POLITECNICO DI TORINO

Master's Degree in Automotive Engineering

Master's Degree Thesis

Imitation learning for path tracking control in a multi-actuated vehicle

---

Supervisor

Prof. Aldo SORNIOTTI

Ing. Cecilia FORMENTO

Prof. Antonio TOTA

Candidate

Salvatore INGRAFFIA

October 2025

Table of Contents

List of Figures

Nomenclature

A	Vehicle frontal area
a	Semi-wheelbase, from CoG to front axle
a_y	Vehicle lateral acceleration
b	Semi-wheelbase, from CoG to rear axle
β	Vehicle sideslip angle
C_x	Drag coefficient
d_{shift}	Space-shifted longline coordinate
e_{ct}	Cross-track error
e_ψ	Heading error
F_x	Tire longitudinal force
F_y	Tire lateral force
FL	Front left tire
FR	Front right tire
GA	Genetic Algorithm
I_z	Vehicle rotational inertia around its vertical axis
IL	Imitation Learning
J_w	Wheel rotational inertia
k_D	Derivative gain
k_P	Proportional gain
k_{ref}	Path curvature
K_{US}	Understeering coefficient
m	Vehicle mass
M_{rr}	Rolling resistance torque
$NMPC$	Nonlinear Model Predictive Control
NN	Neural Network
p	Generic online parameter

R	Wheel radius
ρ	Air density
RL	Rear left tire
RR	Rear right tire
SQP	Sequential Quadratic Programming
t	Vehicle track
τ	Generic time constant
T_{em}	Commanded e-machine torque
u_{act}	Generic executed command input
u_{des}	Generic controller commanded input
v_x	Vehicle longitudinal speed in the vehicle local reference frame
v_y	Vehicle lateral speed in the vehicle local reference frame
W	Weighting matrix
x	Generic vehicle state quantity
x_{ref}	Reference path x-coordinate
y_{prev}	Lateral position preview
y_{ref}	Reference path y-coordinate
$\dot{X}$	Vehicle longitudinal speed in the inertial reference frame
$\dot{Y}$	Vehicle lateral speed in the inertial reference frame
$\dot{\omega}$	Wheel angular acceleration
ψ	Vehicle orientation angle
$\dot{\psi}$	Vehicle yaw rate
$\ddot{\psi}$	Vehicle yaw acceleration
δ	Tire steering angle
δ_{FB}	Feedback contribution to the steering wheel angle
δ_{FFW}	Feedforward contribution to the steering wheel angle
δ_{SW}	Steering wheel angle

1 Introduction

Autonomous driving is poised to reshape modern mobility by eliminating much of the human error component that underlies the majority of road accidents. As reported in [1], the number of fatalities involved in vehicle crashes worldwide in 2020 was 1.35 million, and 90% of cases are related to human error. In this scenario, the usage of autonomous driving can help address the problem, since the expectation is that the autonomous control logic is not affected by typical human issues while driving.

Furthermore, researchers are involved in developing autonomous driving routines capable of performing sophisticated evasive manoeuvres, enabling vehicles to extract themselves safely from hazardous situations that a human driver might be unable to avoid. Some examples are provided in [2] and [3].

In addition to the safety aspect, other benefits come from the usage of autonomous driving. It is possible to smooth traffic flow, reduce congestion, and increase independent travel options for people with disabilities. Furthermore, autonomous driving can provide a significant step toward a definitive transition from private to shared mobility.

The functional architecture of an autonomous-driving stack can be decomposed into four modules. The first is the perception, which involves the fusion of multiple sensors' outputs to reconstruct the surrounding scene. The second module is Localisation and Mapping, which includes the vehicle's capability to localise itself in the space domain with enough accuracy. The third module is the Planning. Once the scenario is acquired and processed, a reference trajectory is designed. The final module is the control, which is responsible for tracking the path using the available actuators and minimising the tracking error.

The control part is the argument of the thesis. There are many different controllers suitable for path tracking. The simplest are the kinematic ones (eg, Stanley

controller) that use pure geometrical considerations for evaluating the steering angle. Then, there are the feedback controllers, which include PID or PD controllers on combinations of lateral and heading errors. A more advanced controller is the LQR, which allows for the use of a linear model of the vehicle. Among the most advanced solutions, probably the most powerful one is the Nonlinear Model Predictive Control. It involves exploiting a nonlinear model of the vehicle to make predictions. Finally, it is possible to combine the solutions for obtaining hybrid controllers.

1.1 Literature Review

Classical feedback controllers, such as Kinematic controllers and PID controllers, have been applied, offering a simple implementation that is robust but shows limits when operating at the limit of handling. Despite that, in [4], an example is provided in which the proposed controller architecture yields highly performing results even close to handling limits.
More advanced linear methods, such as Linear Quadratic Regulators (LQR) and $\mathcal{H}_\infty$ controllers, exploit simplified linear models of the vehicle dynamics, thereby extending the operating range; however, they remain restricted by their reliance on linearization. Nevertheless, there are works, such as [5] and [6], in which these approaches appear to be competitive among the proposed strategies.
In recent years, Nonlinear Model Predictive Control (NMPC) has emerged as one of the most powerful solutions for path tracking. By incorporating a nonlinear dynamic model of the vehicle, NMPC can explicitly handle input/state constraints and optimise performance according to a predefined cost function. However, its applicability in real vehicles is still hindered by the high computational burden. In [7], an NMPC controller is run in real-time with effective results, but not without a proper hardware optimisation.
In general, the NMPC can be run in real-time; for example, in [2], [8], and [9], demonstrations of real-time implementability are provided. However, the models used are Nonlinear ones, albeit not particularly complex ones. For example, the tyre model is not a complete Pacejka model; the vehicle is approximated to a bicycle model, and the number of actuators is reduced.

Regarding Imitation Learning (IL), this approach enables overcoming the limitations associated with the real-time implementability of NMPC. There are examples of articles in which IL is implemented conventionally, fully imitating the NMPC, preserving the complete set of inputs, and using the NN as a clone of the NMPC. Among them, in [10] it is provided a clear example of conventional IL.

There are also examples where the neural network performs imitation learning without requiring the full vehicle state, relying instead on human drivers or other empirical data sources as the reference, rather than an NMPC. One worth citing is [11]. However, this article limits the scenario to highways. Consequently, the controller is never evaluated under demanding conditions. Similarly, in [12], a neural network in which the vehicle dynamics are not explicitly modelled is employed. However, the considered scenarios are purely kinematic, where the influence of vehicle dynamics on overall performance is negligible.

Another aspect related to the effectiveness of path-tracking controllers is the delay associated with sensors and actuators. Conventionally, the solution to compensate for delays between sensors and actuators is the use of dedicated compensation methods. A clear example is provided in [13]. In this article, the benefits of delay compensation are demonstrated; however, the authors also highlight the limitations associated with the tuning of the compensator. Therefore, the performance improvements of the controller are strictly linked to the quality of the compensator and remain dependent on the quality of the sensors.

1.2 Goals and contributions

The thesis's goal is to design a controller that guarantees the best possible tracking performance, even in highly demanding manoeuvres. So, even in high lateral acceleration conditions. As previously expressed, the most effective controller can be considered the NMPC; however, it has limitations primarily linked to high computational cost and secondly linked to the sensor and actuator performances. The IL is a way of overcoming the limitations associated with the required CPU time, but it remains reliant on the quality of the sensors.

Summarising, there are three main limitations to address:

- The very high computational cost associated with an NMPC (in particular for multiple actuator systems) that does not allow for real-time utilisation.
- The need to use the whole state of the vehicle, which must be retrieved by multiple sensors, most of which are costly.
- Even after removing most of the sensors and reducing the required set of inputs (in some way), the performance remains heavily affected by the sensors' performance.

To address the previously described issues the contributions of the thesis work are the following:

- Design of a controller based on Imitation Learning of NMPC simulation outputs. The input set provided to the neural network is significantly reduced, thus avoiding the need for full-state measurements.
- Generation of the training database through models of non-ideal sensors (including delays, coarse sampling, and noise). Rather than compensating for such effects with estimators like Kalman filters, the neural network is directly exposed to them during training, thereby achieving intrinsic robustness to realistic sensing conditions.

Finally, the proposed methodology is implemented and validated on a multi-actuated vehicle platform. The thesis also provides a systematic analysis of the advantages of multi-actuation, demonstrating the proposed approach's capability to ensure consistent and robust performance across a wide range of actuator configurations.

2 | Experimental setup

Figure 2.1: PIX-KIT Hooke 2.0.

The experimental platform, shown in Figure 2.1, adopted in this work is the PIX-KIT Hooke 2.0 Autonomous Driving Development Kit, developed by PIX Moving (Guizhou, China). The vehicle is equipped with a fully electric drive-by-wire chassis with four independent in-wheel motors, enabling independent control of torque at each wheel. Both the front and rear axles are equipped with steer-by-wire actuators. Four disc brakes are present and controlled by applying the braking pressure. A parking brake is present for slope holding.

Two operational modes are available: Normal Control Mode, where the Vehicle

Control Unit (VCU) executes internal motion control algorithms for steering, propulsion, and braking; and Advanced Autonomous Driving Interface Mode, which bypasses the VCU's high-level logic, enabling direct control of all motion actuators.

The vehicle is equipped with multiple sensors that enable the sensing of the complete vehicle state, allowing for the perception of surroundings and centimetre-precise localisation of the vehicle within its surroundings.

The onboard perception and navigation suite includes:

- Three-dimensional LiDAR – RS-Helios-16P, 16-channel scanning, providing a 360° field of view for environment mapping and obstacle detection.
- Monocular RGB camera – Sensing SG2 (or equivalent), forward-facing, GMSL2 interface, high dynamic range, suitable for lane and object detection.
- Integrated GNSS/INS unit – CHCNAV CGI-410, dual-antenna, centimetric-level accuracy, providing precise position, velocity, and attitude estimation.
- Ultrasonic radar array – Chenmu Technology F40-16TR9BL2 (or equivalent) with 8 detection probes, used for short-range obstacle detection and parking assistance.
- Millimetre-wave radar – Continental ARS408-21, long-range FMCW radar, for vehicle and object tracking in various weather conditions (optional module).
- Optical sensor – Kistler Correvit S-Motion, non-contact optical Doppler sensor measuring sideslip angle with ±20° range, 0.01° resolution, and ±0.1° accuracy, sampled at up to 250 Hz.

Onboard computation is managed by an industrial-grade PC, interfaced via CAN and Ethernet to all subsystems. Auxiliary components include an 8-port network switch, a display, a keyboard/mouse, and a wireless remote control unit. The chassis dimensions are approximately 2.52 m × 1.68 m, with a curb weight of 480 kg (including an 11 kWh battery) and a maximum payload capacity of 500 kg. The maximum software-limited speed is 40 km/h, while the nominal driving range is around 130 km under standard operating conditions.

For the control implementation, a dSPACE MicroAutoBox III was used to run the Simulink model, which contained the proposed path-tracking controller, in real-time. The MicroAutoBox is directly connected to the vehicle's CAN bus, ensuring deterministic execution and low-latency actuation of control commands.

The development and compilation of the control algorithms were performed on a Victus laptop equipped with an Intel Core i7 processor, which served as the development workstation for Simulink and dSPACE ControlDesk. This setup allowed a seamless transition from offline simulation to real-time vehicle testing.

3 Vehicle model identification and validation

The goal of the thesis work is to create a neural network for advanced path tracking manoeuvres. The neural network is trained based on simulation outputs. To obtain a NN that is performing well enough, in particular in highly demanding conditions, the simulation outputs must be as close to reality as possible.

Supposing the vehicle model is not a high-fidelity one. In that case, the consequences can be catastrophic, as the NN cannot be fine-tuned in the field like a PID controller or other classical controllers; therefore, it is not possible to account for discrepancies between the model and the real vehicle after training.

3.1 Vehicle model

The vehicle model is defined and run in a Simulink environment, allowing for the exploitation of certain Simulink tools (e.g., the time delay operation). It is possible to divide the model into three main parts:

- vehicle dynamics
- tire dynamics
- actuator and sensors dynamics

The vehicle dynamics consist of seven degrees of freedom. Three of them are the longitudinal, lateral, and yaw motions of the vehicle body, treated as a rigid body. The other four are relative to the rotational motion of the wheels.
The tyre dynamics are described by using a simplified Pacejka model. The camber angle is not considered, as the roll motion is not taken into account. The tyre model evaluates longitudinal and lateral forces, but not the self-aligning moment; the

interaction between longitudinal and lateral slip is considered. The actuator and sensor dynamics are approximated by using zero-order hold blocks to simulate the sensors' sampling times, time delay blocks to account for the delay between actuation request and actual actuation, and first-order transfer functions to represent the dynamics of actuation.

Figure 3.1 offers a schematic of how sensors and actuators are modelled. Figure 3.2 provides a comparison between the steering angle commanded by the controller, the measured actuated angle and the simulated actuated one. It is possible to appreciate how well the model works; in fact, the dashed red line is almost perfectly overlapped by the solid blue line. Fortunately, sensors and actuators appear to function consistently in different scenarios.

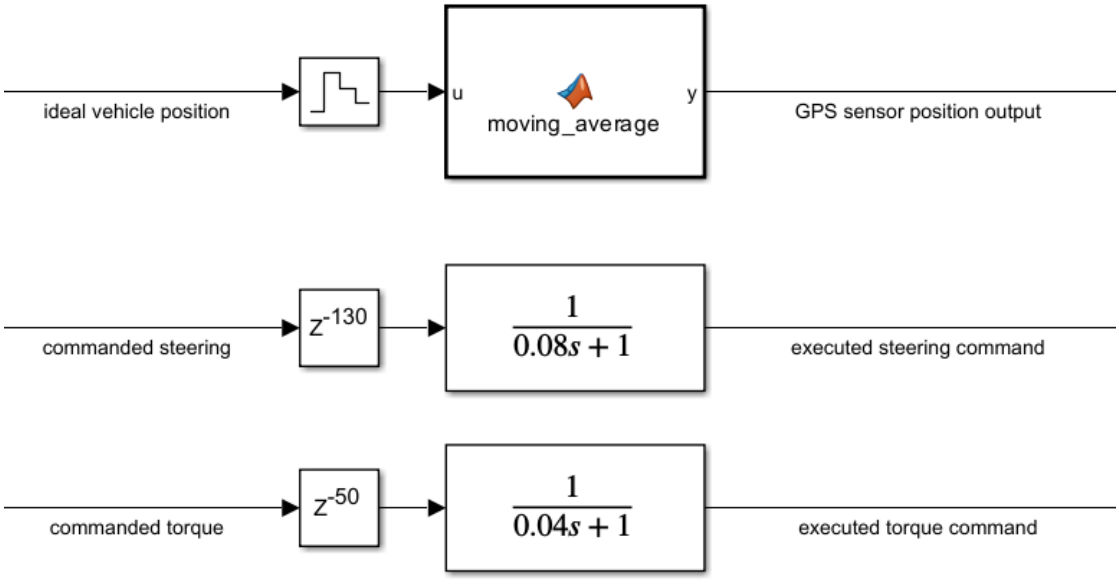

Figure 3.1: schematic of how sensors and actuators are modelled.

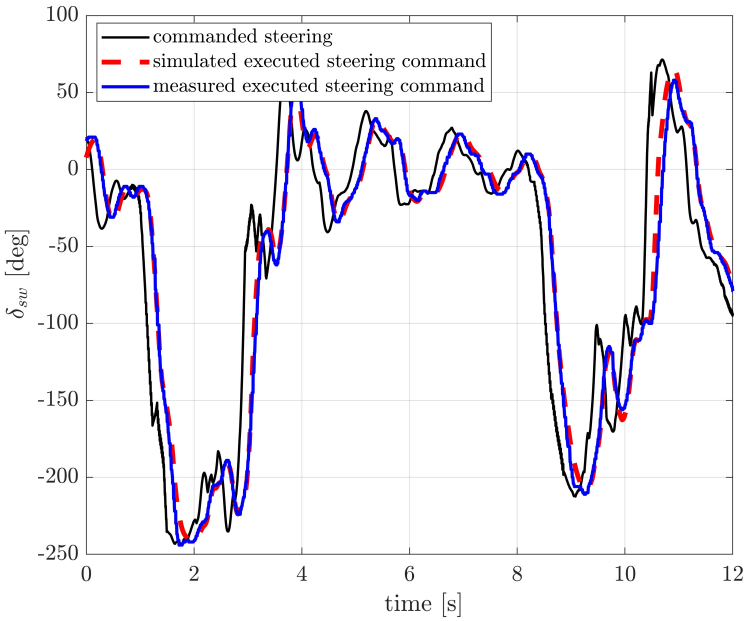

Figure 3.2: steering wheel signals, comparison between commanded steering and actuated ones (simulation vs experiment).

