Master of Science in Automotive Engineering

Master Thesis

Development and Calibration of a Driver Model for Lap-Time Prediction of a 14-DOF Vehicle GT-SUITE Model

University supervisors:
Prof. Federico MILLO
Prof. Luciano ROLANDO

Company supervisors:
Luca CAMBRIGLIA
André FERNANDES

Candidate:
Alessandro LONGHI

November 2025

Abstract

Within the framework of car racing, vehicle dynamics performance is of key importance for maximizing race results on track. The first indicator of vehicle performance is the lap time: the lower it is, the faster the car.

Throughout a vehicle development process, it is valuable to test any changes to ensure they provide the expected results. On-track test drives require continuous costly steps, including components production and installation, track availability, and need to refuel or recharge the vehicle. Furthermore, factors such as upgrade reliability, track and weather conditions, tire wear, and driver precision and feedback substantially affect the results, both in terms of reliability and repeatability.

Such limitations could be minimized – if not eliminated – by replacing on-track testing with virtual environment simulations. This approach ensures high flexibility in parameter and setup changes while avoiding the external disturbances typical of a real environment. In this regard, this thesis focuses on a virtual model consisting of two main subsystems: the vehicle and the driver. The former, provided at the beginning of the project, is a 14-degrees-of-freedom vehicle that serves as the baseline for the tests performed. The latter is the driver model, which is the main focus of this thesis.

The project outlines the workflow followed to develop the individual components of the driver and integrate them within the vehicle and track environment, with the objective of estimating a theoretical optimal lap time. The virtual framework adopted enables co-simulation, allowing the Python-based driver subsystem to exchange information in real time at each time step within the vehicle subsystem provided in the GT-Suite environment.

The model demonstrates positive outcomes through its ability to follow the desired trajectory with minimal lateral error. The pedal inputs are consistent with both the track layout and the vehicle's dynamic capabilities. Reliable results are obtained across different tracks and vehicle configurations, which confirms the robustness of the calibrated parameters under varying conditions.

A driver model of this kind can be a valuable tool in motorsport applications, providing high-fidelity simulation of a racing car's behavior on track. By replicating driving dynamics while exploiting the vehicle's full potential, it can assist in performance analysis, vehicle setup optimization, and theoretical lap time estimation.

Sommario

Nel contesto delle corse motoristiche, le prestazioni in termini di dinamica del veicolo giocano un ruolo chiave nel massimizzare i risultati finali in pista. Il primo indicatore di prestazioni del veicolo è il tempo sul giro: più basso è il valore, maggiore è la velocità.

Durante lo sviluppo del veicolo, è fondamentale testare ogni aggiornamento per verificarne l'efficacia. I test in pista, tuttavia, comportano passaggi dispendiosi come la produzione e l'installazione di componenti, la disponibilità del tracciato e necessità di rifornire o ricaricare il veicolo. Inoltre, fattori come l'affidabilità degli aggiornamenti, le condizioni atmosferiche e del fondo, il consumo degli pneumatici, nonché la precisione e le sensazioni del pilota, influenzano significativamente i risultati, sia in termini di affidabilità che di ripetibilità.

Tali limitazioni possono essere superate sostituendo le prove in pista con simulazioni in un ambiente virtuale. Un approccio di questo genere assicura estrema flessibilità nella variazione dei parametri e dell'assetto, con l'ulteriore vantaggio di evitare disturbi esterni che sono tipici di un ambiente fisico reale. A questo proposito, questa tesi si focalizza su un modello virtuale costituito da due principali sottosistemi: il veicolo e il pilota. Il primo, fornito all'inizio del progetto, è un veicolo a 14 gradi di libertà che costituisce la base per i test condotti. Il secondo è il modello del pilota, ovvero il cuore di questa tesi.

Il progetto descrive il percorso seguito per sviluppare i singoli componenti del pilota e la loro integrazione con il veicolo e il tracciato, con l'obiettivo di stimare il tempo sul giro ottimale. La struttura virtuale adottata consente una co-simulazione in tempo reale, in cui il pilota, sviluppato in Python, scambia informazioni a ogni intervallo di tempo con il sottosistema veicolo, modellato nel software GT-Suite.

Il modello mostra risultati positivi, evidenziati dalla capacità del veicolo di seguire la traiettoria ideale con un errore laterale contenuto. I segnali ricevuti dai pedali risultano coerenti con la configurazione del tracciato e con i limiti dinamici del veicolo. La validità dei risultati è confermata su diverse piste e configurazioni del veicolo, a conferma della robustezza dei parametri calibrati, indipendentemente dalle condizioni operative.

Un modello pilota di questo genere può rappresentare un valido strumento in applicazioni nel campo del motorsport, in quanto capace di produrre simulazioni fedeli del comportamento di un'auto da corsa in pista. Pertanto, replicare le dinamiche di guida sfruttando il massimo potenziale del veicolo, consente di dare supporto alle analisi delle prestazioni, all'ottimizzazione dell'assetto e alla stima del tempo sul giro teorico.

Contents

List of Tables

List of Figures

Introduction

In recent years, market demand for simulation tools focused on vehicle dynamics and lap time analysis has grown considerably, especially in motorsport applications. In such context, every minor improvement becomes extremely valuable, and every tool represents a strategic asset for enhancing performance and reliability.

The design and use of virtual environments in vehicle development and design validation offers the advantage of implementing updates before physical prototypes are manufactured, resulting in both cost and time savings.

Simulation models can vary in accuracy, ranging from steady-state or quasi-static simulations – where computational time is optimized at the expense of accuracy – to multibody simulations that integrate transient effects and vehicle kinematics, offering higher fidelity and more accurate results, particularly useful for aerodynamic or mechanical setup fine-tuning.

These models enable correlations between simulation results and real on-track data, allowing further refinement of the model and improving future predictive capabilities. Virtual simulations can also support driver performance by providing insights into optimal flying-lap execution, or by suggesting race strategies that help reduce fuel and energy consumption while mitigating tire wear. Additional information can be obtained regarding how to approach specific tracks, especially when new layouts are introduced and historical data is unavailable.

In a context where software evolves rapidly, such environments could in the future be enhanced through the integration of artificial intelligence and machine learning, improving predictive accuracy and automated optimization. Nonetheless, physical integration and correlation with real telemetry data will remain essential for validating virtual models.

Within the scope of this thesis, the development of a self-adaptive Driver Model, capable of adjusting to different vehicles and tracks, represents a powerful tool that facilitates

independence and versatility in high performance applications. The goal is therefore to allow end users to couple the Driver to their own vehicle model without any need of intermediate calculations or further development.

The Driver's objective is to minimize the vehicle lap time by exploiting the full potential of the vehicle's performance limits while following a given reference trajectory. However, it is essential for the Driver to develop awareness of both the track layout and the vehicle's dynamic limits, in order to generate the appropriate control signals for the vehicle's actuators. This capability is achieved through pre-processing steps conducted prior to the main lap time simulation.

Through these pre-processing steps, the Driver gains understanding of the environment and vehicle characteristics by performing track analysis for corners identification and testing the vehicle's adherence limits. For instance, track analysis may provide corner position and curvature radii, while vehicle analysis can calculate the corresponding cornering speeds and estimate the braking distances.

Since the driver model is meant to be a plug-and-play subsystem for any vehicle, it does not include additional control logic such as gear shifting. Instead, it focuses on pedal control for longitudinal dynamics and steering control for lateral dynamics.

The driver model is developed in the form of a Python script integrated within GT-Suite environment. This setup enables co-simulation, allowing both systems to exchange information in real time at each time step. GT-Suite handles the simulation of the vehicle and its related subsystem, while the Python script governs the Driver's behavior, including environmental awareness and control strategies.

The following chapters outline the methodology adopted to achieve the desired behavior from scratch. As a preliminary step, the track is pre-processed to align its reference system consistently across both Python and GT-Suite environments. Additionally, curvature radius data is extracted and stored for use in later stages, such as cornering speed estimation.

To support this, analyses of the vehicle's longitudinal and lateral limits are conducted in parallel, aiming to build a GGV diagram – a representation of iso-speed envelopes that define the acceleration limits of the vehicle under driving conditions.

The steering is implemented using a Stanley controller, enhanced with additional parameters such as lateral error dead-zone and steering-rate limits to reduce unpredictable or unstable behavior. These parameters are optimized for both performance and stability.

Accelerator and brake pedal controls are designed using two independent PI con-

trollers, each with its own sensitivity optimization. These controllers aim to maintain tire slip within optimal limits, thereby maximizing force generation for improved performance.

As a result, the pre-processed track data and vehicle limit values are integrated to generate an optimal theoretical speed profile and to design the pedal control logic that the Driver applies to maximize lap performance throughout the track.

1. Software Tools Overview

1.1 Python: Platform for Driver Model

Python is a high-level, interpreted, and multi-paradigm programming language – characteristics that offer great flexibility to users. Moreover, its syntax is clean and readable, making it sound close to spoken English, which simplifies its use and broadens its applicability across various domains, from data analysis to simulation and artificial intelligence.

One of Python's most significant advantages is its **open-source nature**. Developed under the *Python Software Foundation License*, Python is freely available for use, modification, and distribution – even in commercial contexts. This openness has fostered a global community that continuously contributes to Python's evolution, ensuring rapid development, bug fixes, and the creation of powerful third-party libraries.

In this project, Python was selected for its flexibility and ease of integration for co-simulation with GT-Suite environment. Valuable support is provided by specific *third-party libraries* that enhance computational performance and offer user-friendly commands.

NumPy, the foundational package for numerical computing in Python, introduces optimized operations on large arrays and matrices. It enables vectorized computations, significantly reducing execution time compared to native Python functions.

SciPy, built on top of NumPy, extends its capabilities by providing modules for optimization, integration, differential equations and interpolation.

Matplotlib, a versatile plotting library that allows for the creation of high-quality 2D and 3D visualizations of graphs through intuitive and straightforward code.

The use of Python and its libraries enables a versatile and streamlined workflow, seamlessly interfacing with GT-Suite in co-simulation, resulting in a powerful tool for modeling and analysis. [1]

1.2 GT-Suite: Simulation Platform and Base Model

The simulations conducted in this thesis were developed and executed using **GT-SUITE**, a leading multi-physics simulation platform developed by Gamma Technologies. Originally designed for internal combustion engine modeling, the software has evolved in a more extended environment capable of simulating systems across various domains such as *fluid flow*, *acoustics*, *thermal management*, *electrical systems*, *chemistry*, and *mechanical systems*.

GT-Suite supports *multi-resolution modeling*, allowing users to choose from **0D** to **3D** simulations, depending on the required calculation accuracy and the computational resources. This flexibility allows subsystems to be modeled with varying levels of detail, enabling to balance computational cost with simulation accuracy depending on the specific requirements of each analysis. Moreover, GT-Suite enables the co-existence of subsystems of different nature, allowing the integration of mechanical, hydraulic, and thermal subsystems for multiphysics simulation. This makes the software a very powerful tool for complex analysis, not only for automotive applications, but also naval, aerospace, and industry.

For mechanical system analysis, GT-Suite includes a *Mechanical 3d* module which enables **multi-body dynamics (MBD)** simulations. This module is essential for modeling components such as chassis, suspension mechanisms, tires. A new software internal suite – called GT-3DMBD – has recently been introduced to GT-Suite. This addition enhances the software's capabilities in multi-body simulation, and improves how multiple subsystems of the same model can interact and affect vehicle dynamics and stability.

While GT-Suite has long supported *longitudinal vehicle dynamics*, *lateral dynamics simulation* was introduced more recently in the 2024 release, although with some limitations, expanding the capabilities of the platform.

The simulations presented in this thesis were performed using GT-Suite's default solver configuration, already assigned to the vehicle model utilized – and described in the following. The core integrator employed is *DAE-GenAlpha*, a second-order accurate implicit solver for Index-1 Differential-Algebraic Equations (DAEs). Known in academic literature as *CH-Alpha*, this solver is well-suited for stiff systems and ensures numerical stability across a wide range of operating conditions. [2]

The solver is an implicit, one-step time integration scheme derived from the Generalized-

Alpha method [3] and discretizes second-order equations of motion – that include displacements, velocities and accelerations – using weighted approximations [4].

Being a one-step implicit method, the algorithm only relies on its previous time-step solution, allowing variable time-step integration. Step size is based on error control and convergence behavior, increasing when system dynamics vary slowly and reducing when rapid changes occur.

To ensure stability, the residual formulation is modified from the standard Generalized-alpha approach [5]. Dynamic equilibrium is enforced at each time-step, and convergence is achieved when the residual falls below predefined tolerances, after which the solver proceeds to the next iteration.

Before exploring the driver model development, it is essential to describe the specific vehicle model implemented within the GT-Suite environment. The model used is a *14 Degrees of Freedom (DOF)* vehicle [6], introduced in the 2025 version of GT-Suite. It serves as a platform on which all control strategies and driver interactions are tested in this thesis.

The vehicle model is constituted by the sprung mass of vehicle body and four unsprung masses that are the wheels. The sprung mass accounts for 6 DOF, including longitudinal, lateral, vertical, roll, pitch, and yaw motion. Each wheel is allowed to have 2 DOF, consisting of the vertical motion and wheel spin. A schematic representation of the model is shown in Figure1.1

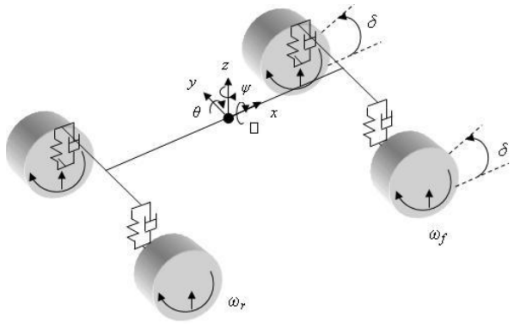

Figure 1.1: 14 Degrees of Freedom Vehicle Model [6].

The model is built, tested and validated through two driving maneuvers, consisting of steeps steer and double lane change, with the integration of Pacejka tire magic formula. Lumped mass is used to represent sprung and unsprung masses. The vehicle body is modeled as rigid, the outer and inner steering angle is assumed to be the same and tires are assumed to have contact with the ground the whole time.

Two distinct sets of equation can be identified, splitting the scheme in two models based on the effect they have on the vehicle's behavior: a *Ride Model* and a Handling Model.

