimation**: A tire-road friction identification algorithm that estimates the parameters of the tire model, specifically the coefficients governing lateral forces.

Both modules rely on model-based formulations and exchange information regarding the vehicle state ($\mathbf{X}$), sensor measurements ($\mathbf{w}$), and model parameters ($\mathbf{p}$). Initial parameter values are set using prior knowledge from simulation, selecting the appropriate friction level based on the vehicle, track, and environmental conditions, following the methodology

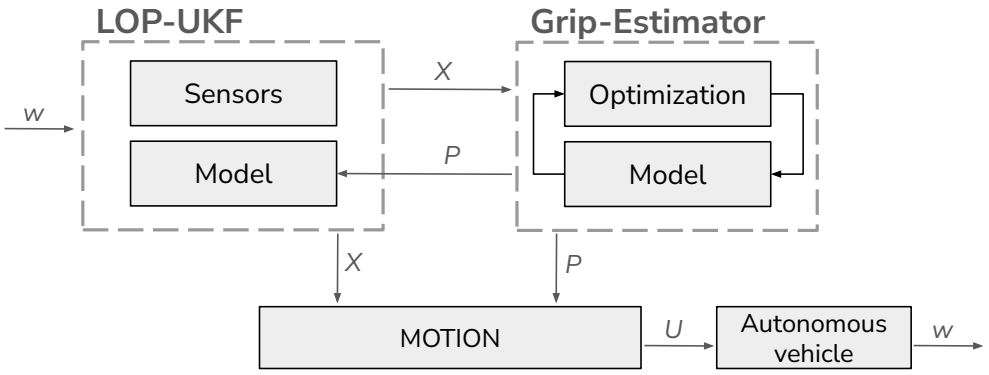


Figure 6.1: Scheme of the system identification framework, including the state-estimation filter and the online tire-road friction model calibration module. In the figure, $\mathbf{w}$, $\mathbf{X}$, and $\mathbf{p}$ are defined as in Sec. 6.0.2.

presented in this chapter. The two modules work synergically following the logical flow below:

1. **Step 1 - State estimation:** The LOP-UKF, supported by onboard sensors, estimates the fundamental lateral-dynamics quantities (v_y, r) and forwards these estimates to the grip-estimation module.

2. **Step 2 - Grip estimation:** Using the estimated vehicle states ($\mathbf{X}$) and the measured inputs, the algorithm identifies the tire model parameters and evaluates deviations from the initial model. If the discrepancy exceeds a configurable threshold, updated parameters are published.

3. **Step 3 - Model update:** The updated parameters ($\mathbf{p}$) are fed back into the UKF to ensure consistent and accurate filtering of sensor data, and to maintain robustness in case sensing sources become unavailable.

Detailed implementation on the state-estimation filter (Sec. 6.1) and the friction-estimation module (Sec. 6.2) are provided in the subsequent sections. Overall, the framework is designed to support three key applications:

1. **Model Predictive Control:** by providing accurate and consistent estimates of lateral velocity and yaw rate, thereby enhancing prediction and control performance.

2. **Performance Parameter Updates:** by identifying deviations from the assumed surface conditions and enabling real-time adaptation of model parameters for all model-based planning and control algorithms.

3. **Mission Planning:** by monitoring vehicle dynamics, such as sideslip and understeer angle, to detect potentially unsafe conditions and adjust high-level performance targets accordingly.

6.0.2 Sensor setup

The autonomous vehicles used for testing are equipped with a variety of sensors, including: yaw rate and accelerations (IMUs), vehicle linear velocities (from optical sensors and LiDAR-based odometry), vertical loads (load cells, used solely for validation), wheel speeds, and steering angle. An overview of the sensors used for system identification, among those available on the vehicle, is provided in Fig. 6.2.

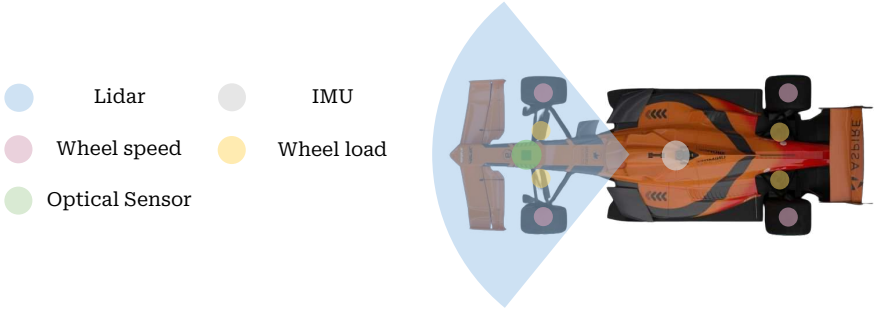


Figure 6.2: Sensor setup of the autonomous vehicle (Dallara EAV24 Super Formula) used for the identification process.

The sensor measurements are collected in the vector $\mathbf{w}$, the filter outputs are collected in the vector $\mathbf{X}$, and the tire-road friction parameters, estimated using a simplified Pacejka model according to (3.14), are collected in the vector $\mathbf{P}$:

$$\mathbf{w} = [a_x,\ a_y,\ v_x,\ v_y,\ r,\ \omega_{\mathrm{FL}},\ \omega_{\mathrm{FR}},\ \omega_{\mathrm{RL}},\ \omega_{\mathrm{RR}},\ \delta]$$

$$\mathbf{X} = [v_y,\ r,\ a_y,\ \alpha_f,\ \alpha_r]$$

$$\mathbf{P} = [B,\ C,\ D,\ E,\ S_h]$$

6.1 Lateral velocity estimation

Accurate estimation of lateral velocity is essential for applications such as planning, control, tire-road friction estimation, and localization. It enables the computation of key vehicle dynamics quantities, such as the sideslip angle β and the understeer angle $\mathrm{U} = |\alpha_f| - |\alpha_r|$, which are critical for vehicle stability and control, especially in challenging scenarios.

Although model-based approaches are theoretically powerful, they are often overlooked due to the difficulty of accurately identifying parameters, particularly tire properties, which depend on precise velocity measurements, therefore introducing strong nonlinearities into the estimation process. While such measurements traditionally required expensive sensors, recent works [45–47, 76] demonstrate that comparable accuracy can be achieved using standard onboard sensors (IMUs, GPS, cameras, LiDAR) combined with advanced state estimation algorithms. In [5] we presented LOP-UKF, a novel method for estimating lateral velocity by combining LiDAR odometry with predictions from the Pacejka tire model, integrated within an Unscented Kalman Filter (UKF).

6.1.1 LOP-UKF overview [5]

The LOP-UKF filter is designed to estimate key lateral dynamics parameters, namely lateral velocity, lateral acceleration, and yaw rate. To achieve this, it fuses measurements from multiple sensors, including two IMUs, two GNSS units, a steering encoder, phonic wheels, and three LiDARs.

The Unscented Kalman Filter is particularly well-suited for this task because, unlike linearized approaches, it does not rely on approximating the state function around a single working point. Instead, it captures the full probability distribution using deterministic sigma points through the unscented transformation. In our implementation, the Pacejka tire model is used for state prediction, while LiDAR odometry provides the measurement updates. This approach is computationally efficient, operating at 125 Hz in real time.

Experimental results demonstrate that combining LiDAR odometry with a Pacejka-based UKF yields reliable lateral velocity estimates, even when measurement updates are unreliable or produce non-physical values, as thoroughly examined in [5]. The filter remains accurate provided the tire parameters are calibrated under similar grip conditions. The main open points concern the integration of real-time adaptable tire models, which would allow the LOP-UKF to generalize across different vehicle types and driving conditions, and the inclusion of additional sensors to improve measurement redundancy and correction robustness.

6.1.2 Current Version

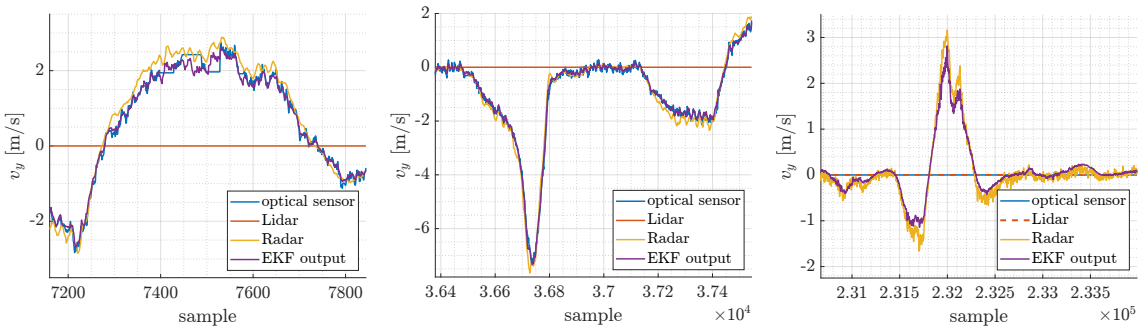
The filter has been further developed to enhance real-time performance and estimation robustness. The current implementation is based on an Extended Kalman Filter (EKF), which benefits from widely available C++ libraries that provide clean, efficient, and maintainable code (essential for ongoing algorithm development). Although this choice sacrifices some accuracy from a fully nonlinear model perspective, it allows greater flexibility in incorporating additional lateral velocity sensors. Key improvements in the current version include:

- **Integration of multiple sensors:** The filter now includes Kistler SF-Motion optical sensors for lateral velocity (initially used only for validation) and radar-based velocity measurements, enhancing robustness to sensor failures.
- **Modular architecture:** The filter input and output interfaces are now flexible, allowing adaptation to different applications and vehicle configurations.
- **Integration with the Grip-Estimation Module:** This module can update tire model parameters in real time when significant discrepancies with initial values are detected. This overcomes the previous limitation where filter performance heavily depended on pre-calibrated tire parameters under similar grip conditions.

In the current tuning, the filter assigns low covariance to the optical sensor measurement, reflecting a high level of trust, as this sensor provides the most accurate and reliable lateral velocity measure. The LiDAR-based velocity is assigned a medium-high covariance, corresponding to a moderate level of trust, since it is generally accurate but less robust under certain conditions (such as map occlusions). Conversely, the radar-based velocity measurements are assigned a high covariance, reflecting low trust, as this sensor is prone to noise for certain speed range, and remains an under-development solution.

6.1.3 Main results

In Fig. 6.3, we present three case studies demonstrating the robustness of the advanced lateral velocity estimation filter. The selected edge-case scenarios focus on sensor failures, high-dynamics maneuvers, and low model accuracy.



(a) Case study 1 (b) Case study 2 (c) Case study 3

Figure 6.3: Lateral velocity estimation performance across three challenging scenarios, including loss of primary sensor sources (a, c) and high-dynamics maneuvers (b).

Case study 1 (Subfig.6.3a): in this dataset, the LiDAR velocity source is not used by the estimator. The filter relies only on the optical sensor and on the radar as direct

measurements of lateral velocity. However, the optical sensor exhibits hardware issues, clearly visible from its step-like signal (blue line). The radar, while consistent, tends to overestimate the velocity. Despite these limitations, the filter maintains an accurate and coherent estimate across the entire corner thanks to the stabilizing effect of the model.

Case study 2 (Subfig.6.3b): This dataset includes a pronounced oversteer event, with lateral-velocity peaks exceeding 7 m/s. The SLAM-based velocity source is disabled in this run. Nevertheless, supported by the optical sensor, the filter successfully tracks the sharp dynamics and provides a precise estimation of v_y, enabling the controller to react appropriately and restore vehicle stability.

Case study 3 (Subfig.6.3c): In this scenario, both the LiDAR and the optical sensor are unavailable due to a sensor failure, leaving the radar as the sole direct source of lateral velocity information. The radar likely overestimates the magnitude of v_y, although no additional ground-truth measurement is available for confirmation. Furthermore, model inaccuracies arise from road configurations, such as a large bump that introduces a mismatch in vertical load estimation. Despite relying only on the model, radar, and secondary vehicle sensors (IMUs), the filter still produces a stable and physically plausible estimate of lateral velocity.

6.2 Grip Estimation Algorithm

The grip estimation module receives as input the vehicle lateral dynamics (v_y, r) from the filter, as well as the linear accelerations (a_x, a_y) and longitudinal velocity v_x from the localization module. These measurements are processed through a simplified single-track vehicle model (Sec. 6.2.1) to compute axle-level characteristics (α_i, μ_{yi}), where $\mu_y = \frac{F_y}{F_z}$ represent the normalized lateral force and $i \in f, r$ denotes the front and rear axles. The resulting data are then used in a nonlinear least-squares optimization (Sec. 6.2.2) to identify the tire model parameters $(\mathbf{p})$, thereby extracting information on the tire characteristics and, consequently, the tire-road friction coefficient.

The resulting data (α_i, μ_{yi}) are stored in structures with a maximum capacity of $N_{\max}$ samples. When a new data point is inserted, it is compared to existing entries: if a previously stored point lies within a certain spatial threshold and the minimum time difference has elapsed, it is replaced by the new point:

$$\mathbf{x}_{\text{new}} \to \mathbf{x}_{\text{old}} \quad \text{if} \quad \|\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{old}}\| < d_{\text{thresh}} \text{ and } t_{\text{new}} - t_{\text{old}} > \Delta t_{\min}. \tag{6.1}$$

This strategy offers two main advantages:

1. The point cloud is continuously updated with the most recent measurements, naturally capturing the evolution of the system.

2. The number of stored points remains bounded, avoiding memory saturation while maintaining a representative dataset.

The structure is implemented as a Cover Tree, allowing efficient nearest-neighbor queries for duplicate checking and replacement. Points are only included if the vehicle speed exceeds a minimum threshold and the slip angles are larger than a configurable minimum $|\alpha_{\min}|$. Thread safety is ensured via a mutex lock when updating the window.

6.2.1 Model formulation

Given the vehicle's constructive parameters (such as mass, inertia, aerodynamic coefficients), the objective is to evaluate tire slips, forces, and loads in order to generate the

experimental curves $\frac{F_y}{F_z} = f(\alpha)$, and subsequently identify the tire model coefficients.

With the vehicle motion field available, tire slip angles are computed using Eqs. (3.8), while vertical loads on each axle are estimated using Eqs. (3.17).

By applying the vehicle equilibrium equations to the lateral (Y) and yaw (Z) dynamics of the single-track model shown in Fig. 3.1, we obtain:

$$\begin{cases} F_{\mathrm{c}} = m \cdot a_y, \\ F'_{yf} = F_{\mathrm{c}} \cdot \frac{l_r}{L} \cos(\delta), \\ F'_{yr} = F_{\mathrm{c}} - F_{yf}. \end{cases} \tag{6.2}$$

where F'_{yf} and F'_{yr} represent the lateral axle forces in a quasi-stationary regime. To extract the pure lateral component from the measured lateral forces, the forces are scaled by a weighting function G_y that accounts for the longitudinal force contribution:

$$G_y = \cos\left(\arcsin\left(\frac{F_x}{F_{x\,\max}}\right)\right), \tag{6.3}$$

where F_x is the actual longitudinal force and $F_{x,\max}$ is the maximum achievable longitudinal force at the axle, which depends on vertical load and available longitudinal grip. The longitudinal forces derive from the equilibrium in the longitudinal (X) direction.

$$\begin{cases} F_{\mathrm{d}} = \frac{1}{2}\,\rho\,S\,c_x\,v_x^2, \\ F_x = m\,a_x + F_{\mathrm{d}}. \end{cases} \tag{6.4}$$

Front-rear distribution during braking is determined by the brake balance, while traction force (positive a_x) act entirely on the rear axle. Finally, the pure lateral forces are computed as:

$$F_y = \frac{F'_y}{G_y}, \tag{6.5}$$

The processed data points (α, μ_y) are stored in a data structures for each axle, and are given to a nonlinear-least-square solver for parameter identification.

The differential moment (M_{diff}) must be included in the equilibrium equations in Eqs. (6.2) for a tricycle model. In this context, it has been excluded from the equations, allowing its effect to be implicitly incorporated into the derived axle characteristics.

6.2.2 Optimization

To estimate the parameters of the tire model, a nonlinear least-squares problem is formulated and solved using Ceres Solver, an open-source framework for large-scale optimization [77].

Given a set of N measurements (α_i, μ_{yi}), the problem is formulated as the minimization of the sum of squared residuals, weighted with a robust loss function:

$$\min_{\mathbf{p}} \; \frac{1}{2} \sum_{i=1}^{N} \rho(r_i(\mathbf{p})^2) \tag{6.6}$$

where $\mathbf{p} = [\,B,\, C,\, D,\, E,\, S_h\,]^T$ is the vector of unknown model parameters, and $r_i(\mathbf{p})$ is the residual for the i-th observation, defined as the difference between the measured lateral grip and the grip predicted by the Pacejka model:

$$r_i(\mathbf{p}) = \mu_{yi,\mathrm{meas}} - \mu_{y,\mathrm{model}}(\alpha_i; \mathbf{p}). \tag{6.7}$$

The function $\rho(\cdot)$ denotes a robust loss function; in this work, a Cauchy loss is employed to reduce the influence of outliers:

$$\rho(r_i) = c^2 \cdot \ln\left(1 + \frac{r_i^2}{c^2}\right), \tag{6.8}$$

where the scale parameter c is kept medium-high, since the input data are already largely noise-free, being primarily derived from filtered measurements. Ceres automatically computes the Jacobians via automatic differentiation ($J = \frac{\partial \mathbf{r}}{\partial \mathbf{p}}$) which are then used by the Levenberg-Marquardt algorithm to iteratively minimize the cost.