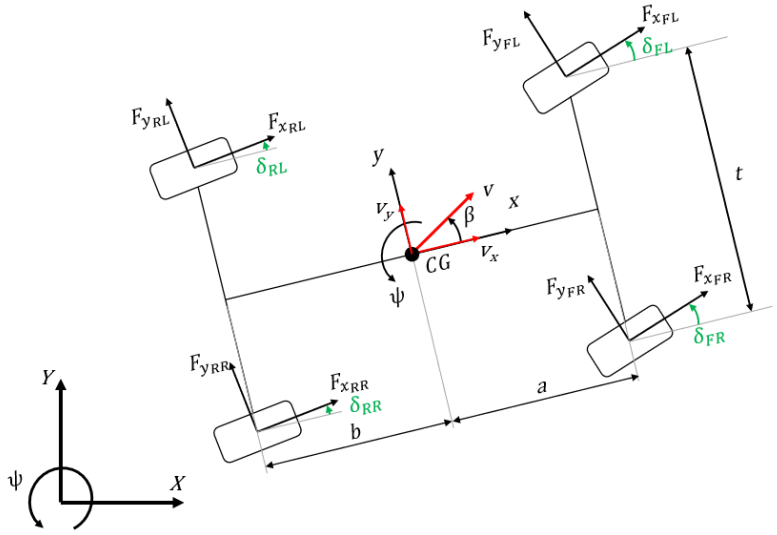

Figure 3.3: schematic of the vehicle dynamic quantities.

Equations from 3.1 to 3.9 are relative to vehicle dynamics.

$$\dot{X} = v_x \cos\psi - v_y \sin\psi \tag{3.1}$$

$$\dot{Y} = v_x \sin\psi + v_y \cos\psi \tag{3.2}$$

$$\begin{aligned}\dot{v}_x = \frac{1}{m}\Big(& F_{x,FL}\cos\delta_{FL} - F_{y,FL}\sin\delta_{FL} + F_{x,FR}\cos\delta_{FR} \\ & - F_{y,FR}\sin\delta_{FR} + F_{x,RL}\cos\delta_{RL} - F_{y,RL}\sin\delta_{RL} \\ & + F_{x,RR}\cos\delta_{RR} - F_{y,RR}\sin\delta_{RR}\Big) - \frac{1}{2}\rho A C_x\, v_x^2 + v_y\dot{\psi}\end{aligned} \tag{3.3}$$

$$\begin{aligned}\dot{v}_y = \frac{1}{m}\Big(& F_{x,FL}\sin\delta_{FL} + F_{y,FL}\cos\delta_{FL} + F_{x,FR}\sin\delta_{FR} \\ & + F_{y,FR}\cos\delta_{FR} + F_{x,RL}\sin\delta_{RL} + F_{y,RL}\cos\delta_{RL} \\ & + F_{x,RR}\sin\delta_{RR} + F_{y,RR}\cos\delta_{RR}\Big) - v_x\dot{\psi}\end{aligned} \tag{3.4}$$

$$\begin{aligned}\ddot{\psi} = \frac{1}{I_z}\Bigg\{ & a\Big[F_{x,FL}\sin\delta_{FL} + F_{y,FL}\cos\delta_{FL} + F_{x,FR}\sin\delta_{FR} \\ & + F_{y,FR}\cos\delta_{FR}\Big] - b\Big[F_{x,RL}\sin\delta_{RL} + F_{y,RL}\cos\delta_{RL} \\ & + F_{x,RR}\sin\delta_{RR} + F_{y,RR}\cos\delta_{RR}\Big] + \frac{t}{2}\Big[- (F_{x,FL}\cos\delta_{FL} \\ & - F_{y,FL}\sin\delta_{FL}) + (F_{x,FR}\cos\delta_{FR} - F_{y,FR}\sin\delta_{FR}) \\ & - (F_{x,RL}\cos\delta_{RL} - F_{y,RL}\sin\delta_{RL}) + (F_{x,RR}\cos\delta_{RR} \\ & - F_{y,RR}\sin\delta_{RR})\Big]\Bigg\}\end{aligned} \tag{3.5}$$

$$\dot{\omega}_{FL} = \frac{1}{J_{w,FL}}(T_{em,FL} - F_{x,FL}\, R_{FL} + M_{rr,FL}) \tag{3.6}$$

$$\dot{\omega}_{FR} = \frac{1}{J_{w,FR}}(T_{em,FR} - F_{x,FR}\, R_{FR} + M_{rr,FR}) \tag{3.7}$$

$$\dot{\omega}_{RL} = \frac{1}{J_{w,RL}}(T_{em,RL} - F_{x,RL}\, R_{RL} + M_{rr,RL}) \tag{3.8}$$

$$\dot{\omega}_{RR} = \frac{1}{J_{w,RR}}(T_{em,RR} - F_{x,RR}\, R_{RR} + M_{rr,RR}) \tag{3.9}$$

3.2 Parameters estimation method

Once the model is defined, the goal is to identify the parameters that are several and of different natures. There are parameters related to the vehicle's construction, such as mass and geometry. These parameters can be measured directly, but it can be very difficult to do so for some of them (e.g. rotational inertia). The difficult-to-measure parameters are estimated and then refined in the identification process. Then, some parameters cannot be directly measured or estimated, for example, the tyre parameters.

To identify the model parameters, experimental acquisitions are used, which are then compared with the simulation outputs to validate the model. The quantities used for the validation of the model are three: the vehicle sideslip angle β, the vehicle yaw rate $\dot{\psi}$, and the lateral acceleration a_y.

The validation manoeuvres chosen are a skidpad and a sinusoidal steering. For the skidpad test, a fixed steering angle is imposed on the vehicle, and it is allowed to accelerate smoothly until it reaches its maximum target speed. The inputs provided to the vehicle, such as the steering angle and the four-wheel torques, are also recorded for use during the simulation phase. Regarding the sinusoidal steering, the vehicle accelerates while it steers sinusoidally.

The idea behind the estimation method is simple. The experimental outputs are compared to the simulation outputs; there exists a certain error between the simulation and experimental signals, which is evaluated as follows:

$$\|error\|_2 = \sqrt{\int_0^T (exp(t) - sim(t))^2 \, dt} \tag{3.10}$$

The error is the norm of the difference between the two time signals. The model parameters are changed until this error is small enough.

From a conceptual point of view, the easiest way to find the correct set of coefficients is to try a vast number of possible combinations and select the set that yields the lowest error. Of course, this method is inefficient and does not guarantee the best possible result. So, it is possible to use an optimisation algorithm. There are many, and they exploit different strategies. The two chosen are Genetic Algorithm (GA) and Sequential Quadratic Programming (SQP). These are two standard optimisation methods. The Genetic Algorithm is a population-based, stochastic technique inspired by natural selection. It uses operations such as selection, crossover, and mutation to evolve a set of candidate solutions toward an

optimal result. GA is beneficial for complex, nonlinear problems where the search space may contain many local minima.
On the other hand, Sequential Quadratic Programming is a deterministic, gradient-based method designed for constrained nonlinear optimisation. SQP solves a sequence of quadratic subproblems that approximate the original problem, updating the solution step by step until convergence is achieved. While GA is robust and global in nature, SQP is efficient and accurate when a good initial guess is available.

To summarise, the GA is effective for finding the global minimum, but it is very slow for a large set of parameters. In contrast, SQP is relatively fast but risks converging to a local minimum.
To obtain the benefits from both methods, they were used in sequence. The GA is initially used with a vast population (containing at least twenty individuals per parameter), but the number of generations is kept small to limit the time required. Then, the result obtained, which is supposed to be very close to the global minimum, is given to the SQP that refines the result.

The cost function of both is based on 3.10. In fact, the error to minimise is the sum of three errors, as many as the validation quantities (β, $\dot{\psi}$, and a_y). As explained below, the longitudinal speed is also used as a validation quantity, but it is considered separately from these three.

Once the optimisation method is clarified, it is possible to explain the optimisation procedure. Fundamentally, the vehicle model can be validated for the longitudinal and lateral dynamics separately. In fact, it is possible to have a model that is for the longitudinal and lateral only. This is a good point to exploit; thanks to this, there exists the possibility to find the longitudinal and lateral parameters separately, which has the benefit of reducing the complexity of the optimisation problem. Therefore, the parameters related to the longitudinal dynamics were first determined by exploiting a coastdown manoeuvre. Then the lateral dynamics parameters were identified using a manoeuvre at a fixed steering angle. The whole system is finally validated using a sinusoidal steering manoeuvre.

3.3 Results

In this section, the results are shown. It is possible to appreciate the high level of fidelity of the model. In particular, the model performs well even in high longitudinal and lateral accelerations. As shown by Figure 3.6, the model is good even when the vehicle is accelerating while performing the steering action, which is considered to be a critical condition. Figure 3.7 shows a circuit lap performed

by the vehicle; the speed is approximately constant, but the lateral acceleration is medium-high. Also, in this condition, the model appears to be highly faithful to reality. The latest result is considered proof that the model works, as it was not used for the optimisation problem, unlike the others.

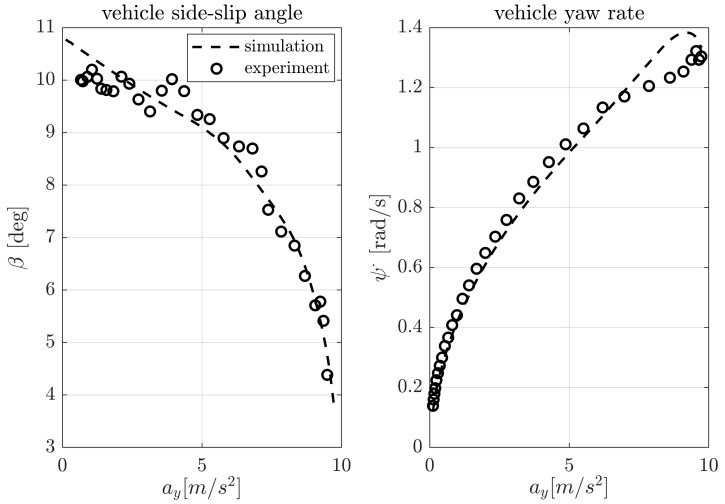

Figure 3.4: constant steering angle manoeuvre.

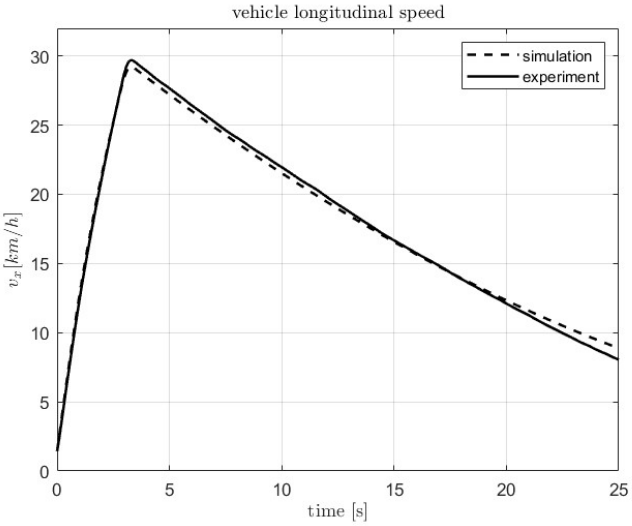

Figure 3.5: Coastdown manoeuvre.

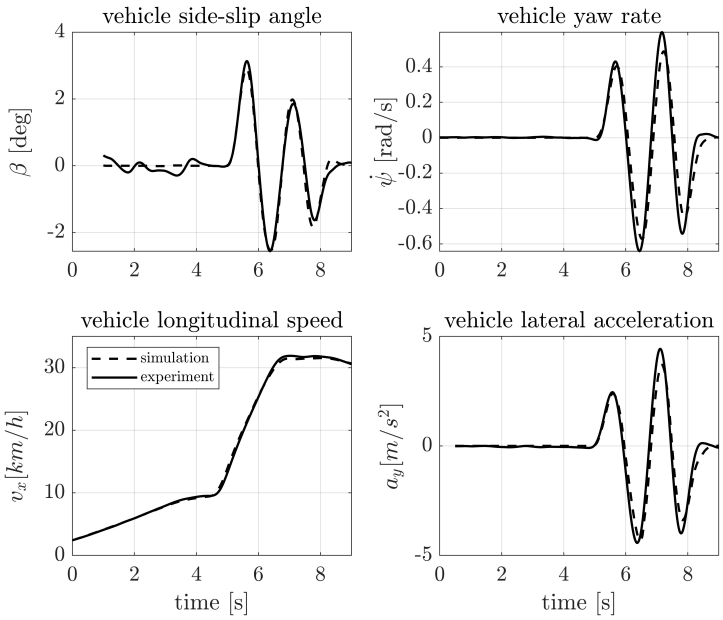

Figure 3.6: Transient steering manoeuvre.

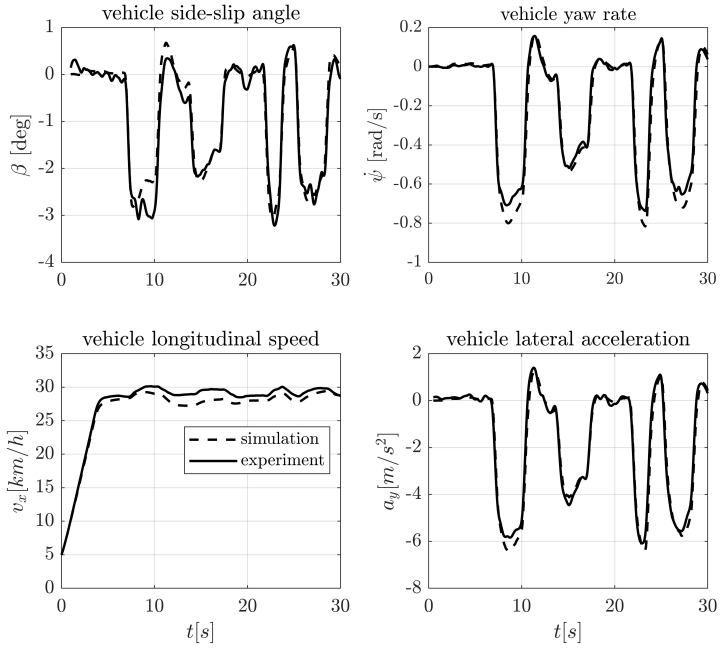

Figure 3.7: circuit lap.

4 Path-Tracking problem formulation, trajectories, and Key Performance Indicators

In this chapter, several relevant aspects of the path tracking problem are presented. Specifically, it introduces the error definitions used in the path tracking formulation, the trajectories employed for testing the controllers in both simulation and experimental scenarios, and, finally, the Key Performance Indicators (KPIs) used to evaluate the controller's performance.

4.1 Path-Tracking errors

The errors used for the path tracking problem are the following:

$$e_{ct} = (y_{ref} - y)\cos(\psi_{ref}) - (x_{ref} - x)\sin(\psi_{ref}) \tag{4.1}$$
$$\dot{e}_{ct} = v_y \cos(e_\psi) - v_x \sin(e_\psi) \tag{4.2}$$
$$e_\psi = \psi_{ref} - \psi \tag{4.3}$$
$$\dot{e}_\psi = k_{ref} \cdot v_x - \dot{\psi} \tag{4.4}$$
$$\tag{4.5}$$

They are the cross-track error (4.1), the derivative of it (4.2), the heading error (4.3) and the derivative (4.4). It is essential to note that the four are evaluated independently according to the proposed formulas, so the cross-track error is never

evaluated as the integral of the derivative of the cross-track error. In fact, due to non-idealities typical of the experimental field, it is possible that the integral of 4.2 is not equal to 4.1. Furthermore, the term $\dot{e}_\psi$ is not the complete formulation, but it is only an approximation that is true in the case that the heading error and the cross-track error are small. Figure 4.1 aims to provide a geometrical representation of (4.1) and (4.3).

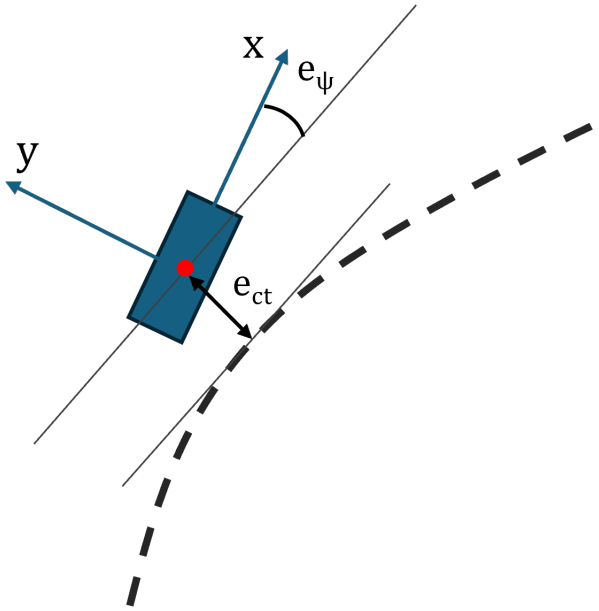

Figure 4.1: graphical representation of the path-tracking errors.