The *Ride Model* shown in Figure 1.2 describes 7 degrees of freedom. The vehicle body is allowed to have 3 DOF consisting of vertical, roll, and pitch motion, while for the tires only the vertical motion is considered

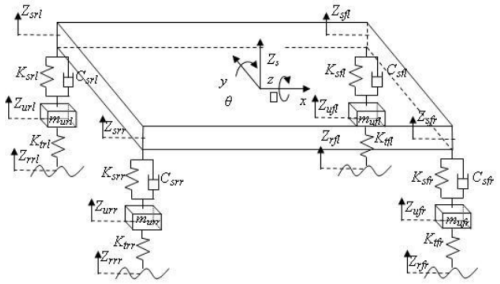

Figure 1.2: Vehicle Ride Model [6].

The *Handling Model* shown in Figure1.3 consists of 7 DOF. The vehicle body describes 3 DOF, including longitudinal, lateral and yaw motion, while the remaining 4 DOF correspond to the spin of each wheel – which is represented by their angular velocity.

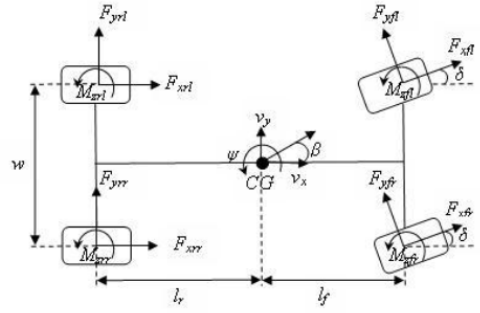

Figure 1.3: Vehicle Handling Model [6].

The described model is validated against real data for lateral acceleration, yaw rate, roll angle, and tire side-slip through a step steer maneuver and a double lane change maneuver.

For the purpose of this work, the vehicle model is provided in an optimized structure in GT-Suite environment for *fast-running simulations*, significantly outperforming conventional 3D models and even faster than real-time execution. This speed enables rapid iteration and parameter adjustments, and efficient analysis of handling and stability be-

havior, when ensuring that simulation outcomes align with expectations. As such, it is a powerful tool for development and validation of control algorithms related to vehicle stability, handling and dynamics.

To the describe structure of the vehicle model, other necessary subsystems are attached. The brake system is simplified, by including a brake template applied uniformly to all four corners of the vehicle. It receives the activation control signals from the driver model, and provides it to the wheels. The powertrain side includes two independent electric motors, each driving one of the half-shafts. This layout enables for activating and controlling torque vectoring technique – if needed or wanted.

The tire model is based on a simplified Pacejka tire model, and is described with a set of longitudinal and lateral force maps, enabling realistic and fast-running tire behavior within the simulation.

Ultimately, the vehicle model needs to interface with the Python Driver Model developed in this work, via a *Python template*. The communication between GT-Suite and Python includes several signals, flowing both in input and output:

Inputs from Python to GT-Suite

- *Steering input*: front wheels angle
- *Accelerator input*: torque request
- *Brake input*: torque percentage

Outputs from GT-Suite to Python

- *Vehicle speed*
- *Vehicle position and heading*
- *Tires longitudinal slip*
- *Tires side slip angle*

2. Track and Vehicle pre-processing

The main model's ability to work effectively relies on the driver's understanding of the surrounding environment, which includes both the track and the vehicle. To ensure a reliable and consistent setup, a series of pre-processing steps are required.

The first step focuses on the track file. It is analyzed to verify that its reference system aligns with the GT-Suite's default coordinate system. Additionally, key data such as *trajectory points discretization*, *angular orientation of track points*, *curvature radius* and *corner position* are processed. This information is made available to the driver during the lap time simulation.

Alongside track preparation, a pre-processing step is also performed on the vehicle model. This step is essential to determine the vehicle's longitudinal and lateral acceleration limits. These limits are evaluated in separate environments using specific maneuvers designed to define the vehicle's performance envelope.

The resulting dataset can be stored in the form of a GGV diagram – where *GGV* stands for *longitudinal acceleration (G)*, *lateral acceleration (G)*, *vehicle speed (V)*. This diagram consists of acceleration curves distributed across various speed bins and provides a comprehensive visualization of how the vehicle's acceleration limits change as a function of speed.

These curves often resemble an elliptical form, but this is a simplification. In reality the structure is influenced by complex interactions, mainly governed by tire characteristics – including non-linearities – and vehicle load transfer effects. In fact, as the vehicle accelerates, decelerates and steers, vertical load shifts between wheels, changing the available grip at each vehicle corner and thus changing the corresponding maximum acceleration limit. These effects could be further affected by aerodynamics and suspension geometry, making the GGV diagram a vehicle setup-specific powerful tool for understanding its performance.

The information of the GGV diagram is crucial for correlating the vehicle's performance

limits with the characteristics of current track, enabling the generation of a reliable *speed profile* that the driver must follow during the main lap time simulation.

2.1 Track Analysis

During track processing, it is assumed that the provided trajectory represents an already optimized driving line. This assumption is necessary due to the absence of detailed information regarding track lateral limits, such as curbs or off-track areas. Therefore, the driver is designed to minimize lateral deviation, considering the reference trajectory as ideal; the track is processed accordingly.

The track file is first analyzed to ensure that its reference coordinate system aligns with the default system used by GT-Suite, guaranteeing consistency in subsequent simulations. A finer discretization of the trajectory is then applied to enhance the resolution of the road profile, which contributes to improved ride smoothness. The *angular orientation* and *curvature radius* of the path are calculated to provide spatial awareness for the driver and to support dynamic analysis. Corner identification is then performed along the trajectory, focusing on the detection of critical low-radius segments. The resulting data – such as curvature values and related metrics – will later be associated with vehicle performance limits to support the estimation of safe cornering speeds.

2.1.1 Track Translation and Rotation

GT-Suite and Python track alignment is crucial to maintain consistency in the data exchanged between the two software, such as vehicle position and trajectory deviation. Specifically, the start/finish line of the track must coincide with the origin of the coordinate system, and the track direction should be parallel to the positive x-axis. Figure 2.1 shows the top view of the two main tracks tested for the purpose of this work, named Track A and Track B.

If the track data does not meet these criteria by default, a transformation is applied – consisting of a translation and a rotation – in the Python environment, to adjust the track points accordingly. This ensures that the environment is correctly oriented and ready for further processing.

For the track *translation*, the coordinates of the first point are subtracted from all the other track points, to make sure the reference trajectory starts from the origin.

For the *rotation*, the angle that the first segment of the trajectory forms with the x-axis is calculated – Equation (2.1) – and the resulting value defines the rotation apply to all

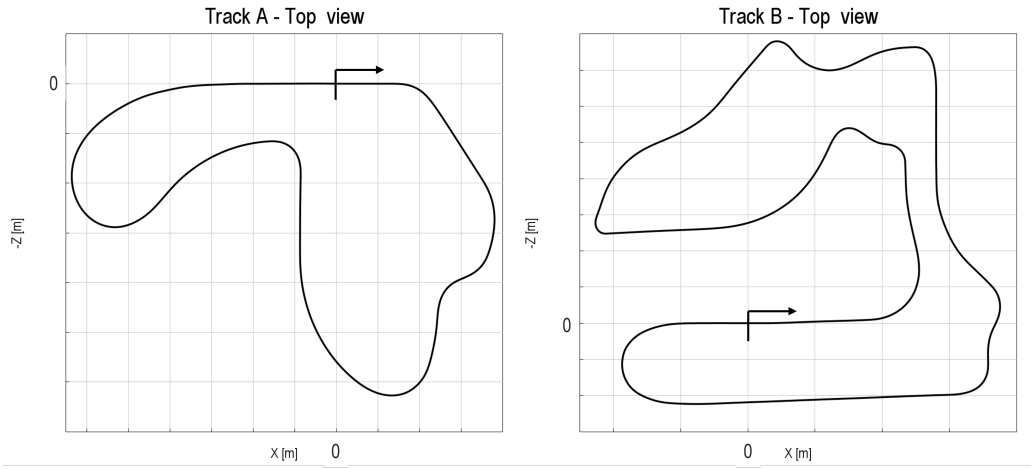

Figure 2.1: Track A and Track B reference trajectory, aligned - Top View.

the trajectory points – Equations (2.2) and (2.3).

$$\theta = \arctan\left(\frac{y_1}{x_1}\right) \tag{2.1}$$

$$x_{rot} = x\cos(\theta) + y\sin(\theta) \tag{2.2}$$

$$y_{rot} = -x\sin(\theta) + y\cos(\theta) \tag{2.3}$$

2.1.2 Cubic Interpolation and Finer Discretization

To mitigate the risk of simulating a trajectory with excessive roughness and abrupt heading changes, the path first undergoes cubic interpolation. This method proves effective with the available GPX files, which feature a relatively high spacing between recorded points, helping to smooth the trajectory. After interpolation, the resulting curve is discretized with a finer resolution – less than one meter – to better support realistic driver behavior modeling.

The retrieved finer trajectory becomes the reference line for the Driver model in the Python environment, overriding the initial GPX file that only works as a pointer for the track definition in GT-Suite at the beginning of each simulation.

2.1.3 Track Angular Orientation

To compute the heading angle along the trajectory, denoted as ψ_{track}, the track is first represented by two vectors containing the x and y coordinates of the discretized track points. To estimate the local orientation, the differences between consecutive points are calculated, and segment components are then used to compute the heading angle at each point, as shown in Equation (2.4).

$$\psi_{track} = \arctan\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right) \tag{2.4}$$

A visual representation is shown in Figure 2.2.

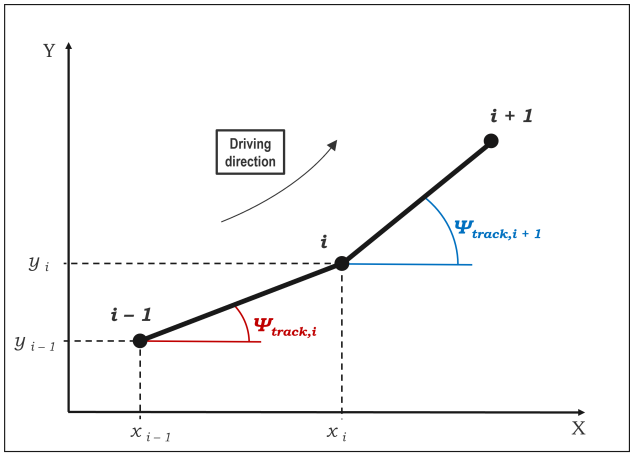

Figure 2.2: Representation of reference trajectory heading angle.

2.1.4 Track Curvature Radius

The calculation of the *Curvature Radius* plays a key role in detailed track analysis, enabling the identification of critical sections and supporting the final stages of pre-processing aimed at enhancing driver awareness.

Equation (2.5) presents the parametric formulation used to compute the *Track Curvature*, where the trajectory is defined by spacial coordinates x and y along the track discretized path:

$$\kappa = \frac{|\dot{x}\ddot{y} - \dot{y}\ddot{x}|}{(\dot{x}^2 + \dot{y}^2)^{3/2}} \tag{2.5}$$

In this context:

- $\dot{x}, \dot{y}$ are the first derivatives of the spatial coordinates with respect to the path index.

- $\ddot{x}, \ddot{y}$ are the corresponding second derivatives.

The numerator:

$$|\dot{x}\ddot{y} - \dot{y}\ddot{x}|$$

represents the rate of change of the direction along the path, thus how sharply the trajectory bends at each point. The denominator:

$$\left(\dot{x}^2 + \dot{y}^2\right)^{3/2}$$

normalizes the curve bending by the cube ot the local *path direction derivative*, ensuring the curvature is purely geometrical.

For simplicity, the reciprocal of the curvature – Equation (2.6) – is used in the following sections.

$$R = \frac{1}{\kappa} \tag{2.6}$$

This quantity is referred to as the *Curvature Radius*, and corresponds to the radius of the osculating circle at each point along the path. By osculating circle, it is intended the best-fitting circle that approximates the curve locally. In fact, it is tangent to the curve at the considered point, it shares the same curvature and its center lies along the normal vector to the curve.

A visual representation the osculating circle and the corresponding curvature radius on a generic trajectory is shown in Figure 2.3.

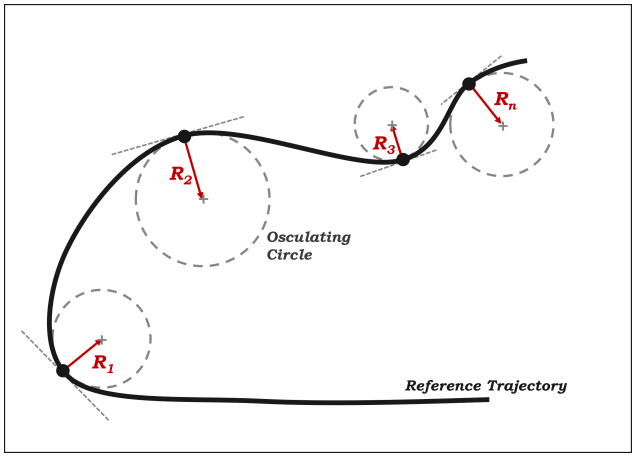

Figure 2.3: Curvature radius representation with tangential osculating circles.

Smaller values of curvature radius indicate tighter turns, which are relevant for corner

identification and driving prediction. Figures 2.4 and 2.5 respectively illustrate how track curvature κ and curvature radius R vary along Track A, highlighting critical sections of high directional change, which are particularly relevant for driver-awareness modeling and trajectory optimization.

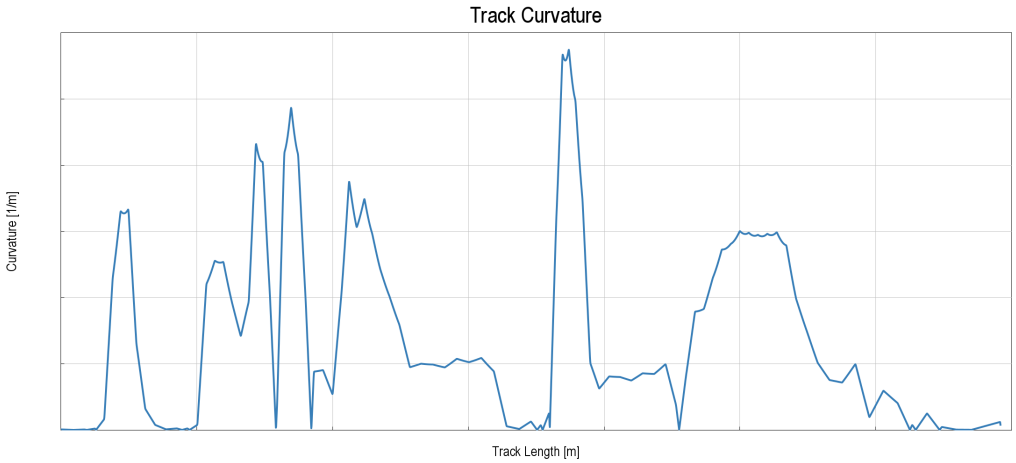

Figure 2.4: Track curvature κ along Track A reference trajectory.