An initial guess for the parameter vector, $\mathbf{p_0}$, is provided to the solver to facilitate convergence. Generally, these values are derived from simulator experiments or previous testing data. Parameter bounds are enforced to restrict the optimization to physically meaningful regions, and some coefficients (e.g., E) can be held constant to improve convergence. This is particularly advantageous to reduce the overall computational load, especially since parameters describing the behavior beyond the grip peak are unlikely to be represented in the N measurements.

The solver is configured to use a dense QR decomposition, suitable for the small parameter dimension and moderate number of observations. A maximum of 100 iterations is allowed. Upon convergence, Ceres returns the optimized parameter set:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \sum_{i=1}^{N} r_i(\mathbf{p})^2, \tag{6.9}$$

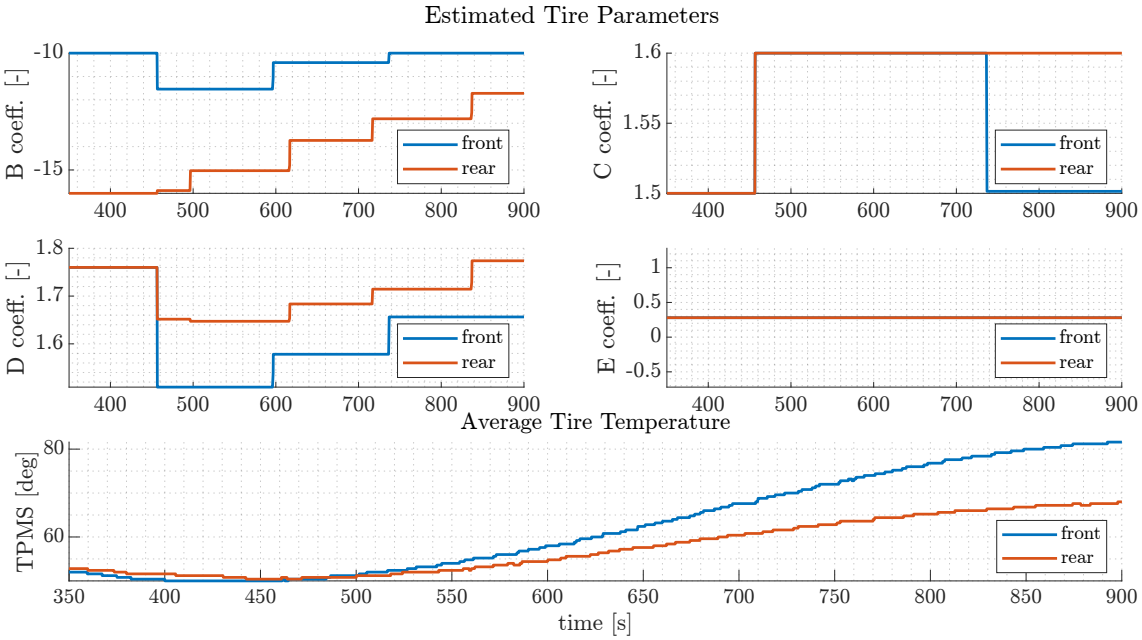
which represents the best-fit Pacejka coefficients for the given tire dataset. In the actual version, the solver is executed every 20 s, and a minimum set of N = 2000 samples is required.

6.2.3 Main results

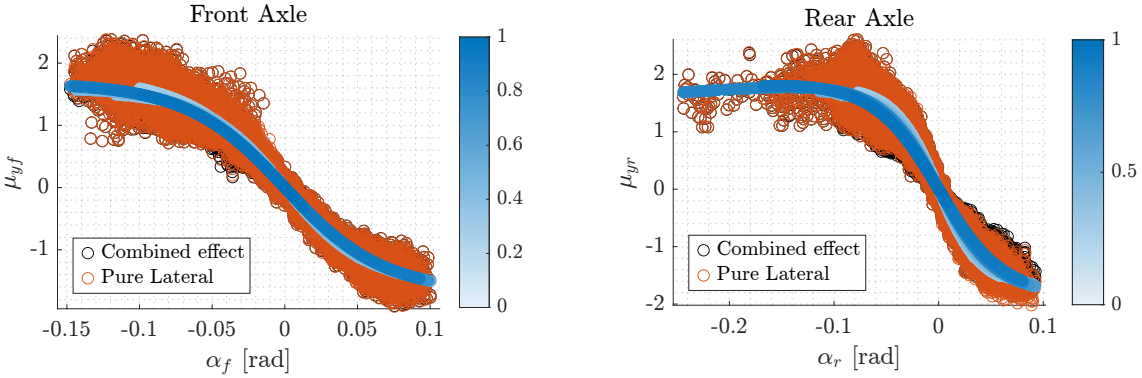
The grip estimation module enables online adaptation of the tire model and, consequently, the tire-road friction levels, improving both planning and control performance. This section presents the main results obtained using the module, structured in two parts. First, we analyze the grip estimation results over time and across evolving tire conditions. Second, we examine the impact of the updated tire model on the performance of the MPC.

Grip evolution. The upper subplots in Fig. 6.4a show the evolution of the Pacejka parameters $\mathbf{p}$, i.e., the module output over time. As discussed above, the parameter E is kept constant, while C is allowed to vary within a narrow range, thus ensuring consistency with the baseline simulator. Larger freedom is assigned to B and D, which respectively capture the initial stiffness and the overall grip of the axle. The lower plot reports the average axle tire temperature, measured from the inner tire core (TPMS sensors). Temperature emerges as the dominant factor driving parameter variation: as temperature increases, the overall grip rises while the stiffness decreases (an expected and physically consistent behavior of the racing tire [78, 79]).

Starting from an initial guess of $D_f = D_r = 1.75$, the module triggers an update when the estimated friction deviate by more than 5% from these nominal values ($time \approx 450$ s). From the updated parameter set, the model evolves progressively until converging to steady-state values of approximately $D_f = 1.66$ and $D_r = 1.76$, consistent with the stabilization of the tire temperature. Compared to the initial guess, the resulting parameters



(a) Time evolution of the identified tire model coefficients, starting from the initial parameter guess. The lower plot reports the average tire temperature per axle.



(b) Point cloud of the experimental axle characteristics obtained through the data manipulation procedure described in Sec. 6.2.1. The blue-gradient curves represent the identified tire model evolution over time, with darker shades corresponding to later estimation steps.

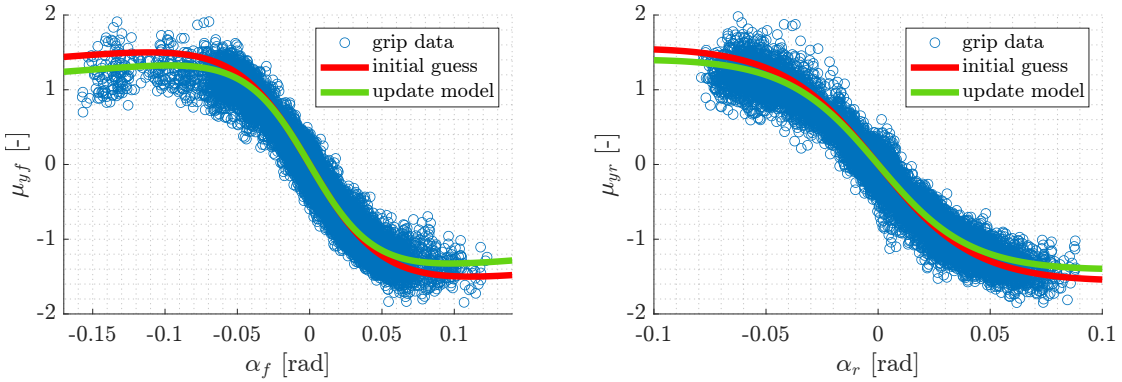
Figure 6.4: Results of the tire grip estimation and its dependency on tire's operating conditions.

indicate a more understeering vehicle balance, which is coherent with the high mileage of the tires used for this test (over 250 km).

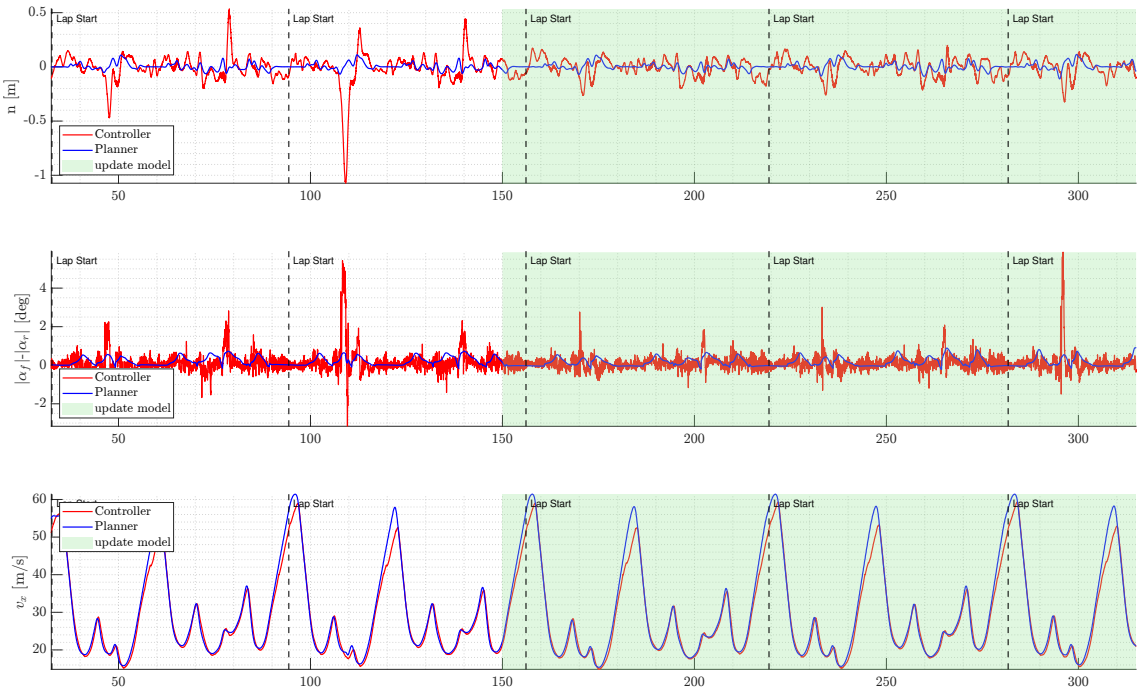
In Fig. 6.4b we report the complete dataset collected during the test, consisting of approximately N $\approx$ 7800 sample points (α_i, μ_{yi}) per axle. The black and orange point clouds represent the full set of measurements stored in the moving window, while the blue-gradient curves show the identified axle characteristics, with color intensity increasing over time. Because the maneuvers involved nearly pure lateral dynamics, only negligible differences arise between the black and orange datasets. The shift coefficient S_h was set to zero, as the vehicle setup is expected to be symmetric. Nevertheless, a slight asymmetry appears in the front axle data, causing the identified parameters to converge toward a trade-off that balances positive and negative slip angle behavior.

Planning and Control Results. This section presents an experimental test in which the initial tires parameters (new condition) significantly differed from the actual tires used (around 250 km). The test was conducted at the Marzaglia racetrack using the Dallara AV24 from the Indy Autonomous Challenge.

Figure 6.5a shows the grip estimation module results: the blue point cloud represents data collected in the moving window over the test session (≈ 320 s), the red line shows the initial model, and the green line represents the updated model estimated by the module after convergence (≈ 150 s). The updated model differs from the initial guess by approximately 12% for both front and rear axles. This threshold allows the motion framework to be updated with the newly identified parameters, better matching the measured data and improving overall vehicle performance. Figure 6.5a compare the controller performance



(a) Grip estimation results: blue points show the full dataset collected during the test session, the red line indicates the initial tire model parameters, and the green line represents the final model after convergence of the estimation algorithm.



(b) Controller performance before and after tire parameter update: lateral and longitudinal tracking significantly improves after updating the model.

Figure 6.5: Results of the tire grip estimation and its effect on the controller (MPC).

before and after the parameter update. From the planner's perspective, the trajectory remains unchanged, as does the understeer angle, while the planned speed is slightly lower. After updating the parameters, the controller tracking performance significantly improves: lateral errors previously ranging from 50 cm to over 1 m are reduced below 30 cm, allowing the controller to more accurately follow the planned reference in terms of both path tracking and vehicle dynamics.

Notably, turn 2 at t $\approx$ 110 s and the same turn at t $\approx$ 295 s show the effect clearly. Despite identical speed and understeer angle, the lateral tracking error drops from 1.1 m to just over 30 cm, demonstrating the substantial improvement in closed-loop performance enabled by the online grip parameter update.

6.3 Pratical considerations and Future improvements

Lateral velocity filter and grip estimation work synergically to guarantee the stack an accurate knowledge and fast adaption to the operating conditions. While the proposed is effective in real-world racing scenarios, a few practical considerations should be highlighted:

- To guarantee consistency across all modules (including the motion framework presented in Cha.5), the same vehicle model must be used throughout the framework. Otherwise, the identified parameters may become meaningless when applied to a different model formulation (e.g., a tricycle model). While it is possible to adopt a different vehicle model, more complex formulations generally require significantly more effort during the estimation process (for example, M_{diff} in the case of a tricycle model). Our findings indicate that a simple nonlinear single-track model offers the best trade-off among control, planning, state estimation, and parameter identification tasks, unless specific requirements, such as strong differential effects, need to be considered.

- Using raw sensor measurements directly in the grip estimation module is theoretically possible but often unsafe, as racing applications tend to push sensors to their operational limits. This may result in corrupted readings (Subfig.6.3a), which can significantly degrade the quality of the identified parameters. In our approach, potential filtering errors caused by imperfect initial model calibration can indeed propagate into the grip estimation module. However, the EKF relies not only on the model but also on multiple independent sensing sources, which preserves robustness and ensures accurate predictions even when certain signals degrade. In practice, this setup may take longer to converge, but it offers more reliability.

Main limitations and future improvements may be:

- **Explicit temperature dependency**: The parameter identification does not explicitly incorporate tire temperature. Instead, the model adapts purely based on sensor and filter feedback, indirectly capturing temperature effects through the observed forces. This creates a delay: when optimal parameter set is applied, the system may have already evolved. A natural extension for future work would involve constructing a 3D parametric surface in which the tire model parameters evolve as a function of both slip angle and measured tire temperature.

- **Dependence on the initial parameters accuracy**: The framework assumes that the initial parameter set is reasonably accurate. This assumption holds in our case

thanks to the high-fidelity simulator, but in general, larger discrepancies between the initial guess and the true tire behavior increase the convergence time. This effect becomes more pronounced when key sensors required for measurement correction are unavailable or unreliable.

- **Longitudinal tire parameters estimation**: The model presented in Sec. 6.2.1 assumes that the longitudinal tire grip is known and constant, particularly in the computation of the weight function in Eq. (6.3). While this simplification is acceptable for the proposed application, extending the module to estimate the tires' longitudinal characteristics would be highly valuable. This extension represents an important avenue for future improvements.

Chapter 7

Simulation results

Simplified modeling approaches offer several advantages, including easier formulation and calibration, and improved computational efficiency. For these reasons, selecting the simplest possible vehicle model may appear to be the most natural choice. However, in high-dynamics autonomous racing scenarios, excessive simplification can compromise model validity, then control performance, especially when operating close to the tire friction limits. The key challenge is to identify the minimum level of model complexity required to reliably capture the relevant vehicle dynamics for the intended application.

In this chapter, we evaluate simplified vehicle models within an autonomous racing context, following the structure below:

1. First, open-loop simulations are conducted to compare simplified models against a high-fidelity multibody reference, focusing on dynamic accuracy and stability.

2. Second, the models are integrated into the Model Predictive Controller (MPC) to assess their effectiveness within a closed-loop path-tracking context.

This analysis is essential for validating the operative range of the models and their underlying assumptions, and for selecting the most appropriate vehicle representation to be included in the model-based motion framework prior to real-world deployment.