4.2 Trajectories

The trajectories used for testing the controllers are several. Figure 4.2 shows them. The first two are circuits, created at the Torino Aeroclub, and experiments are conducted using them; the difference between A and B lies in the second turn. In the B version, the second turn is modified to be trickier. They are composed of a sequence of constant-radius turns and straight lines. The most severe turn is a 10-meter radius turn (circuit B), which guarantees a challenging scenario with high lateral acceleration. Supposing that the turn is perfectly tracked at a constant speed, the lateral acceleration reached is 7 m/s^2. It is noted that for lateral accelerations exceeding approximately 6 m/s^2, the tyres typically operate in the non-linear region, approaching saturation (which occurs at around 8 to 9 m/s^2), after which it depends on a series of other factors. Therefore, the controller is used in highly demanding conditions, including the experimental field.

The second trajectory is used only in a simulation environment. It can be considered an extreme manoeuvre for two main reasons. The first is related to the fact that the two consecutive turns have minimal radii of 10 and 8 meters, respectively. The second reason is that the passage from the first to the second turn is direct, without a solution of continuity, which guarantees a stressful condition for the vehicle, as well as strong lateral accelerations of up to almost 1g.
The third path is used for conducting a robustness analysis; it is like the second one, but it has a lower level of difficulty. A small straight line separates the two turns, and the radii are both 10 meters in length.

4.3 Key Parameter Indicators (KPIs)

KPIs are an essential instrument for making comparisons between the results obtained using different controllers/configurations. The most important KPI is, without a doubt, the cross-track error, since the goal of the work is to develop a controller that performs as well as possible in tracking the path. Therefore, in general, the controller is more effective the closer it can stay to the trajectory. However, there are other important indicators; for sure, the heading error is one of them, but it is also important to evaluate the sideslip angle. Generally, it is a good practice to maintain a slight sideslip angle. Finally, to evaluate the controller activity, the steering angle and its derivative.

Following is a list of the KPIs used:

- $e_{ct,\mathrm{MAX}}$, peak value of the cross-track error.
- $e_{ct,\mathrm{RMS}}$, rms value of the cross-track error.
- $e_{\psi,\mathrm{MAX}}$, peak value of the heading error.
- $e_{\psi,\mathrm{RMS}}$, rms value of the heading error.
- β_{MAX}, peak value of the sideslip angle.
- β_{RMS}, rms value of the sideslip angle.
- $\delta_{SW,\mathrm{RMS}}$, rms value of the steering angle.
- $\dot{\delta}_{SW,\mathrm{RMS}}$, rms value of the derivative of the steering angle signal.

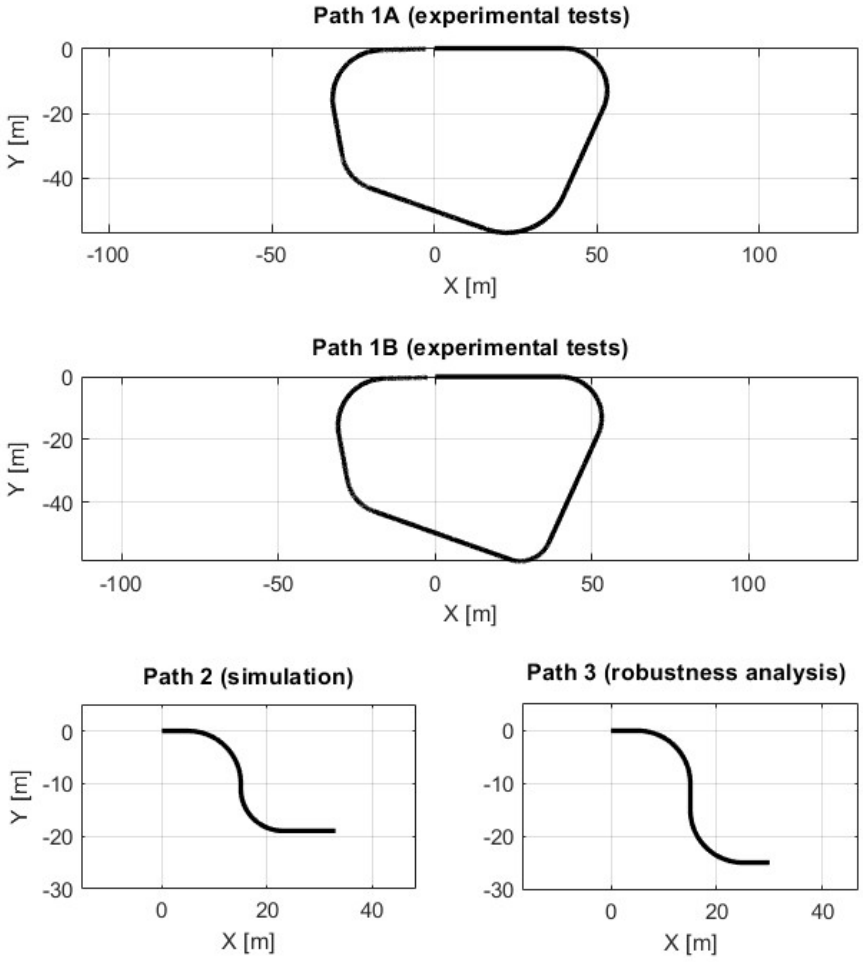

Figure 4.2: trajectories used for controller testing.

5 Design of a PD controller for benchmarking purposes

As stated, the thesis's goal is to design a controller that guarantees the best possible tracking performance. However, to understand how well the controller performs, it is necessary to have a benchmark. There exist different controllers, and their performance has been largely debated in the literature. The simplest controller architecture includes a PID controller that operates in feedback with the cross-track error, the heading error, or a combination of the two, along with a feedforward contribution based on knowledge of the trajectory curvature.
This structure is simple but effective, as demonstrated in [4].

The controller proposed in this chapter is not exactly equal to the one proposed by Kapania and Gerdes. Still, it is similar, and it works even in highly demanding conditions.

5.1 Controller definition

The controller is defined through the following equations:

$$\delta_{SW} = \delta_{FFW} + \delta_{FB} \tag{5.1}$$

$$\delta_{FFW} = \left(1 + K_{US}\, v_x^2\right)(a+b)k(s_{\text{shift}}) \tag{5.2}$$

$$\delta_{FB} = K_P\, e_{ct} + K_D \frac{de_{ct}}{dt} \tag{5.3}$$

There are two distinguished contributions, the feedback one and the feedforward. The feedback contribution is a PD controller on the cross-track error. The feedforward contribution is the kinematic steering (wheelbase times curvature)

that multiplies the term, including the understeering coefficient, to account for the vehicle's dynamic behaviour.

Looking carefully at (5.2), it is possible to note that the curvature k is a function of a term called s_{shift}, which is the longline coordinate shifted forward according to the vehicle's speed. It is like using a look-ahead distance, and the utility is to account for the GPS signal delay and the steering actuator delay. This expedient is not merely a detail for improving the controller's performance; it is essential for its proper functioning, and fine-tuning is required to achieve the optimal spatial shifting. In other words, it is not sufficient to sum the GPS delay and the steering actuator delay (in the spatial domain).

Figure 5.1 shows a simulation result. It underlines the importance of using the derivative term. It contributes to making the controller more stable, avoiding the vehicle from starting to oscillate around the reference.
Figure 5.2 refers to experimental tuning of K_{US}. The manoeuvre is conducted using only the feedforward contribution (no feedback). The contribution of the understeering coefficient is crucial to account for the oversteering behaviour of the vehicle. Looking at the manoeuvre conducted using $K_{US} = 0s^2/m$, it is possible to note that the vehicle is oversteering in the case that it is provided with only the kinematic contribution. $K_{US} = -0.0015s^2/m$ helps to account for the oversteering behaviour of the vehicle.

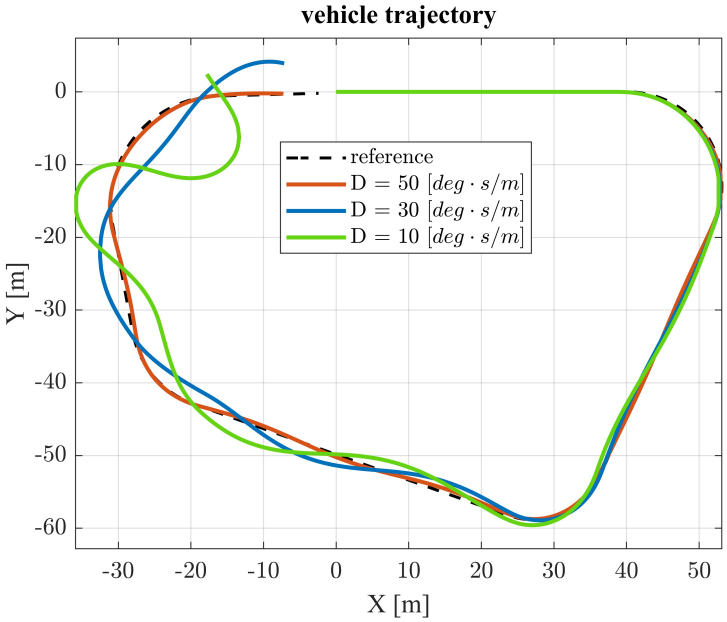

Figure 5.1: Vehicle trajectory changing the derivative contribution.

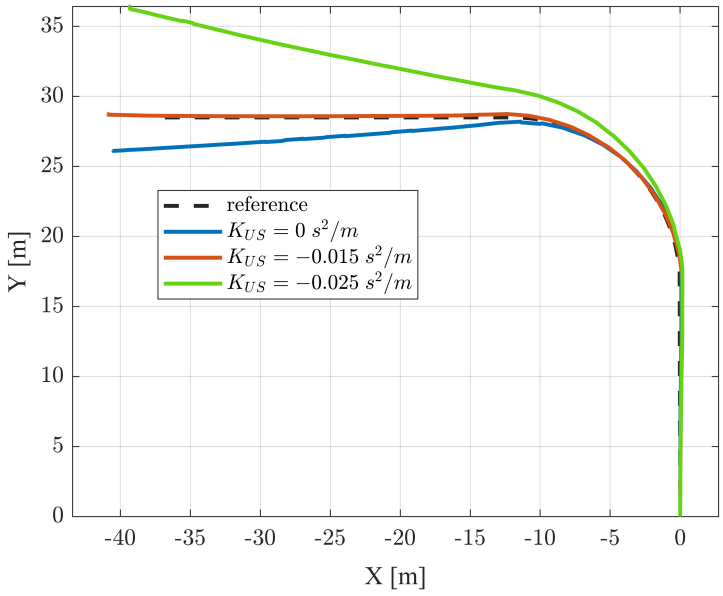

Figure 5.2: Vehicle trajectory changing K_{US}, experimental data.

5.2 Experimental results

In this section, the experimental results are presented. These results are based on a tuning process in which the main controller parameters were adjusted in accordance with the proposed test outputs. The circuit adopted is the 1B.

Examining Figures 5.3-5.5, the controller appears to function effectively even under demanding conditions. The vehicle does not oscillate around the reference, indicating that it does not start in unstable conditions. Looking at Figure 5.4, it is possible to observe that for almost 50% of the time, the vehicle is exactly on the reference.

Another important aspect concerns disturbances. These are present due to the sensors; they could be detrimental, particularly for the derivative contribution. Thanks to proper filter tuning, the disturbances are rejected without compromising path-tracking performance too much.

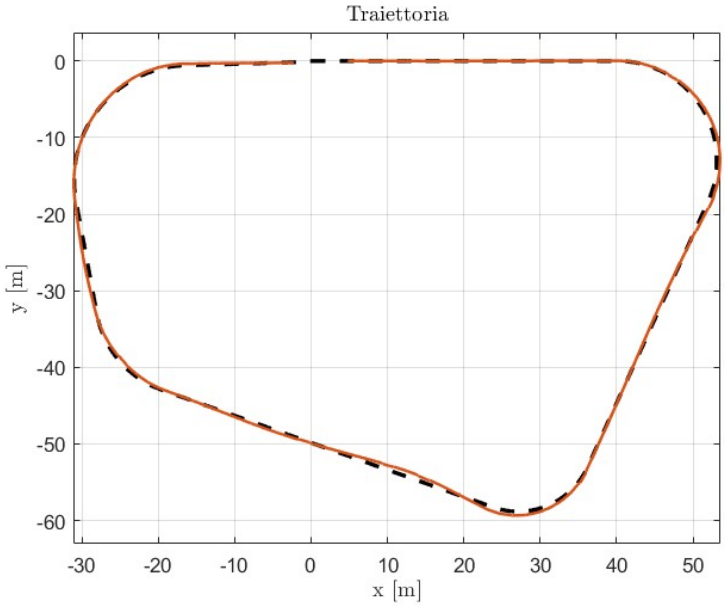

Figure 5.3: Vehicle trajectory.

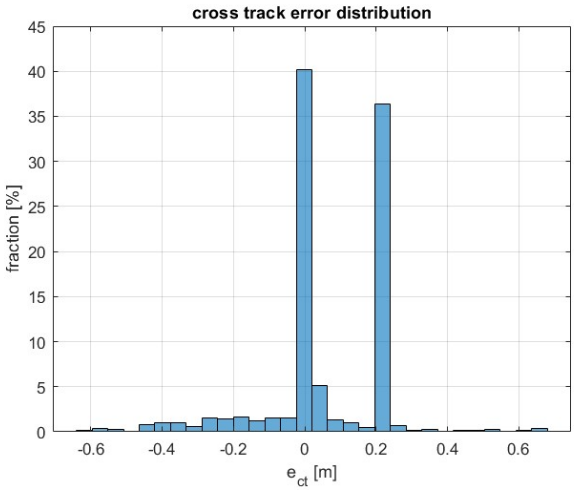

Figure 5.4: Distribution of the cross-track error.

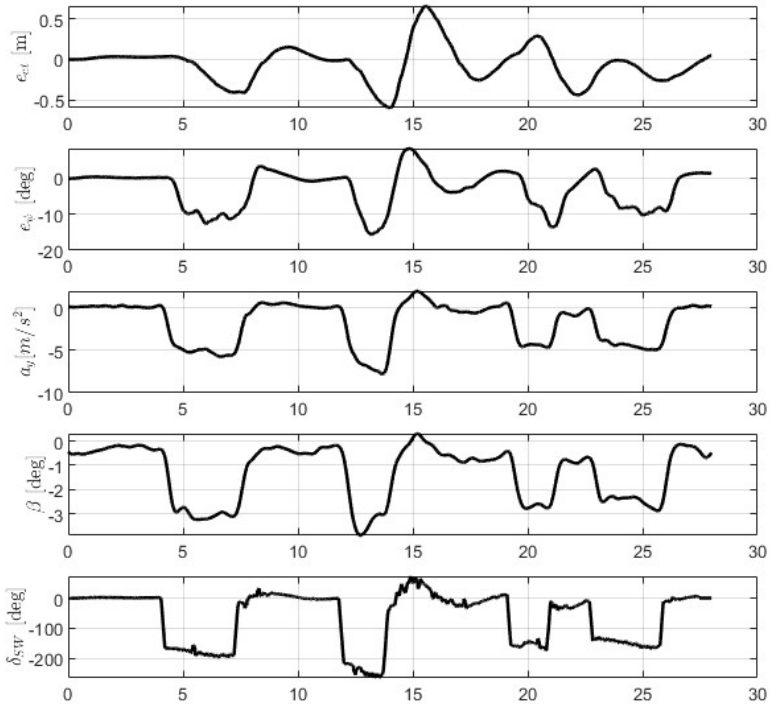

Figure 5.5: Main vehicle quantities.

6 Nonlinear Model Predictive Control (NMPC)

As declared, the NMPC is used to produce a database of inputs and outputs, which is then used for training the neural network. For this reason, there is no interest in the computational efficiency of the controller; if it is required, it can be heavy and slow. In fact, the goal is to optimise the controller settings for the best possible performance or almost the best one. In fact, it is interesting to note that if the NMPC performs extremely well, it produces an ideal dataset, in which the lateral error is always very low. However, this is not beneficial for training the NN, which is not exposed to a sufficiently wide range of possible scenarios.

The internal model used by the NMPC is very similar to the external model (described in Chapter 3). Even the tyre model used is the same. The only difference with respect to the external model is the simplification of the actuator dynamics; in fact, it is not possible to include the pure time delay in the internal model (in the external model, the actuator dynamics are expressed by exploiting Simulink blocks).

$$\tau \, \dot{u}_{act}(t) + u_{act}(t) = u_{\text{des}}(t) \tag{6.1}$$

The equation is a typical first-order equation used for approximating the actuator dynamics in the internal model. However, this is only a simplification, which will lead to trade-offs in terms of prediction capability. The subscript act stands for actuated, so it exists a delay between the desired and actuated command.