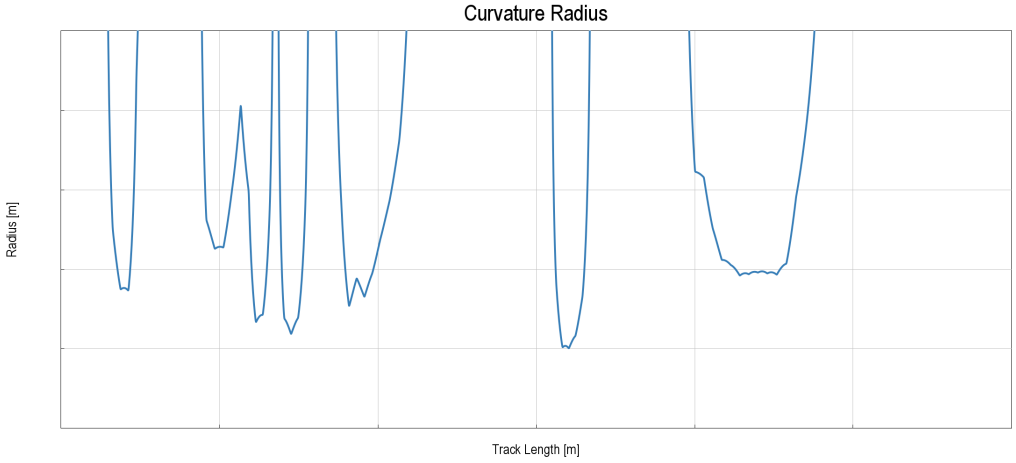

Figure 2.5: Curvature radius R along Track A reference trajectory.

2.1.5 Track Corner Identification

Once the *Curvature Radius* is calculated and stored in a vector, this data can be analyzed to distinguish curves from straights. To achieve this, a *threshold radius* value must be defined to represent the onset of a corner. The curvature radius data of the whole track is then scanned: whenever the radius drops below the threshold, the start of a corner is

marked; when the radius rises back above the threshold, the end of the corner is identified.

For each identified corner, a set of equally spaced points is selected along the segment. This discretization is intentionally coarser than that of the original trajectory points, to reduce computational load. The selected points include both the start and end of the corner and must satisfy specific criteria:

- a minimum number of points is required to ensure the corner is adequately represented;
- A maximum allowable distance between consecutive points is imposed to maintain sufficient resolution, even with the coarser sampling.

This approach balances accuracy in corner representation with efficiency in computation.

After the point selection is completed for each corner, the local minima of the curvature radius within each curve are identified and stored. These points are significant because they typically correspond to the *apexes* of the corners, locations where the vehicle reaches its *minimum speed* and follows the tightest radius. This interpretation is based on the assumption earlier introduced, according to which the provided trajectory is not the track centerline, but an already optimized driving line.

At these apex points, longitudinal acceleration is assumed as negligible, meaning that the vehicle is only subjected to lateral forces. As a result, these apexes serve as key reference points for calculating the *cornering reference speed curve*, once the vehicle dynamic limits have been determined.

A top view with the identified corners is shown in Figure 2.6, while the corresponding points on the Curvature Radius plot are displayed in Figure 2.7. In the latter plot, the highest values among these points tend to align around a similar level, almost forming a visible boundary. This pattern effectively illustrates the presence of the threshold used for corner detection, as the points themselves seem to outline it.

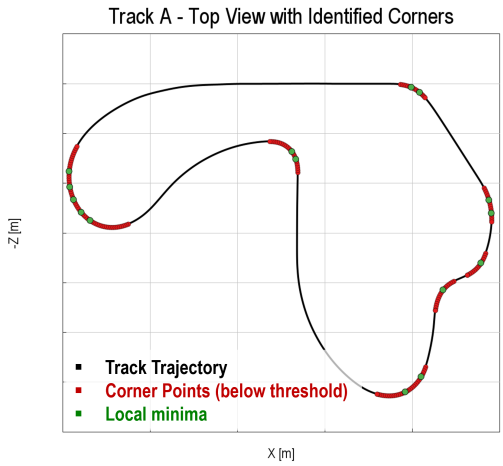

Figure 2.6: Track A Reference Trajectory with detected corners, Top View.

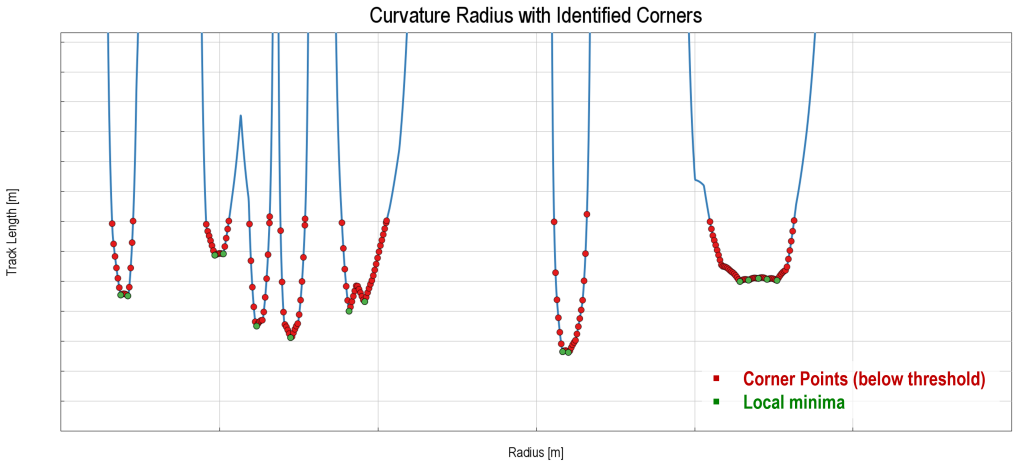

Figure 2.7: Curvature radius R along Track A trajectory with Identified corners points.

2.2 Vehicle Limits Analysis and GGV Diagram

Analyzing vehicle limits is essential for determining how far a driver can push the car without encountering instabilities or exceeding track boundaries by deviating significantly from the reference trajectory. In particular, it is important to determine the maximum longitudinal and lateral acceleration capabilities of the vehicle, which can be used to build combined acceleration boundaries at various driving speeds. These boundaries are crucial for estimating the vehicle's cornering behavior and for helping the driver adopt the optimal speed profile throughout a lap.

In particular, the focus of the GGV ellipses is placed on negative longitudinal accelerations, which are associated with trail-braking maneuvers. By trail-braking it is intended sections – mainly identified through corner entry – where both a brake and a steering signal are active. These regions are critical for the vehicle's driving dynamic, due to the combined action of longitudinal and lateral forces, and must be accurately processed to ensure the driver model will be able to navigate them stably.

Conversely, the combined effects of positive longitudinal acceleration – primarily governed by throttle input – will be analyzed separately in the following chapter, where the throttle control model is introduced and evaluated.

Additionally, for each speed, the corresponding minimum curvature radius that the vehicle can sustain is investigated through a dedicated maneuver. This analysis is essential for the next section, where the process will be reversed: estimating the minimum speed at each corner apex by associating the curvature radius (extrapolated in the previous track pre-processing) with the maximum speed the vehicle can maintain without losing grip.

2.2.1 Lateral Limits for Curvature Radius vs Speed

By setting up simulations where the vehicle maintains a constant speed while progressively increasing the steering angle, it is possible to investigate the minimum curvature radius achievable at each speed. The resulting trajectories are illustrated in Figure 2.8.

By measuring and correlating each curvature radius with its corresponding speed, a lookup table can be generated (Figure 2.9). This table enables the estimation of the maximum speed the vehicle can sustain at a given corner apex, based on the radius of curvature at that point.

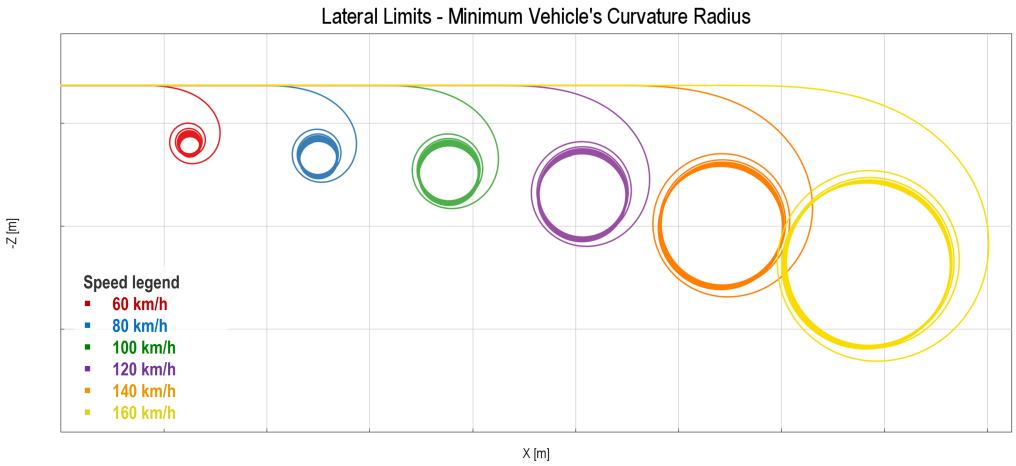

Figure 2.8: Simulation trajectories of Curvature Radius vs Speed pre-processing.

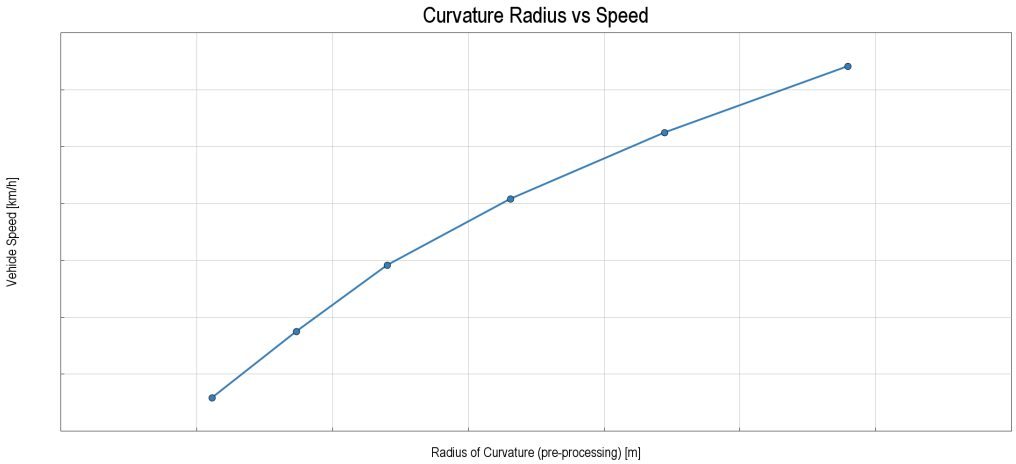

Figure 2.9: Curvature Radius vs Speed – Look up Table.

2.2.2 GGV Diagram: Longitudinal Limits

To define the longitudinal acceleration limits of the vehicle, an independent simulation environment is created. The vehicle begins at a high initial speed with zero steering angle and performs a braking maneuver until it comes to a complete stop. The resulting speed profile from the simulation is shown in Figure 2.10.

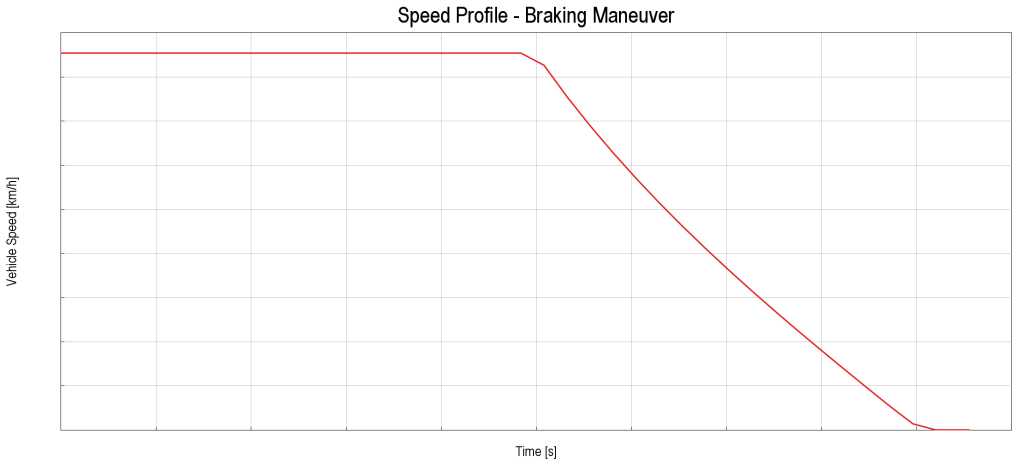

Figure 2.10: Longitudinal braking maneuver – vehicle speed profile.

The braking maneuver is managed to ensure that the longitudinal tire slip remains close to the optimal operating point where the tire generates its maximum longitudinal force. This allows the measurement of the maximum achievable longitudinal deceleration.

As the simulation progresses, each speed value recorded during the braking phase is associated with its corresponding longitudinal acceleration. These correlated data points can be used to build a lookup table for estimating braking distances at various speeds.

From the definition of acceleration:

$$a = \frac{dv}{dt} \quad \Rightarrow \quad dt = \frac{dv}{a} \tag{2.7}$$

From the definition of speed:

$$v = \frac{dx}{dt} \quad \Rightarrow \quad dx = v \cdot dt \tag{2.8}$$

Substituting dt from Equation (2.7) into Equation (2.8):

$$dx = \frac{v}{a} dv \tag{2.9}$$

Integrating this expression over the speed range provides the braking distance as a function of initial and final speeds. The resulting plot, shown in Figure 2.11, can be used to estimate the braking distance required to decelerate from a given speed to a target speed.