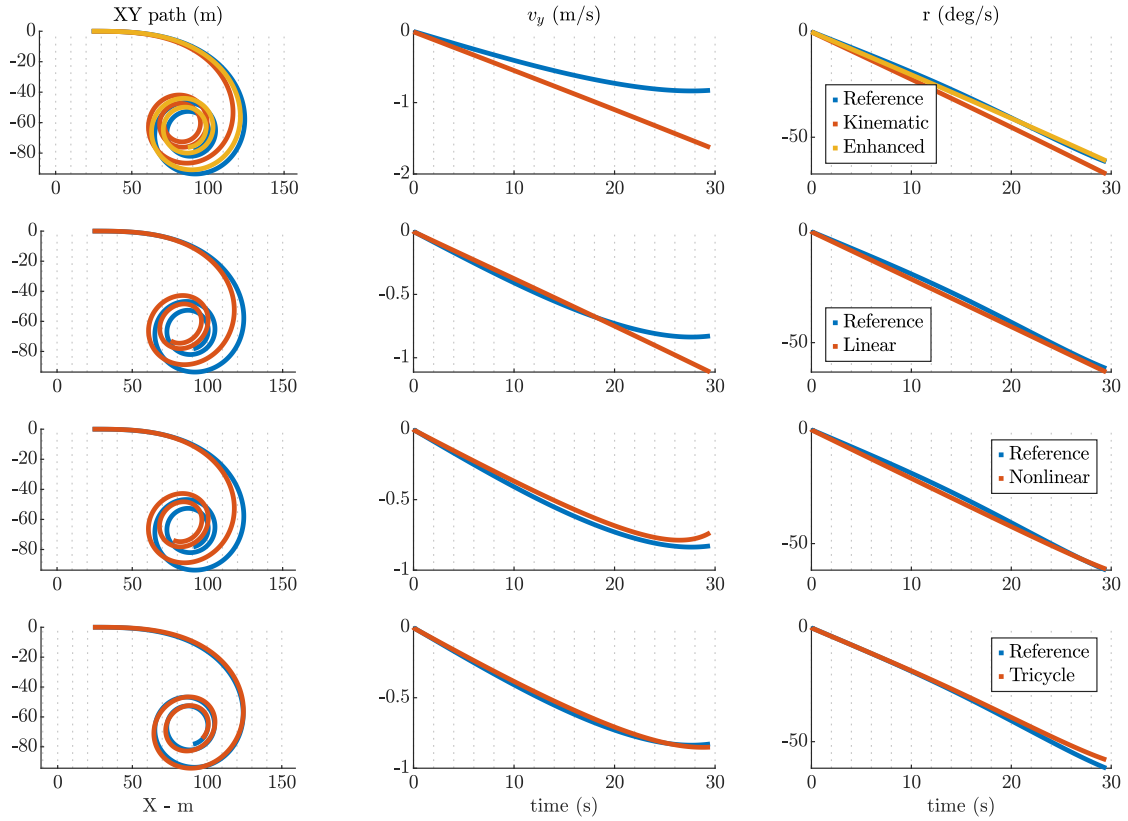
7.1 Open-Loop simulations

All models are initialized using the same state (x_0), and the simulation evolves by applying steering and a constant velocity. The steering angle is gradually increased from zero to its maximum value at a rate of approximately $\dot{\delta} = 1\,\mathrm{deg/s}$ (measured at the wheel hub), resulting in a quasi-stationary maneuver. Simulations are conducted at both low and high longitudinal velocities to assess models performance across a wide range of driving conditions.

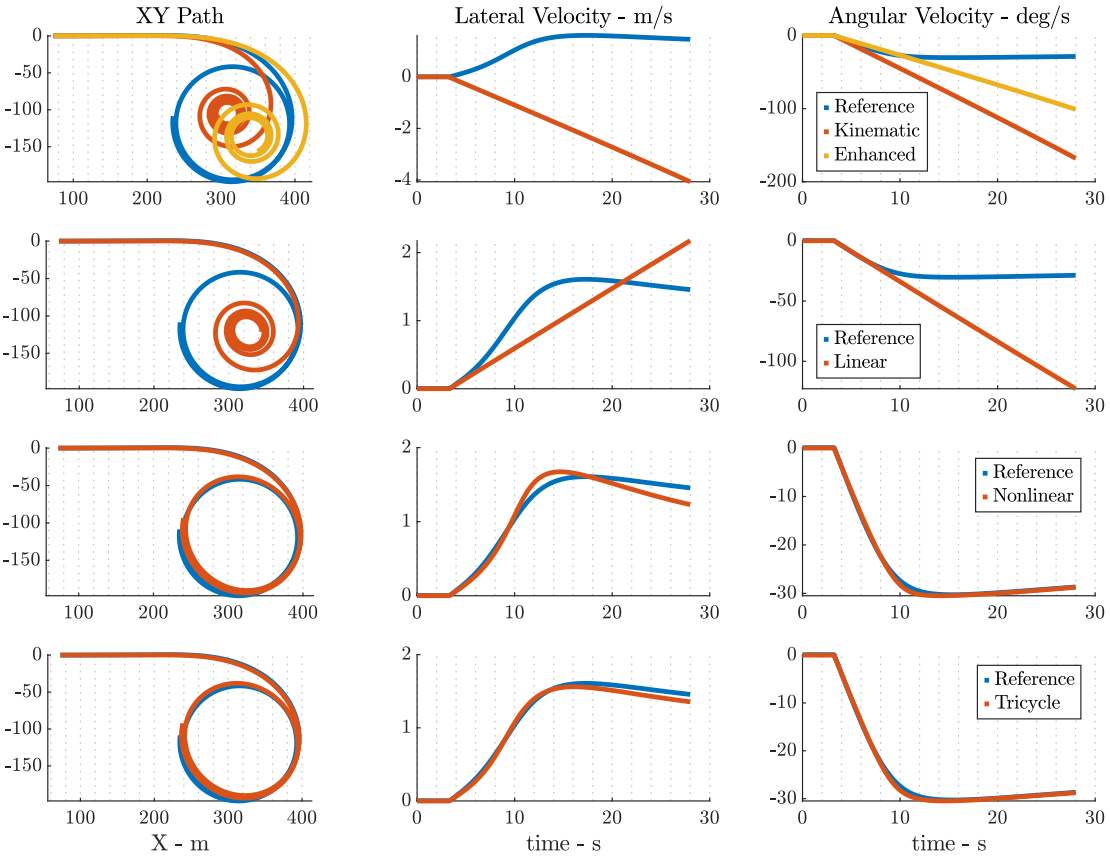
7.1.1 Model Physics

This section presents the results on model accuracy in reproducing vehicle dynamics quantities and the resulting path. Results are reported In Fig.7.1.

Fig.7.1a illustrates a low-speed scenario ($v_x = 14\,\mathrm{m/s}$). All models are expected to accurately replicate vehicle dynamics, as the speed remains low enough to avoid tire saturation, evidenced by the nearly linear relationship between yaw rate and steering input. This keeps the operating conditions within the valid range for all models. The inclusion



(a) Low longitudinal speed, $v_x = 14\,\mathrm{m/s}$.



(b) High longitudinal speed, $v_x = 42\,\mathrm{m/s}$.

Figure 7.1: Path and velocity comparison between the simplified models and the multibody reference model. Each row shows a specific model, from top to bottom: kinematic single-track, linear and nonlinear single-track, and tricycle models.

of the stability factor enhances the kinematic model's accuracy by accounting for the vehicle's nose-out behavior. Linear and nonlinear single-track models perform similarly, though the nonlinear formulation provides a more accurate representation of lateral velocity. The tricycle model outperforms the others, as it captures the influence of the LSD, which is particularly significant at low speeds and in high-curvature maneuvers.

Fig.7.1b illustrates a high-speed scenario ($v_x = 42\,\mathrm{m/s}$). Kinematic models neglect tire slip and provide only a geometry-based approximation of motion, which prevents them from accurately capturing lateral velocity sign changes, Eqs.(3.1). In dynamic formulations, v_y emerges from the balance of lateral forces and can change sign depending on driving conditions (e.g., longitudinal velocity [59]). In both kinematic and linear dynamic models, lateral velocity and yaw rate are not constrained by tire force limits, leading to unrealistic growth beyond the linear range of the tires. Both the nonlinear single-track and "tricycle" models achieve a high level of accuracy. However, the nonlinear single-track model struggles to fully replicate the lateral velocity beyond the tire limit. Both models accurately represent the yaw rate. The limited improvement of yaw rate prediction with the tricycle model is due to the moderate locking angles of the differential, the small rear wheel speed and torque differences at high velocity, and the understeer behavior that prevents tighter turning.

A summary of the overall performance is presented in Tab.7.1a and Tab.7.1b, corresponding to low and high longitudinal speed scenarios, respectively, over a time horizon of $T = 28\,\mathrm{s}$. The evaluation includes Maximum Absolute Error and Root Mean Square Error (RMSE) for the lateral deviation n (m), the lateral velocity v_y ($\mathrm{m\,s^{-1}}$), and the yaw rate r ($\mathrm{rad\,s^{-1}}$). The Pearson coefficient quantifies signal correlation, where values near 1 indicate strong agreement and zero or negative values reflect poor correlation. Thanks to its ability to capture dynamics across varying turn radii, the tricycle model proves to be the most accurate single-parameter model for reproducing vehicle behavior across the full speed range. The nonlinear single-track model, which balances performance across

Model	Max Abs. Error			RMSE Error			P. Correlation	
	n	v_y	r	n	v_y	r	v_y	r
Kinematic	9.54	0.80	0.10	4.69	0.36	0.067	0.9815	0.9995
Enhanced	**3.69**	0.80	**0.023**	2.10	0.36	**0.015**	0.9815	0.9995
Linear s-track	8.41	0.285	0.045	4.07	0.085	0.032	0.9815	0.9994
Nonlinear s-track	8.36	0.093	0.044	4.06	0.019	0.029	**0.9994**	0.9989
Nonlinear tricycle	3.81	**0.029**	0.064	**0.89**	**0.018**	0.027	0.9987	**0.9998**

(a) Low longitudinal speed, $v_x = 14\,\mathrm{m/s}$

Model	Max Abs. Error			RMSE Error			P. Correlation	
	n	v_y	r	n	v_y	r	v_y	r
Kinematic	47.6	5.52	2.43	30.5	3.40	1.18	-0.858	0.7740
Enhanced	60.7	5.52	1.26	27.5	3.40	0.564	-0.858	0.7738
Linear s-track	48.3	0.72	1.64	24.3	0.367	0.776	0.8613	0.7806
Nonlinear s-track	8.03	0.226	0.016	2.98	0.106	0.0057	0.9859	0.9997
Nonlinear tricycle	**7.04**	**0.101**	**0.015**	**2.58**	**0.056**	**0.0055**	**0.9981**	**0.9998**

(b) High longitudinal speed, $v_x = 42\,\mathrm{m/s}$

Table 7.1: Error metrics of the simplified models with respect to the multibody model.

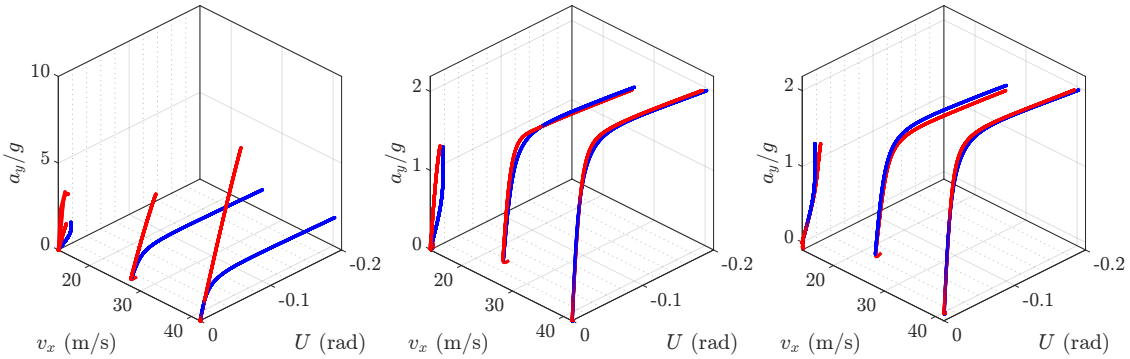


Figure 7.2: Handling diagram as 3D map representation for the dynamic models, where $U = \|\alpha_f\| - \|\alpha_r\|$ is the understeer angle. Blue lines represent the multi-body ground truth, red lines are the simplified models. From left to right: linear single-track, nonlinear single-track, and "tricycle" model.

multiple scenarios through a trade-off parameter, still provides a highly faithful representation of vehicle dynamics. Simpler model prove effective until the tires enter nonlinear operating regions.

7.1.2 Handling diagrams

Handling diagrams are shown in Fig. 7.2. The curves are plotted as a function of longitudinal velocity to capture the effects of aerodynamics and the limited slip differential. An intermediate velocity of 30 m/s is also simulated to provide a more representative handling map. The vehicle consistently exhibits understeer behavior across all analyzed speeds. The linear single-track model struggles to fully capture the understeer characteristics but performs well in the linear region of the handling curves. The nonlinear single-track model shows a slight delay in replicating understeer saturation at both 30 m/s and 42 m/s, yet it still provides a reasonable representation of both the linear and saturation regions. At 14 m/s, it fails to accurately capture the initial understeer angle, primarily due to neglecting the effect of the differential. The "tricycle" model effectively replicates the initial stiffness across most scenarios. Compared to the nonlinear single-track model, it slightly underestimates the maximum performance at 30 m/s, but still offers a good overall approximation. Handling diagrams intrinsically contain all necessary information to identify the stability threshold derived from the linearization of the vehicle model across all equilibrium configurations. Therefore, accurately reproducing the handling diagram with simplified models also implies accurate prediction of linear stability.

The results reflect each model's theoretical performance limits. The observed discrepancies primarily originate from simplified models neglecting setup variations (including Ackermann compensation, toe and camber changes), incomplete load transfer dynamics, and fixed aerodynamic coefficients that don't account for ride height variations. Despite these limitations, nonlinear models maintain satisfactory accuracy across all test scenarios.

7.2 Closed-Loop simulations

The MPC is tasked with completing a lap of the Yas Marina F1 full circuit, "driving" the high-fidelity model of the autonomous vehicle Dallara EAV24 Super Formula. The simplified vehicle models are integrated into the MPC framework to evaluate how their accuracy impact control performance, including the minimization of lateral and heading deviations

from a predefined trajectory while ensuring vehicle stability. The reference racing line is precomputed using the lap-time optimization algorithm described in 5.2.1. Different speed profiles are associated with the optimal reference path using the longitudinal planner, as detailed in 5.2.3.
To evaluate the controller's performance, the following metrics are defined:

1. Oscillation counter: this metric counts the number of significant zero-crossings of the steering deviation from its local mean within a fixed time window, ignoring small deviations below a predefined threshold, and providing a quantitative measure of controller stability.

2. Lateral error constraints: a maximum lateral deviation of $n_{\max} = 1\,\mathrm{m}$ is enforced. Thus, the goal is to track the reference while maintaining stability. This is crucial in autonomous racing, where respecting track boundaries and avoid other agents is essential.

3. Understeer angle: in the case of quasi-steady-state maneuvers, and together with lateral acceleration, yields an incremental estimation of the understeer gradient, and consequently feedback on understeer and oversteer behavior.

The weights of the cost function are set as shown in Tab.7.2. Tuning slightly differs from the one proposed in Tab.5.5, as in this context the MPP is not active. Primary objectives are path following, mainly through q_n and q_μ, and vehicle stability, through the sideslip angle cost q_β and soft constraint for combined slip s_{tires}. Note that the tire soft constraint is relevant only when a nonlinear tire model is employed, as neither the kinematic model nor the linear tire model can capture tire saturation.

Table 7.2: MPC costs and soft constraints used for the simulations. Table notation follows the same rule as in Tab.5.5.

q_n	q_μ	q_{v_x}	q_r	q_β	$r_{d\delta}$	r_{J_x}	s_{track}	s_{tires}
++++	+++	+++	+	++++	++	+	++	+++

7.2.1 Kinematic single-track model.

The results obtained using the kinematic single-track model (including the enhanced formulation) are shown in Fig.7.3. The standard formulation fails to complete a full lap without exceeding the lateral error constraints, particularly in high-speed turns, where the deviation from the path increases significantly. Additionally, some oscillations are visible in the fast turn 3 ($s \simeq 600$). The maximum achieved performance is limited to 70%. Introducing the stability factor helps to mitigate oscillations, leading to an overall improvement in performance. Nevertheless, the model remains limited to 78% of the vehicle's potential, as the lateral error constraint is still exceeded. The enhanced formulation shows noticeable improvement in both lateral deviation and steering smoothness.

7.2.2 Linear single-track model.

The results obtained using the single-track model with a linear tire formulation (modeled through constant cornering stiffness) are shown in Fig.7.4. This approach significantly outperforms the kinematic models, raising the performance threshold to 88%. The improvement is mainly due to a more accurate representation of lateral velocity, as the

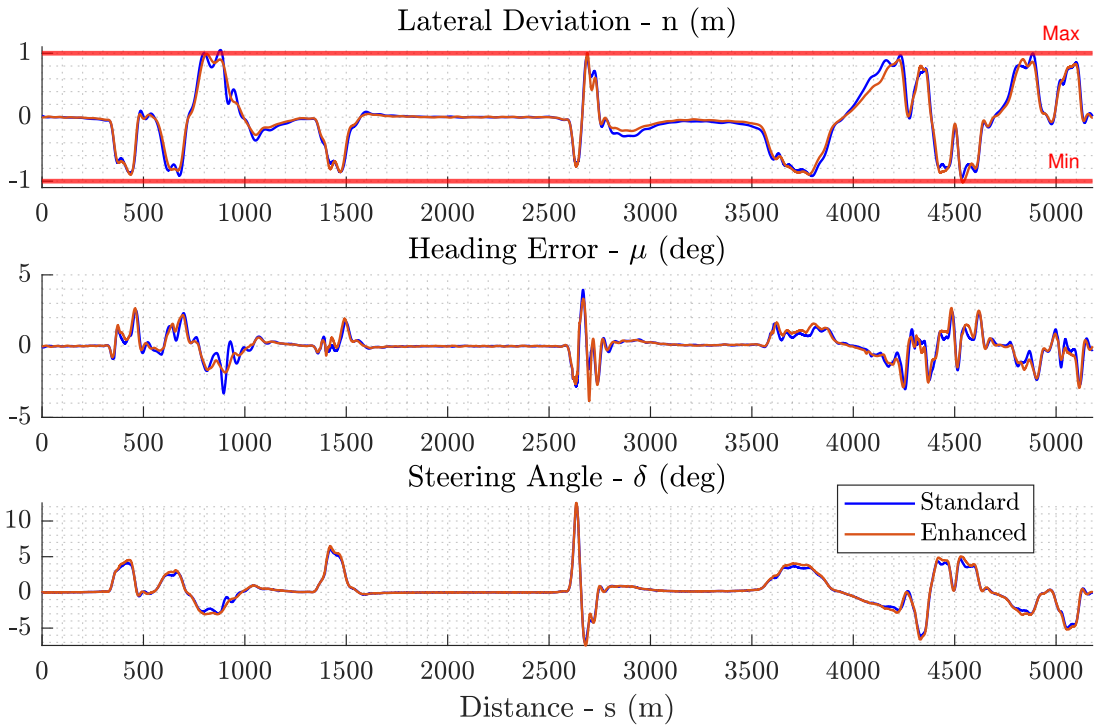


Figure 7.3: Lap results using kinematic models. Standard formulation achieved 70% of the vehicle's performance, while the enhanced version reached 78%.