6.1 Problem definition and Cost function

The NMPC works by minimising a particular cost function along a prediction horizon. The controller has an internal model that receives initial conditions at a specified initial time. It then makes predictions using different combinations of command inputs. Finally, the controller chooses the set of command inputs that minimises the cost function. After applying only the first control input of the optimal sequence, the time horizon is shifted forward, the system states are updated with new measurements, and the optimisation problem is solved again. This process, known as the receding horizon strategy, ensures feedback and allows the controller to adapt to model uncertainties and external disturbances continuously.

Within the cost function, it is possible to include the model states, command inputs, and combinations of states with exogenous inputs. These last, also referred to as online or time-varying parameters, are quantities that are measured or provided to the controller and are not part of the system's state. They are treated as known (never predicted) sequences along the prediction horizon, i.e., they are not decision variables of the optimisation. For example, in the developed controller, some online parameters include the trajectory information (coordinates and curvature) and the cruise controller torque, as the NMPC does not compute these but instead enters them as known inputs/disturbances into the prediction model. Also, the derivative of the command inputs is included in the cost function. This is important in case it is necessary to limit the actuator dynamics; however, it primarily serves as a solution to the chattering that can occur in the signal.

Following, the optimal problem is formalised:

$$\begin{aligned}
\min_{\{x_k\}_{k=0}^{N},\{u_k\}_{k=0}^{N-1}} \quad & J = \ell_0(x_0,u_0,p_0) + \sum_{k=1}^{N-1} \ell(x_k,u_k,p_k) + \ell_N(x_N,p_N) \\
\text{s.t.} \quad & x_{k+1} = F_k(x_k,u_k,p_k), \qquad k=0,\dots,N-1, \\
& x_{min} \le x_k \le x_{max}, \qquad k=0,\dots,N, \\
& u_{min} \le u_k \le u_{max}, \qquad k=0,\dots,N-1.
\end{aligned}$$

$$\begin{aligned}
\ell_0(x_0,u_0,p_0) &= \tfrac{1}{2}\left(y_0 - y_{\text{ref},0}\right)^\top W_0 \left(y_0 - y_{\text{ref},0}\right), \\
\ell(x,u,p) &= \tfrac{1}{2}\left(y_k - y_{\text{ref},k}\right)^\top W \left(y_k - y_{\text{ref},k}\right), \\
\ell_N(x) &= \tfrac{1}{2}\left(y_e - y_{e,\text{ref}}\right)^\top W_e \left(y_e - y_{e,\text{ref}}\right),
\end{aligned}$$

$$y_0 = \begin{bmatrix} e_{\psi,0} \\ e_{ct,0} \\ \dot{e}_{\psi,0} \\ \dot{e}_{ct,0} \\ u_0 \\ \dot{u} \end{bmatrix}, \qquad y_k = \begin{bmatrix} e_{\psi,k} \\ e_{ct,k} \\ \dot{e}_{\psi,k} \\ \dot{e}_{ct,k} \\ u_k \end{bmatrix}, \quad k=1,\dots,N-1, \qquad y_e = \begin{bmatrix} e_\psi(x_N,p_N) \\ e_{ct}(x_N,p_N) \end{bmatrix}.$$

States, command inputs and online parameters are the following:

$i = F(front), R(rear)$, $j = L(left), R(right)$

$$x = \begin{bmatrix} X \\ Y \\ v_x \\ v_y \\ \dot{\psi} \\ \omega_{ij} \\ \delta_{i,\text{act}} \\ T_{em,ij,act} \end{bmatrix}, \qquad u = \begin{bmatrix} \delta_{i,des} \\ T_{\text{em,ij,des}} \end{bmatrix}, \qquad p = \begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \\ \psi_{\text{ref}} \\ K_{\text{ref}} \end{bmatrix}$$

Some clarifications are required regarding $\dot{u}$. This term is included only in the initial stage cost: ℓ_0, as the penalisation must be done only at the beginning of the prediction horizon. In fact, the central role of this term is related to preventing chattering. Nevertheless, it is possible to tune it in case it helps in giving the NMPC behaviour some desired characteristics. For example, the torque vectoring can be forced to be used mainly in the central part of the cornering (because the derivative is more penalised and the highest torque levels are reached only in the central part of the turn), reducing its usage in the other conditions, making the controller more robust in experimental scenarios. It is essential to note that the vehicle's stability is inherently incorporated into the NMPC, and these modifications do not compromise the vehicle's stability in any way, while also providing benefits in handling.

6.2 Solver and numerical aspects

The implementation of the nonlinear model predictive controller requires the definition of a mathematical model and the configuration of suitable numerical solvers. In this work, this process is carried out using CasADi and ACADOS. CasADi provides a symbolic environment in which the vehicle dynamics are defined in terms of states, inputs, and online parameters. The dynamics are provided in the implicit form:

$$f_{\text{impl}}(x, \dot{x}, u, p) \;=\; f_{\text{expl}}(x, u, p) \;-\; \dot{x}$$

. This comes from the need to use the implicit Runge-Kutta (IRK) integration within ACADOS. The advantages of IRK are related to a higher numerical stability, even for large integration steps, and to the management of stiff dynamics.
The continuous-time problem is discretised using a multiple shooting scheme. The prediction horizon is divided into twenty intervals (N = 20) with a non-uniform distribution obtained via a polynomial scaling function with exponent 2. This strategy concentrates more nodes near the beginning of the horizon, improving accuracy in the short term while keeping the overall computational burden moderate.

The cost function is formulated in the nonlinear least-squares form. The linear form does not allow the use of quantities that are not states or command inputs in the definition of the cost (so, it is not possible to use directly the path tracking errors, for example). The cost is evaluated separately at the first integration step, at the last one and in the middle. Therefore, there are three weighting matrices: W_0, W_e, and W. Consequently, it is possible to assign different weights to the three moments of the prediction horizon.

The resulting nonlinear problem is solved using a Sequential Quadratic Programming

(SQP) scheme. The solver adopted is the embedded one, HPIPM, in full condensing mode. Warm-starting of the QP solver is enabled to reduce computation time in closed-loop operation, and the maximum number of NLP iterations is set to 1500. Even if it can appear to be a very high number of iterations, and it is, it ensures that the solver always converges to a solution that satisfies the residual tolerance. It is worth noting that reaching the convergence tolerances is not mandatory; in some cases, this may not occur without affecting the control, but in rare instances, it can significantly impact the continuation of the controller action. Since computation time is not a requirement, it is preferable to maintain a high maximum number of iterations.
Convergence tolerances for complementarity, equality, inequality feasibility, and stationarity are set to 10^{-4}. To enhance robustness, a globalisation strategy based on a merit function with backtracking line search is applied, with sufficient descent conditions enforced to avoid divergence in the presence of nonlinearities or poor initial guesses.

Finally, ACADOS generates tailored C code for the solver, which is integrated into Simulink as a custom S-function block. This setup enables closed-loop simulations in which the solver receives reference trajectories, bounds, and online parameters as inputs, and outputs predicted states, control trajectories, cost values, and Karush–Kuhn–Tucker (KKT) residuals.

6.3 Simulation results

In this section, the simulation results are shown. It is possible to appreciate the results obtained using the most challenging path among the ones designed for testing the controllers. The manoeuvres are conducted at 30 km/h, which is almost the maximum vehicle speed allowed (due to software limitations). In general, from now on, all manoeuvres are conducted at 30 km/h. The speed during the manoeuvres is maintained at a constant level by a cruise controller.

The results are compared to a tailor-made PD+FF controller that is used to set a reference. It is imperative to emphasise that the controller, in its structure, is the one described in the previous chapter. Despite that, it is specifically tuned to perform well in this challenging manoeuvre. Representing, in this way, the best benchmark possible.

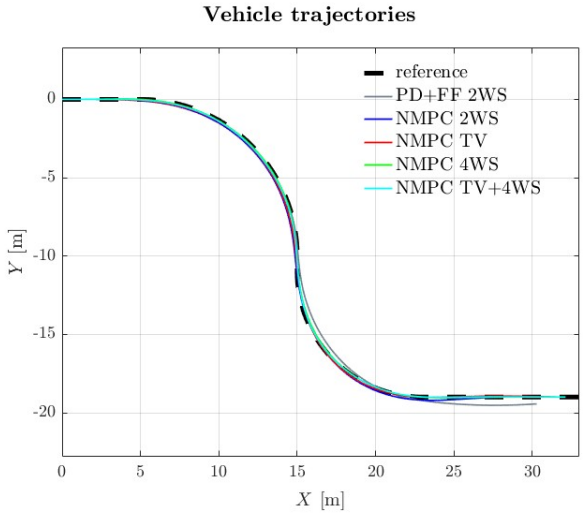

Figure 6.1: Vehicle trajectories and path reference.

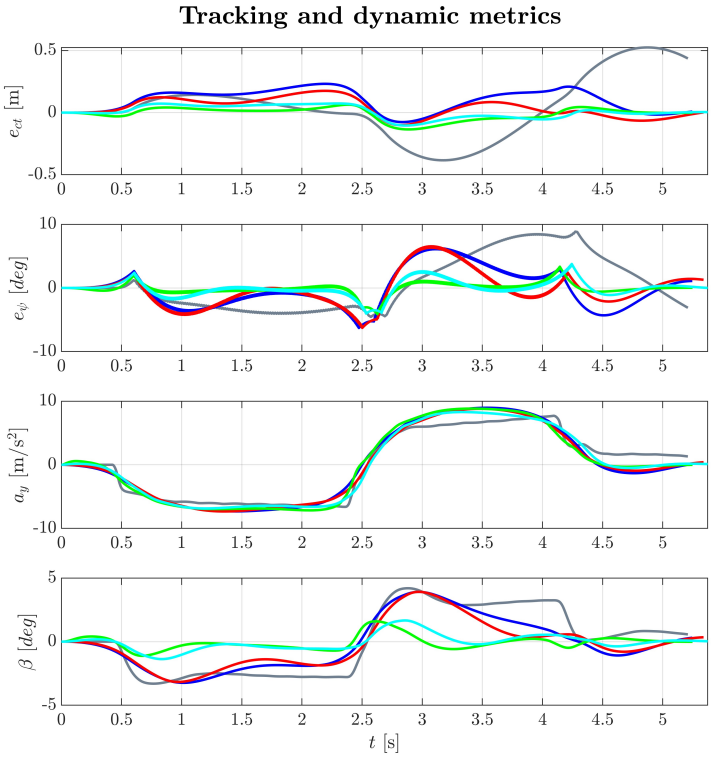

Figure 6.2: Tracking and dynamic quantities.

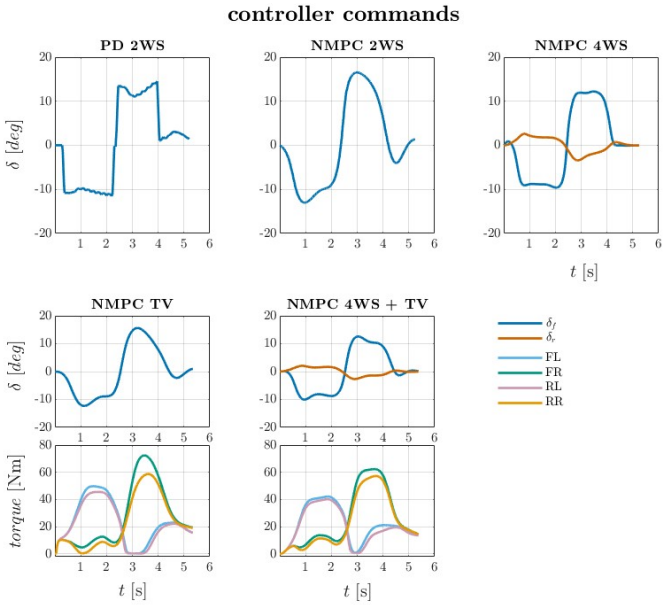

Figure 6.3: Command inputs of all controller configurations. The steering command is intended to be at the wheel.

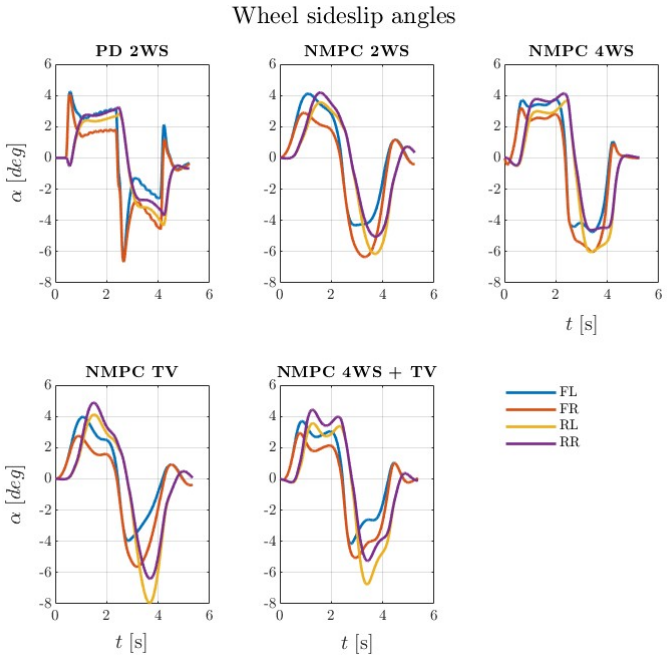

Figure 6.4: Wheel sideslip angles.

Examining the figures showing the results, it is possible to begin comparing the PD+FF controller with the NMPC one. The latter shows control commands (Figure 6.3) that are smoother and more progressive, while the PD+FF controller has sharper and more violent commands. The reason for this is that the steering command depends mainly on the feedforward component, which is directly proportional to the path curvature. This curvature has a staircase shape, as the path is composed of arcs of constant curvature; therefore, the steering command also has a shape that resembles a staircase. The NMPC, knowing the vehicle dynamics and, equally importantly, the actuator dynamics, as well as the path features, can provide a command that adapts to both vehicle behaviour and minimises the cross-track error, taking into account both current and future conditions, since it is predictive. Furthermore, the PD+FF controller performs very well in the first turn and is absolutely comparable to the NMPC controllers. However, since it is not predictive, it is not as effective at suddenly changing direction and ultimately realigning with the last straight line.

Passing to analysing the actuators' influence, the main focus must be on the change of direction. In fact, again, there are no particular differences among the configurations in the first turn. The significant difference is visible between 2.5 and 4.5 seconds; here, it is possible to appreciate the capability of the configurations to rotate the vehicle in the opposite direction suddenly. The TV configuration and the 4WS one (4WS is superior to TV) offer a higher capability to maintain vehicle inertia and change direction more rapidly. The configuration 4WS+TV is the best possible, but it is not much better than either 4WS or TV alone.

7 | Imitation learning

In this chapter, all the themes related to Imitation learning are tackled. The training procedure and the Network's main features are explained. Simulation and experimental outputs are provided and analysed.

7.1 NN and NMPC architectures

In this section, a comparison is provided between the control architecture of the NN and NMPC. As mentioned, the NN is trained based on the NMPC simulation data, a process known as Imitation Learning (IL). The IL, typically, consists of substituting the NMPC controller with a Neural Network that is capable of imitating it, so it receives the entire set of inputs (including the complete vehicle state) of the NMPC and, ideally, provides the same set of outputs (or a very similar one), with the difference that the NMPC is, computationally speaking, too heavy while, the NN is light and can be run in real time.

But, as explained in chapter 1, there are drawbacks, following a recall:

- The whole vehicle state must be measured and provided to the NN.
- The performance of the NN depends on the performance of the sensors. Therefore, the various sensor accuracies and delays can significantly impact the tracking performance. In the case presented in this work, it is challenging to model GPS delay, as it arises from the relatively large sample time (0.2 ms). Therefore, the delay does not originate from signal filtering; consequently, it is difficult to compensate for this.
- The inputs fed to the neural network must match those used by the NMPC. However, when it comes to reference quantities, providing them in the global

reference frame can be inefficient, since very similar driving situations may correspond to widely different absolute values (e.g., global position and orientation).

To overcome these disadvantages, an IL is proposed that trains the NN using the same NMPC outputs but a different, reduced set of inputs, evaluated in parallel with the NMPC data during database production. Therefore, while the NMPC operates in the Simulink environment, the set of inputs to be fed to the NN is generated in parallel. Figure 7.1 is a schematic that shows the training procedure and what was explained soon before about the set of inputs produced for the training of the NN.

