Politecnico di Torino

Master's degree in Mechatronic Engineering
A.y. 2023/2024

Politecnico
di Torino

A Comparative Analysis of Two Model Predictive Control Architectures for Autonomous Driving Systems

Supervisors:
Prof. Alessandro RIZZO
Eng. Francesco D'INTRONO - BrainTechnologies

Candidate:
Manuel FISCATO

Acknowledgements

Mi prendo queste piccolo spazio per ringraziare tutte le persone che mi hanno aiutato e mi sono state vicine in questi ultimi anni ricchi di successi, sconfitte, momenti importanti e momenti difficili. Vorrei partire ringraziando la mia famiglia che è stata sempre presente nel momento del bisogno, sempre disponibile a portarmi e venire a prendere in stazione, sempre pronta ad accogliermi a casa anche dopo mesi di lontananza con dei fantastici pranzi e sempre disposta a fare qualcosa per rendermi felice. Grazie anche alla nonna che mi è stata vicina, con dei messaggi, con delle chiamate e con il cibo che con tanto amore preparava prima di ogni mia partenza. Non meno importante è la compagnia di casa, partendo dalla componente più storica ci tengo a ringraziare Simone, Carlo, Jack e Massi amici da una vita con i quali ho avuto la fortuna di crescere condividendo qualsiasi tipo di situazione, semplici uscite settimanali, litigi, vacanze, campi scuola o semplici discussioni, tutte affrontate con uno spiccato senso del rispetto, della coesione e dell'amicizia. La seconda parte della compagnia anche se più recente è comunque riuscita a concentrare negli ultimi anni una molteplicità di bei ricordi. Mi riferisco a Giorgia, Arianna, Aurora e Nicole validissime ascoltatrici e consigliere, disponibili in qualsiasi momento a parlarle, discutere e snocciolare anche il più piccolo dettaglio presente nelle nostre vite. Grazie anche ai membri del team RoboTo, Paolo, Franz, Vittorio, Corrado, Marco e Jhonny che mi hanno accolto fin da subito con un grandissimo entusiasmo permettendomi di crescere a livello professionale e personale. Sono stati per quasi due anni i migliori compagni di avventura che potessi immaginare, trasmettendomi la loro infinita passione, dedizione e caparbietà. In ultima ringrazio l'azienda BrainTechnologies, il mio relatore Francesco e il professore Alessandro Rizzo per la disponibilità dimostrata durante tutto il percorso di tesi.

"*Se cerchi di essere qualcuno prova con te stesso*"

Salmo, Mic Taser

Abstract

This thesis presents a comparative analysis of two Nonlinear Model Predictive Control (NMPC) architectures applied to autonomous driving for the automotive industry. In response to the growing demand for reliable and efficient autonomous driving systems, NMPC has emerged as a prominent control strategy, capable of handling both linear and nonlinear constraints while predicting future states. The objective of this study is to develop and evaluate two distinct NMPC architectures for lane centering and obstacle avoidance. The key difference between the two architectures is that in the base architecture, both path planning and control computation are performed by the same NMPC, whereas in the double-layer NMPC, the higher layer handles path planning while the lower layer computes the control inputs. The decision to use Model Predictive Control for path planning stems from its model-based nature, this ensures that the generated path is not only feasible but also dynamically consistent with the vehicle's capabilities. Detailed vehicle models, including a four wheeled model, a two-wheeled model, and a point mass model, are developed starting from the vehicle dynamics and then used in simulations. The two NMPC architectures are assessed for their efficacy in lane centering and managing both static and dynamic obstacle avoidance. Performance is evaluated in terms of computational efficiency, driving comfort, and overall control effectiveness. The results demonstrate that while both architectures effectively manage vehicle control, the double layer architecture exhibits superior performance, it is faster, produces smoother trajectories, and performs better at higher velocities compared to the single-layer architecture.

Table of Contents

List of figures

List of tables

Acronyms

ADAS

Advanced driver assistant system

MPC

Model predictive control

NMPC

Nonlinear model predictive control

ISO

International organization of standardization

CoG

Center of gravity

OSM

Open street map

DSD

Driving scenario designer

LC

Lane centering

SOA

Static obstacle avoidance

DOA

Dynamic obstacle avoidance

HIL

Hardware in the loop

LIDAR

Light detection and ranging

RADAR

Radio detection and ranging

GPS

Global positioning system

Chapter 1

Introduction

1.1 Motivations

In recent years, the automotive industry has seen a significant advancement in field of advanced driver assistance systems. These systems aim to improve vehicle safety and driving comfort by introducing automation features that reduce human error, one of the leading causes of road accidents. ADAS is categorized into various levels, starting from level 0, where no automation is present, up to level 5, representing full autonomous driving capabilities. This classification, outlined by the society of automotive engineers, helps distinguish the degree of autonomy and control present in a vehicle, with each level adding more assistance and autonomy to driving task.