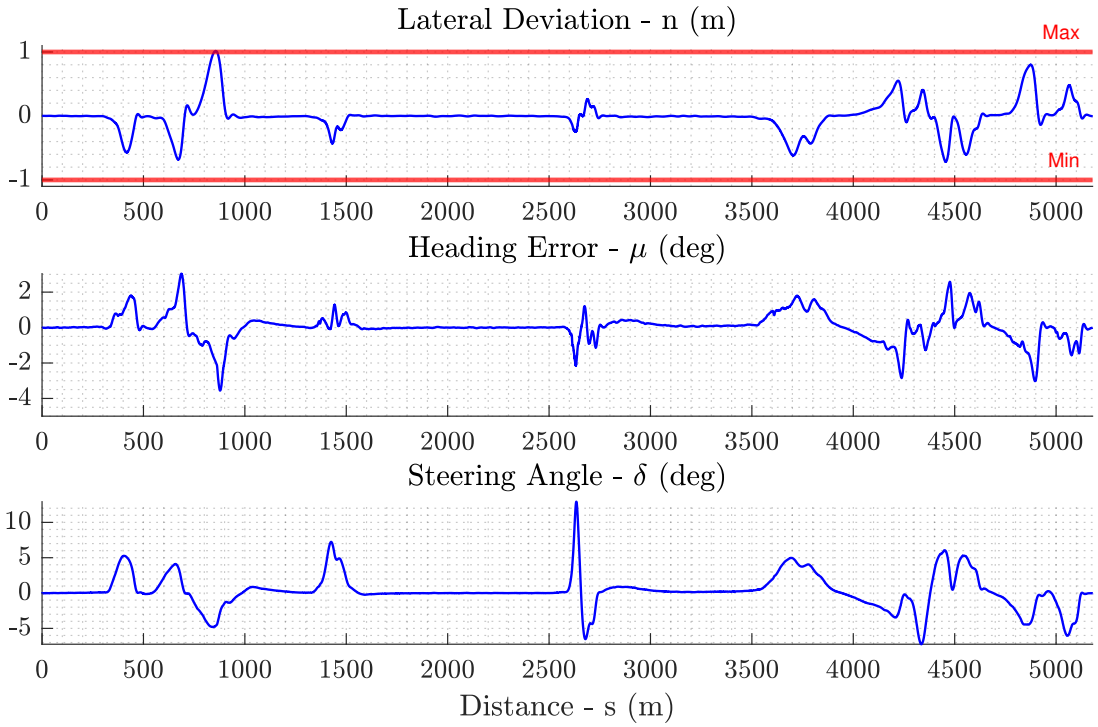


Figure 7.4: Lap results using single-track model with linear tire formulation. The controller achieved 88% of the vehicle's performance.

dynamic model can correctly capture its sign. The controller struggles to keep the lateral error within predefined constraints in high-speed sections, i.e., in turn 3 ($s \simeq 800$). Steering control remains relatively smooth, as does the resulting heading error. Additionally, the controller exhibits the ability to handle conditions slightly beyond the linear tire region, though it accumulates up to 0.8 m of lateral error in turn 2 ($s \simeq 400$), turn 13 ($s \simeq 4400$), and turn 15 ($s \simeq 4800$). In contrast to the kinematic model, the linear dynamic model performs well in low-speed corners such as turn 5 ($s \simeq 1400$), turn 6-7 ($s \simeq 2700$), and turn 12 ($s \simeq 4300$).

7.2.3 Nonlinear single-track model.

The results obtained using the single-track model with a nonlinear tire formulation, incorporating the Pacejka model and the combined slip effects, are shown in Fig.7.5. In this scenario, the controller successfully completes all speed profiles while remaining within the predefined constraints. The steering angle remains sufficiently smooth, while the heading error exhibits some oscillations in the turn 6-7 sequence ($s \simeq 2700$). This is likely due to the high slip ratios in this section of the track, leading to frequent activations of the low level traction control. Nevertheless, the lateral error remains within an acceptable range (as shown in Tab.7.3), and vehicle stability (from a technical point of view) is consistently preserved. Despite its simplifications, the single-track model ensures reliable path-following performance by adapting to different cornering conditions through tire load dependency.

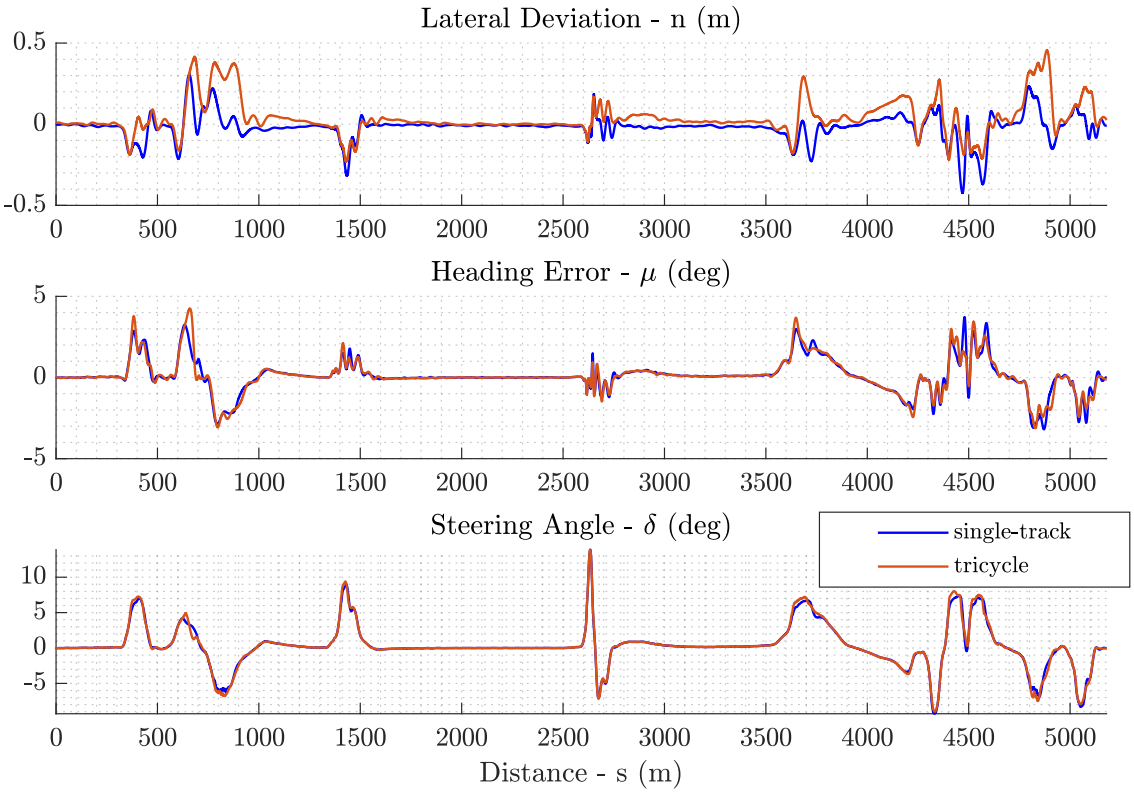


Figure 7.5: Comparison between the single-track and the tricycle model, with nonlinear tire formulation. Both controllers achieved the full vehicle's performance.

7.2.4 Nonlinear "tricycle" model.

The results obtained using the "tricycle" model with a nonlinear tire formulation, incorporating the Pacejka model, the combined slip and the LSD effects, are shown in Fig.7.5. The performance achieved by the MPC is comparable to those obtained using the nonlinear single-track formulation. Notable improvements in path tracking are observed in low-speed corners, such as turn 5 ($s \simeq 1500$) and turns 13 and 14 ($s \simeq 4400$), where the model can better exploit the yaw moment prediction. Conversely, performance slightly deteriorates in high-speed sections, in particular turn 3 ($s \simeq 800$) and turn 15 ($s \simeq 4800$). It should be noted that the tricycle model relies on a simplified LSD formulation, which can struggle to accurately capture the actual differential behavior, especially since it does not incorporate wheel speed feedback.

A summary of each model path-tracking accuracy and overall performance is provided in Tab.7.3.

Model	**Performance**	**Lap Time**	**Max n**	**Max μ**
kinematic	70 %	2:00:99	1.05	3.95
kinematic enhanced	78 %	1:57:41	1.02	3.84
linear single-track	88 %	1:53:94	1.01	3.55
nonlinear single-track	100 %	1:49:79	0.42	3.72
nonlinear "tricycle"	100 %	1:49:69	0.46	4.25

Table 7.3: Summary table of the performance achieved by all the models.

7.2.5 MPC Execution Time

A summary of the MPC execution times for a full lap is provided in Tab. 7.4. All models were able to run in real-time, with observed runtimes consistently below the execution time (0.01 s). The simulations were performed on an Ubuntu 24 system with an Intel Core i9-14900HX CPU (32 threads). Maximum values should be interpreted as isolated spikes rather than representative of typical performance. Compared with the other models, the tricycle exhibits slightly reduced performance, since it requires the engine torque additional variable for LSD force computation.

Runtime	**Kinematic**	**Enhanced**	**Linear**	**Nonlinear**	**Tricycle**
Minimum	1.006	1.052	1.054	1.216	1.385
Maximum	3.588	3.415	4.430	4.448	4.851
Average	1.513	1.537	1.768	1.891	2.169

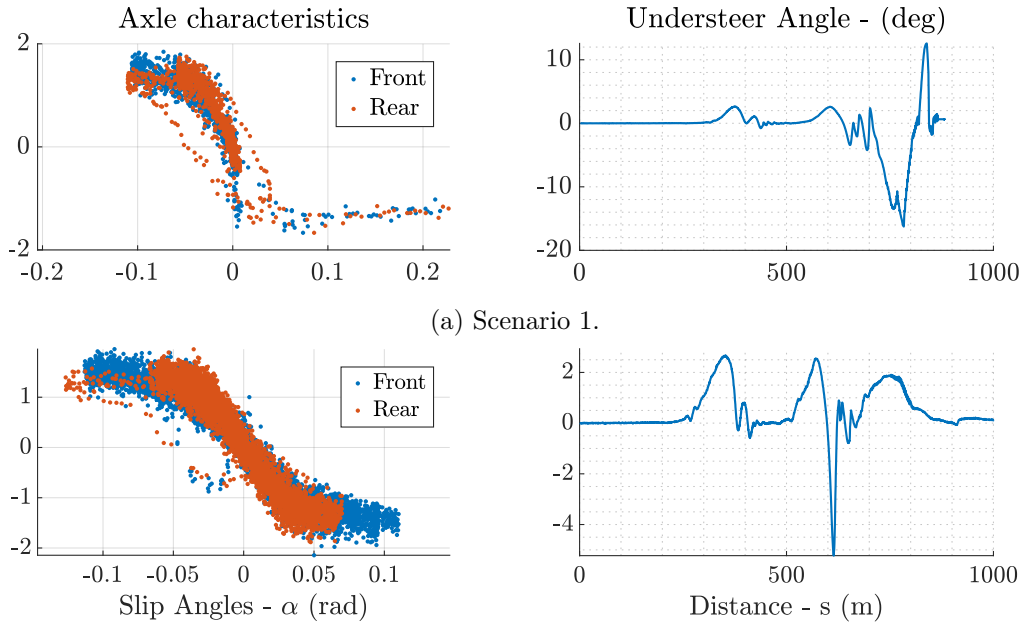
Table 7.4: Summary of MPC execution times (ms) for all vehicle models.

7.2.6 Edge-Case Scenarios

In this subsection, the controller using the single-track model with linear or nonlinear tire formulations is evaluated under edge-case scenarios. Scenario 1 tests the linear tire model at 94% of maximum performance, thus pushing it beyond its theoretical validity. Scenario 2 tests the nonlinear model at full performance (100%) under model mismatch to replicate a more realistic scenario.

Scenario 1. When operating beyond the tire's linear region, the lateral error increases significantly, causing the controller to overcompensate by increasing the steering command.The simulated tire characteristics and understeer angle are shown in Fig. 7.6a. In turn 2 ($s \simeq 600$), the vehicle's dynamic behavior shifts from nose-out to nose-in. This abrupt transition, combined with the model's inability to accurately capture tire saturation, induces a series of oscillations that push the system into an unstable zone, leading to loss of control. Increasing the weight q_r enhances stability, but reduces path-tracking accuracy. On the other hand, increasing the sideslip angle weight q_β proves less effective, as the model cannot accurately predict the actual velocity field under spin conditions.

Scenario 2. The multibody tire parameters were adjusted to induce oversteer by progressively reducing rear tire grip, creating a mismatch with the single-track model and causing oversteer due to the controller's inability to fully predict the vehicle dynamics. Simulation results are shown in Fig.7.6b. Since tire grip reduction remains within 92% of its nominal value, the controller is able to successfully restore vehicle stability. This is mainly due to the slope characteristics of the nonlinear tire model, which help mitigate the impact of parameter mismatches. Additional factors include the tire soft constraints (s_{tires}), the weight on the sideslip angle (q_β), and the weight on the heading error (q_μ). A greater mismatch in tire grip leads to loss of control, as the countersteering action fails to generate enough steering input to correct the yaw rate error. This reflects the findings of [22], where the authors demonstrated that even a 10% error in friction estimation can compromise controller accuracy.



(a) Scenario 1.

(b) Scenario 2.

Figure 7.6: Grip curves and understeer angle using the linear (a) and nonlinear (b) single-track model.

7.3 Summary of Simulation Results

This chapter clarifies the trade-off between model's complexity and the performance of an autonomous racing controller in closed-loop path-following context, using an MPC framework and a structured quantitative analysis. The results align with literature: more accurate models that better represent actual vehicle dynamics lead to improved path-tracking performance, and oversimplified models cannot be reliably used beyond their theoretical range.

Each simplified model exhibits strengths suited to different applications. In low-dynamic or urban scenarios, the kinematic single-track model often suffices, offering low computational cost. Introducing a few dynamic parameters, as in enhanced kinematic models, can improve dynamic response while preserving simplicity and manageable calibration effort. Linear dynamic models can further enhance performance but may introduce numerical instability at low speeds. When operating near tire grip limits, more accurate models become necessary, both in racing and in critical urban situations such as loss of grip or emergency obstacle avoidance. Even modest increases in model complexity, such as adopting a nonlinear single-track model, can yield significant improvements, justifying the additional modeling and calibration effort. Including tire combined-slip effects, even in a simplified form using Eq.(6.3) and Eq.(5.13), proved crucial for maintaining high tracking performance during both corner entry and exit phases. Beyond this point, increasing model complexity may require substantially more calibration effort while offering progressively smaller performance gains, except for specific requirements such as torque vectoring or drifting maneuvers. This effect is particularly pronounced in closed-loop control, where continuous sensor feedback can partially compensate for minor modeling inaccuracies.

Overall, closed-loop path-following performance is highly sensitive to model fidelity, and careful selection and calibration of the vehicle model are critical for high-dynamics driving scenarios.

In practice, the single-track model with a nonlinear Pacejka tire formulation represent an optimal candidate for real-world testing, as it provides an excellent balance between accuracy and calibration effort. Conversely, the kinematic and linear single-track models are not considered for real-world testing: models that perform poorly in simulation cannot be safely deployed on a physical vehicle. Simulation is, in fact, intended to filter out unsuitable models before any track tests are conducted. This is particularly crucial in autonomous racing, where experimental variability (e.g., cold tires, dusty track, localization noise) makes operating within linear tire conditions unrealistic. Based on the preliminary results, the tricycle model could be considered as a candidate for experimental validation. However, due to limited track time, the significantly higher calibration effort required, and the relatively small performance gain observed for this specific vehicle platform, we decided not to pursue its real-world deployment.

Although these conclusions are drawn from controller-focused path-tracking tests, they provided a clear criterion for selecting the most suitable vehicle model for the entire motion framework. In particular, as discussed in Sec. 5, model consistency between planning and control layers is essential to guarantee feasibility, robustness, and reliable convergence.

Chapter 8

Real-world results

This chapter reports the results of the autonomous driving software developed in this thesis, validated during the A2RL competition, where it demonstrated performance beyond the current state of the art in full-scale autonomous racing. Experiments were conducted with Car 6 (Unimore Racing team), a Dallara EAV-25 Super Formula, on the North layout of the Yas Marina Formula 1 circuit. For the experiments, the single-track model with a nonlinear Pacejka tire formulation was adopted.