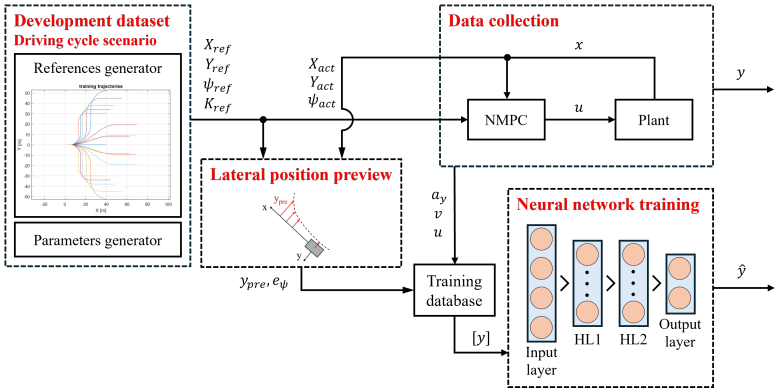

Figure 7.1: Controller training scheme.

The training quantities provided to the NN are:

- vehicle lateral acceleration, to include information about the vehicle lateral dynamics.
- vehicle longitudinal speed, since the command inputs, for the same path, change according to the vehicle speed (even if the manoeuvres are conducted at 30 km/h, the speed is not always exactly constant).
- heading error.
- lateral position preview.
- command inputs

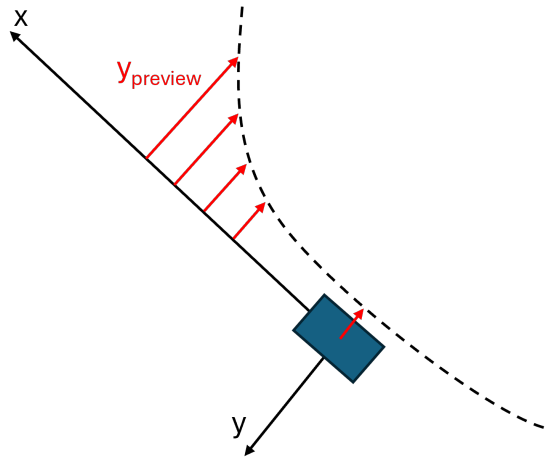

Figure 7.2: Controller training scheme.

Some words are needed to explain the meaning of the lateral position preview. Figure 7.2 provides a schematic of this quantity. The lateral position preview represents the lateral distance (i.e., the y-coordinate) of twenty points along the reference trajectory, computed with respect to the vehicle's coordinate system, over a distance of 10 meters ahead. The vehicle's lateral error is implicitly included in the road preview, as it corresponds to the first point of the lateral preview vector.

7.2 Policy model and training database

The control policy is implemented as a fully connected feed-forward neural network. The architecture uses multiple hidden layers with element-wise nonlinearities (e.g., `tansig` for hidden layers). The output activation function can be pure linear or tanh, depending on which guarantees the best performance during the simulation validation;

The training database is organised as a single matrix partitioned column-wise into features and targets. A pre-processing applies either min–max scaling to $[-1,1]$ (naturally paired with tanh outputs).
Data generation spans straight segments and constant radius curves with broad curvature coverage and an explicit emphasis on challenging regimes. All the paths are made alternating between two consecutive turns with opposite directions. There are paths with sudden changes in direction and others with straight segments between successive turns. The reference paths used are shown in Figure 7.3. The database is further enriched with off-nominal conditions such as lateral force disturbances, yaw-moment injections, and initial pose misalignments (an example is

provided by Figure 7.4) to increase the density of corrective examples and improve robustness.

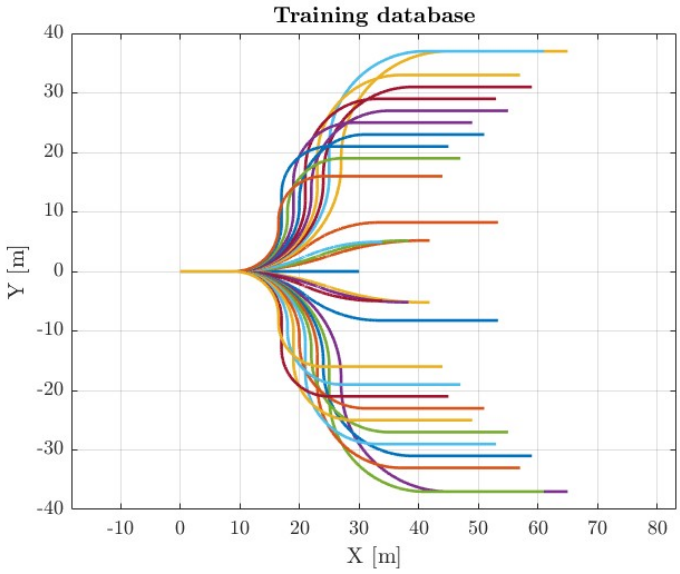

Figure 7.3: reference paths in the database.

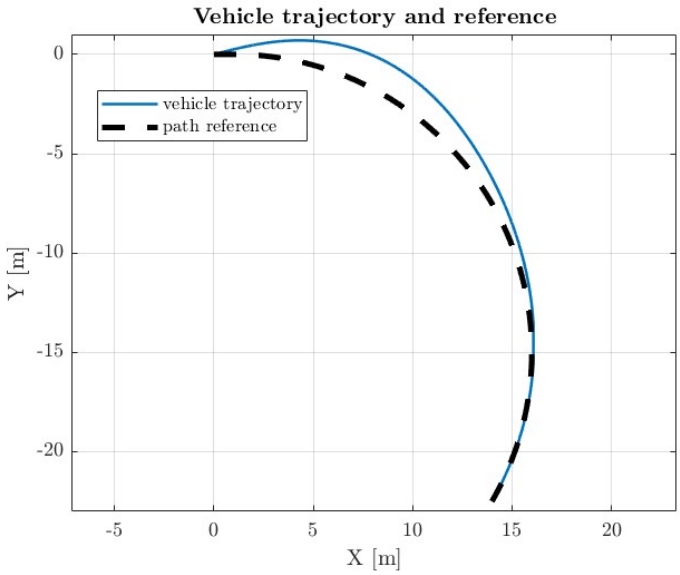

Figure 7.4: Vehicle trajectory during a manoeuvre with an initial vehicle heading error.

For reproducibility, the dataset is globally shuffled with a fixed random seed and split by indices into training, validation, and test sets (e.g., about 65%/30%/5%). The learning objective minimizes mean squared error on the command vector with L2 regularization to enhance generalization and numerical conditioning, optimization leverages an efficient second-order routine suitable for medium-scale FFNNs (e.g., Levenberg–Marquardt) with a cap on epochs and patience-based early stopping driven by the validation set, and available compute resources are detected at runtime so that multi-core pools and GPUs can be exploited to accelerate training.

7.3 Simulation results

In this section, the simulation results are shown. The procedure for testing is precisely the same as that used for testing the NMPC.

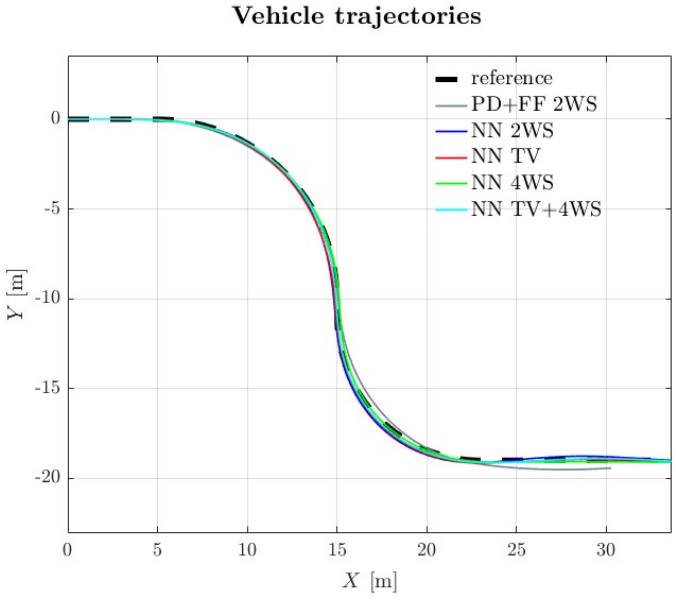

Figure 7.5: Vehicle trajectories and path reference.

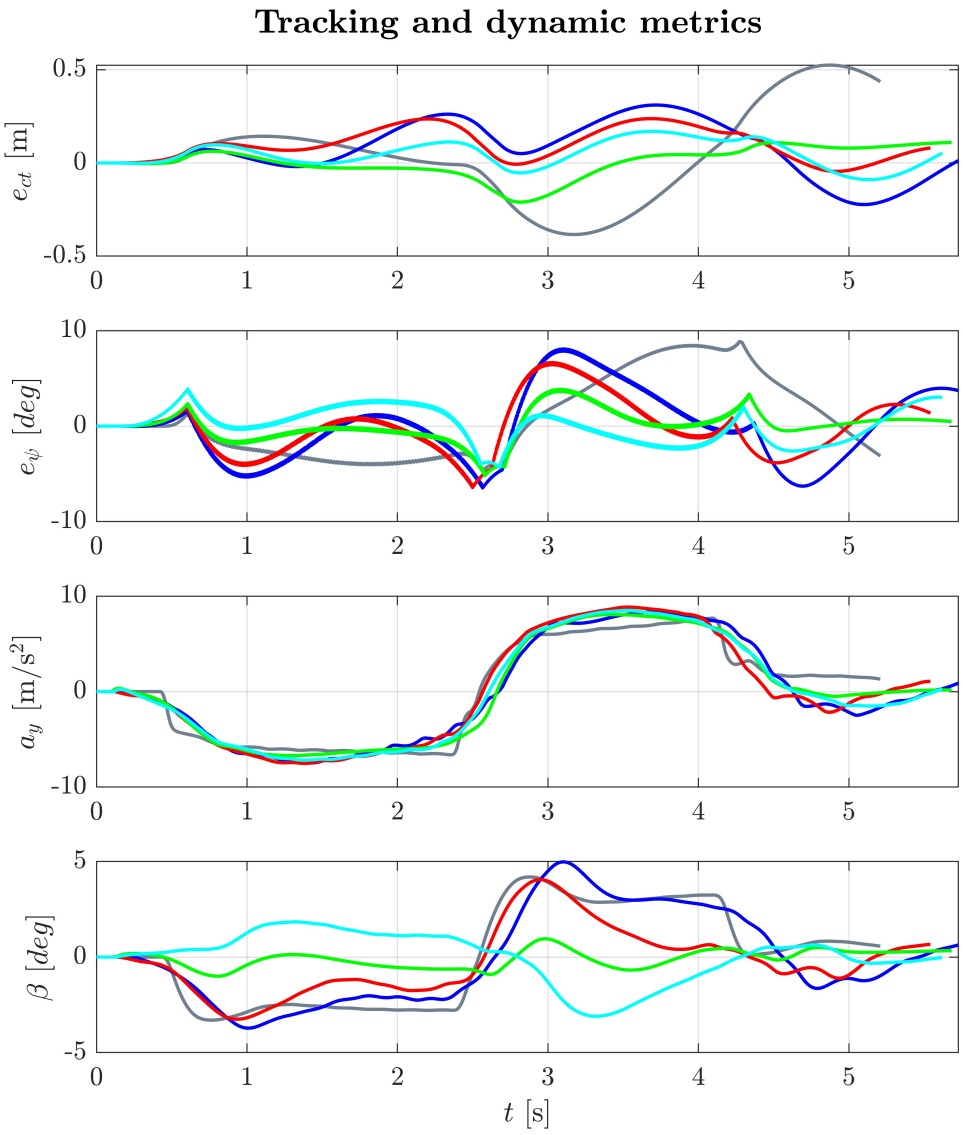

Figure 7.6: Tracking and dynamic quantities.

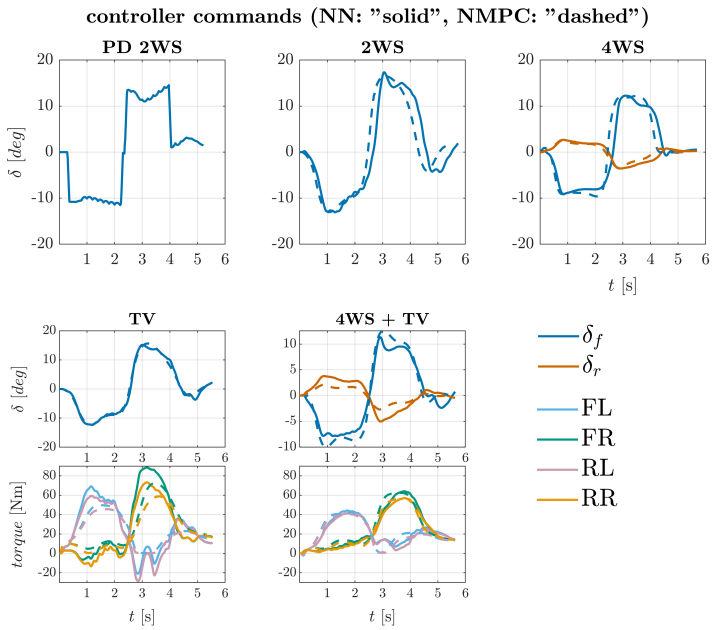

Figure 7.7: Command inputs of all controller configurations.

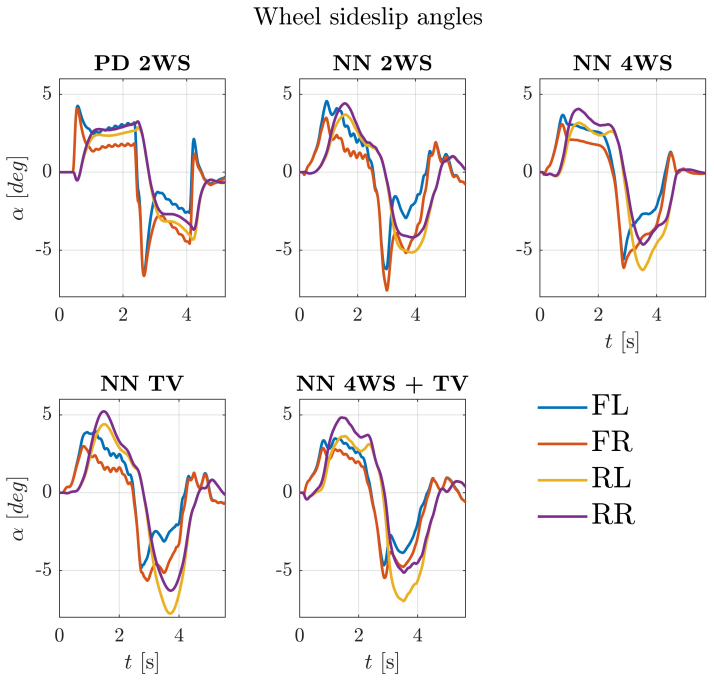

Figure 7.8: Wheel sideslip angles.

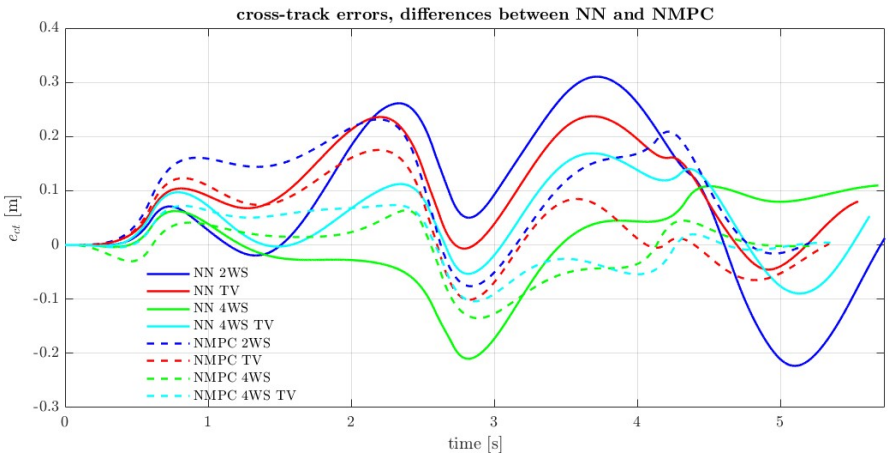

Figure 7.9: Comparison between the cross-track errors of NN and NMPC.

Upon examining the figures, several key considerations are made. The first one is that the neural networks are capable of effectively imitating the NMPC. So, the training procedure, including the usage of a reduced input set, is effective. Figure 7.7 shows how the NN commands are very similar to the NMPC ones. A further consideration is that, even though the commands are very similar, the performances of the NMPC are consistently better than those of the NN; nevertheless, they remain similar and can be considered acceptable. Figures 7.6 and 7.9 show this. Focusing on Figure 7.9, a consideration must be made, NN and NMPC do not behave exactly in the same way, but the shapes of the signals are comparable, and the controller ranking is maintained too.

7.3.1 Robustness analysis

In this section, two types of robustness analysis are shown: A Montecarlo analysis and a simulation manoeuvre using a path including out-of-nominal scenarios.

The Monte Carlo analysis involves conducting numerous simulations (approximately 250 for each configuration) in which the path remains constant while some vehicle parameters are randomly altered to observe their impact on the controller's performance. The parameters chosen to be varied are:

- adherence coefficient μ.
- cornering stiffness scaling factor k_y.
- longitudinal slip stiffness scaling factor k_x.
- vehicle mass Δ_m.