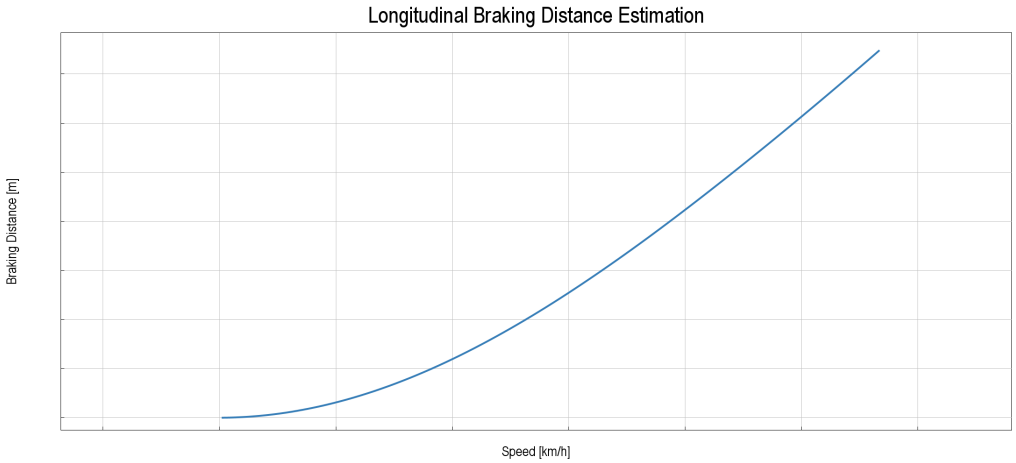

Figure 2.11: Longitudinal braking distance as a function of speed – Look up table.

2.2.3 GGV Diagram: Lateral Limits

To evaluate the lateral acceleration limits of the vehicle and complete the GGV diagram ellipses, a dedicated simulation environment is developed. This involves running multiple test cases with constant steering angles, each following a slow ramp-up speed profile.

By combining the results of lateral acceleration versus speed, it becomes possible to identify the maximum lateral acceleration the vehicle can sustain at various speeds. Most importantly, this analysis is independent of the specific steering angle applied in each case, as the simulation is designed to reveal the fundamental handling limits of the vehicle.

However, certain limitations may arise in terms of maximum achievable speeds and steering angles for reliable results. These constraints depend on vehicle-specific characteristics such as aerodynamic downforce, tire model, weight distribution, and suspension parameters.

For instance, tires may become unstable and unpredictable at higher speeds and higher steering angles, as shown in Figure 2.12. The plot represents lateral accelerations as function of vehicle speed, for different steering angles applied. It is evident how the results become unreliable at higher speeds, where the boundaries of the tire model are

reached and the lateral acceleration suddenly drops.

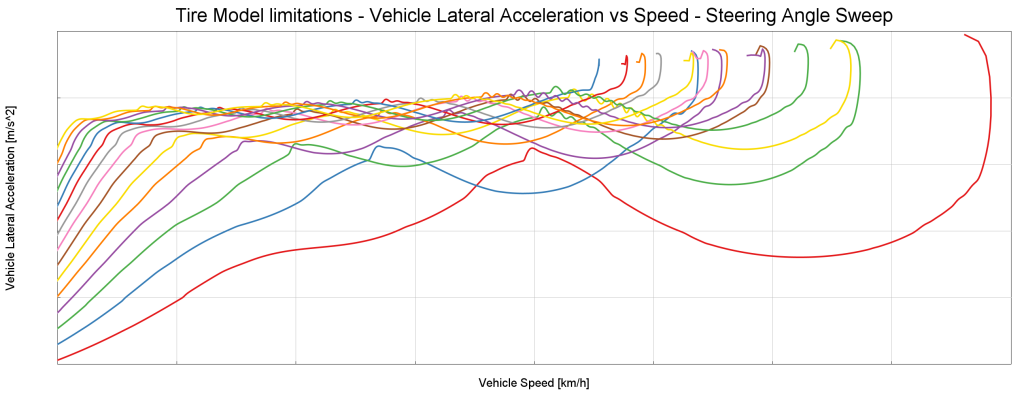

Figure 2.12: Pre-processing vehicle instabilities at high speeds and large steering angles.

2.2.4 Final GGV Diagram

By combining the longitudinal and lateral acceleration limits calculated for each speed, it is possible to construct an elliptical envelope that represents the vehicle's braking dynamic capabilities. These envelopes are used to estimate the corner capability, allowing the prediction of the vehicle's potential speed at specific points within a corner. Once the corresponding lateral acceleration is known, the envelope provides the remaining longitudinal deceleration margin that the driver can exploit to slow the vehicle down without losing control or deviating significantly from the reference trajectory.

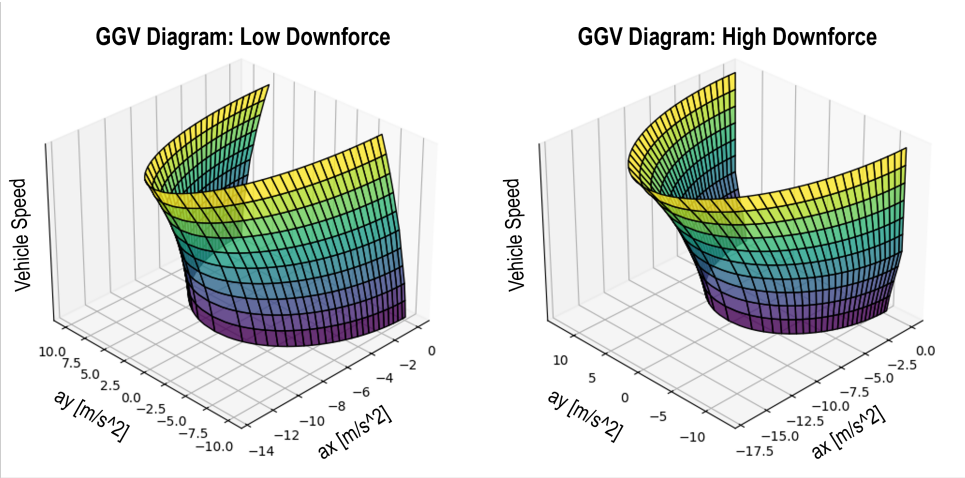

Figure 2.13: GGV Diagrams with different vehicle downforce.

Figure 2.13 shows two GGV plots generated with different downforce configurations.

The variation in shape reflects how setup changes interact with the vehicle subsystems, revealing limitations imposed by factors such as suspension geometry, tire characteristics, and ride dynamics. These differences highlight the sensitivity of the vehicle's performance envelope to aerodynamic and mechanical parameters.

2.3 Prediction of Speed Profile

The reference speed profile is constructed as a signal mapped to track position, which serves as input for the driver model during the main simulation. This profile guides the driver toward lap-time optimization by providing a physically feasible speed target.

This process relies on two key datasets obtained from previous sections: the *track characteristics* from the track pre-processing phase, and the *vehicle limits* stored in the form of GGV diagram.

To ensure robustness, several assumptions are made, and safety factors are introduced to prevent the predicted profile from exceeding the physical limits defined earlier through the pre-processing.

From the discretized track dataset, each corner is analyzed individually. Figure 2.14 illustrates a generic curve, with nomenclature used throughout the prediction process.

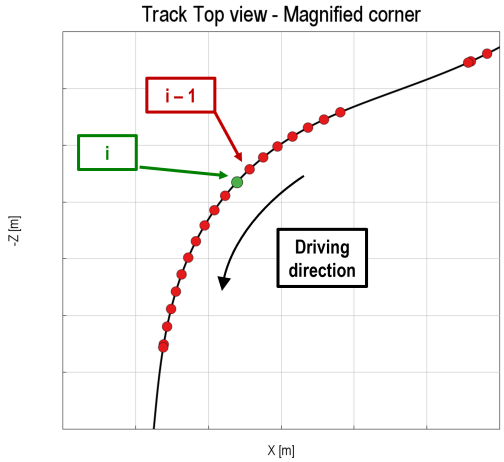

Figure 2.14: Generic track corner – Identified points and nomenclature.

The starting point for each corner is the local minimum of curvature radius – i.e. the apex – denoted as point i. Since this represents the slowest point in the corner, it is assumed that the vehicle speed is slightly higher immediately before and after. At the apex, the vehicle is expected to operate at maximum lateral acceleration, thus with

negligible longitudinal force.

Therefore, the curvature radius at the apex is mapped to the corresponding vehicle speed using the lookup table previously described (Figure 2.9):

$$R_i \quad \Rightarrow \quad v_i \tag{2.10}$$

From point i, the analysis proceeds backward to point $i-1$, scanning the track in reverse driving direction. Initially, the vehicle speed at $i-1$ is assumed equal to v_i, and the provisional lateral acceleration is computed:

$$a_{y,i-1} = \frac{v_i^2}{R_{i-1}} \tag{2.11}$$

Using the GGV diagram, the corresponding longitudinal acceleration limit is interpolated:

$$a_{x,i-1} = interp(a_{y,i-1}, v_i) \tag{2.12}$$

This value is used to estimate the speed at point $i-1$ starting from the speed at point i and the longitudinal deceleration extracted in (2.12), adding a safety factor (SF) to ensure conservative predictions:

$$v_{i-1} = \sqrt{v_i^2 + 2 \cdot a_{x,i-1} \cdot SF \cdot ds} \tag{2.13}$$

Next, the estimated lateral acceleration at $i-1$ is verified against the GGV limit, using the new predicted speed, making sure that the value falls inside the envelope limits:

$$a_{y,estim} = \frac{v_{i-1}^2}{R_{i-1}} < a_{y,lim} = interp(a_{x,i-1}, v_{i-1}) \tag{2.14}$$

If valid, the corresponding longitudinal acceleration is retrieved:

$$a_{x,estim} = interp(a_{y,i-1}, v_{i-1}) \tag{2.15}$$

Finally, the actual deceleration between $i-1$ and i is checked against the GGV limit:

$$\frac{v_{i-1}^2 - v_i^2}{2 \cdot ds} < a_{x,estim} \tag{2.16}$$

Once validated, the process continues iteratively from point $i-1$ to $i-2$, and so on, completing the prediction for each corner and eventually for the entire track. [7]

A visual representation of the final reference speed points is shown in Figure 2.15. It clearly illustrates how the descending speed profile at each corner provides the necessary reference for trail-braking maneuver, which the driver model must manage during the lap time simulation.

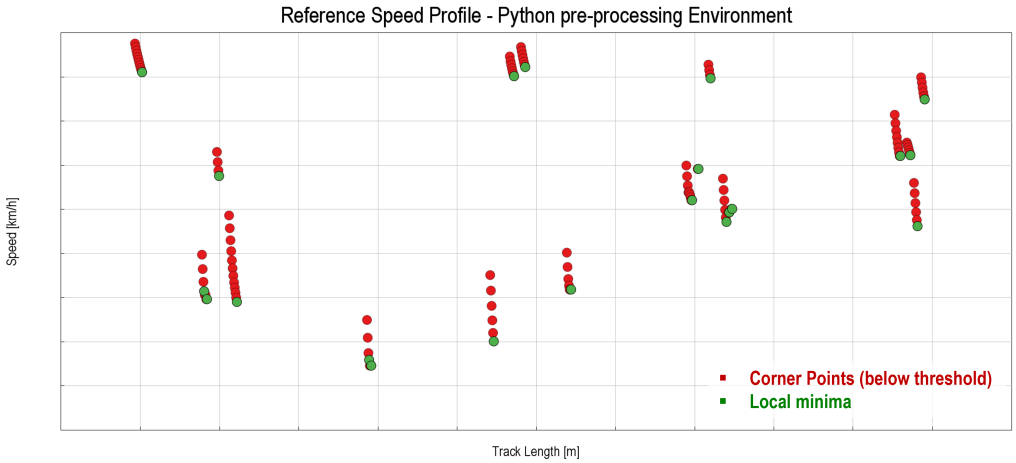

Figure 2.15: Pre-processed target speed predictions through track corners.

3. Driver modeling

A significant component of the lap time simulator is dedicated to the development of submodels that generate control signals for the steering and pedal actuators. These submodels form the core logic that enables the vehicle to be driven around the track in a realistic and performing way.

This chapter focuses on the structure of these subsystems, the mathematical formulations they rely on, the parameters involved, and the real-time signals measured from the vehicle during simulation.

3.1 Steering Design

The steering subsystem is based on the *Stanley* controller formulation [8], which has been adapted and tuned to properly suit the needs of the lap time simulation. Following an initial parameter setup, an optimization process is performed around the track to extract robust parameter values which may be implemented across different track layouts and vehicle configurations.

3.1.1 Stanley Model and Steering Parameters

The Stanley steering model implemented in the driver simulation is expressed as:

$$\delta = (\psi_{veh} - \psi_{ahead}) + atan\left(\frac{k_{stan} \cdot e}{V_{long} + k_{soft}}\right) \tag{3.1}$$

Where:

δ: Steering command signal.

ψ_{veh}: Vehicle heading angle.

ψ_{ahead}: Track heading angle at a *lookahead* point ahead of the vehicle. This anticipates the desired trajectory and compensates for system inertia.

k_{stan}**:** Stanley gain constant, controlling the aggressiveness of lateral correction actuation.

$e_{deadzone}$**:** Lateral deviation from the reference trajectory, measured in correspondence of the middle position of the front axle. Values below a defined *deadzone* threshold are set to zero to avoid unnecessary corrections.

V_{long}**:** Vehicle longitudinal speed.

k_{soft}**:** Softening constant to prevent division errors at low speeds.

The Stanley model consists of two main components:

Heading error correction, that aligns the vehicle's orientation with the desired path:

$$(\psi_{veh} - \psi_{ahead})$$

Lateral error correction, that adjusts the steering based on lateral deviation and vehicle speed, with higher speeds resulting in smoother corrections:

$$atan\left(\frac{k_{stan} \cdot e}{V_{long} + k_{soft}}\right)$$

To enhance realism and prevent abrupt steering inputs that could make the vehicle unstable, two additional parameters are introduced between the Stanley controller output and the steering actuator: the *rising steering rate* and *falling steering rate*. These parameters limit how quickly the steering angle can increase or decrease, ensuring smooth transitions. By limiting the rate of change, the system avoids unrealistic or abrupt steering commands that could lead to overcorrection, loss of control, or vehicle spin.

3.1.2 Optimization of Steering Parameters

The initial steering model is configured using parameters retrieved from literature or reference models, when available [9]. This setup provides a preliminary baseline and enables early simulation runs to evaluate basic system behavior.

Based on the results from these initial runs, a rough Design of Experiments (DOE) is prepared. This involves sweeping broad ranges of key parameters and analyzing the resulting simulations to identify configurations that provide stable and reliable behavior. The most promising parameter sets are then used to define narrower ranges for further optimization.