8.0.1 Race overview

A total of six cars were admitted to the final race, depending on several robustness and performance evaluation criterion. The starting grip order was: car 33, car 6, car 3, car 7, car 5, and car 8. The race begins after two formation laps and lasts for 20: all cars except the leader may use two *push-to-pass* (P2P) activations per lap, lasting 15 seconds each. The lap times of the six cars are reported in Tab. 8.1, where an asterisk marks laps without P2P. Lap times highlighted in green indicate each autonomous driver's personal best, while purple denotes the overall fastest lap. For consistency and comparability, we report only the first 11, since an accident on lap 12 forced our car to retire. For fairness, we mention that on lap 18, car 33 improved the best lap previously set by our vehicle, eventually establishing a new autonomous lap record on lap 20 with a lap-time of 58.183 seconds.

Table 8.1: Lap times per driver, where the order reflects the starting grip position (left to right).

Lap	**car33 (TUM)**	**car6 (Unimore)**	**car3 (Kinetiz)**	**car71 (TII)**	**car5 (Polimove)**	**car8 (Constructor)**
1	1:02.149	1:03.272	1:06.698	1:09.733	1:13.036	1:23.845
2	1:02.581 *	1:00.307	1:01.540	1:04.063	1:04.297	1:09.317
3	1:00.197	59.677 *	1:00.993	1:02.987	1:02.472	1:06.708
4	59.549	58.999 *	59.994	1:02.358	1:02.180	1:04.709
5	58.952	58.765 *	59.598	1:01.701	1:02.705	1:18.441
6	59.459	1:00.922 *	59.535	1:01.282	1:01.829	1:17.652
7	59.301	59.209 *	1:01.186	1:01.127	1:01.762	1:17.549
8	58.903	58.929 *	59.389	1:01.884	1:02.396	1:04.374
9	59.089	59.168 *	59.462	59.993	1:01.785	1:08.949
10	59.274	59.287 *	59.433	1:00.333	1:01.763	
11	59.046	59.094 *	59.407	1:08.803	1:18.033	

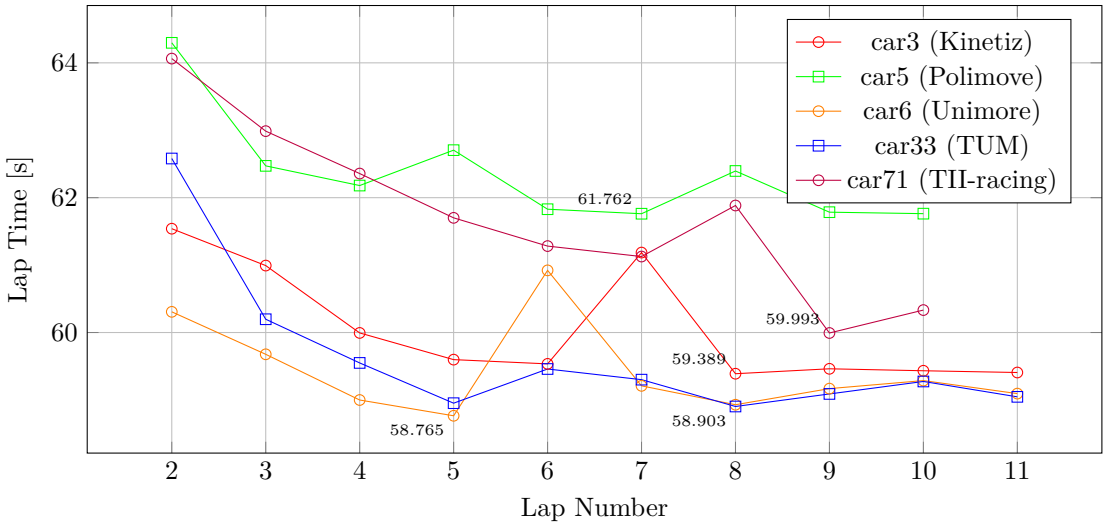


Figure 8.1: Lap Time Progression per Driver. Too slow timing were filtered out to preserve the clarity of the figure. Among all teams, Car 6 performed the fastest lap-time progression, despite could not count on P2P. In Lap 6, it is evident the time deterioration due to the lapping maneuver on car 8.

Car 6 gained first position after overtaking car 33 during lap 2, then maintained the lead until the crash with car 8 on lap 12. Note that the P2P system was deactivated from lap 3. The autonomous vehicle set the best lap-time within four laps. During lap 5, the lap time deteriorated significantly due to a lapped car (car 8) overtake. Afterward, car 33 gained advantage and came very close. Unfortunately, this prevented further lap-time improvements, as car 33 acquired the right of way[1] on each lap, disturbing our ideal trajectory and hindering performance gains. During the final laps (8-11), car 33 continued to chase car 6 at the limits of vehicle handling. However, despite benefiting from slipstreaming and having push-to-pass available, car 33 was not faster enough to attempt an overtaking maneuver and reclaim the lead position. On turn 1 of lap 12, lapped vehicle car 8 becomes stationary on the racing line during an overtaking maneuver by car 6, ultimately leading to an unavoidable collision that ended our race.

8.1 Warm-up Progression

This section analyzes the lap-time progression (shown in Fig.8.1) leading up to race pace, highlighting the performance of the mission planner.

Prior to the race, the planner is provided with a reference performance progression, experimentally derived to represent the ideal evolution of vehicle performance over the race distance. Deviations from this nominal progression are handled online by the mission planner using vehicle and state estimator feedback (Sec. 5.2.2). Performance is managed lap-by-lap by dividing the track into zones corresponding to groups of corners. Figure 8.2 shows performance gains and combined coefficients in the turn 5 and turn 6-7 zones. Initial performance is set close to maximum, as thermal blankets were used. Any subsequent tire cool-down is effectively handled by the mission planner. The ideal progression saturates at

[1]Right of way: is obtained by the attacking car with respect to the defending car when the former enters within a 15-meter longitudinal margin. The defending car must leave a lateral gap of three meters from the track edge.

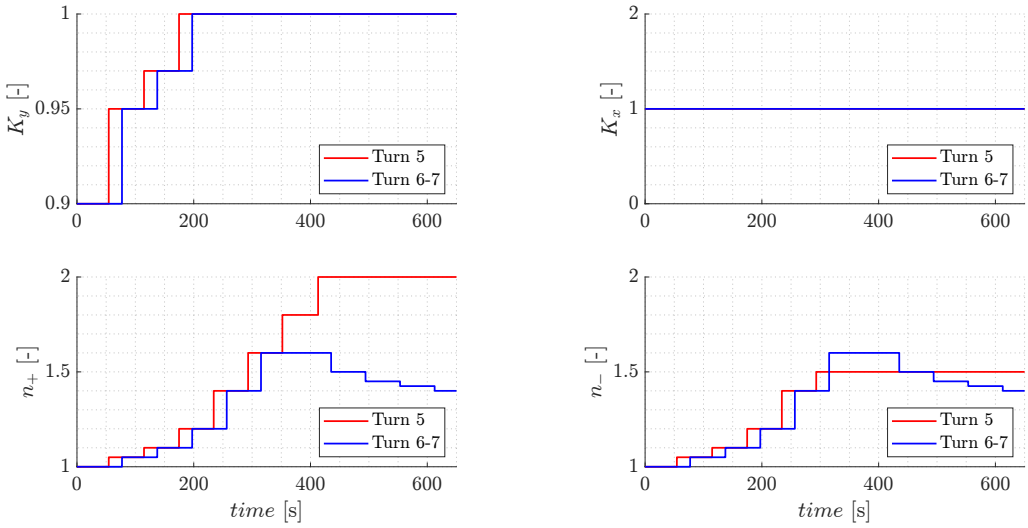


Figure 8.2: Progression over race time of mission planner's performance target: pure lateral (K_y), pure longitudinal (K_x), positive (n_+) and negative (n_-) combined slip. For clarity, only turn 5 and turn 6-7 sections are shown.

lap 5. On lap 6, a lapping maneuver results in a temporary loss of approximately 1.8 s/lap, after which no further improvement is observed. This is primarily due to:

1. Interaction with car 33: shortly after turn 6 entry, car 33 occupies the right-hand side of the track, preventing the autonomous vehicle from following the ideal racing line and forcing a suboptimal trajectory.

2. Planner and controller response in turn 6-7: the evasive maneuvers required to react to car 33 induce abrupt steering corrections and understeer. The resulting feedback causes the mission planner to reduce the target performance in this zone (blue signal at $\approx$ 400 s), limiting further lap-time improvement despite favorable grip conditions.

The last point highlights a limitation of the current mission planner formulation: it cannot distinguish between nominal on-the-edge maneuvers and forced deviations caused by opponent interactions.

8.2 Best-lap analysis

This section analyzes the best lap extracted from race data. The lap was completed with tires and brakes not yet at their operating temperature, therefore further performance improvements were still possible. Despite these non-ideal conditions, the achieved lap time lies within approximately 2% of the human driver reference lap (57.569 s).

8.2.1 MPP and MPC results

Fig. 8.3 reports a detailed comparison between planner references and controller performance, represented as vehicle state feedback.

Velocity tracking (v_x). The first subplot shows a noticeable mismatch between the longitudinal velocity planned by the local planner and the velocity actually achieved by the MPC, particularly at high-speed sections ($s \approx$ 100 m, 1000 m, and 2000 m). This

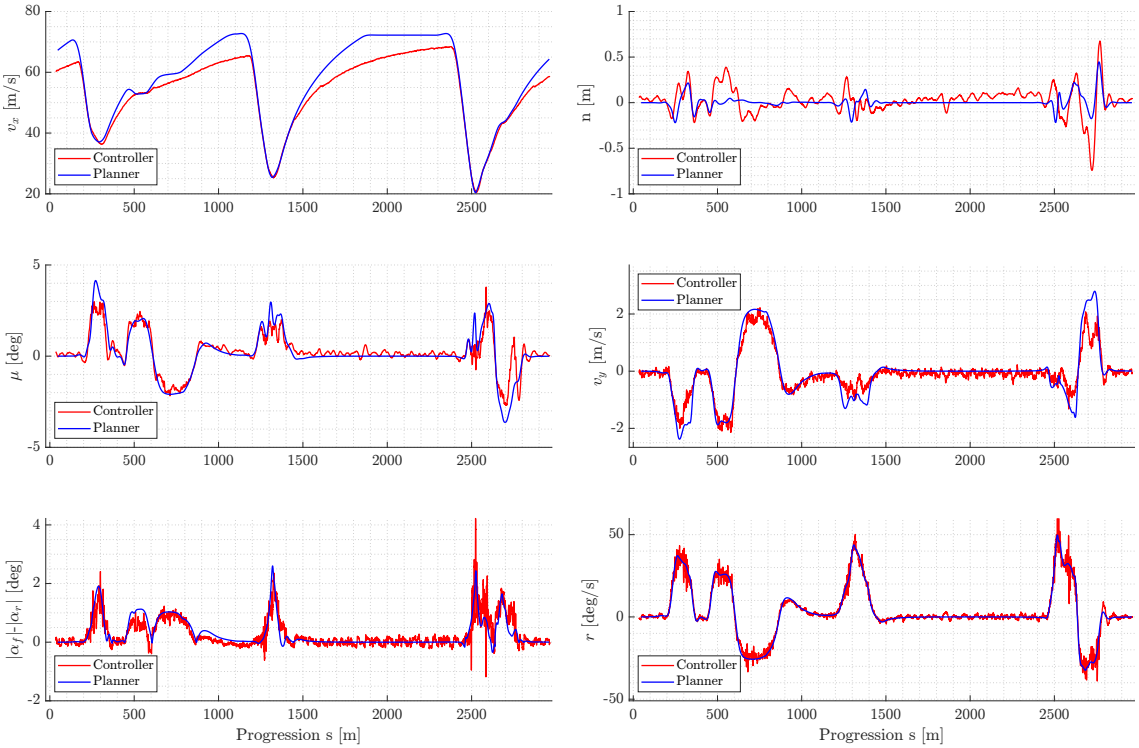


Figure 8.3: Comparison between MPP and MPC control-related states and vehicle dynamics quantities, indicating the difference between planned and feedback signals.

behavior is the result of a deliberate design choice: the planner overestimates the available longitudinal acceleration (a_x as a function of v_x), effectively assuming higher engine torque capabilities.

This strategy ensures that the controller consistently commands full throttle when this mismatch occurs. The torque overestimation creates a gap between the planned and actual velocity, which the controller attempts to close by requesting maximum acceleration. Without this bias, the controller could reduce the throttle while tracking the reference, for example in the presence of feedback overshoots or when it is close to the target velocity. Moreover, it guarantees a more jerk-free velocity references at braking points for the controller, which represent a weak point of the design, as detailed in Sec.5.4.4.

The same mismatch also manifests in some medium-speed ($s \approx 300$ m) and high-speed (700 m) corners, where the engine, despite operating at full load, cannot reach the ideal velocity planned upstream. This mismatch also incorporates longitudinal controller modeling inaccuracies, as investigated in Sec.8.2.2.

Control-related states (n, μ). Although the local planner operates in ideal open-loop conditions without feedback on tracking errors, it generates a path that slightly deviates from the global racing line. This difference mainly stems from model parameters differences between the global and local planning layers, combined with a slightly different cost function tuning.

The MPC exhibits strong lateral tracking performance: the lateral error closely matches the reference generated by the local planner. The largest errors occur around $s \approx 500$ m (turn 3), where a significant velocity mismatch is present and linearization errors are expected to be more pronounced, and around $s \approx 2700$ m (turn 8). The latter represents a clear outlier compared to the average and maximum lateral error observed elsewhere on

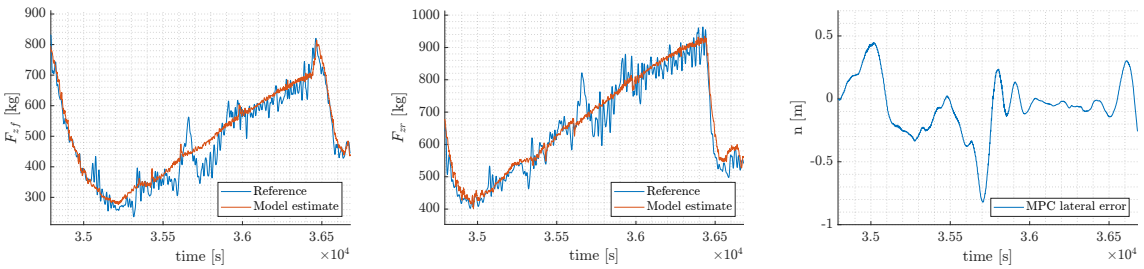


Figure 8.4: Mismatch between load cell sensors data and MPC model predicted vertical load. At abscissa ≈ 3.56 an unmodeled road bump, therefore not captured by the MPC simplified model.

the track. In Fig.8.4, the discrepancy between actual and predicted vertical loads per axle is shown, along with the lateral error in the same track-portion. Further analysis revealed that this anomaly is caused by track geometry. While the MPC single-track accounts for banking effects, it does not model rapid elevation changes. In turn 8, a pronounced crest induces a sudden increase in vertical tire loads (F_z), leading to higher lateral forces (F_y) and causing the vehicle to cut the ideal line by more than 60 cm. This effect is therefore attributable to unmodeled three-dimensional track features.

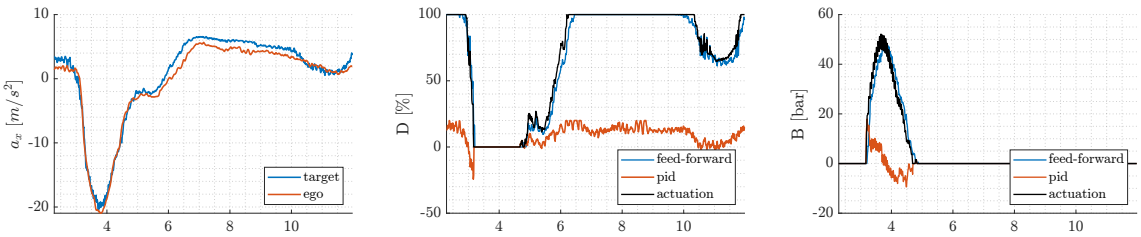
A similar trend is observed in the heading error. Across most of the lap, the heading error remains consistent with the planner reference, indicating good agreement between the vehicle model and actual vehicle behavior. However, in turn 8 the heading error exhibits an opposite trend compared to the planned reference, further confirming the dominant influence of local track elevation effects.

Dynamic-related states (v_y, r) and understeer angle. A close agreement between the planned and measured lateral velocity v_y, yaw rate r, and understeer angle further supports the goodness of the adopted vehicle model. Aside from measurement noise (particularly evident in the yaw rate feedback) the controller is able to accurately reproduce the states predicted by the planner through appropriate control actions.