In level 1, the vehicle can assist with either steering or acceleration/braking, but not at the same time. The driver remains fully responsible for all other driving tasks and must always stay active. While at level 2, the vehicle can manage both steering and acceleration/braking simultaneously. As vehicles progress to higher levels, automation becomes increasingly capable of handling complex driving tasks, eventually allowing the driver to detach completely under specific conditions. Achieving higher levels of autonomy requires advanced control algorithms that can make safe and efficient decisions in real-time, especially in scenarios involving potential collisions or obstacles.

The motivation behind the implementation of advanced driver assistance systems is deeply rooted in improving road safety by mitigating human error, which is the leading cause of traffic accidents. Distraction, fatigue and misjudgment frequently leads lead to collisions and ADAS aims to minimize these risks through automation. Technologies like sensors, cameras, machine learning and control algorithms enable quicker, more precise responses than a human driver can typically provide. This is supported by a lot of studies that report the reduction of incidents thanks to the ADAS [1], [2], [3], [4].

The data in [5] highlights the benefits of collision avoidance technologies, with a 50% reduction in front-to-rear collisions with automatic emergency braking or a 78% reduction in rear-end collisions thanks to a rear automatic braking system.

1.2 Design choices

MPC has been chosen because it has emerged as a widely adopted control technique in the automotive industry thanks to its unique ability to manage multivariable systems, optimize performance and handle complex constraints in real-time. Its popularity is supported by numerous studies [6], [7], [8], [9] highlighting its application in various vehicle subsystems, including spark-ignition engine speed control, vehicle stability control, transmission and suspension.

One of the main advantages of MPC is its ability to predict the future behavior of a system using models. By generating an optimal sequence of control actions over a finite horizon, MPC also ensures the control of multivariable nonlinear systems. This makes it highly effective in automotive applications, where systems are characterized by nonlinearities and are often interconnected.

Given its flexibility and effectiveness in managing multivariable systems while respecting constraints and system dynamics, MPC has become a core component in advanced automotive control systems. Its application spans from traditional systems, already mentioned, to more advanced domains, such as autonomous driving, where real-time decision and optimization are crucial. All these features explain the motivation behind choosing MPC as the control strategy for tasks like lane centering and obstacle avoidance.

However, despite these benefits, its computational burden remains a key challenge, especially in real-time applications where adherence to deadline is essential. Solving the optimization problem at each time step can be computationally intensive, particularly when dealing with complex vehicle dynamics and tight constraints.

To address this challenge, the implementation of an enhanced architecture offers a solution by distributing the computational workload across two layers. In this setup, the higher layer handles the trajectory planning using a simpler model. The lower layer on the other hand focuses on computing the control inputs with a more detailed model. This hierarchical structure reduces the computational burden on each layer, allowing the system to maintain high accuracy without sacrificing real-time feasibility.

A crucial factor in choosing the path planner algorithm[10] stays in the online or offline generation of the path. Offline path planner generates a complete path before the vehicle starts moving, typically using global knowledge of the environment. These planners work well in structured environments, but they lack flexibility in dynamic situations such as obstacle avoidance. In contrast, online path planners, like those using MPC [11],

continuously compute the optimal path as the vehicle moves, reacting to dynamic changes in the environment.

In summary, the use of MPC for path planning was motivated by its ability to operate online, making continuous adjustments based on data sampled over time. This flexibility, combined with its capacity to optimize trajectories and respect dynamic constraints, made MPC the optimal choice for achieving both safe and efficient path planning in autonomous driving scenarios.

1.3 Thesis objectives

The main aim of this thesis is to develop an NMPC-based control framework capable of handling key aspects of autonomous driving, specifically lane centering, static and dynamic obstacle avoidance. In terms of lane centering the focus is on designing a control system that ensures the vehicle maintains an optimal trajectory, minimizing deviations from the middle point. For static and dynamic obstacle avoidance, the goal is to develop an architecture that react quickly to both stationary and moving obstacles, ensuring the vehicle can navigate safely around them.

By comparing two NMPC architectures, a single layer and more complex double-layer system, this thesis evaluate the time efficiency and driving comfort provided by each approach, offering insights into the trade-offs between computational complexity and control performance.

Chapter 2

Vehicle and environment modelling

2.1 Introduction

In the context of Nonlinear Model Predictive Control, accurate and detailed modeling of vehicle dynamics is of critical importance. The control algorithm's ability to predict future states and compute optimal control inputs is directly tied to the quality of this models. By providing an accurate representation of the vehicle's physical behavior, the models allow the NMPC to anticipate how the vehicle will respond to various inputs and constraints. However, it is equally important to balance the complexity of the models used, as MPC is already computationally intensive. Incorporating overly complex models could worsen performance by increasing computational demands, potentially compromising real-time control. Thus, the development of suitable vehicle dynamics models [12] is not only a matter of accuracy but also efficiency, ensuring that the system can operate effectively within the given computational constraints.

In this thesis, the simulation will account for a case study in which the vehicle is operated in a real-world environment. Simulating the scenario requires generating both the environmental data and the sensor data. This means that the road geometry, lane boundaries and potential obstacles must be artificially created, while simulated sensors provide continuous feedback on these elements.