With these refined ranges, a more robust optimization process is executed using a *genetic algorithm*. This approach allows for the simulation of hundreds of model configurations, thanks to the definition *optimization objectives* that the algorithm aims to minimize.

For this optimization, two primary objectives are defined:

Cumulative lateral error: Minimizing this value over a lap time ensures that the vehicle remains as close as possible to the reference trajectory.

Cumulative steering signal derivative: Minimizing this value reduces noisy or unrealistic steering oscillation, which could otherwise lead to instability or loss of control.

Although it may seem counterintuitive, lap-time value is deliberately excluded from the optimization objectives. Including it could lead to unrealistic or aggressive steering behavior, potentially leading to corner cutting, vehicle instability, or loss of control – compromising the reliability and accuracy of the driver model.

It's important to emphasize that corner cutting is explicitly undesirable in this context. The trajectory provided to the driver does not represent the track's centerline or a flexible line, but rather an already optimized path intended to be followed precisely. Therefore, any deviation from this trajectory even if resulting in a faster lap time, would not be compliant with the expected control behavior, undermining the integrity of the simulation.

3.2 Pedals Design

The initial development of the pedal subsystems is carried out in the GT-Suite environment, even though the final goal is to implement the driver model in Python. GT-Suite offers a faster and more flexible platform for building, optimizing and adjusting pedal subsystems structure during early development stages.

Once a stable and reliable subsystem is achieved in GT-Suite, it is translated into Python. This passage includes fine-tuning to account for differences in signals handling between the two environments.

Both accelerator and brake pedal are based on independent PI (Proportional-Integral) controllers. These controllers are activated and deactivated alternatively during lap time simulation runs, based on pedal logic that will be detailed in the following chapter. The objective is to maximize vehicle performance by controlling longitudinal tire slip, keeping

it as close as possible to the optimal value while avoiding instability. To achieve such stability, the system monitors the divergence between front and rear tire side-slip angles, using it as a limiting factor for pedal actuation in combined longitudinal-lateral conditions. Additionally, the difference between the vehicle's measured speed and the reference speed at the current track position contributes to the control strategy, ensuring that the driver model remains within the vehicle's dynamic limits.

3.2.1 Brake Pedal

The brake pedal controller is based on a PI (Proportional-Integral) control strategy, which is tuned through an optimization process aimed at maintaining the longitudinal slip of the most critical tires near their optimal value.

Figure 3.1 shows the normalized longitudinal slip values of the front and rear axles during a hard braking maneuver (normalized with respect to the optimal slip value). As expected, due to the load transfer toward the front axle during braking, the rear axle becomes lighter and thus more critical. Consequently, the controller targets the optimal slip value on the rear tires, which are the more prone to locking. The front axle, while not fully exploiting all the available grip, is not adjusted further – e.g. by varying the front/rear brake balance – as the goal of this work is not to optimize the vehicle setup itself, but rather to develop a driver model capable of performing optimally under given conditions.

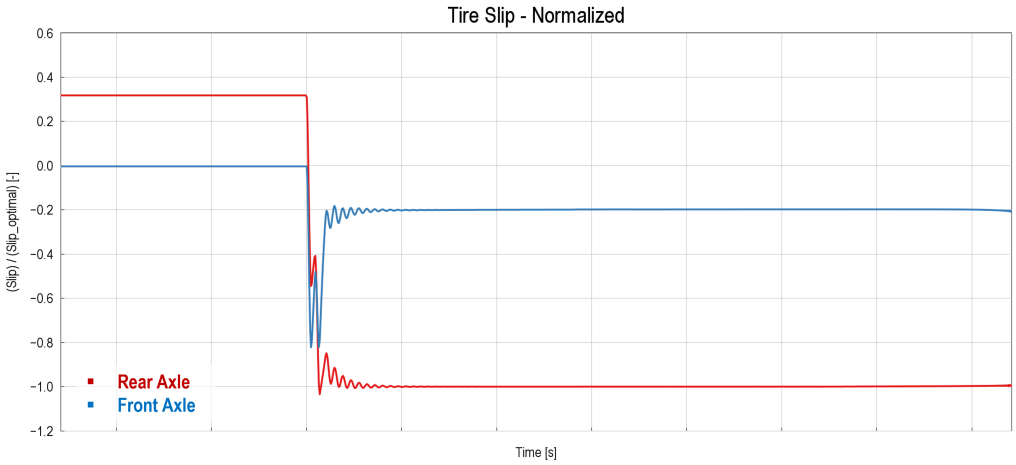

Figure 3.1: Normalized longitudinal slip of front and rear axles during braking.

Once the PI is tuned and its parameters are integrated in the model, it operates in two main configurations:

Longitudinal braking: The controller monitors the longitudinal slip of all four tires and adapts to the most critical one to prevent lock-ups, thereby maximizing the available longitudinal force under current conditions.

Trail braking: In this mode, the controller simultaneously monitors three signals:

- The longitudinal slip of all tires – to prevent lock-up.
- The side slip angles of the front and rear tires – to avoid oversteer or understeer.
- The reference speed during corner entry and mid-corner – as predicted during pre-processing.

It then adapts to the most critical among these inputs and generates the appropriate braking signal for the actuators.

Figure 3.2 shows the final structure of the brake pedal. The scheme gives evidence of how the first "if" check verifies whether the driver needs to initiate trail-braking or longitudinal braking, based on the current position on track. In case the vehicle is navigating through a corner, the signal feeding the PI controller is also considering tire side-slip angles and target speed, to guarantee stability and accuracy.

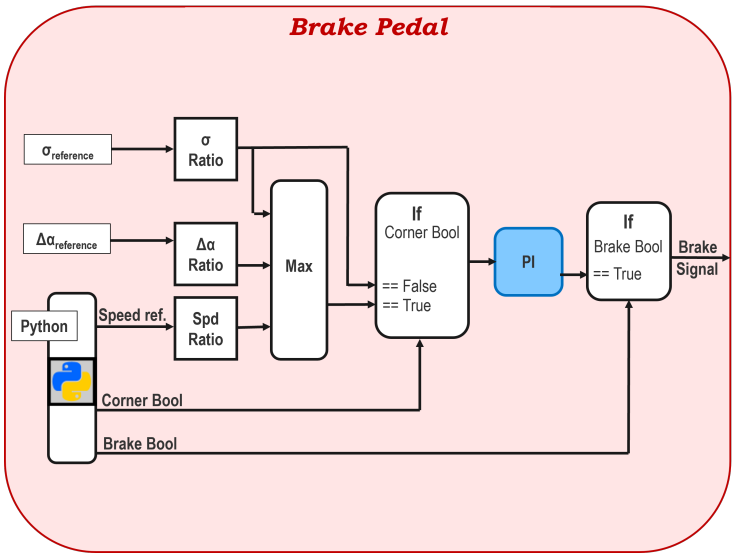

Figure 3.2: Brake Pedal structure.

3.2.2 Accelerator Pedal

The accelerator pedal shares similarities in its structure with the brake pedal system. It is also based on a PI controller, which, when active, monitors three key inputs signals [9] – similar to the brake pedal during trail-braking:

- The longitudinal slip of traction tires only – to prevent wheel spin during acceleration.
- The side-slip angles of the front and rear tires – to avoid oversteer or understeer.
- The reference speed during mid-corner and corner exit – to ensure the vehicle operates near its dynamic limits as predicted in the pre-processing phase.

To tune the PI controller, a dedicated driving maneuver is designed. The reference profiles are shown in Figure 3.3.

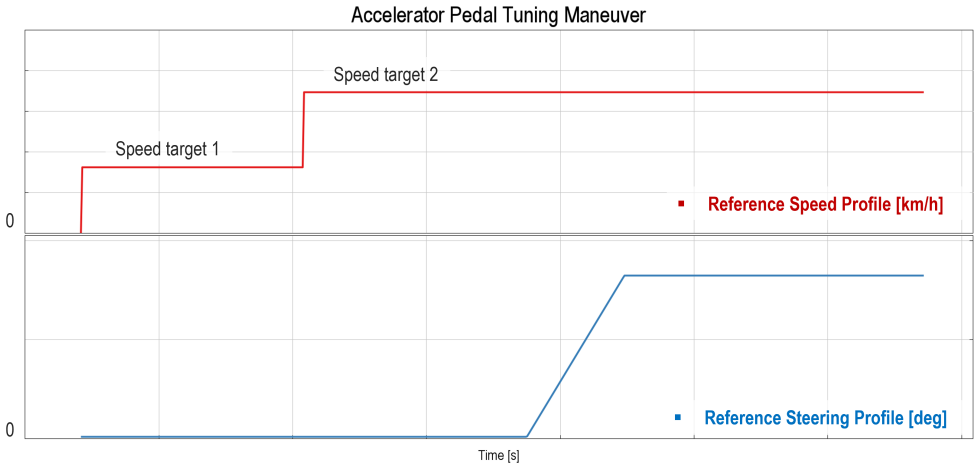

Figure 3.3: Maneuver profiles for accelerator pedal tuning.

In this maneuver, the vehicle starts from rest with zero steering angle and accelerates toward a first target speed (*Speed target 1*), which it maintains to stabilize. After a few seconds, a second, higher target speed (*Speed target 2*) is introduced and held for the remainder of the simulation. Once the second speed target is reached, the vehicle maintains constant speed while the steering angle ramps up from zero to a predefined value.

This maneuver stresses all critical input signal of the controller – tire longitudinal slip, tire side-slip angle, reference speed – at each transition, making it ideal for tuning.

During optimization, it becomes evident that controller performance depends on both the difference between actual and target speed, and vehicle's acceleration. As a result,

the accelerator PI parameters must be dynamic, adapting based on speed error and acceleration. By identifying the best-performing PI values for each speed target phase, a parameter sweep between these extremes can be introduced, resulting in a controller that adapts effectively across conditions.

Figure 3.4 illustrates this concept. The dashed signals represent two separate simulations with fixed PI parameters – performing well for one speed target but poorly for the other, either being too slow to reach or overshooting the target. The solid red line shows the result of the variable-parameter approach, demonstrating consistent and reliable performance across both conditions.

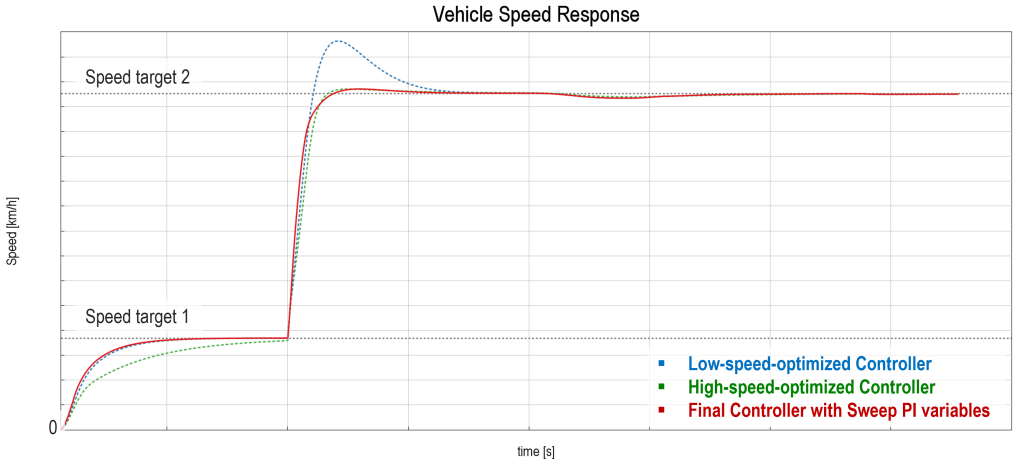

Figure 3.4: Vehicle speed responses to accelerator pedal tuning maneuver with different PI parameters.

Figure 3.5 shows the final structure of the accelerator pedal. The scheme appears similar to that implemented for the brake subsystem. However, in this case the PI controller always receives the most critical signal between longitudinal tire slip, tire side-slip angle and target speed, ensuring stability and accuracy both while navigating through corner and for high speed traction conditions.

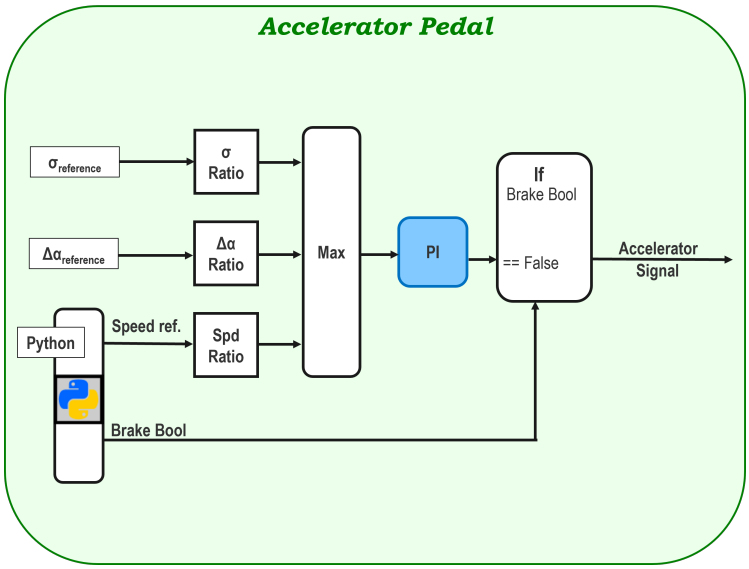

Figure 3.5: Accelerator Pedal structure.

4. Co-Simulation and Driver Adaptation

To enable seamless interaction between GT-Suite and Python subsystems during simulations, a robust signal exchange setup must be constructed. This ensures that both environments communicate effectively throughout runtime, allowing the driver model to remain aware of its surroundings – specifically, the vehicle's dynamic state and its position on the track at each time-step.

With this setup ensured, the logic governing pedal control can be developed and integrated into the driver model. The following sections detail the precautions taken to ensure reliable co-simulation, and describe how pedal control strategies are executed.

Software Signals Interface

GT-Suite provides a co-simulation template that enables seamless communication with external Python scrips. This interface allows bidirectional signal exchange between GT-Suite and Python at every simulation step. As a result, the Python script containing the driver model logic is executed simultaneously with the GT-Suite simulation, ensuring that both systems operate with consistent time discretization.

This setup is necessary for the correct driver model operations. At each time step, GT-Suite sends vehicle state information – such as speed, position, yaw angle, tire longitudinal slip and side-slip angle, acceleration – to Python. The driver model processes these inputs and returns control signals for steering, throttle and braking, which are then applied to the actuators of the vehicle model in GT-Suite.