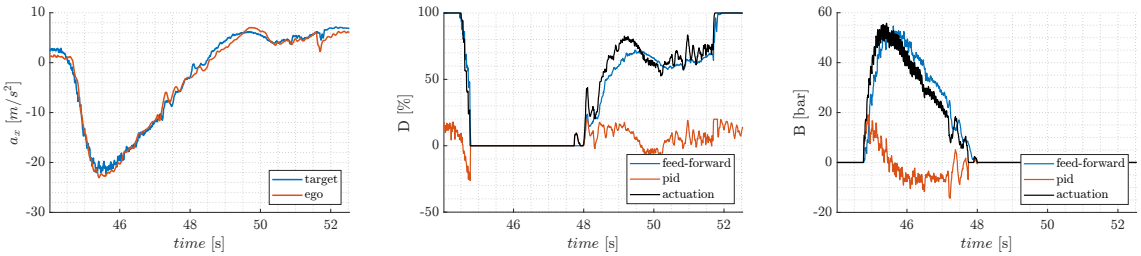
Larger mismatches are observed in turn 1 ($s \approx 300$ m), where the vehicle operates close to the combined-slip limit. More generally, the most pronounced deviations occur during combined-slip phases of cornering, notably in the v_y profile around $s \approx 300$ m, $s \approx 1300$ m, and $s \approx 2600$ m. Regarding turn 1, the longitudinal velocity mismatch further contributes to generate discrepancies in lateral dynamics. Nevertheless, the lateral velocity error remains sufficiently small that the understeer angle closely matches the planned trajectory, indicating consistent global handling behavior. The mismatch observed in turn 8 ($s \approx 2700$ m) is instead attributable to unmodeled variations in vertical load. Since this effect is not explicitly captured by the planner or controller models, the vehicle exhibits a trajectory deviation that was not foreseen in the planning phase.

8.2.2 Longitudinal controller results

As discussed in Sec. 8.2.1, the poor longitudinal velocity tracking is primarily caused by a mismatch between the planned speed profile and the actual vehicle capabilities. However, margins for improvement in the longitudinal controller exists: despite its overall effectiveness in steady-state driving and braking maneuvers, the longitudinal controller exhibits reduced accuracy during transient phases. The results associated with poor longitudinal tracking are shown in Fig. 8.5.



(a) Turns 1 and 2.



(b) Turns 6-7.

Figure 8.5: Longitudinal tracking error between target, set by the MPC, and feedback, along with the longitudinal controller commands (throttle, D, and brake, B).

In Fig. 8.5a, the braking point at $t \approx 3$ s is slightly delayed due to turbo-lag: even after the throttle command is set to zero, the turbocharger continues to generate boost pressure, resulting in residual positive engine torque. This unmodeled feedforward effect introduces a delay that the throttle PID (subplot 2) and brake PID (subplot 3) are able to partially, but not fully, compensate. Overall, the braking maneuver is well handled by the longitudinal controller, with a peak acceleration tracking error of approximately 1.5 m/s^2. Additional minor PID corrections are attributed to the unmodeled thermal behavior of the braking system, which is particularly relevant for carbon-carbon brakes. In this case, the friction coefficient between pad and disc (see Eq. (5.21)) can vary significantly with brake temperature. The largest tracking error occurs during the re-acceleration phase at $t \approx 6$ s. Here, the target and feedback signals exhibit a temporal offset, again caused by turbocharger dynamics. Turbo-lag introduces a substantial delay that is difficult to model, as it depends on several variables (e.g., engine speed and wastegate pressure). A more accurate representation of this effect within the controller model could enable anticipation of the throttle command, effectively compensating for the turbo lag and reducing the temporal misalignment observed in subplot 1.

To partially mitigate this, the throttle lookahead is set to 0.12 s, corresponding to the third step of the MPC prediction horizon, so that the controller tracks a future acceleration target. Larger lookahead values are possible, but potentially unsafe: excessive anticipation of positive torque can activate the limited-slip differential too early, inducing premature yaw moments and causing the vehicle to rotate ahead of the intended turn apex. Conversely, the brake lookahead is set to the second horizon step (0.08 s), as the braking system exhibits faster dynamics compared to the powertrain.

A similar behavior is observed in turns 6-7, as shown in Fig. 8.5. As the acceleration demand is less aggressive, the mismatch between reference and response is smaller. An overshoot, attributable to turbo-lag, is visible around *time* ≈ 49. Additionally, at *time* ≈ 47, a small spike in the brake actuation (Front Left) can be observed, caused by the ABS intervention.

8.3 Overtake Analysis

This section presents the results of two representative overtaking maneuvers performed during the race. As discussed in Sec. 5, the control layer is not explicitly aware of multi-vehicle interactions, as its role is limited to accurately tracking the reference trajectory generated by the local planner. For this reason, the analysis in this section primarily focuses on the planning predicted path.

In both scenarios, the local planner successfully generated dynamically feasible trajectories while safely managing interactions with surrounding vehicles and avoiding any contact, producing collision-free solutions that respected both vehicle dynamics and race regulations. Two overtaking scenarios are considered:

1. **Lap 2 - On-track overtake.** Car 6 performs an inside overtake on Car 33 at turn 6.

2. **Lap 12 - Lapping maneuver.** Car 6 overtakes a slower vehicle (Car 8) during a lapping scenario.

Overall, these results highlight the ability of the planning framework to handle multi-agent racing scenarios in a principled and robust manner, leveraging predictive obstacle representations and dynamic constraints to generate safe and competitive overtaking trajectories.

Lap 2 - On-track overtake. Both vehicles nominally follow their respective racing lines, with local deviations induced by mutual interaction and collision-avoidance constraints. The results are shown in Fig. 8.6, organized into eight subplots representing successive time instants (first row left-to-right, then second row).

Initially, the planner directed the trajectory toward the right side as the navigable tunnel opened in that direction (subplots 1-2). When the opponent moved to the braking

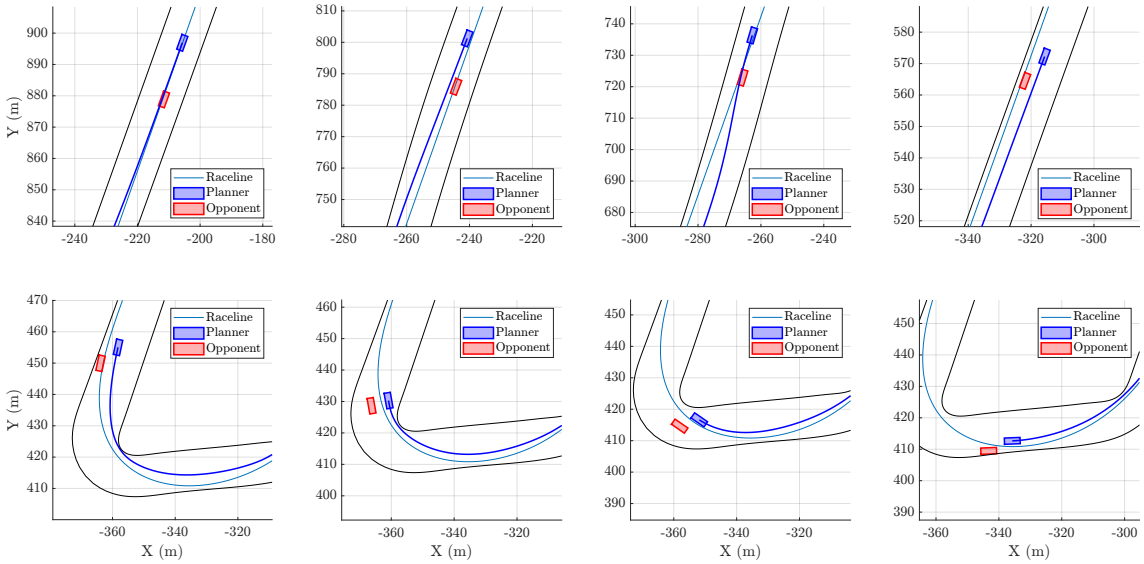


Figure 8.6: Local planning results during on-track overtake. The blue box represents our vehicle's position, while red box represent the opponent's position. The light-blue line represents the ideal path, while blue thick line represent the new generated, collision-free path.

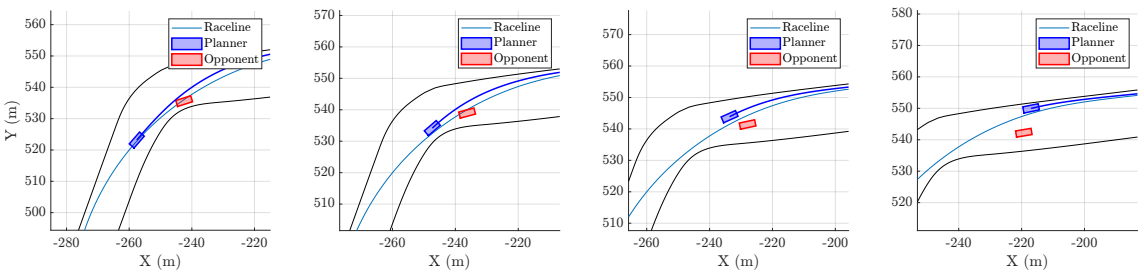


Figure 8.7: Local planner results during the lapped car 8 overtake. The legend follows the same rules as in Fig. 8.6.

point on the right side, the planner recomputed a trajectory occupying the inside of the track (subplots 3-4). During braking, the vehicle gained the right-of-way on the inside line of the corner (subplot 5). The overtake is completed successfully in the final subplot. The overtake logic ensures lateral safety margins (approximately 3 m) to avoid obstructing the opponent, allowing the planner to smoothly reconnect to the optimal racing line.

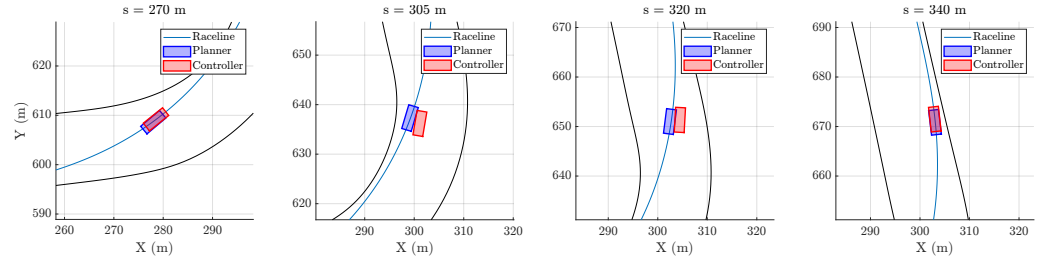
Although the opponent left sufficient maneuvering space, the planner slightly exceeds the track margins. This occurs when the costs on sideslip angle (β) and tire soft constraints dominate over the soft track-margin penalties, highlighting a limitation of soft-constraint-based track representations versus hard bounds. A further reduction in speed would allow the vehicle to remain within the track limits. However, in this situation braking was already at full performance, and additional deceleration was not dynamically feasible due to the inner-line geometry of the maneuver. As a result, a minor violation of the track margin represented the only viable solution.

Lap 12 - Lapping maneuver. According to race regulations, the lapped vehicle is required to move to an outer line, increasing the available navigable region. Car 6 nominally follows the racing line, with local deviations induced by interaction with the opponent. The results are shown in Fig. 8.7, organized into four subplots representing successive time instants (left-to-right).

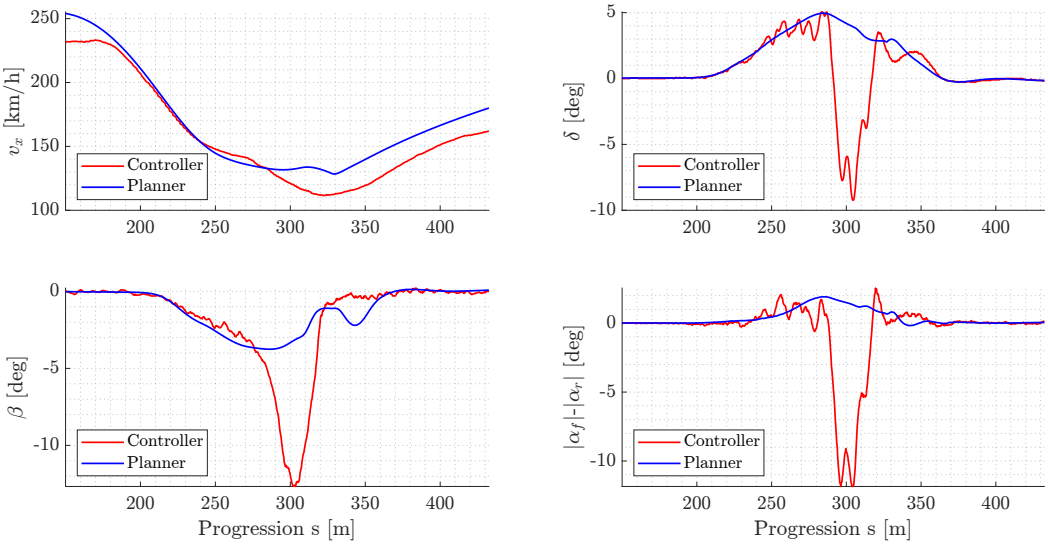
Due to the performance difference between the two vehicles, this scenario is particularly challenging, as the collision-free zones are less stable and harder to predict. Nevertheless, the local planner generated a smooth outer trajectory for the overtake, minimizing disruption to the nominal racing line and successfully completing the maneuver.

8.4 High-dynamics scenarios

In high-performance autonomous driving, achieving high speed is important, but even more critical is the robustness to disturbances. This aspect becomes paramount in autonomous racing applications, where operating conditions are often unpredictable and the system may approach instability at any instant, especially when driving close to the tire friction limits. One of the most significant outcomes of the proposed solution is precisely this robustness. In nearly all oversteer-prone scenarios, the controller successfully recovered vehicle stability, effectively containing both the sideslip and understeer angles. This behavior is primarily enabled by the high execution rate of the MPC, which operates at 100 Hz and provides reaction times more than one order of magnitude faster than those achievable by a human driver. Additionally, the accuracy of the vehicle model and a cost function specifically tuned for high-dynamics conditions further contribute to a reliable



(a) Planned and actual (controller feedback) position and heading in turn 1.



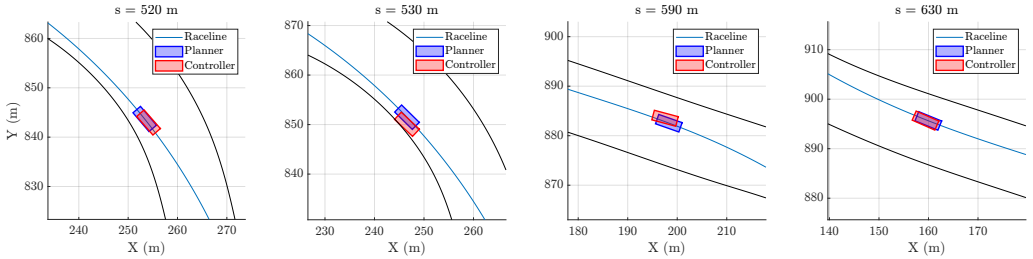
(b) Planned and actual (controller feedback) vehicle dynamics and steering command in turn 1.

Figure 8.8: Difference between predicted (planner) and actual (controller) postition, heading, steering command and vehicle dynamics quantities.

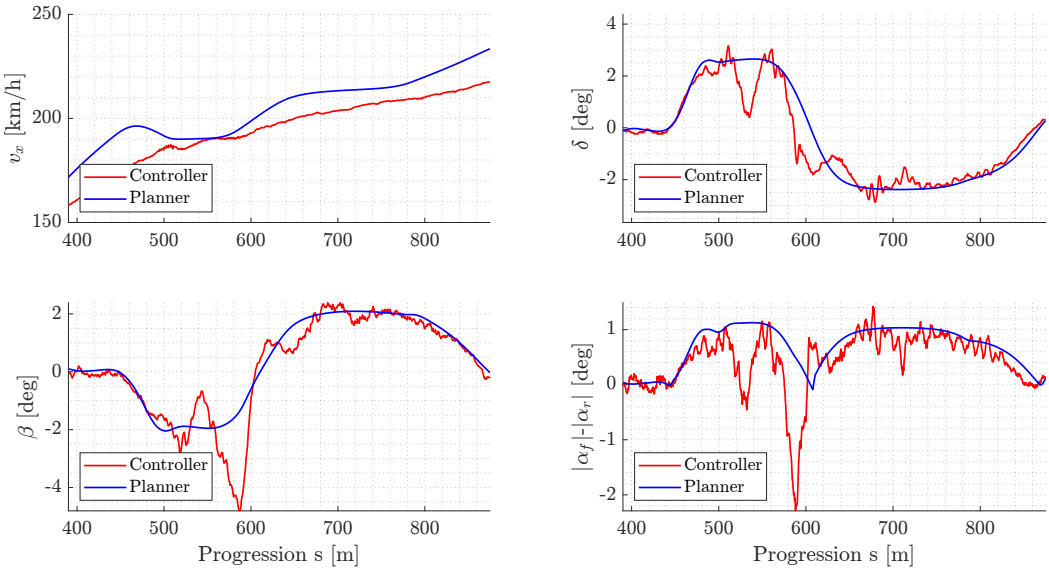
yet highly effective control system. In this section, two edge-case scenarios are presented to validate the robustness and stability properties of the control layer.