- steering actuator time coefficient τ_{str}

Therefore, more than 200 samples have been created, including combinations of these parameters that vary by +/-20% (with a Gaussian distribution), and all have been tested. The path chosen for this test is the S-shape one with a lower difficulty coefficient, as the idea is to test the controller's performance with a vehicle model that has been altered, rather than increasing the difficulty related to path shape. Nevertheless, the path has to have a non-negligible level of difficulty; otherwise, it helps to compensate for the vehicle model alterations. Figure 7.10 helps in visualising how the samples are made. The parameters are Gaussian distributed, so there are many parameters close to the nominal conditions and fewer that are far from them. Even though most of the samples are not too far from the nominal conditions, the entire domain is well-explored.

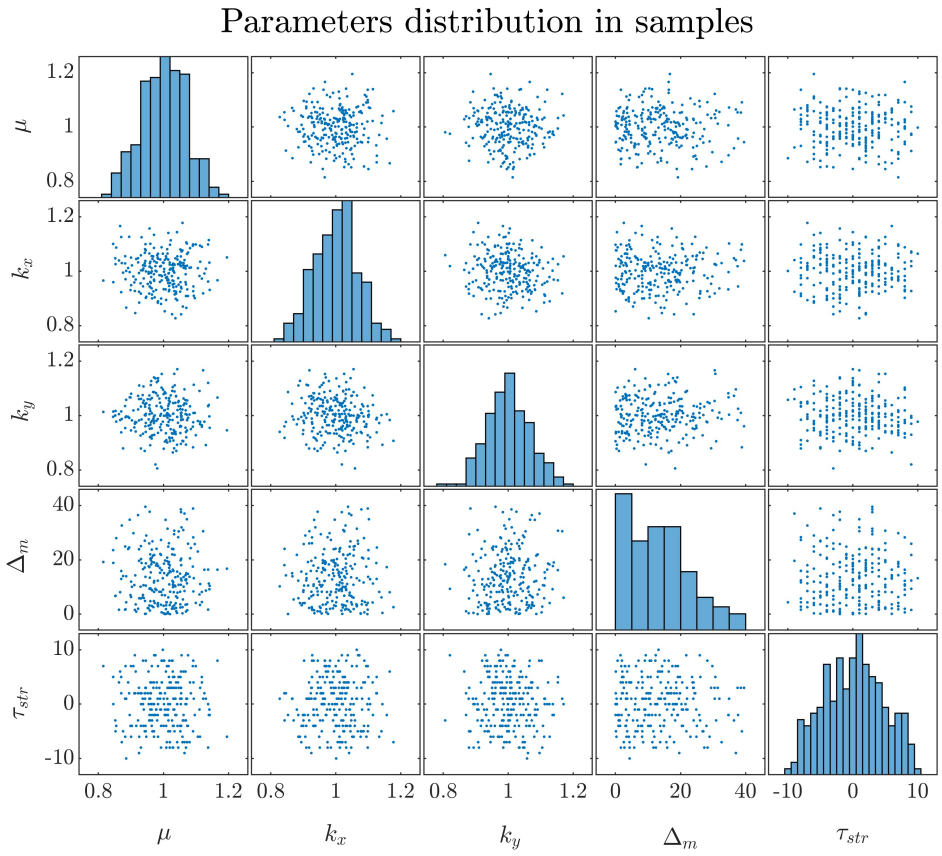

Figure 7.10: Figure showing how the parameters vary within the samples and the parameters distribution.

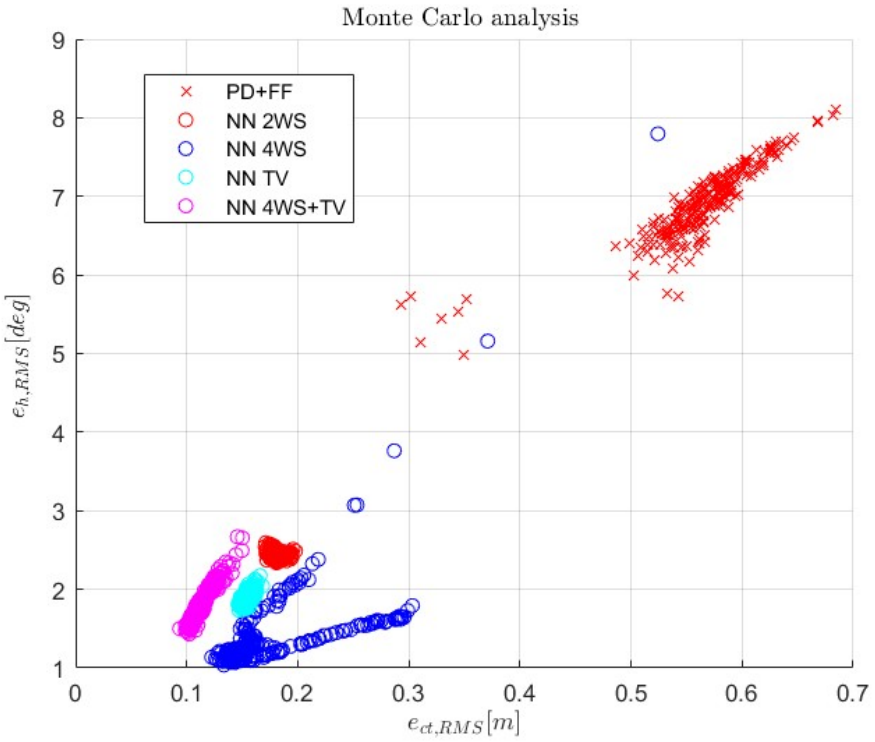

Figure 7.11: Outputs of the Montecarlo analysis.

Figure 7.11 shows the results of the analysis. It is possible to make two considerations. The first is related to the fact that neural networks are robust and their performance does not degrade in out-of-nominal conditions of the vehicle, compared to the PD+FF controller. The second consideration is related to the fact that, even though the PD+FF controller is tuned to achieve the best performance in the manoeuvre, it exhibits both lower performance compared to the NN and lower robustness, as the cloud of points is wider compared to those of the NN. Except for NN 4WS, which appears to be more sensitive to the parameter alteration.

The second analysis aims to show the effectiveness of the training procedure. Upon reviewing the training database, it appears to be insufficient to describe all possible scenarios. Firstly, the trajectories are limited to around forty (although they are symmetric, including the same turns at left and right, so the paths are 20+20). Secondly, they consist only of short straight segments and constant-radius turns. Furthermore, the inclusion of off-nominal scenarios is limited to a few cases (approximately 33% of the database) in which the vehicle starts misaligned or external disturbances are applied. Therefore, it appears to be only adequate in cases where the NN operates in scenarios similar to those present in the database.

Figure 7.12 shows how the NN 2WS (but also all the others, even with improved performances. The NN 2WS has been chosen since it is the less performing) can tackle with very high performances a path in which turns have very different characteristics. In particular, there are more than two turns in sequence, with different curvatures, and turns at linearly increasing curvatures (clothoids), scenarios that are absent from the database. It is possible to conclude that the training procedure is effective in producing Neural networks that are performing, robust and usable in various scenarios.

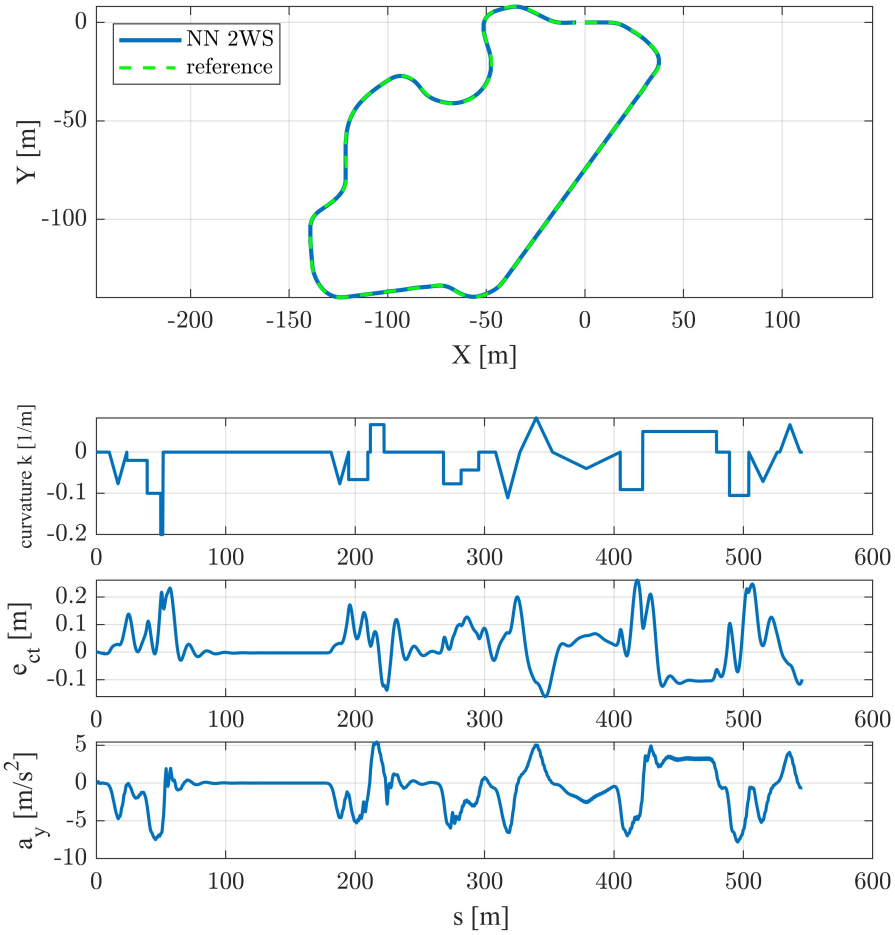

Figure 7.12: Maneouvre including multiple off-nominal condition scenarios.

7.4 Experimental results

In this section, the experimental results are presented and discussed. The data related to the classical controllers are taken from another thesis work focused on designing classical performing controllers to serve as a benchmark. The data reported for the second circuit are relative to the second turn only, the one at high lateral acceleration, to avoid redundancy, since the remaining part of the circuit is exactly equal to the first circuit.

Before commenting on the results, it is important to clarify the rationale behind the NN+PD configurations. As the name suggests, these setups combine the Neural Network controller with a PD controller whose gains were experimentally tuned. From the analysis of the experimental outputs — and in particular from the observation of the NN behaviour — it appears that some networks exhibiting progressive and smooth steering actions tend to tolerate or even benefit from the corrective contribution of a parallel PD controller. This behavior was also confirmed in simulation. Consequently, the NN 2WS and NN 4WS configurations proved to work properly when coupled with a PD controller in parallel. In contrast, the NNs associated with torque-vectoring configurations generated steering signals that were incompatible with the use of a PD controller in parallel. Nevertheless, the gains associated with the PD controller are extremely small. As a result, the PD action becomes beneficial only when the cross-track error is large. Conversely, under nominal conditions, the PD controller neither worsens the performance nor provides any significant improvement.

Some observations are possible. The first concerns the cross-track error obtained on circuit 1A, as reported in Figure 7.14 and Table 7.1. The differences among the various configurations — both classical and NN-based — are rather limited. This outcome is reasonable, since circuit 1A represents a relatively undemanding scenario where all controllers, including the conventional PD-based ones, are capable of maintaining accurate trajectory tracking. Consequently, the KPIs do not highlight a clear hierarchy among the controllers, nor a distinct separation between neural and classical approaches. Under these nominal conditions, the advantages of employing multiple actuators, such as four-wheel steering or torque vectoring, remain marginal. A closer look at Figure 7.14 and Table 7.1, however, reveals an interesting secondary effect. While the tracking accuracy is comparable across all configurations, the NN-based controllers generally exhibit lower RMS steering activity — both at the front and rear axles — and, when available, lower steering rate RMS values. This suggests that the NNs, thanks to their predictive capability, generate smoother and more progressive control actions. They do not rely solely on instantaneous error correction, but rather exploit knowledge of the upcoming path segment.

As a result, they achieve similar levels of precision as classical controllers, while demanding less steering effort and potentially improving overall driving smoothness.

The results reported in Figure 7.17 and Table 7.2 refer to the sharp corner of circuit 1B, characterized by higher lateral acceleration and a more demanding vehicle dynamics condition. In this case, the differences among controllers become much more evident compared to circuit 1A. The NN-based controllers achieve markedly lower cross-track and heading errors, with reductions of up to 68% in the maximum lateral deviation and more than 60% in its RMS value. A similar improvement is observed for the sideslip angle, whose RMS decreases by approximately 70% in the best NN configurations. These results clearly indicate that the Neural Networks are capable of maintaining a more stable and precise vehicle attitude under highly nonlinear conditions. Moreover, the steering activity is consistently reduced. Both the RMS steering angle and steering rate decrease significantly, up to 25–30% compared to the baseline PD+FF 2WS, confirming that the NN controllers achieve smoother and more efficient control actions. The most performing configurations are those exploiting multi-actuation, particularly the NN 4WS and NN+PD 4WS, which combine the predictive behavior of the network with the extended vehicle actuation authority.

In summary, while the differences in circuit 1A were marginal, circuit 1B highlights the actual advantage of the NN-based approach. When the maneuver involves strong lateral dynamics, the Neural Networks provide superior tracking accuracy, improved stability, and reduced control effort — thus demonstrating their capability to effectively generalize and to exploit their predictive nature in highly dynamic scenarios.

A further confirmation of the trends discussed above can be observed in the distribution of the cross-track error. As shown in Figures 7.15 and 7.18, the error distributions obtained on circuit 1A are nearly identical for all controllers, confirming that under mild conditions the differences are marginal. Conversely, in circuit 1B the Neural-Network-based controllers exhibit narrower and more centered histograms, indicating both reduced tracking error and improved consistency. This suggests that, in highly dynamic turns, the NNs not only decrease the mean error but also stabilize its instantaneous variability, leading to smoother and more predictable control behavior.

A final comment can be provided speaking about the usage of the rear steering axle. In fact, the NN uses the rear steering axle much more than the classical controllers. This brings to an impressive reduction of the vehicle sideslip angle. In particular, about the configuration NN 4WS, the curve representing the β signal is almost flat.

This reduction is not visible in the classical controllers exploiting the rear steering axle. The reasons are mainly related to a much reduced rear steering activity.

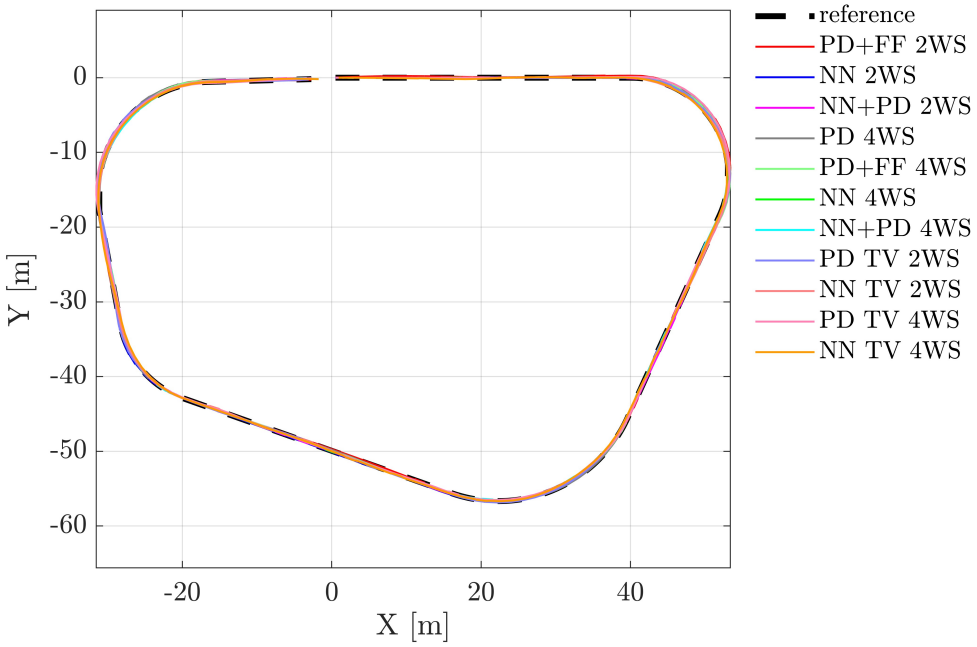

Figure 7.13: Trajectories measured during experimental tests on circuit 1A.