Driver Position Awareness

The vehicle's relative position with respect to the track trajectory is measured at the midpoint of the front axle. This point is used to calculate the lateral error, consistent with standard applications of the Stanley controller.

An additional mathematical adjustment is introduced for handling the heading angle

error. As discussed in the steering section of the driver modeling chapter, the raw heading error is computed as:

$$(\psi_{veh} - \psi_{ahead})$$

where both angles are measured in the range $[0° \quad 360°)$, counterclockwise from the positive x-axis. This representation can lead to meaningless resulting values near the wrap-around point. For example, if the vehicle heading is just above $0°$ and the track heading is just below $360°$, the subtraction produces a large negative value, which is incorrect in context.

To resolve this, the heading error is converted in the range $[-\pi \quad \pi)$ using the following expression:

$$\textit{heading error} = [(\psi_{veh} - \psi_{ahead} + \pi) \bmod 2\pi] - \pi \tag{4.1}$$

This operation ensures:

- The first subtraction gives the raw heading angle.
- Adding π and applying modulo 2π ensures the result is wrapped within the range $[0 \quad 2\pi)$.
- Subtracting π shifts the result to the range $[-\pi \quad \pi)$
- As a result, positive values represent counterclockwise deviations, and negative values represent clockwise deviations, relative to the track heading.

4.1 Pedal Logics and Live Track Interaction

To determine pedal activation, the driver model references the pre-processed track data. Whether the accelerator or the brake pedal is engaged, the corresponding torque demand or brake pressure is computed based on the limitations and logic described in the pedals modeling section.

Longitudinal Braking

When the vehicle is not cornering, the default behavior activates the accelerator pedal. Simultaneously, the driver model monitors the upcoming corner entry point, comparing the current vehicle speed with the target entry speed. These values are checked against

the braking distance lookup table (Figure 2.11) to estimate the required deceleration distance. If the estimated braking distance exceeds the actual distance to the corner entry, the brake signal overrides the accelerator. In this phase, the brake subsystem operates in pure longitudinal mode, aiming to maximize deceleration within the available limits.

A safety margin is added to the estimated braking distance to account for uncertainties and ensure the vehicle does not enter the corner with excessive speed.

Trail-braking and Cornering

At the corner entry point, the accelerator remains the default active pedal form a formal perspective. However, the presence of a corner speed profile introduces an additional control layer on the brake pedal structure. If the vehicle speed exceeds the target cornering speed, the brake signal overrides the accelerator, and brake subsystem structure transitions from pure longitudinal – used during initial braking phase – to a more complex trail-braking mode. The brake subsystem now considers the tire side-slip angles and the corner speed profile, similar to the logic used by the accelerator subsystem.

As the vehicle decelerates through the corner, brake pressure gradually fades, and once the vehicle speed drops below the reference value, the accelerator pedal is reactivated. Throughout the corner, brake and accelerator signals may alternate to closely follow the target speed profile, ensuring the vehicle remains within its dynamic limits.

Corner Exit

As the curve radius increases again beyond the corner threshold, the reference speed ramps up, allowing the vehicle to accelerate on the straight. The driver model switches the brake subsystem back to its default longitudinal configuration and shifts focus to the next corner entry, initiating a new braking distance estimation cycle while the vehicle accelerates.

5. Results

Several configurations have been tested throughout the development process to obtain a comprehensive overview of the driver's capabilities under varying track and vehicle conditions.

Once the pedals subsystems developed in the GT-Suite environment were confirmed to be robust and reliable, they were translated into a Python script. The results from both implementations were compared to ensure no loss in performance occurred during the translation.

After validating the Python translation and confirming the subsystem's reliability, additional configurations were prepared to assess the versatility of the Driver model in different conditions.

These configurations were cross-fitted as follows:

- *Track Layout:* Two different tracks, "Track A" and "Track B", were tested.
- *Vehicle Aerodynamics:* Two downforce configurations were evaluated: "Low downforce" and "High downforce".
- *Vehicle Mass:* Two mass configurations were tested: "Mass 1" (default) and "Mass 2" (15% increased).

5.1 Python Translation Validation

The final Driver model, translated from GT-Suite into Python, is validated by comparing vehicle behavior under identical conditions. The GT-Suite implementation serves as an initial reference model, used to define the core structure and perform basic tuning. The Python version is constructed upon this foundation, incorporating refined logic and parameter fine adjustments, providing the basis for all the subsequent simulations.

Validation is performed by running both models with the same vehicle setup and track configuration. This ensures that the structure and parameters derived from GT-Suite model are correctly implemented in the Python environment.

Table 5.1 presents the lap times obtained for both models.

GT-Suite Model	***Python Model***
1:23.668	1:23.087

Table 5.1: Lap-time simulation results: GT-Suite vs Python driver models.

Figure 5.1 shows the comparison though a bar plot, highlighting a minimal difference between the two models. The Python-based model achieves a slightly faster lap time, primarily due to the finer tuning, confirming the validity of the translation.

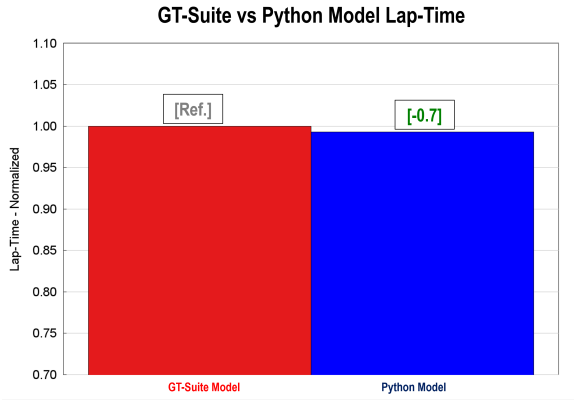

Figure 5.1: Bar plot with lap times and percentage variation.

Furthermore, telemetry data for each simulations will be presented, following the same format used for the validation runs shown in Figures 5.2 and 5.3.

The *left side* of each plot displays the track top view, including the reference trajectory and the actual driven path. Colored markers indicate longitudinal braking and trail-braking sections, separated by corner entry points, as expected from the control logic

correlated to the track layout.

The *right side* reports the four key telemetry signals used to assess model quality: reference and actual speed, lateral error, pedals usage, and vehicle side-slip angle.

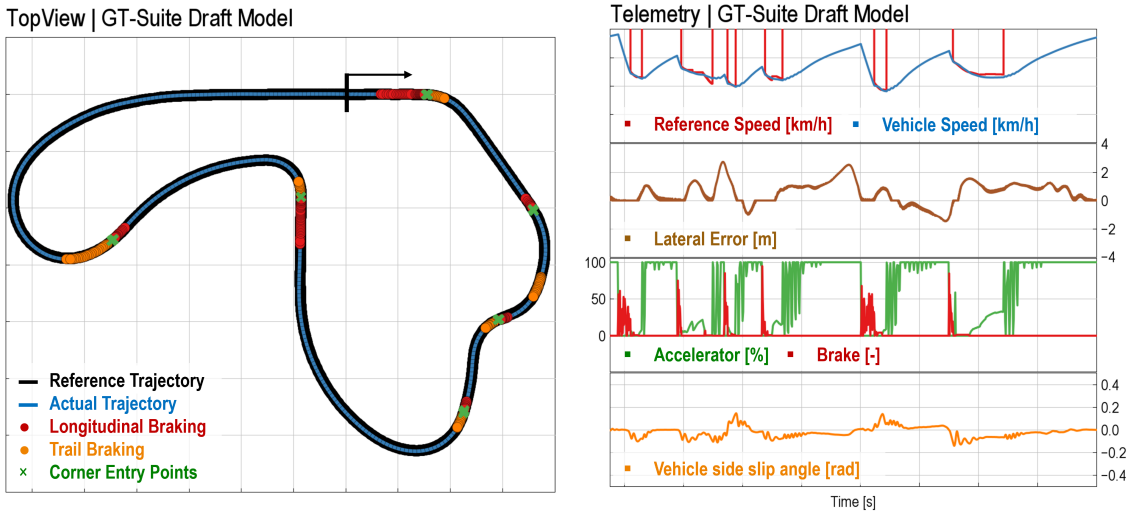

Figure 5.2: Simulation results for the GT-Suite Model.

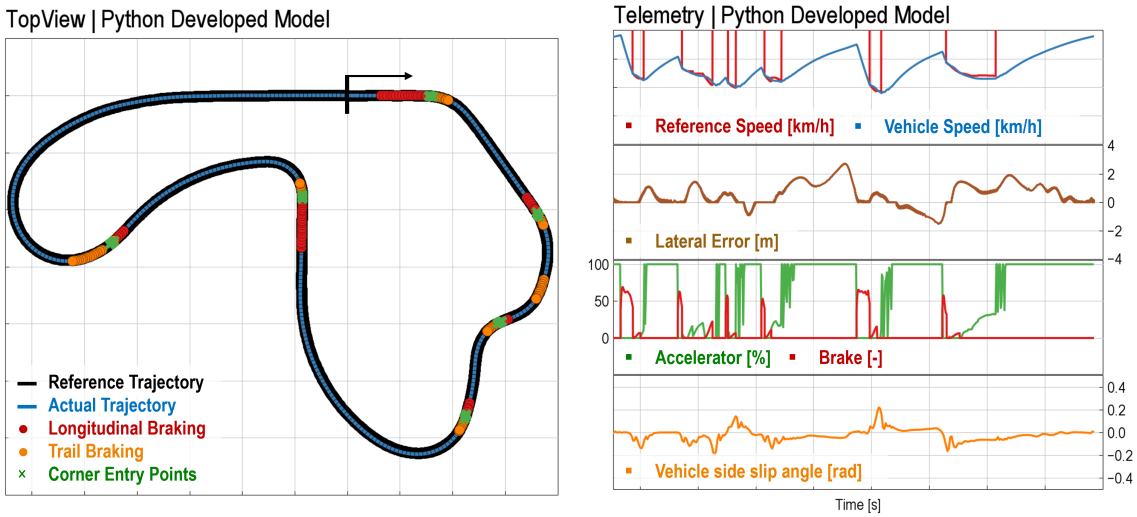

Figure 5.3: Simulation results for the Python Model.

For the current comparison, both models exhibit excellent trajectory tracking. However, the Python model is demonstrating smoother pedal control, producing cleaner actuator commands, especially for braking and acceleration during corner exits. This contributes at reducing abrupt transition and improves overall stability.

Additionally, the Python model shows a noticeable reduction in vehicle side-slip angle

oscillations, with only a few isolated spikes of higher magnitude, which are well controlled and do not propagate instability. The observed decrease in lateral error confirms the robustness of the Python implementation.

5.2 Final Lap Time Simulations

	Track A		Track B	
	Low Downforce	*High Downforce*	*Low Downforce*	*High Downforce*
Mass 1	1:23.087 Ref.	1:22.329 -0.758	2:43.440 Ref.	2:42.000 -1.440
Mass 2 (+15%)	1:24.068 +0.981	1:23.373 +0.286	2:46.320 +2.880	2:44.880 +1.440

Table 5.2: Lap-time simulation results across different configurations simulations.

Table 5.2 summarizes the lap-time results for all simulation configurations. Each lap time is compared against the baseline setup – *low downforce* with *Mass 1* – for both Track A and Track B. The relative performance variations are indicated as time differences, with improvements shown in green and degradation in red.

Figure 5.4 visualizes these relative variations, highlighting the performance impact of increased downforce and increased mass across both tracks.

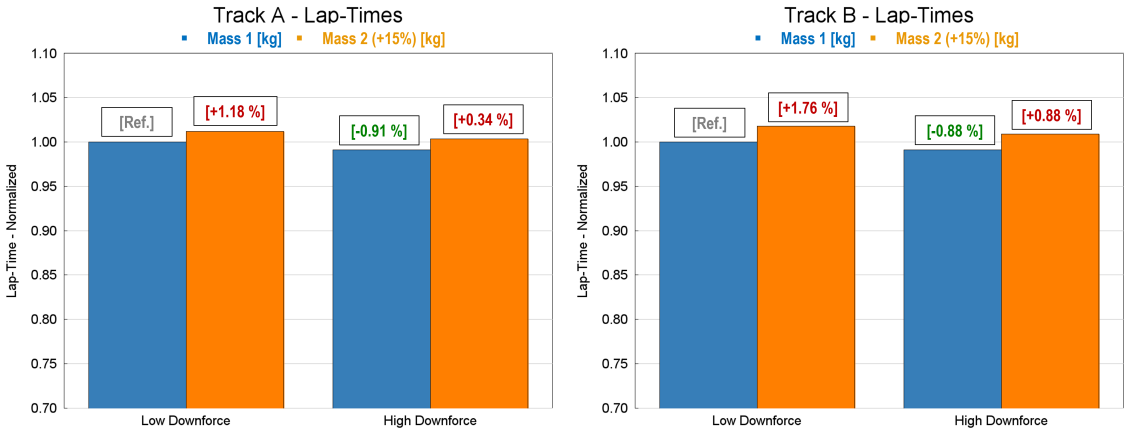

Figure 5.4: Relative lap-time variations compared to baseline configurations.

The results demonstrate consistent trends in response to parameter changes. For both tracks, lap times improve with higher downforce and worsen with increased mass. These findings validate the accuracy of the Driver model's predictions and its adaptability to varying vehicle and environmental conditions.

However, a limitation emerges on both tracks, as highlighted by the telemetries reported in the following simulation analysis, particularly regarding the vehicle side-slip angle. The final corner entry of both tracks shares a similar characteristic: a relatively wide curvature radius during the initial phase of the longitudinal braking. This section

is not classified as a corner by the track pre-processing algorithm, which leads the model to apply braking as if it were a purely longitudinal braking maneuver. In reality, the lay-out introduces a non-negligible lateral component in this section, reducing the vehicle's lateral capability and causing instability as corner approaches.

On Track A , these instabilities remain controllable, but on Track B a manual override is required to anticipate braking and lower the reference entry speed to prevent running wide or spinning. Residual effects of this limitation are evident in vehicle side-slip angle peaks exceeding 0.3-0.4 radians, compared to the 0.2 rad threshold generally considered indicative of stable behavior. While these spikes are managed without loss of control, they highlight an area for improvement in the corner detection and braking strategy.

5.2.1 Simulation (a): Track A, Mass 1, Low Downforce

	Track A	
	Low Downforce	*High Downforce*
Mass 1	1:23.087 Ref.	1:22.329 -0.758
Mass 2 (+15%)	1:24.068 +0.981	1:23.373 +0.286

Table 5.3: Track A Results Matrix: Simulation (a).