Figure 8.8 shows an oversteer event in a medium-speed corner, specifically turn 1. The upper subplots (8.8a) display the difference between the planned (blue) and actual (red) position and heading. The local planner stays consistent with the global racing-line, beside small yet unnoticeable adjustment. The controller exhibits a high heading error (subplot 2), caused by the strongly increased understeer angle absolute value. The lower subplots (8.8b) show the planned (blue) dynamic quantities and steering input versus the actuated ones (red). The planned trajectory serves as the baseline reference. In this scenario, the vehicle enters the corner at a slightly higher speed than the local planner computed, inducing instability. Consequently, the understeer angle briefly inverts trend, becoming highly negative around $s \approx 300$ m. The controller promptly countersteers, limiting the sideslip angle to approximately $\beta \approx 13°$. Stability is recovered by $s \approx 320$ m, with minimal oscillations thereafter.

In this scenario, cost function blending proven to be crucial: during the oversteer event, the vehicle drifts outward, generating a negative lateral error while maintaining the reference line to the left. Without cost function adaptation, the lateral tracking term would dominate, forcing the steering to attempt to return the vehicle to the reference line. This could prevent the controller from effectively countersteering to reduce the sideslip. With



(a) Planned and actual (controller feedback) position and heading in turns 2-3.



(b) Planned and actual (controller feedback) vehicle dynamics and steering command in turns 2-3.

Figure 8.9: Difference between predicted (planner) and actual (controller) postition, heading, steering command and vehicle dynamics quantities.

cost function blending, however, the lateral error term is temporarily down-weighted, allowing the controller to prioritize stabilizing the vehicle. As a result, the steering command focuses on recovering stability rather than strictly following the reference path.

Figure 8.9 shows two oversteer events occurring in a high-speed corner, specifically turn 2. This track section is particularly challenging due to the significant track slope and the high nominal cornering speed, approximately $v_x \approx 195$ km/h. While approaching the apex, a mild nose-in behavior, visible as the first peak in understeer angle at $s \approx 530$ m, induces a positive lateral error, causing the vehicle to cut the corner (Fig. 8.9a). A prompt steering correction brings the understeer angle back in line with the planner baseline. Shortly thereafter, a second oversteer event occurs during the change of direction when approaching turn 3, at $s \approx 590$ m. Although the heading error is less pronounced in this case, the high longitudinal speed results in a large lateral velocity associated with the observed sideslip angle. The controller reacts by significantly anticipating the steering input to recover vehicle stability, as clearly shown in Fig. 8.9b.

Once again, the planner (operating in open loop) remains unaffected by vehicle feedback and maintains a stable and consistent reference trajectory. We observed that this configuration further facilitates stability recovery: the fixed path reference allows the heading error to grow naturally, enabling the controller to anticipate instability and react promptly. In contrast, a closed-loop planner configuration, where the planned state is re-

set to the actual vehicle position at each iteration, would continuously reset the perceived tracking error, thereby limiting the controller's ability to react effectively in high-error, high-dynamics scenarios.

Chapter 9

Conclusions and Future Works

This thesis advances model-based planning and control for autonomous racing by integrating the proposed contributions into a fully operational and experimentally validated framework.

A first key contribution is the definition of a coherent motion planning and control architecture that consistently leverages a vehicle dynamics model across all layers. Extensive simulation and experimental testing were conducted using models of varying complexity, leading to the identification of a formulation that represents a balanced trade-off between accuracy, robustness, and real-time feasibility and calibration effort. Although originally motivated by high-performance autonomous racing, the resulting architecture proved effective across a wide range of operating conditions, from low-speed maneuvers to highly dynamic scenarios, and is therefore applicable beyond the racing domain. The proposed framework was designed and validated against previous approaches, demonstrating a substantial performance improvement at the system level. In particular, the mission planner emerged as a solid foundation toward full vehicle autonomy, operating in conjunction with the longitudinal planner to generate physically consistent references that continuously adapt to vehicle feedback. The MPP module operates in open-loop and is capable of providing stable and consistent references to the controller, independently of feedback quality. Conversely, the MPC operates in closed-loop and demonstrated strong robustness in edge-case scenarios, maintaining vehicle stability and high path-following performance even under highly dynamic conditions. Particular attention was devoted to classical challenges in optimization-based approaches, such as cost function tuning and model linearization. Within this framework, a dynamic cost function is introduced, allowing the planner and controller to smoothly balance competing objectives and operate effectively across different driving conditions. This formulation enables the system to adapt its behavior without structural changes, addressing a common limitation of optimization-based methods that rely on fixed or scenario-specific objectives. Moreover, the analysis of model linearization shows that, under suitable conditions (such as high execution frequency and sufficiently informative feedback) linearized formulations can achieve performance comparable to fully nonlinear models, while significantly improving computational efficiency. Overall, this work highlights the importance of a consistent motion framework in which planning, control, and estimation are tightly coupled through a shared vehicle dynamics representation.

A second key contribution lies in the development of a simple yet effective grip and parameter estimation strategy aimed at tracking time-varying tire characteristics. This module enables the framework to adapt online to changing operating conditions, such as transitions between low- and high-grip environments, without requiring extensive re-

tuning. Experimental results demonstrate that both planning and control benefit significantly from this capability: trajectory and speed references adapt to the identified dynamics at the planning level, while control actions naturally adjust their aggressiveness and driving style, resulting in improved tracking performance and enhanced stability.

Finally, the thesis presents a validated methodology for the development of a high-fidelity, real-time-capable simulator. This simulator played a central role throughout the work, supporting model calibration, controller validation, and systematic testing in scenarios that are difficult, unsafe, or impractical to reproduce on track. Beyond its role as a development tool, the simulator also proved valuable for interpreting on-track behavior, enabling the analysis of effects later observed in real experiments, such as steering anticipation during turn-in, lateral errors induced by road geometry, or velocity mismatches at corner entry.

9.1 Future Works

The developments presented in this thesis open several avenues for future research, spanning simulation, grip estimation, and model-based predictive planning and control. The following subsections outline potential directions for further enhancement of the proposed framework.

9.1.1 High-Fidelity Simulation

- Extend the current single-point tire-road contact formulation to a multi-point (e.g., five-point) contact model. While the present simulator adopts a single contact point per tire, a multi-point representation would allow a more accurate description of load distribution over the tire, especially over curbs, uneven surfaces, and three-dimensional track geometries. Such an extension would improve the fidelity of tire-road interaction in highly dynamic conditions and better capture local variations in contact patch behavior. Multi-point contact formulations are well established in the state of the art and have been adopted in commercial and research-grade vehicle dynamics simulators[1] for several years; integrating them would thus represent a natural and mature evolution of the current models.

- The simulator has already been used to generate high-quality datasets for the calibration of simplified mathematical models. A natural and impactful extension would be the systematic generation of large-scale datasets to support data-driven and neural network-based planning and control approaches. Recent works have demonstrated that learning-based methods can achieve super-human performance in fast-lap when trained in arcade-style simulators [54–56]. However, most of these results are obtained in proprietary commercial simulators. When these approaches are applied within a slightly more realistic environments, they still fail to outperform humans consistently under all conditions, as shown in [8]. Compared to game-oriented simulators, a physics-based, high-fidelity simulator provides more representative vehicle dynamics, tire behavior, feedback noise and actuator responses, reducing the sim-to-real gap that often limits learning-based methods. Leveraging such a simulator would enable the training of neural policies on physically consistent data, including extreme and safety-critical scenarios that are difficult or unsafe to reproduce on track. The expected outcome is the development of learning-based components that

[1] https://trinacriasimracing.wordpress.com/acc-the-5-point-tyre-model/

generalize better to real vehicles and can be meaningfully integrated or compared with model-based solutions.

9.1.2 Grip and Tire Parameter Estimation

- Extend the current estimation module to explicitly incorporate tire temperature, constructing a 3D parametric surface for tire parameters as a function of slip angle and measured temperature. This would allow predictive adjustment of vehicle control strategies rather than purely reactive adaptation, enabling the planner and controller to anticipate grip changes, avoid entering high-dynamics states unexpectedly, and optimize trajectory and speed profiles while maintaining safety.
- Integrate longitudinal tire characteristics estimation alongside lateral parameters, improving the estimation of the tire behavior as already detailed in Sec.6.2.
- Although based on a classical nonlinear regression approach, our solution represents a valuable contribution to the literature by demonstrating effective estimation of tire-road friction parameters and significant improvements in path planning and control performance. Future work could explore alternative estimation and learning methods proposed in the literature, including neural network-based models, to capture complex, nonlinear tire behavior under varying conditions [40–43, 49]. This would also help account for unmodeled effects that are difficult to isolate or reproduce within a simplified model formulation.

9.1.3 Model Predictive Planning and Control (MPPC)

While current results (measured in absolute lap times) already achieve performance within approximately 2% of a professional Formula 1 racing driver, targeted improvements in braking and cornering could potentially reduce this gap to below 1%. Many research groups with strong expertise in vehicle dynamics and modeling are moving towards neural solutions, often physics-free, i.e., end-to-end networks that do not rely on an explicit vehicle model. However, in the context of full-stack autonomous racing, where additional aspects must be carefully addressed, such as absolute robustness, flexibility across different driving styles, interaction with other competitors, compliance with racing regulations (e.g., track flags, full-course yellows), rapid adaptation to changing conditions, and limited, costly on-track testing, solutions like the one proposed in this work remain highly competitive and have not yet been surpassed. As already detailed in Sec. 5.4, there is still room for improvement. Briefly, the main avenues for enhancement are:

- Address limitations of the current MPP-MPC jerk cascade, particularly during aggressive braking. Future iterations could incorporate adaptive jerk weighting or alternative smoothing techniques to prevent overly conservative longitudinal behavior.
- Explore the integration of low-level vehicle controllers, such as ESP, directly into the high-level MPC formulation, potentially using brake torques as explicit control inputs.
- Improve the handling of slip ratio within the predictive controller to enhance robustness in high-dynamics maneuvers. Incorporating this physical behavior at the MPC level is expected to provide a good trade-off, as the controller can rely on vehicle feedback to react promptly in most scenarios.

- Investigate data-driven or neural-network-assisted MPC approaches. Neural methods could complement the current cost-function blending strategy to push performance closer to the theoretical limit. In particular, data-driven approaches may help in defining a more accurate and complete vehicle model, avoiding most of classic problems in vehicle modeling (for example, the numerical stability Sec.3.3)

9.1.4 Summary

Overall, while the current framework demonstrates strong performance and versatility, there is significant potential to extend its predictive, adaptive, and learning capabilities. The combination of high-fidelity simulation, advanced tire modeling and identification, and improved MPPC algorithms could further bridge the gap between autonomous and professional human driving performance, offering a path for continued research and innovation.

Appendix A

Appendix

A.1 Point-mass model formulation

The point-mass model represents the simplest vehicle dynamics approximation. It assumes the vehicle can be represented as a single material point moving on a plane, neglecting rotational dynamics such as yaw. The model is derived from the equations of uniform circular motion. The lateral and longitudinal dynamics are described by:

$$\begin{cases} a_y = & \frac{v_x^2}{R}, \\ a_x = & \dot{v}_x. \end{cases} \tag{A.1}$$

Acceleration limits can be expressed either as a function of a tire model, as detailed in Sec. 5.2.3, or defined via data-driven approaches. Lateral and longitudinal accelerations are generally coupled using the friction ellipse, as shown in Eq. (5.18).

Since the point-mass model does not include yaw dynamics, the yaw rate $\dot{r}$ is not explicitly modeled. Consequently, the model is not a full vehicle representation. Its outputs (e.g., lateral acceleration) cannot be directly translated into low-level vehicle commands, such as steering angle or wheel torques. Additional mapping or control strategies are required to transform the point-mass references into feasible actuator commands suitable for a real vehicle or a high-fidelity simulator. For this reason, although the point-mass model is used in certain layers of the planner in this thesis, it is not extensively discussed or compared with other vehicle models proposed in the literature.

A.2 Locked Differential

The yaw moment is derived from the rotational equilibrium of the rear axle in the Z-axis direction:

$$M_{\text{diff}} = (F_{xrr} - F_{xrl}) \cdot \frac{t_r}{2}. \tag{A.2}$$

The longitudinal forces include both the contribution of the locked differential and the traction dynamics: $F_{xrj} = F'_{xrj} + F^0_{xrj}$. The longitudinal forces experienced during vehicle coasting are estimated using a non-linear Pacejka tire model (Eqs.(3.14)). We implement the simpler nonlinear version, which reads:

$$\begin{aligned} F'_{x,rl} &= F_{z,rl}\, D_{xr}\, \sin\Big(C_{xr} \tan^{-1}(B_{xr}\, \kappa_{x,rl})\Big) \\ F'_{x,rr} &= F_{z,rr}\, D_{xr}\, \sin\Big(C_{xr} \tan^{-1}(B_{xr}\, \kappa_{x,rr})\Big) \end{aligned} \tag{A.3}$$

where D_{xr}, B_{xr} and C_{xr} are the macro-parameters of the magic formula [58]. $F_{z,rl}$ and $F_{z,rr}$ represent the vertical load on each rear wheel. The slip ratio is expressed as

$$\begin{aligned} \kappa_{x,rl} &= \frac{v_x - v_{x,rl}}{v_x} \\ \kappa_{x,rr} &= \frac{v_x - v_{x,rr}}{v_x} \end{aligned} \tag{A.4}$$

which depends on the ideal speed of the rear wheels, taking into account the vehicle's geometry and its yaw rate:

$$\begin{aligned} v_{x,rl} &= v_x - r \cdot \frac{t_r}{2} \\ v_{x,rr} &= v_x + r \cdot \frac{t_r}{2} \end{aligned} \tag{A.5}$$

To account for the traction forces, we have included $F_{xr}^0 = ma_x$, which is distributed between the rear wheels through the lateral load transfer. In this way, we can check for a possible nose-in during the phase of turn-exit.

A.3 Simplified Limited Slip Differential

A simplified LSD model was implemented to approximate the torque transfer between the rear wheels during the traction and release phases of the tricycle model. For compactness, only the traction phase is described. The effective locking force generated by the differential is modeled as:

$$F_{\text{diff}} = \xi \, \max(F_0, \, \epsilon_d \, F_{\text{eng}}) \, \text{sign}(r), \tag{A.6}$$

where F_{eng} is the drive (powertrain) force at the differential input, F_0 is the preload force ensuring a minimum locking effect, ξ is an engagement factor that suppresses the locking force when the vehicle is traveling straight, and $\text{sign}(r)$ determines the turning direction. The coefficient ϵ_d represents the fraction of torque that can be transferred between the rear wheels to maintain traction when one loses grip (with 0% corresponding to an open differential and 100% to a fully locked one). Similarly, ϵ_c (in Tab.3.3) denotes the locking percentage during coasting conditions.

The longitudinal forces at the rear wheels are computed as:

$$\begin{aligned} F_{x,rl} &= \tfrac{1}{2}(F_{\text{eng}} - F_{\text{diff}}), \\ F_{x,rr} &= \tfrac{1}{2}(F_{\text{eng}} + F_{\text{diff}}), \end{aligned} \tag{A.7}$$

If one wheel exceeds its traction limit, the surplus longitudinal force is redistributed to the opposite side, up to its maximum capacity:

$$\begin{aligned} F_{x,rl}^{\text{ex}} &= F_{x,rl} - F_{x,rl}^{\max}, \\ F_{x,rr}^{\text{ex}} &= F_{x,rr} - F_{x,rr}^{\max}. \end{aligned} \tag{A.8}$$

where the maximum force of each tire is evaluated as $F_{x,rj}^{\max} = F_{z,rj} \, D_{xr}$. Finally, the wheel forces are first saturated at their respective traction limits:

$$\begin{aligned} F_{x,rl} &= \min(F_{x,rl}, F_{x,rl}^{\max}), \\ F_{x,rr} &= \min(F_{x,rr}, F_{x,rr}^{\max}), \end{aligned} \tag{A.9}$$

and any excess force is redistributed according to:

$$\begin{aligned} &\text{if } F_{x,rl} = F_{x,rl}^{\max} \Rightarrow F_{x,rr} = \min(F_{x,rr} + F_{x,rl}^{\text{ex}}, F_{x,rr}^{\max}), \\ &\text{if } F_{x,rr} = F_{x,rr}^{\max} \Rightarrow F_{x,rl} = \min(F_{x,rl} + F_{x,rr}^{\text{ex}}, F_{x,rl}^{\max}), \end{aligned} \tag{A.10}$$

which allows computing the differential moment according to the formulation in Eq. (A.2).

This simplified formulation, validated against the multi-body reference model, reproduces the main effects of an LSD while maintaining computational efficiency suitable for real-time optimal control applications.

Bibliography

[1] Johannes Betz, Hongrui Zheng, Alexander Liniger, Ugo Rosolia, Phillip Karle, Madhur Behl, Venkat Krovi, and Rahul Mangharam. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488, 2022.

[2] Tantan Zhang, Yueshuo Sun, Yazhou Wang, Bai Li, Yonglin Tian, and Fei-Yue Wang. A survey of vehicle dynamics modeling methods for autonomous racing: Theoretical models, physical/virtual platforms, and perspectives. *IEEE Transactions on Intelligent Vehicles*, 9(3):4312–4334, 2024.

[3] Ayoub Raji, Alexander Liniger, Andrea Giove, Alessandro Toschi, Nicola Musiu, Daniele Morra, Micaela Verucchi, Danilo Caporale, and Marko Bertogna. Motion planning and control for multi vehicle autonomous racing at high speeds. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, October 2022.