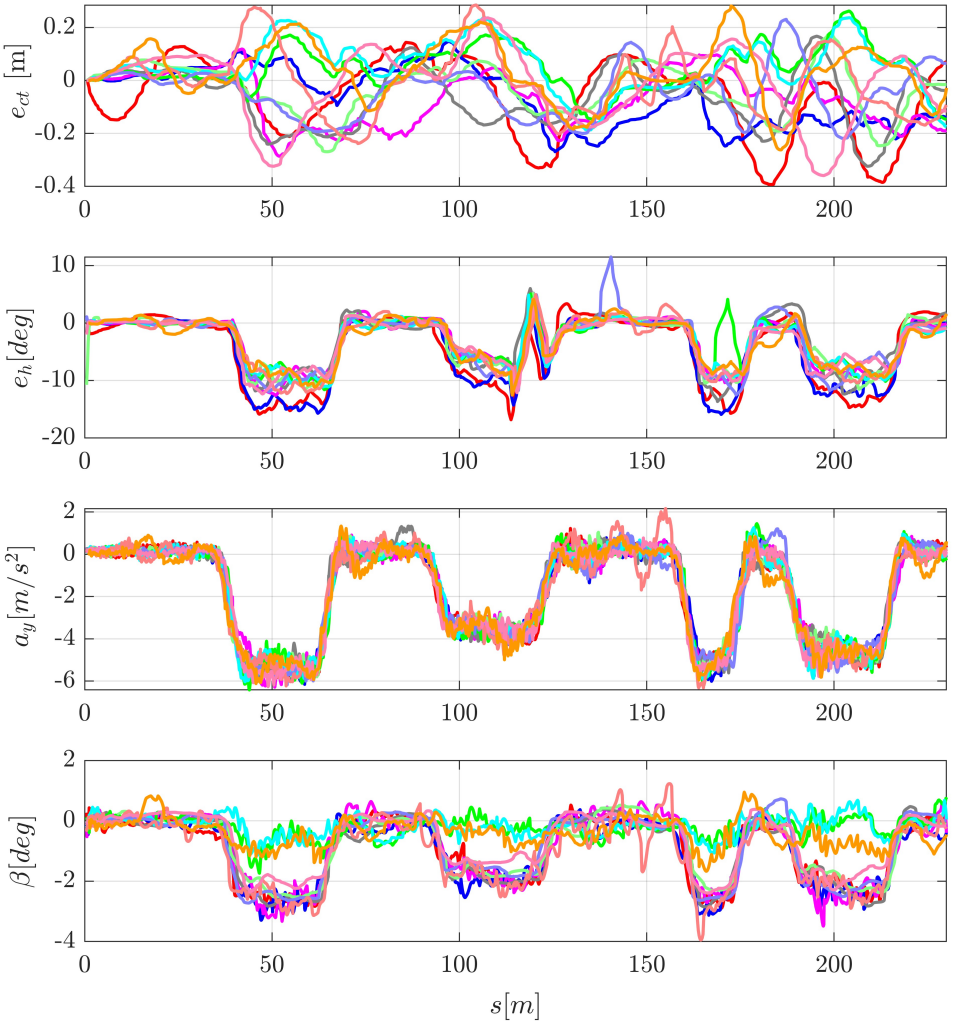

Figure 7.14: Tracking and dynamic quantities measured during experimental tests on circuit 1A.

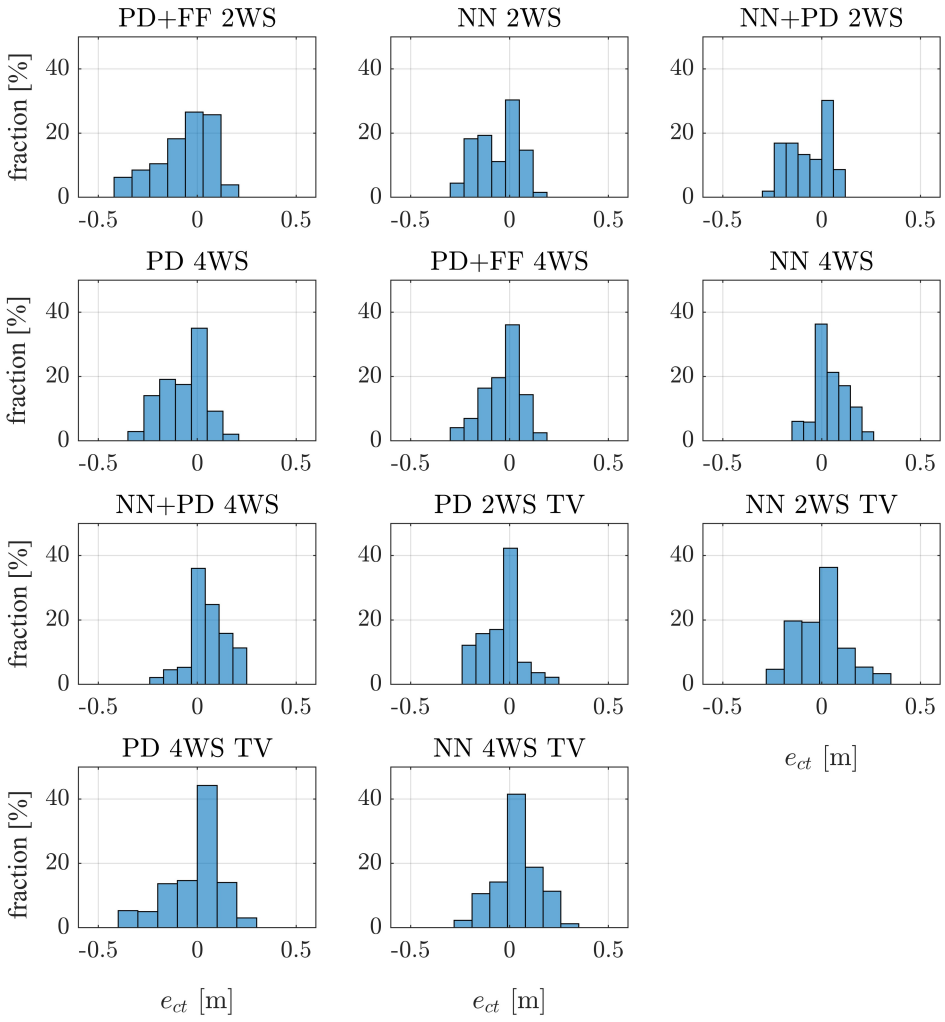

Figure 7.15: Distribution of the cross-track error, in percentage. experimental tests on circuit 1A.

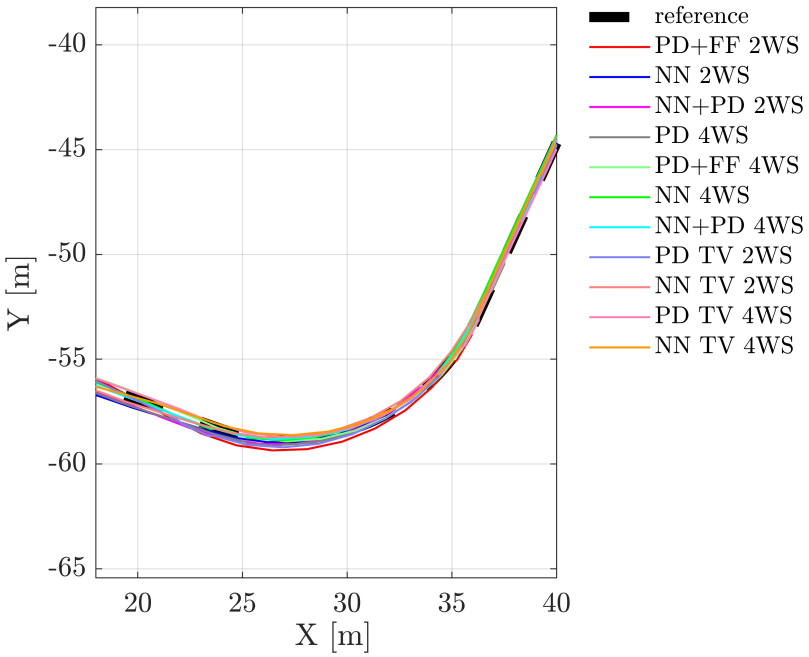

Figure 7.16: Trajectories measured during experimental tests on circuit 1B, results relative to the high lateral acceleration turn.

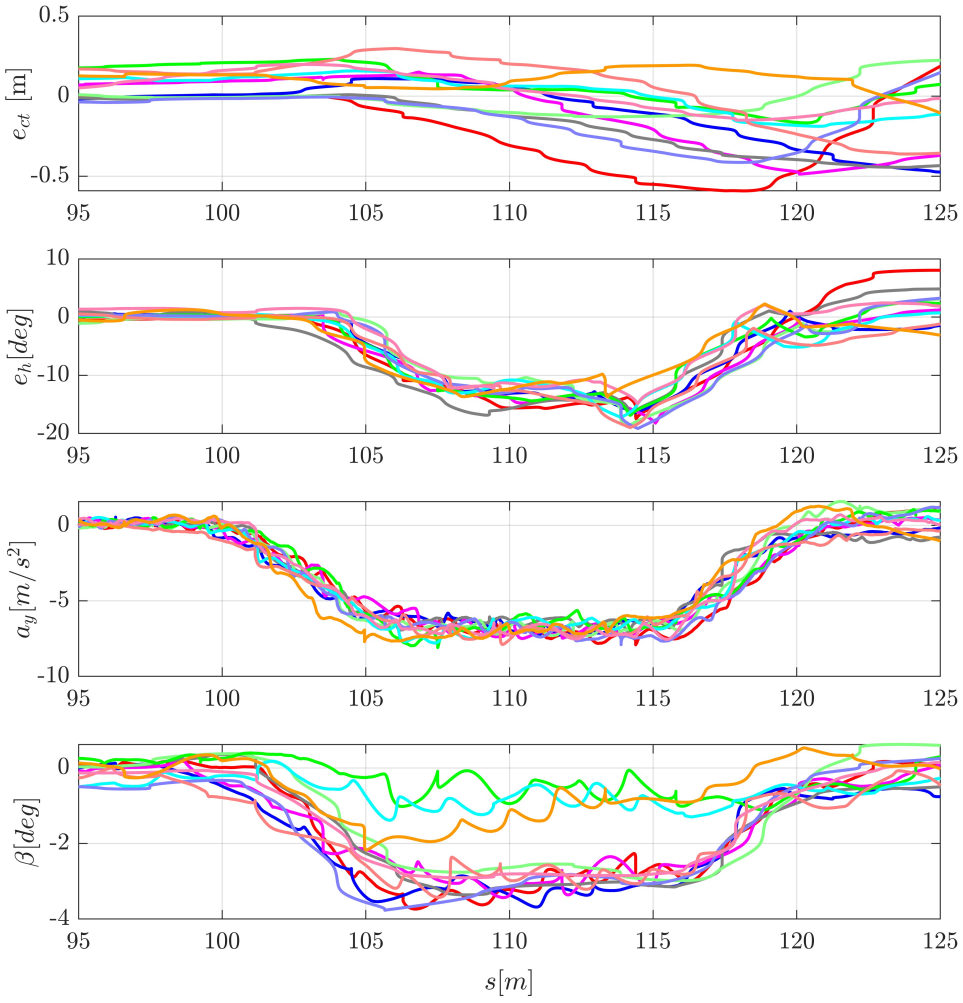

Figure 7.17: Tracking and dynamic quantities measured during experimental tests on circuit 1B, results relative to the high lateral acceleration turn.

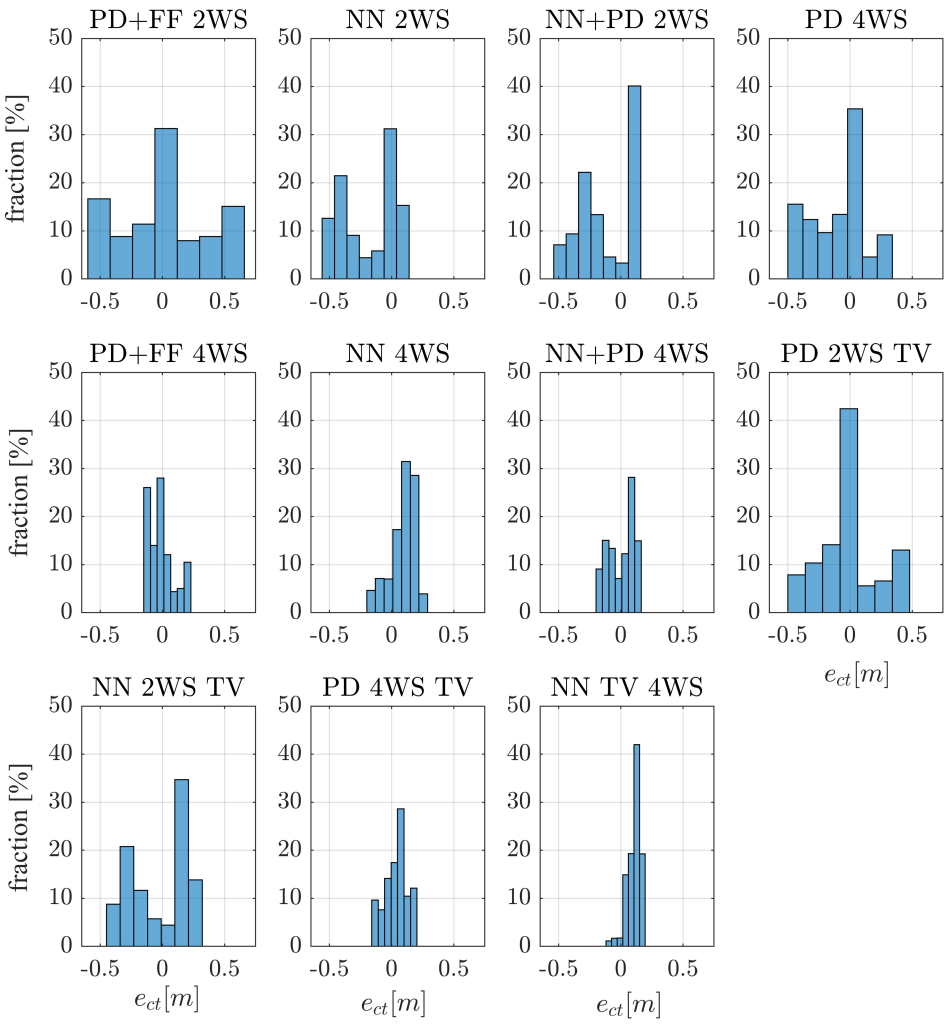

Figure 7.18: Distribution of the cross-track error, in percentage. experimental tests on circuit 1B, results relative to the high lateral acceleration turn.

controller	MAX e_{ct} [m]	RMS e_{ct} [m]	MAX e_h [deg]	RMS e_h [deg]	MAX β [deg]	RMS β [deg]	RMS δ_f [deg]	RMS δ_r [deg]	RMS δ_{PD} [deg]	RMS $\dot{\delta}_f$ [deg/s]	RMS $\dot{\delta}_r$ [deg/s]
PD+FF 2WS	0.393 (+0%)	0.157 (+0%)	16.81 (+0%)	7.51 (+0%)	2.89 (+0%)	1.37 (+0%)	94.2 (+0%)	-	-	93 (+0%)	-
PD TV 4WS	0.363 (-8%)	0.126 (-20%)	11.80 (-30%)	4.53 (-40%)	2.09 (-28%)	0.91 (-34%)	87.4 (-7%)	5.8	-	90 (-4%)	12
PD 4WS	0.330 (-16%)	0.120 (-23%)	13.10 (-22%)	4.99 (-34%)	2.83 (-2%)	1.30 (-5%)	93.5 (-1%)	4.6	-	91 (-2%)	12
NN+PD 2WS	0.286 (-27%)	0.120 (-24%)	12.58 (-25%)	5.12 (-32%)	3.49 (+20%)	1.40 (+2%)	92.8 (-1%)	-	-	87 (-7%)	-
NN TV 2WS	0.285 (-27%)	0.118 (-25%)	12.26 (-27%)	5.27 (-30%)	3.95 (+36%)	1.47 (+7%)	92.6 (-2%)	-	-	97 (+4%)	-
NN 4WS TV	0.282 (-28%)	0.113 (-28%)	12.54 (-25%)	4.95 (-34%)	1.67 (-42%)	0.61 (-56%)	74.5 (-21%)	20.6	-	87 (-7%)	25
PD+FF 4WS	0.272 (-31%)	0.094 (-40%)	11.30 (-33%)	4.72 (-37%)	2.14 (-26%)	0.99 (-28%)	90.2 (-4%)	6.1	-	89 (-5%)	12
NN 2WS	0.269 (-32%)	0.123 (-21%)	15.89 (-6%)	7.33 (-2%)	3.29 (+14%)	1.45 (+6%)	94.6 (+0%)	-	-	90 (-3%)	-
NN 4WS	0.263 (-33%)	0.093 (-41%)	11.58 (-31%)	4.73 (-37%)	1.74 (-40%)	0.40 (-71%)	72.6 (-23%)	24.1	-	85 (-9%)	27
NN+PD 4WS	0.242 (-38%)	0.109 (-31%)	11.86 (-29%)	4.91 (-35%)	1.14 (-61%)	0.38 (-73%)	71.8 (-24%)	24.4	-	84 (-10%)	26
PD TV 2WS	0.236 (-40%)	0.097 (-38%)	12.10 (-28%)	4.93 (-34%)	2.69 (-7%)	1.24 (-10%)	91.9 (-2%)	-	-	92 (-2%)	-