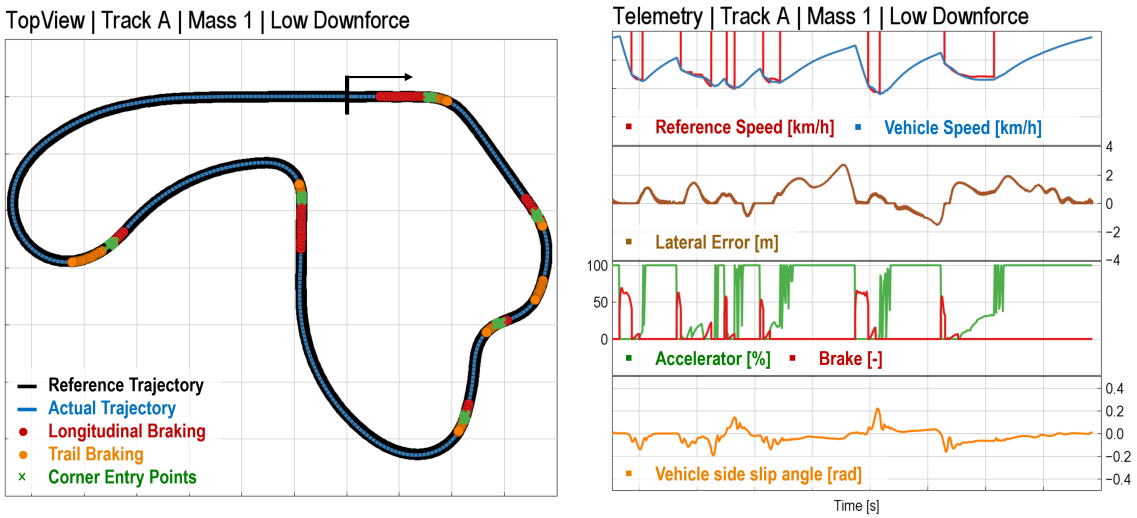

Figure 5.5: Telemetry data for Simulation (a): Track A, Mass 1, Low Downforce.

Simulation (a), conducted on Track A with the vehicle's default setup – *low downforce* and *Mass 1* – serves as the benchmark for evaluating the impact effects of parameter variation (Benchmark lap time is defined as *Ref.* in Table 5.3).

The telemetry data (Figure 5.5) reveal that lateral error remains well-controlled, with only a few spikes exceeding 2 meters and an average error of approximately 1 meter or less throughout most of the lap. The vehicle side-slip angle also shows promising behavior, peaking at around 0.2 radians in the most demanding cornering sections.

It is noticeable how the driver maintains braking input even beyond the processed corner entry point, enabling a higher momentum approach and contributing to lap time optimization. When feasible, the accelerator pedal is applied with values below 20%, allowing the vehicle to navigate corners while maintaining target speed and effectively exploiting its dynamic limits.

5.2.2 Simulation (b): Track A, Mass 1, High Downforce

	Track A	
	Low Downforce	*High Downforce*
Mass 1	1:23.087 Ref.	1:22.329 -0.758
Mass 2 (+15%)	1:24.068 +0.981	1:23.373 +0.286

Table 5.4: Track A Results Matrix: Simulation (b).

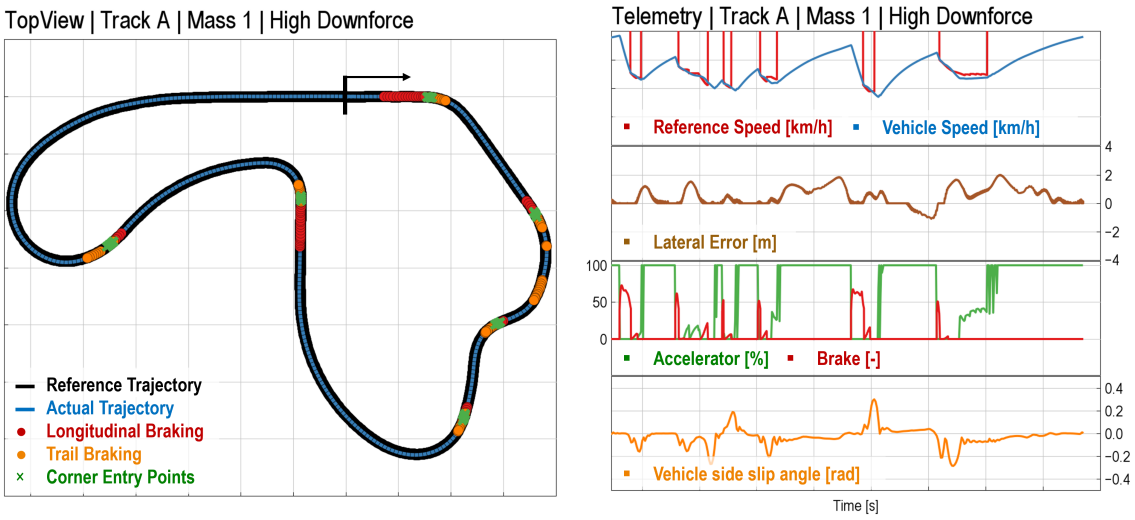

Figure 5.6: Telemetry data for Simulation (b): Track A, Mass 1, High Downforce.

As shown in Table 5.4, Simulation (b) achieves a lap time improvement of approximately seven tenths of a second compared to the baseline, thanks to the increased downforce applied to the vehicle. This result aligns with expectations, confirming that both the pre-processing and driving model are effectively taking advantage of the enhanced aerodynamic performance.

Lateral error spikes are further reduced, with the maximum deviation remaining around 2 meters. This indicates improved trajectory tracking and cornering precision.

Conversely, the vehicle side-slip angle exhibits higher peaks, likely due to the increased tire load resulting from higher downforce. Despite this, the driver model successfully manages these minor instabilities, allowing the vehicle to rotate efficiently without compromising control and further contributing to the improved lap time.

Pedal coasting (i.e. both pedals off) is minimized, and the accelerator input remains smooth throughout mid-corner and corner exit phases. This behavior is particularly evident in the last corner, as shown in the telemetry data in Figure 5.6

5.2.3 Simulation (c): Track A, Mass 2, Low Downforce

	Track A	
	Low Downforce	*High Downforce*
Mass 1	1:23.087 Ref.	1:22.329 -0.758
Mass 2 (+15%)	1:24.068 +0.981	1:23.373 +0.286

Table 5.5: Track A Results Matrix: Simulation (c).

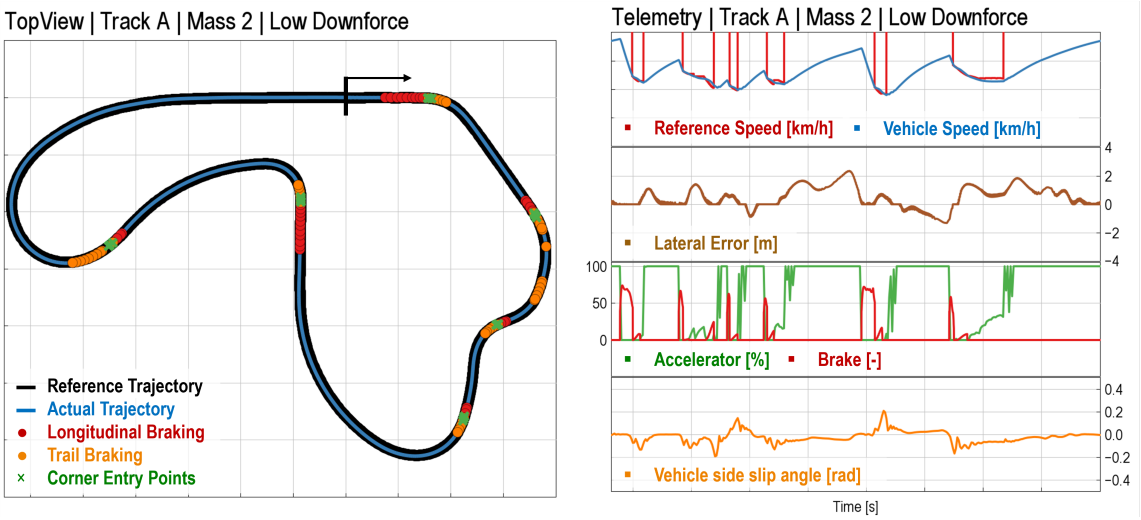

Figure 5.7: Telemetry data for Simulation (c): Track A, Mass 2, Low Downforce.

Simulation (c) results in a lap time approximately one second slower than baseline (Table 5.5). The increased vehicle mass leads to higher inertia, requiring earlier braking and lower corner entry speed, which in turn extends the braking distance.

This behavior is reflected in the telemetry data shown in Figure 5.7, where the accelerator pedal input is noticeably smoother and lower, especially in middle section of the final corner.

Despite the performance drop, the vehicle maintains good stability. Both lateral error and side-slip angle remain within acceptable limits, indicating that the driver model adapts effectively to the increased mass without compromising control.

5.2.4 Simulation (d): Track A, Mass 2, High Downforce

	Track A	
	Low Downforce	*High Downforce*
Mass 1	1:23.087 Ref.	1:22.329 -0.758
Mass 2 (+15%)	1:24.068 +0.981	1:23.373 +0.286

Table 5.6: Track A Results Matrix: Simulation (d).

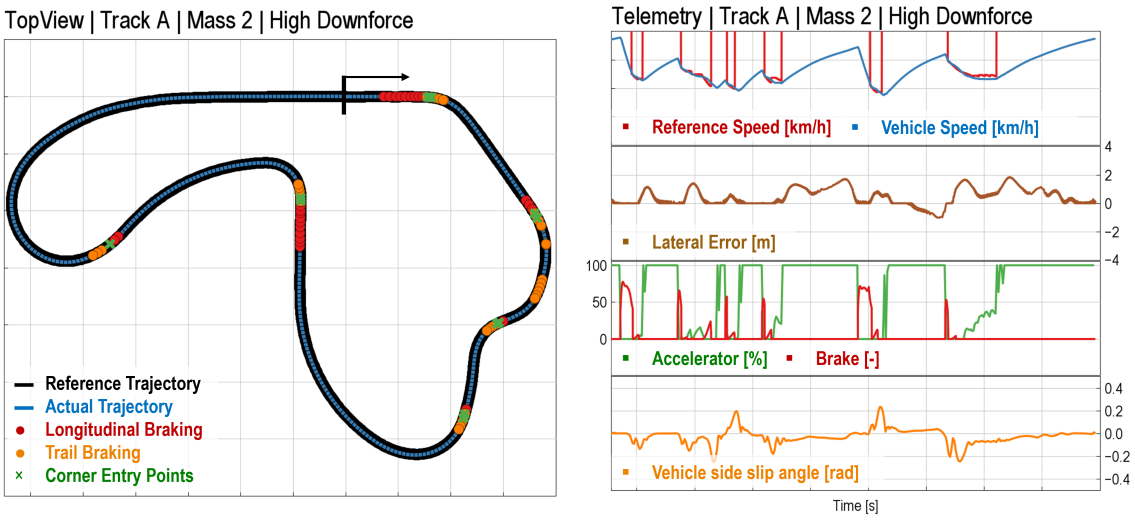

Figure 5.8: Telemetry data for Simulation (d): Track A, Mass 2, High Downforce.

Similar to the improvement observed from Simulation (a) to (b), Simulation (d) benefits from increased downforce, resulting in a faster lap time compared to Simulation (c), where only mass was increased. The lap time approaches the baseline (about three tenths of a second observing Table 5.6), confirming the positive impact of aerodynamic enhancement eve under higher mass conditions.

Analyzing telemetry data of Figure 5.8, braking distances are slightly reduced, and lateral error remains below 2 meters, indicating improved cornering precision. As seen in previous *high downforce* configurations, the vehicle side-slip angle show slightly elevated peaks – up to 0.3 radians – likely due to increased tire load. However, the value remains within controllable limits, and the driver effectively manages the vehicle dynamics.

Overall, the simulation confirms that increased downforce helps soften the negative effect of added mass, maintaining stability and improving performance.

5.2.5 Simulation (e): Track B, Mass 1, Low Downforce

	Track B	
	Low Downforce	*High Downforce*
Mass 1	2:43.440 Ref.	2:42.000 -1.440
Mass 2 (+15%)	2:46.320 +2.880	2:44.880 +1.440

Table 5.7: Track B Results Matrix: Simulation (e).

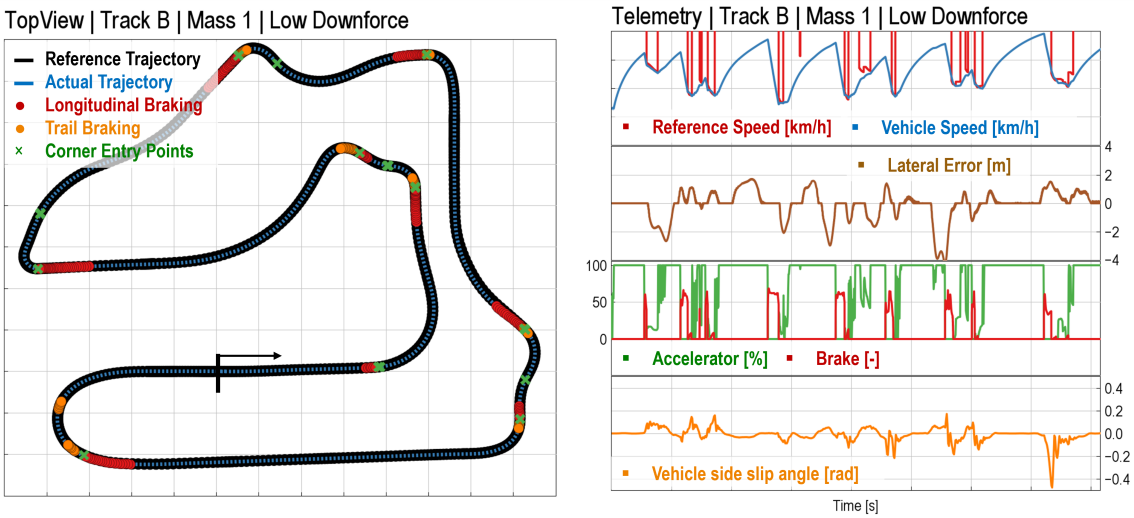

Figure 5.9: Telemetry data for Simulation (e): Track B, Mass 1, Low Downforce.