[4] Ayoub Raji, Danilo Caporale, Francesco Gatti, Andrea Giove, Micaela Verucchi, Davide Malatesta, Nicola Musiu, Alessandro Toschi, Silviu Popitanu, Fabio Bagni, Massimiliano Bosi, Alexander Liniger, Marko Bertogna, Daniele Morra, Francesco Amerotti, Luca Bartoli, Federico Martello, and Riccardo Porta. er.autopilot 1.0: The full autonomous stack for oval racing at high speeds. *Field Robotics*, 4:99–137, 01 2024.

[5] Alessandro Toschi, Nicola Musiu, Francesco Gatti, Ayoub Raji, Francesco Amerotti, Micaela Verucchi, and Marko Bertogna. Guess the drift with lop-ukf: Lidar odometry and pacejka model for real-time racecar sideslip estimation. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 885–891, 2024.

[6] Ayoub Raji, Nicola Musiu, Alessandro Toschi, Francesco Prignoli, Eugenio Mascaro, Pietro Musso, Francesco Amerotti, Alexander Liniger, Silvio Sorrentino, and Marko Bertogna. A tricycle model to accurately control an autonomous racecar with locked differential. In *2023 IEEE 11th International Conference on Systems and Control (ICSC)*, pages 782–789. IEEE, December 2023.

[7] Ayoub Raji, Danilo Caporale, Francesco Gatti, Alessandro Toschi, Nicola Musiu, Micaela Verucchi, Francesco Prignoli, Davide Malatesta, AndrÃ© Fialho Jesus, Andrea Finazzi, Francesco Amerotti, Fabio Bagni, Eugenio Mascaro, Pietro Musso, and Marko Bertogna. er.autopilot 1.1: A software stack for autonomous racing on oval and road course tracks. *IEEE Transactions on Field Robotics*, 1:332–359, 2024.

[8] Adrian Remonda, Nicklas Hansen, Ayoub Raji, Nicola Musiu, Marko Bertogna, Eduardo E. Veas, and Xiaolong Wang. A simulation benchmark for autonomous racing

with large-scale human data. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[9] Nicola Musiu, Eugenio Mascaro, Ayoub Raji, Alessandro De Felice, Silvio Sorrentino, and Marko Bertogna. A comprehensive benchmark of vehicle dynamics models for autonomous racing: a deep dive into mpc. *Vehicle System Dynamics*, 0(0):1–33, 2026.

[10] Pietro Musso. Developmemt of a tyre thermal model for an autonomous race car. Master's thesis, Università degli studi di Modena e Reggio Emilia, 2023.

[11] Eugenio Mascaro. Sviluppo del modello multi-body della vettura autonoma dallara av21 e ottimizzazione per applicazioni real-time. Master's thesis, Università degli studi di Modena e Reggio Emilia, 2023.

[12] Ayoub Raji. *Model Predictive Planning and Control for Autonomous Racing, from HPC to Embedded Platforms*. PhD thesis, Università degli Studi di Parma, Parma, Italy, 2024.

[13] John K. Subosits and J. Christian Gerdes. Impacts of model fidelity on trajectory optimization for autonomous vehicles in extreme maneuvers. *IEEE Transactions on Intelligent Vehicles*, 6(3):546–558, 2021.

[14] Karl Berntorp, Bjorn Olofsson, Bo Bernhardsson, Kristoffer Lundahl, and Lars Nielsen. Models and methodology for optimal vehicle maneuvers applied to a hairpin turn. In *2013 American Control Conference*, pages 2139–2146, 2013.

[15] Simon Sagmeister, Panagiotis Kounatidis, Sven Goblirsch, and Markus Lienkamp. Analyzing the impact of simulation fidelity on the evaluation of autonomous driving motion control. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 230–237, 2024.

[16] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.

[17] JosÃ© L. VÃ¡zquez, Marius BrÃŒhlmeier, Alexander Liniger, Alisa Rupenyan, and John Lygeros. Optimization-based hierarchical motion planning for autonomous racing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2397–2403, 2020.

[18] Stan Meijer, Alberto Bertipaglia, and Barys Shyrokau. A nonlinear model predictive control for automated drifting with a standard passenger vehicle, 2024.

[19] Basilio Lenzo, Tushar Goel, and Joseph Christian Gerdes. Autonomous drifting using torque vectoring: Innovating active safety. *IEEE Transactions on Intelligent Transportation Systems*, 25(11):17931–17939, 2024.

[20] Michael Thompson, James Dallas, Jonathan Y. M. Goh, and Avinash Balachandran. Adaptive nonlinear model predictive control: Maximizing tire force and obstacle avoidance in autonomous vehicles. *IEEE Transactions on Field Robotics*, 1:318–331, 2024.

[21] Takao Kobayashi, Trey P. Weber, and J. Christian Gerdes. Trajectory planning using tire thermodynamics for automated drifting, 2024.

[22] Trey P. Weber and J. Christian Gerdes. Modeling and control for dynamic drifting trajectories. *IEEE Transactions on Intelligent Vehicles*, 9(2):3731–3741, 2024.

[23] Luqi Tang, Fuwu Yan, Bin Zou, Kewei Wang, and Chen Lv. An improved kinematic model predictive control for high-speed path tracking of autonomous vehicles. *IEEE Access*, 8:51400–51413, 2020.

[24] Philip Polack, Florent AltchÃ©, Brigitte d'AndrÃ©a Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017.

[25] Joshua Spisak, Andrew Saba, Nayana Suvarna, Brian Mao, Chuan Tian Zhang, Chris Chang, Sebastian Scherer, and Deva Ramanan. Robust modeling and controls for racing on the edge, 2022.

[26] Hassan Jardali, Durgakant Pushp, Youwei Yu, Mahmoud Ali, Ihab S. Mohamed, Alejandro Murillo-Gonzalez, Paul D. Coen, Md. Al-Masrur Khan, Reddy Charan Pulivendula, Saeoul Park, Lingchuan Zhou, and Lantao Liu. From zero to high-speed racing: An autonomous racing stack, 2025.

[27] Alexander Wischnewski, Thomas Herrmann, Frederik Werner, and Boris Lohmann. A tube-mpc approach to autonomous multi-vehicle racing on high-speed ovals. *IEEE Transactions on Intelligent Vehicles*, 8(1):368–378, 2023.

[28] Matthias Rowold, Levent Ogretmen, Ulf Kasolowsky, and Boris Lohmann. Online time-optimal trajectory planning on three-dimensional race tracks. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–8. IEEE, June 2023.

[29] Mattia Piccinini, Sebastiano Taddei, Edoardo Pagot, Enrico Bertolazzi, and Francesco Biral. How optimal is the minimum-time manoeuvre of an artificial race driver? *Vehicle System Dynamics*, 0(0):1–28, 2024.

[30] Matteo Corno, Alex Gimondi, Giulio Panzani, Federico Roselli, Andrea Alessandretti, and Sergio M. Savaresi. A non-optimization-based dynamic path planning for autonomous obstacle avoidance. *IEEE Transactions on Control Systems Technology*, 31(2):722–734, 2023.

[31] Arun Muraleedharan. Randomized model predictive control for autonomous racing. 04 2025.

[32] Bei Zhou, Cheng Hu, Jun Zeng, Zhouheng Li, Johannes Betz, Lei Xie, and Hongye Su. Adaptive learning-based model predictive control strategy for drift vehicles, 2025.

[33] Ningyuan Guo, Basilio Lenzo, Xudong Zhang, Yuan Zou, Ruiqing Zhai, and Tao Zhang. A real-time nonlinear model predictive controller for yaw motion optimization of distributed drive electric vehicles. *IEEE Transactions on Vehicular Technology*, 69(5):4935–4946, 2020.

[34] Marsie T. Peterson, Tushar Goel, and J. Christian Gerdes. Exploiting linear structure for precision control of highly nonlinear vehicle dynamics. *IEEE Transactions on Intelligent Vehicles*, 8(2):1852–1862, 2023.

[35] Nathan A. Spielberg, Maximilian Templer, John Subosits, and J. Christian Gerdes. Learning policies for automated racing using vehicle model gradients. *IEEE Open Journal of Intelligent Transportation Systems*, 4:130–142, 2023.

[36] Craig Earl Beal and J. Christian Gerdes. Model predictive control for vehicle stabilization at the limits of handling. *IEEE Transactions on Control Systems Technology*, 21(4):1258–1269, 2013.

[37] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[38] Vincent A. Laurense, Jonathan Y. Goh, and J. Christian Gerdes. Path-tracking for autonomous vehicles at the limit of friction. In *2017 American Control Conference (ACC)*, pages 5586–5591, 2017.

[39] Alberto Bertipaglia, Davide Tavernini, Umberto Montanaro, Mohsen Alirezaei, Riender Happee, Aldo Sorniotti, and Barys Shyrokau. Model predictive contouring control for vehicle obstacle avoidance at the limit of handling using torque vectoring*. In *2024 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1468–1475, 2024.

[40] Kemal Koysuren, Ahmet Faruk Keles, and Melih Cakmakci. Online parameter estimation using physics-informed deep learning for vehicle stability algorithms, 2023.

[41] Stefania Santini, Nicola Albarella, Vincenzo Maria Arricale, Renato Brancati, and Aleksandr Sakhnevych. On-board road friction estimation technique for autonomous driving vehicle-following maneuvers. *Applied Sciences*, 11(5), 2021.

[42] Onur Dikici, Edoardo Ghignone, Cheng Hu, Nicolas Baumann, Lei Xie, Andrea Carron, Michele Magno, and Matteo Corno. Learning-based on-track system identification for scaled autonomous racing in under a minute. *IEEE Robotics and Automation Letters*, 10(2):1984–1991, February 2025.

[43] Sven Goblirsch, Benedikt Ruhland, Johannes Betz, and Markus Lienkamp. Bayesian optimization-based tire parameter and uncertainty estimation for real-world data, 2025.

[44] Daniel Chindamo, Basilio Lenzo, and Marco Gadola. On the vehicle sideslip angle estimation: A literature review of methods, models, and innovations. *Applied Sciences*, 8(3), 2018.

[45] Alberto Bertipaglia, Mohsen Alirezaei, Riender Happee, and Barys Shyrokau. An unscented kalman filter-informed neural network for vehicle sideslip angle estimation. *IEEE Transactions on Vehicular Technology*, PP:1–15, 09 2024.

[46] Elvis Villano, Basilio Lenzo, and Aleksandr Sakhnevych. Cross-combined ukf for vehicle sideslip angle estimation with a modified dugoff tire model: design and experimental results. *Meccanica*, 56, 09 2021.

[47] Sven Goblirsch, Marcel Weinmann, and Johannes Betz. Three-dimensional vehicle dynamics state estimation for high-speed race cars under varying signal quality. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3371–3378. IEEE, October 2024.

[48] Nathan A. Spielberg, Matthew Brown, and J. Christian Gerdes. Neural network model predictive motion control applied to automated driving with unknown friction. *IEEE Transactions on Control Systems Technology*, 30(5):1934–1945, 2022.

[49] Franck Djeumou, Jonathan Y. M. Goh, Ufuk Topcu, and Avinash Balachandran. Autonomous drifting with 3 minutes of data via learned tire models, 2023.

[50] John Chrosniak, Jingyun Ning, and Madhur Behl. Deep dynamics: Vehicle dynamics modeling with a physics-constrained neural network for autonomous racing. *IEEE Robotics and Automation Letters*, 9(6):5292–5297, June 2024.

[51] Nicolas Fraikin, Kilian Funk, Michael Frey, and Frank Gauterin. A fast and accurate hybrid simulation model for the large-scale testing of automated driving functions. *Proceedings of the Institution of Mechanical Engineers, Part D*, 234(4):1183–1196, 2020.

[52] Stephan Rhode, Fabian Jarmolowitz, and Felix Berkel. Vehicle single track modeling using physics guided neural differential equations, 2024.

[53] Akshay Bhoraskar and P. Sakthivel. A review and a comparison of dugoff and modified dugoff formula with magic formula. In *2017 International Conference on Nascent Technologies in the Engineering Field (ICNTE)*, Mumbai, India, 2017. IEEE.

[54] Peter Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael Thomure, and Hiroaki Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602:223–228, 02 2022.

[55] Miguel Vasco, Takuma Seno, Kenta Kawamoto, Kaushik Subramanian, PeterR. Wurman, and Peter Stone. A super-human vision-based reinforcement learning agent for autonomous racing in Gran Turismo. In *Reinforcement Learning Conference (RLC)*, August 2024.

[56] Florian Fuchs, Yunlong Song, Elia Kaufmann, Davide Scaramuzza, and Peter Durr. Super-human performance in gran turismo sport using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4257–4264, July 2021.

[57] Jonathan Y. M. Goh, Michael Thompson, James Dallas, and Avinash Balachandran and. Beyond the stable handling limits: nonlinear model predictive control for highly transient autonomous drifting. *Vehicle System Dynamics*, 62(10):2590–2613, 2024.

[58] Hans B. Pacejka. *Tyre and Vehicle Dynamics*. Butterworth-Heinemann, 3rd edition, 2012.

[59] Massimo Guiggiani. *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*. Springer, 2nd edition, 2018.

[60] C. Innocenti. Questioning the notions of roll center and roll axis for car suspensions. In *Proceedings of the Deuxième Congrès International Conception et Modélisation des Systèmes Mécaniques (CMSM 2007)*, Monastir, Tunisia, March 19–21 2007. CD-ROM Proceedings, paper No. 84.

[61] Claytex. Stability of explicit euler solvers. https://www.claytex.com/tech-blog/stability-of-explicit-euler-solvers/, n.d. Accessed: 2025-06-04.

[62] Mike Dempsey. Dymola for multi-engineering modelling and simulation. In *2006 IEEE Vehicle Power and Propulsion Conference*, pages 1–6, 2006.

[63] Mike Dempsey, Garron Fish, and Juan Gabriel Delgado Beltran. High fidelity multi-body vehicle dynamics models for driver-in-the-loop simulators. In *Proceedings of the 11th International Modelica Conference*, pages 273–280, Versailles, France, 2015. Linköping University Electronic Press.

[64] Mike Dempsey, Garron Fish, and Alessandro Picarelli. Using modelica models for driver-in-the-loop simulators. In *Proceedings of the 9th International Modelica Conference*, pages 571–578, Munich, Germany, 2012. Linköping University Electronic Press.

[65] ASAM. Asam opencrg toolbox, version 1.2.0. https://www.asam.net/index.php?eID=dumpFile&t=f&f=3950&token=21a7ae456ec0eb0f9ec3aee5bae3e8c9ebaea140, 2025. Accessed: 2025-11-27.

[66] Flavio Farroni. T.r.i.c.k.-tire/road interaction characterization & knowledge - a tool for the evaluation of tire and vehicle performances in outdoor test sessions. *Mechanical Systems and Signal Processing*, 72-73:808–831, 2016.

[67] Flavio Farroni, Raffaele Lamberti, Nicolò Mancinelli, and Francesco Timpone. Trip-id: A tool for a smart and interactive identification of magic formula tyre model parameters from experimental data acquired on track or test rig. *Mechanical Systems and Signal Processing*, 102:1–22, 2018.

[68] Oliver Lenord, Martin Otter, Christoff Burger, Michael Hussmann, Pierre Bihan, JÃ¶rg Niere, Andreas Pfeiffer, Robert Reicherdt, and Kai Werther. efmi: An open standard for physical models in embedded software. pages 57–71, 09 2021.

[69] T. Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. 09 2012.

[70] CATIA-Systems. FMIKit-Simulink: Import and export Functional Mock-up Units with Simulink. https://github.com/CATIA-Systems/FMIKit-Simulink, 2025. Archived repository (read-only).

[71] NTNU-IHB. Fmi4cpp: A c++ library for working with fmi. https://github.com/NTNU-IHB/FMI4cpp, 2023. Accessed: 2025-11-27.

[72] FranÃ§ois E. Cellier and Hilding Elmqvist. Object-oriented modeling of biochemical processes. In *Proceedings of the European Simulation Multiconference (ESM'95)*, Prague, Czech Republic, 1995.

[73] Gianluca Frison and Moritz Diehl. Hpipm: a high-performance quadratic programming framework for model predictive control, 2020.

[74] JoÃ£o Leal. CppADCodeGen: A c++ algorithmic differentiation code generator based on cppad. https://github.com/joaoleal/CppADCodeGen, 2017. Accessed: 2025-05-17.

[75] Alessandro Toschi, Francesco Prignoli, and Marko Bertogna. Modular decision-making and drivable areas for multi-agent autonomous racing. pages 12435–12441, 10 2025.

[76] Leonardo Serena, Mattia Bruschetta, Giovanni Righetti, Ricardo de Castro, and Basilio Lenzo. Real time camera-based sideslip angle estimation: design and experiments. *IFAC-PapersOnLine*, 58(28):750–755, 2024. The 4th Modeling, Estimation, and Control Conference - 2024.

[77] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 10 2023.

[78] Flavio Farroni, Michele Russo, Aleksandr Sakhnevych, and Francesco Timpone. Trt evo: Advances in real-time thermodynamic tire modeling for vehicle dynamics simulations. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 233(1):121–135, 2019.

[79] Luigi Teodosio, Giuseppe Alferi, Andrea Genovese, Flavio Farroni, Benedetto Mele, Francesco Timpone, and Aleksandr Sakhnevych. A numerical methodology for thermo-fluid dynamic modelling of tyre inner chamber: towards real time applications. *Meccanica*, 56(3):549–567, 2021.