Table 7.1: Table showing the Key Performance Indicators (KPIs) obtained from the experimental tests carried out on circuit 1A.

controller	MAX e_{ct} [m]	RMS e_{ct} [m]	MAX e_h [deg]	RMS e_h [deg]	MAX β [deg]	RMS β [deg]	RMS δ_f [deg]	RMS δ_r [deg]	RMS δ_{PD} [deg]	RMS $\dot{\delta}_f$ [deg/s]	RMS $\dot{\delta}_r$ [deg/s]
PD+FF 2WS	0.592 (+0%)	0.317 (+0%)	17.50 (+0%)	9.30 (+0%)	3.73 (+0%)	2.12 (+0%)	130.3 (+0%)	-	-	165 (+0%)	-
NN+PD 2WS	0.487 (-18%)	0.245 (-23%)	18.21 (+4%)	8.71 (-6%)	3.43 (-8%)	1.97 (-7%)	118.2 (-9%)	-	-	137 (-17%)	-
NN 2WS	0.475 (-20%)	0.201 (-37%)	16.42 (-6%)	7.84 (-16%)	3.68 (-1%)	2.20 (+4%)	115.7 (-11%)	-	-	138 (-16%)	-
PD 4WS	0.445 (-25%)	0.240 (-24%)	16.85 (-4%)	9.17 (-1%)	3.36 (-10%)	2.11 (+0%)	119.1 (-9%)	6.7	-	114 (-31%)	52
PD TV 2WS	0.414 (-30%)	0.201 (-36%)	19.14 (+9%)	8.51 (-8%)	3.77 (+1%)	2.26 (+6%)	122.5 (-6%)	0.2	-	109 (-34%)	1
NN TV 2WS	0.362 (-39%)	0.213 (-33%)	18.98 (+8%)	8.38 (-10%)	3.43 (-8%)	1.99 (-6%)	113.2 (-13%)	-	-	138 (-16%)	-
NN 4WS	0.229 (-61%)	0.138 (-56%)	16.87 (-4%)	8.08 (-13%)	1.11 (-70%)	0.52 (-76%)	99.2 (-24%)	26.9	-	144 (-12%)	39
PD+FF 4WS	0.223 (-62%)	0.094 (-70%)	18.75 (+7%)	8.02 (-14%)	2.98 (-20%)	1.85 (-13%)	119.3 (-8%)	8.6	-	111 (-33%)	15
PD TV 4WS	0.198 (-67%)	0.119 (-62%)	14.88 (-15%)	6.90 (-26%)	2.91 (-22%)	1.90 (-10%)	110.8 (-15%)	8.6	-	107 (-35%)	14
NN TV 4WS	0.192 (-67%)	0.121 (-62%)	13.71 (-22%)	7.06 (-24%)	2.17 (-42%)	0.94 (-56%)	129.4 (-1%)	27.76	-	156 (-5%)	41
NN+PD 4WS	0.189 (-68%)	0.116 (-63%)	17.22 (-2%)	7.66 (-18%)	1.38 (-63%)	0.73 (-65%)	95.4 (-27%)	25.0	-	125 (-24%)	38

Table 7.2: Table showing the Key Performance Indicators (KPIs) obtained from the experimental tests carried out on circuit 1B, results relative to the high lateral acceleration turn.

7.4.1 Computational time

Looking at Figure 7.19, it is possible to observe that NN requires a CPU time that is orders of magnitude lower than the NMPC one; furthermore, the NMPC time tends to increase by increasing the number of actuators, which is not true for the NN.

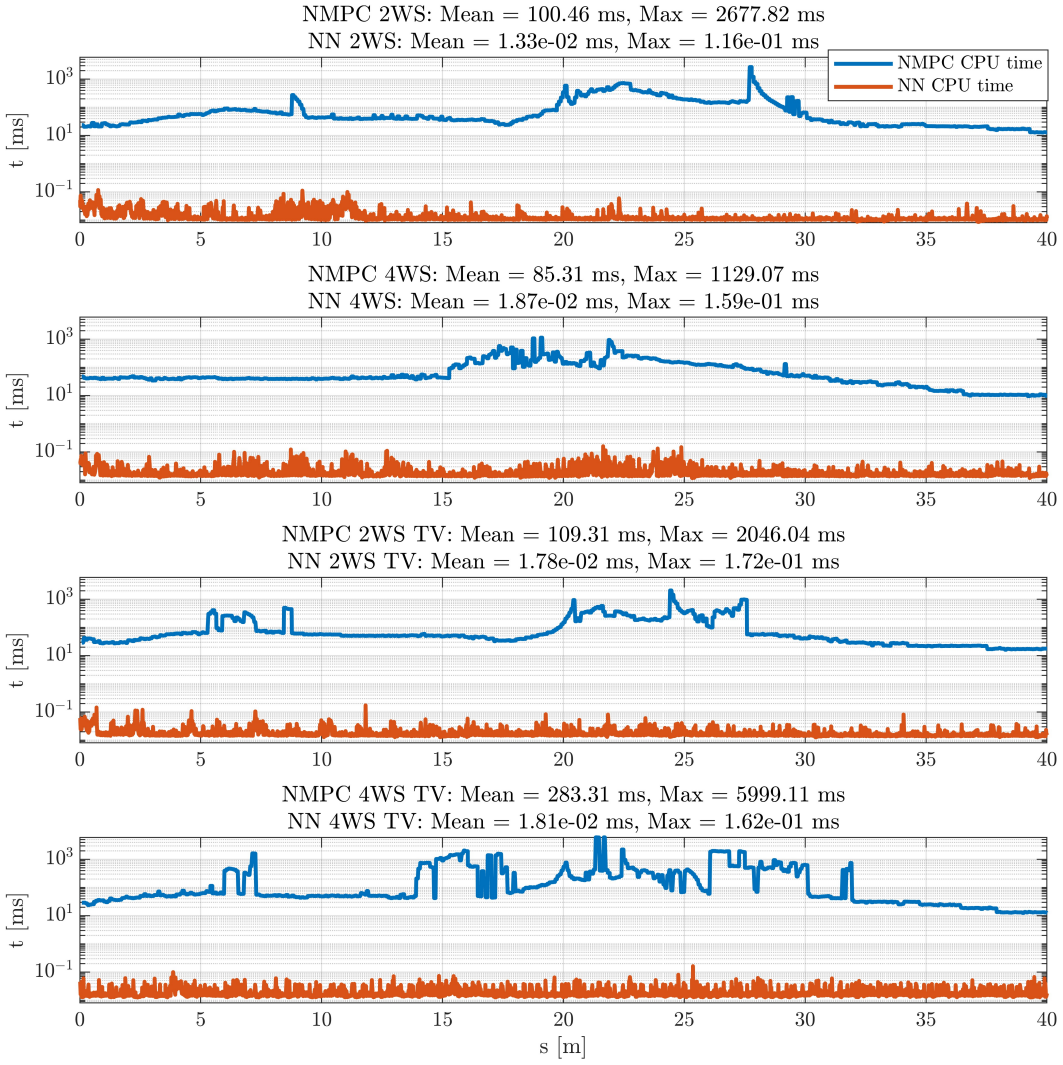

Figure 7.19: CPU time required by controllers.

8 | Conclusions

In conclusion, this thesis demonstrates how the integration of model-based and data-driven techniques can provide an effective solution to the path-tracking problem in vehicles. The work combines Nonlinear Model Predictive Control with Neural Networks in a way that enables high performance while significantly reducing computational effort, a key challenge in real-time applications.

A first novelty of this work is the systematic comparison of different actuator configurations—ranging from the standard two-wheel steering to four-wheel steering and torque vectoring—within a single framework, supported by experimental evidence. This approach enabled the highlighting, fairly and homogeneously, of the advantages and limitations of each configuration, providing a clear benchmark across multiple scenarios.

A second important contribution is the creation of a compact yet well-structured training dataset, generated directly from NMPC simulations. This enabled the training of neural networks that could replicate optimal control actions with remarkable accuracy and very low computational cost. Unlike many existing approaches, the training procedure was not limited to simply reproducing the NMPC output. Still, it was carefully designed to embed the effects of sensor and actuator delays directly into the dataset. In practice, instead of compensating for delays through additional algorithms, an "engineered dataset" was created that already incorporated these delays. This choice significantly simplifies the final controller architecture and ensures that the Neural Network learns to deal with realistic operating conditions from the very beginning.

Another novelty lies in the way imitation learning was applied. The Neural Network policies were trained using a minimal set of input features, carefully selected to capture the essential information for path tracking without introducing redundancy. This not only reduced the training complexity but also ensured faster inference times and greater generalisation capability, making the approach more practical

for real-time implementation.

Overall, the thesis demonstrates that combining predictive control with learning-based methods is a promising approach for vehicle path tracking. The Neural Network policies trained in this work retain the quality of the NMPC actions while being much faster to compute, and they do so without requiring additional mechanisms to handle sensor delays or actuator dynamics. This confirms that data-driven approaches, when properly integrated with model-based strategies, can play a crucial role in developing reliable, efficient, and practical solutions for autonomous driving applications.

8.1 Future developments and possibilities

Although this work demonstrates a feasible method for achieving effective path tracking by exploiting multiple actuators, several potential developments remain open. A first and crucial direction concerns the design of a Neural Network capable of handling the entire vehicle speed range. While this may appear to be only a matter of repeating the training procedure at different speeds to enrich the database, it is actually more complex. The weighting matrices of the NMPC are strongly speed-dependent, and their variations influence the training process. Ensuring consistency across different speeds, in terms of command inputs, is essential to obtain a Neural Network that is both effective and robust. Moreover, when extending the approach to the full speed range, it may be necessary to include additional features in the network inputs, such as longitudinal acceleration. In fact, it is supposed that the NN can also work during an acceleration phase, which heavily affects the vehicle dynamics.

Another potential line of development involves the selective use of actuators based on driving conditions. Experimental results have highlighted increased variability in performance, even with identical tuning parameters, suggesting that actuator activation strategies may need to be more adaptive. A promising approach could be to rely on a baseline 2WS configuration under nominal conditions, while enabling additional actuators—such as rear steering or torque vectoring—only when necessary, for example, in emergency scenarios or, better, when higher stability margins are required (which is the scenario tested on the field). Such a strategy could combine the simplicity and robustness of reduced actuation with the flexibility and enhanced performance of multi-actuator systems.

A further possible development concerns the use of a Neural Network to provide only a feedforward contribution. The idea is to design a hybrid controller that combines

the advantages of classical approaches, such as PID control, with those offered by Neural Networks. In this scheme, the PID controller is responsible for the feedback action, ensuring stability and robustness, while the Neural Network generates the feedforward contribution, improving performance and adaptability. Such a hybrid structure offers a wider tuning margin compared to a purely NN-based controller, effectively merging reliability with learning-based flexibility.

Acknowledgements

I would like to extend my warm thanks to Professor Aldo Sorniotti, who gave me the opportunity to work on a truly inspiring project. I would like to thank him for the trust he placed in me, and I hope to have fully met his expectations. Another thanks goes to Professor Tota for the support provided.

A sincere thank you goes to Cecilia, who followed me step by step, placing immense trust in me throughout these months — a constant source of motivation and support.

I continue to extend my thanks to all my friends: Ciccio, Domenico (JD), Gaetano (Mr Olympia/Thanos), Francesco (Laba), Salvatore (Totino/il maestro), Andrea, Lorenzo, Valerio, and Alberto. Those who have expressed interest in my work and consistently believed in me have provided an additional source of motivation.

In particular, I would like to thank Ciccio, my longest-time friend, who constantly shared his approval of my person and work. From the beginning.

Finally, a heartfelt thank you goes to those who have always believed in me and my work. Before anyone else, the engine of soul: my family, essential and unavoidable contributors to my growth and achievements.

Grazie.

Bibliography

[1] C. Wang, Y. Wang, and K. Alexander. «Quantifying Human Error in Road Accidents». In: *SAE Technical Paper* 2023-01-1234 (2023). WCX SAE World Congress Experience. DOI: 10.4271/2023-01-1234 (cit. on p. 1).

[2] Pietro Stano, Davide Tavernini, Umberto Montanaro, Manuela Tufo, Giovanni Fiengo, Luigi Novella, and Aldo Sorniotti. «Enhanced Active Safety Through Integrated Autonomous Drifting and Direct Yaw Moment Control via Nonlinear Model Predictive Control». In: *IEEE Transactions on Intelligent Vehicles* 9.2 (2024), pp. 4172–4190. DOI: 10.1109/TIV.2023.3340992 (cit. on pp. 1, 2).

[3] Shiyue Zhao, Junzhi Zhang, Chengkun He, Yuan Ji, Heye Huang, and Xiaohui Hou. «Autonomous vehicle extreme control for emergency collision avoidance via Reachability-Guided reinforcement learning». In: *Advanced Engineering Informatics* 62 (2024), p. 102801. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2024.102801. URL: https://www.sciencedirect.com/science/article/pii/S147403462400449X (cit. on p. 1).

[4] N. R. Kapania and J. C. Gerdes. «Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling». In: *Vehicle System Dynamics* 53.12 (2015), pp. 1687–1704. DOI: 10.1080/00423114.2015.1055279. URL: https://doi.org/10.1080/00423114.2015.1055279 (cit. on pp. 2, 20).

[5] Dequan Zeng, Shicong Pan, Yinquan Yu, Yiming Hu, Jinwen Yang, Peizhi Zhang, Lu Xiong, Giuseppe Carbone, and Letian Gao. «A comparative study on trajectory tracking control methods for automated vehicles». In: *Scientific Reports* 15 (2025), p. 17073. DOI: 10.1038/s41598-025-01365-9 (cit. on p. 2).

[6] M. Rokonuzzaman et al. «Review and performance evaluation of path tracking controllers of autonomous vehicles». In: *IET Intelligent Transport Systems* (2023). DOI: 10.1049/itr2.12051 (cit. on p. 2).

[7] Fang Xu, Xu Zhang, Hong Chen, Yunfeng Hu, Ping Wang, and Ting Qu. «Parallel Nonlinear Model Predictive Controller for Real-Time Path Tracking of Autonomous Vehicle». In: *IEEE Transactions on Industrial Electronics* 71.12 (2024), pp. 16503–16513. DOI: 10.1109/TIE.2024.3390738 (cit. on p. 2).

[8] Trey P. Weber and J. Christian Gerdes. «Modeling and Control for Dynamic Drifting Trajectories». In: *IEEE Transactions on Intelligent Vehicles* 9.2 (2024), pp. 3731–3741. DOI: 10.1109/TIV.2023.3340918 (cit. on p. 2).

[9] J. Y. M. Goh, M. Thompson, J. Dallas, and A. Balachandran. «Beyond the stable handling limits: nonlinear model predictive control for highly transient autonomous drifting». In: *Vehicle System Dynamics* 62.10 (2024), pp. 2590–2613. DOI: 10.1080/00423114.2023.2297799 (cit. on p. 2).

[10] Y. Liu, H. Zhao, X. Zhang, Y. Chen, and L. Wang. «A DDPG-based Path Following Control Strategy for Autonomous Vehicles by Integrated Imitation Learning and Feedforward Exploration». In: *Chinese Journal of Mechanical Engineering* 38.1 (2025). Open Access, p. 1336. ISSN: 2192-8258. DOI: 10.1186/s10033-025-01336-1. URL: https://cjme.springeropen.com/articles/10.1186/s10033-025-01336-1 (cit. on p. 2).

[11] Shen Liu and Steffen Müller. «Reliability of Deep Neural Networks for an End-to-End Imitation Learning-Based Lane Keeping». In: *IEEE Transactions on Intelligent Transportation Systems* 24.12 (Dec. 2023), pp. 13768–13785. ISSN: 1558-0016. DOI: 10.1109/TITS.2023.3299229. URL: https://ieeexplore.ieee.org/document/10208229 (cit. on p. 3).

[12] Joonwoo Ahn, Minsoo Kim, and Jaeheung Park. «Autonomous driving using imitation learning with look-ahead point for semi-structured environments». In: *Scientific Reports* 12.21285 (2022). DOI: 10.1038/s41598-022-23546-6. URL: https://www.nature.com/articles/s41598-022-23546-6 (cit. on p. 3).

[13] Shaobing Xu, Huei Peng, and Yifan Tang. «Preview Path Tracking Control With Delay Compensation for Autonomous Vehicles». In: *IEEE Transactions on Intelligent Transportation Systems* 22.5 (2021), pp. 2979–2989. DOI: 10.1109/TITS.2020.2978417. URL: https://ieeexplore.ieee.org/document/9097131 (cit. on p. 3).