Simulation (e) uses the baseline vehicle setup as Simulation (a), but on a different track. As shown in Table 5.7, the lap time is significantly longer due to the increased track length and complexity. The layout of Track B, visible in Figure 5.9, features tighter corners, consecutive chicanes, and more demanding braking zones, providing a valuable test of the driver model's adaptability to varied track conditions.

Telemetry data confirms the model's consistent performance, with only one notable stability issue occurring at the entry of the final corner, where a vehicle side-slip angle spike exceeding 0.4 radians is observed.

Another minor deviation is seen in a long, fast turn where the lateral error reaches 4 meters and is sustained for several seconds. Despite this, the vehicle maintains full throttle in that section, indicating a trade-off between trajectory accuracy and speed.

Pedals usage pattern differ significantly from those observed on track A, showing more pronounced on-off behavior. This further highlights the influence of track layout on driving dynamics and vehicle control strategy adaptability.

5.2.6 Simulation (f): Track B, Mass 1, High Downforce

	Track B	
	Low Downforce	*High Downforce*
Mass 1	2:43.440 Ref.	2:42.000 -1.440
Mass 2 (+15%)	2:46.320 +2.880	2:44.880 +1.440

Table 5.8: Track B Results Matrix: Simulation (f).

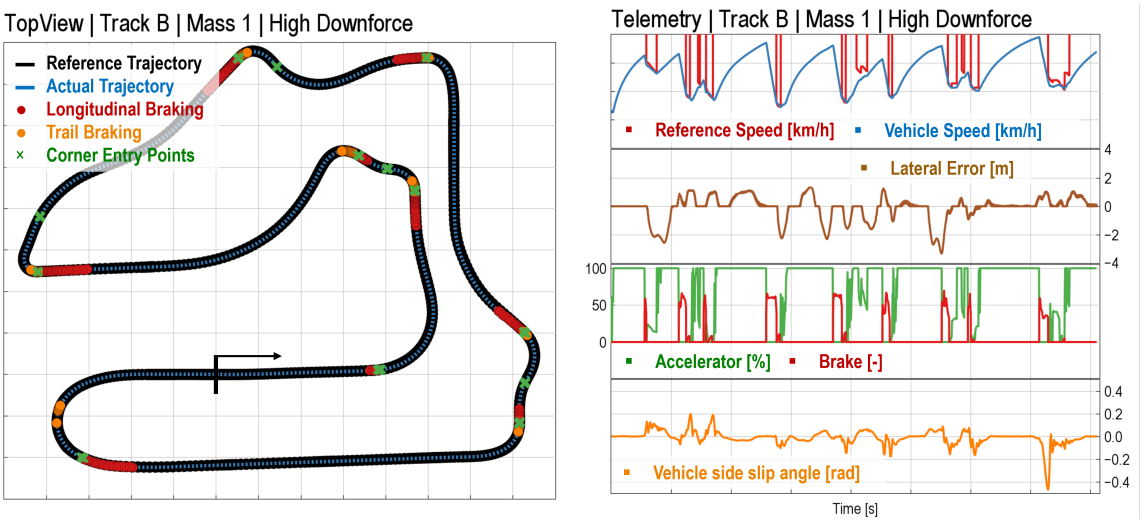

Figure 5.10: Telemetry data for Simulation (f): Track B, Mass 1, High Downforce.

Simulation (f) shows a significant lap time gain, as demonstrated in Table 5.8, primarily due to the increased downforce compared to Simulation (e). This improvement mirrors the trend observed on Track A, confirming the positive impact of aerodynamic enhancement.

The lateral error is notably reduced, particularly in the wide-radius full-throttle left-hand corner during the second half of the lap. As shown in the telemetry data (Figure 5.10) the maximum deviation occurs in the same region as in Simulation (e), but the error is approximately one meter lower, indicating an improved capability at holding the target line.

The final corner entry still exhibits a spike in vehicle side-slip angle, similar to previous runs. However, the driver model successfully recovers without deviating from the reference trajectory, maintaining control and stability. Overall, the simulation confirms that higher downforce enhances vehicle performance, even on technically demanding sections of Track B.

5.2.7 Simulation (g): Track B, Mass 2, Low Downforce

	Track B	
	Low Downforce	*High Downforce*
Mass 1	2:43.440 Ref.	2:42.000 -1.440
Mass 2 (+15%)	2:46.320 +2.880	2:44.880 +1.440

Table 5.9: Track B Results Matrix: Simulation (g).

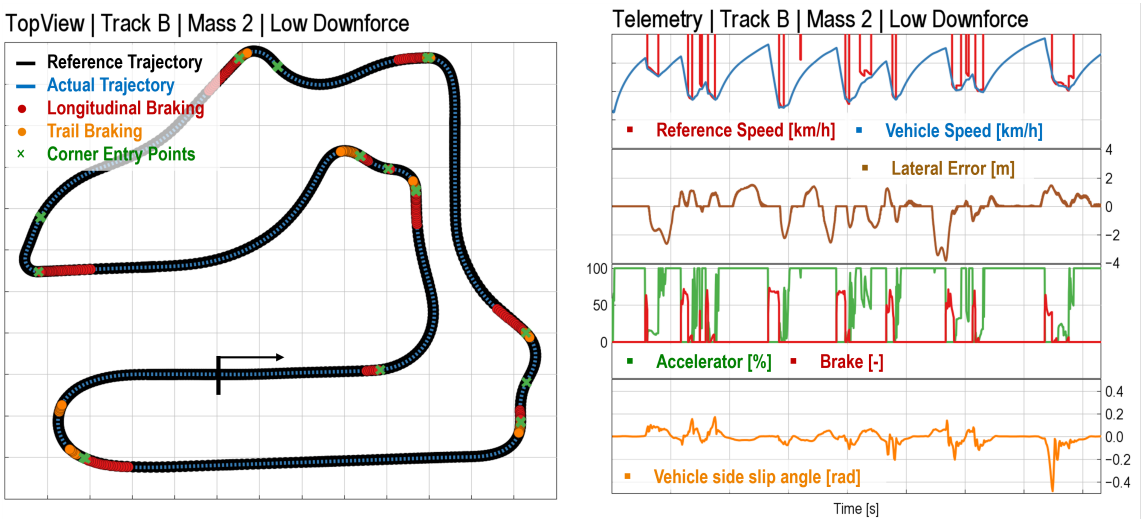

Figure 5.11: Telemetry data for Simulation (g): Track B, Mass 2, Low Downforce.

Simulation (g) presents the slowest lap time for Track B, as shown in Table 5.9. This result aligns with expectations, given that the vehicle configuration combines increased mass with the lower aerodynamic load tested – the least performing setup across all simulations.

Despite performance drop, the driver model adapts effectively to the new conditions, demonstrating the robustness of both the pre-processing and control strategy. This adaptability is crucial, as it confirms the model's ability at resulting effective across varying vehicle configurations and track environments.

The telemetry data (Figure 5.11) shows that the driver navigates through the circuit consistently, positively following the target speed while presenting similar pedal usage patterns as before. Lateral error and vehicle side-slip angle signals remain within acceptable limits, further validating the model's stability and control under less favorable conditions.

5.2.8 Simulation (h): Track B, Mass 2, High Downforce

	Track B	
	Low Downforce	*High Downforce*
Mass 1	2:43.440 Ref.	2:42.000 -1.440
Mass 2 (+15%)	2:46.320 +2.880	2:44.880 +1.440

Table 5.10: Track B Results Matrix: Simulation (h).

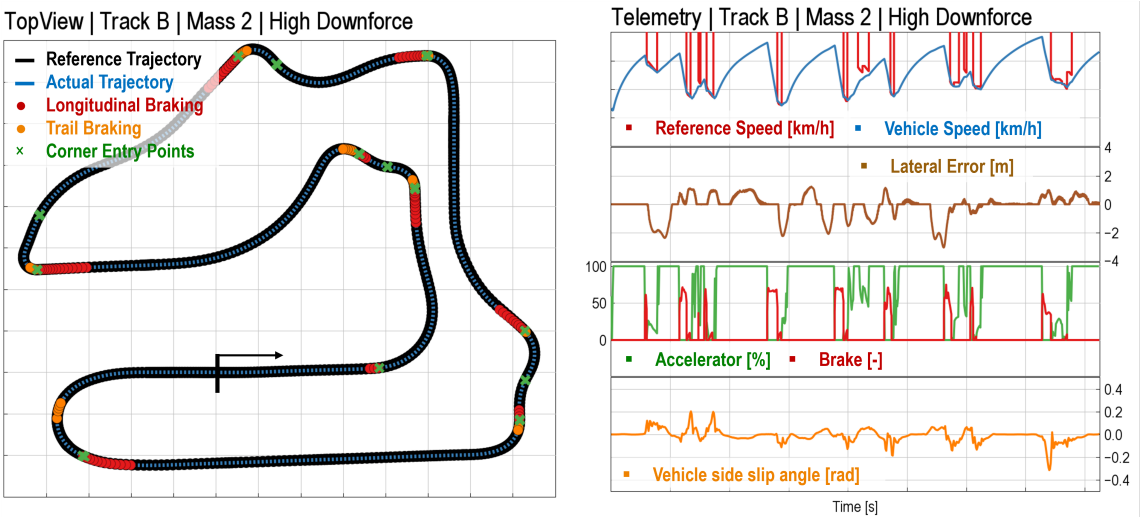

Figure 5.12: Telemetry data for Simulation (h): Track B, Mass 2, High Downforce.

Simulation (h) completes the parameter sweep analysis, using the vehicle configuration with increased mass and high downforce. Although the aerodynamic improvements contribute positively to performance, the added weight results in a slower lap time compared to the baseline, as shown in Table 5.10.

However, when compared to Simulation (g), which shares the same mass but uses a low-downforce setup, the lap time improvement of 1.440 seconds is consistent with the gains observed in the lower mass configuration. This confirms the beneficial role of downforce in mitigating the drawback of the increased vehicle weight.

The telemetry data (Figure 5.12) shows a reduction in lateral error, particularly in high speed corners, and a more stable vehicle side-slip angle profile. Pedal usage patterns remain consistent, with plenty on-off transitions – typical of Track B simulations – and reduced coasting.

Conclusions

The initial goal of this work was to design a driver model from scratch, implemented in Python and capable of operating in co-simulation with GT-Suite. Development began using a few reference datasets and existing driver templates as guidance, but the final objective was to create an independent and robust model. The development started with a preliminary driver structure built and tuned in GT-Suite, which served as a foundation for translating the logic into Python. This translation was followed by fine-tuning, resulting in a final model that integrates pre-processing capabilities for both track and vehicle. These pre-processing steps allow the driver to estimate the dynamic limits autonomously, making the model independent of the specific vehicle or of the environmental configuration. The driver operates based on three main subsystems – steering, throttle, and brake – that work together to optimize lap performance.

The results obtained were highly positive. The transition from GT-Suite to Python not only resulted straightforward, but also improving, enhancing stability and control over the vehicle. Co-simulation with GT-Suite was seamless, enabling the Python driver to potentially interact with any vehicle model as a black box. This configurations ensures generalization, making the driver adaptable to a wide range of applications.

The effectiveness of the model is evident in its ability to consistently follow the desired trajectory with minimal lateral error, while assuring that pedal inputs act consistently with both the track layout and the vehicle's dynamic capabilities. This demonstrates that the same validation criteria can be successfully met across different tracks and vehicle configurations, confirming the robustness of the parameter calibration under varying conditions. The driver also showed strong adaptability when vehicle mass and aerodynamic settings were modified. The trajectory tracking was optimal, as the model treated the provided path as an ideal racing line to be followed without deviation.

Nevertheless, some limitation emerged. Variations in setup and environment did not

compromise the pre-processing and simulation heavily, but few specific sections required manual correction override. These adjustments were necessary on confined limited sections with intermediate curvature values – neither straight lines nor tight corners – where the combination of high speeds and vehicle's model limitations prevented reliable pre-processing predictions. This issue highlights an area of potential improvement in the current track pre-processing algorithm, that tends to identify these zones as purely braking sections, ignoring the contribution of lateral dynamics. This limitation occasionally led to instability, especially during corner entries with high radii, eventually requiring manual intervention to prevent loss of control.

These limitations can be an opportunity for improvement. Enhancing track pre-processing algorithm to better classify intermediate curvature zone would improve braking precision and overall stability in sections where the contribution of lateral forces cannot be neglected.

Looking further ahead, future development could involve a variation on how track data is provided. Instead of supplying an ideal trajectory, the driver could receive the track midline, along with track width and curb positions. This would require the model to compute its own optimal racing line based on vehicle characteristics, increasing adaptability – particularly if different vehicles are tested on the same track – and potentially improving performance.

In summary, the objectives set at the beginning of this work have been successfully achieved. The Python driver results robust, adaptable, and able to operate seamlessly in co-simulation with GT-Suite. It demonstrate strong performance across different vehicle conditions and tracks, validating the model structure design choices and the corresponding control strategies. Despite the known limitations – mainly in handling intermediate track curvature areas – the work provides a reliable platform for future enhancement. With pre-processing improvements and trajectory generation, the driver model can evolve in a more powerful and versatile tool for vehicle dynamic simulation.

References

[1] *Python Documentation*. `www.python.org/doc/`. Accessed: 2025-10-17.

[2] Gamma Technologies, LLC. *GT-SUITE User's Manual*. Version: GT-SUITE v2025. Gamma Technologies. 2025.

[3] Chung J. and Hulbert G.M. *A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-alpha method*. J. Appl. Mech. 60: 371—375 (1993).

[4] Newmark N.M. *A method of computation for structural dynamics.* J. Engrg. Mech. Div. ASCE 85(EM3): 67—94 (1959).

[5] Martin Arnold and Olivier Brüls. *Convergence of the scheme for constrained mechanical systems.* Multibody System Dynamics, Springer Verlag 85, pp.187-202. (2007).

[6] Joga Dharma Setiawan, Mochamad Safarudin, and Amrik Singh. *Modeling, Simulation and Validation of 14 DOF Full Vehicle Model.* Faculty of Engineering, Diponegoro University, Semarang, Indonesia & Faculty of Mechanical Engineering, University Teknikal Malaysia Melaka, Malaysia.

[7] Alberto Reinero. "Lap Time Simulator for a Single-Seater Electric Car". MA thesis. Politecnico di Torino, 2022.

[8] Gabriel M. Hoffmann, Claire J. Tomlin, Michael Montemerlo, and Sebastian Thrun. *Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing*. Aeronautics and Astronautics – Computer Science Departments. Stanford University, Stanford, CA 94305, USA, 2007.

[9] Luca Venerdì. "High Performance Driver Model". MA thesis. Politecnico di Torino, 2021.