It is essential to establish a clear reference frame to describe the vehicle's motion. This thesis adopts the ISO vehicle coordinate system (2.1), which is widely recognized for defining the orientation and the positioning of vehicles in a standardized manner [13]. According to this standard, the X-axis points forward along the vehicle's longitudinal axis, the Y-axis extends to the left in lateral direction and the Z-axis points upward. In

addition to the orientation of the axes the vehicle's motion is described using the yaw angle, denoted as ψ, which represents the rotation about the Z-axis.

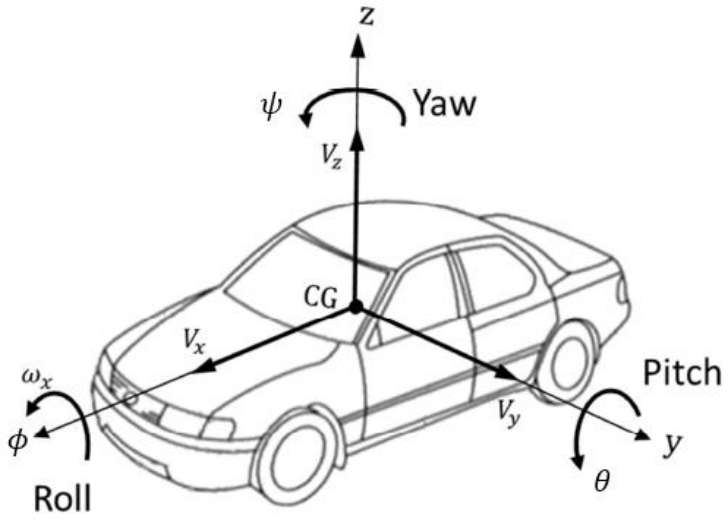

Figure 2.1: ISO vehicle coordinate system

2.2 Nonlinear four-wheel model

The nonlinear four-wheel model plays a crucial role in accurately simulating vehicle dynamics. This model provides a comprehensive representation of the car's behavior, capturing the interactions between all the four wheels and the road, and incorporating the roll dynamics within the yaw motion. This comprehensive representation ensures that the simulation reflects real world behavior, allowing for more reliable analysis of the control performances.

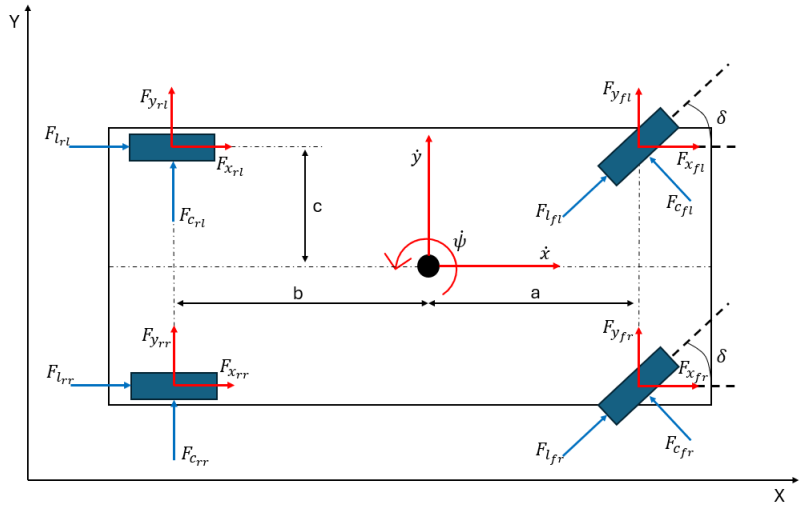

Figure 2.2: Four-wheel forces and reference frames

The dynamics of the system is described in a compact form by the following equation:

$$\dot{\xi}(t) = \mathrm{f}\big(\xi(t), u(t)\big) \tag{2.1}$$

where $\xi(t)$ represent the states vector $\left[\dot{x},\ \dot{y},\ \dot{\psi},\ \psi,\ X,\ Y\right]^T$ and $u(t)$ represent the input vector $\left[T_{fl},\ T_{fr},\ T_{rl},\ T_{rr},\ \delta\right]^T$. In this context the state vector includes: the longitudinal velocity and the lateral velocity in the body frame, the yaw rate and the yaw angle, and the global frame positions. The input vector comprises the torque applied to each wheel and the steering angle.

The notation of the torque must be clarified, the first subscript indicates whether the wheel is in front or rear position. The second subscript indicates whether the wheel is in left or right position. This notation is also applied to the forces.

The system's dynamics with respect to the vehicle's center of gravity and the coordinate transformation from the body frame to the global frame are described by:

$$m\ddot{x} = m\dot{y}\dot{\psi} + F_{x_{fl}} + F_{x_{fr}} + F_{x_{rl}} + F_{x_{rr}} \quad (2.2a)$$

$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_{y_{fl}} + F_{y_{fr}} + F_{y_{rl}} + F_{y_{rr}} \quad (2.2b)$$

$$I\ddot{\psi} = a\left(F_{y_{fl}} + F_{y_{fr}}\right) - b\left(F_{y_{rl}} + F_{y_{rr}}\right) + c\left(-F_{x_{fl}} + F_{x_{fr}} - F_{x_{rl}} + F_{x_{rr}}\right) \quad (2.2c)$$

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \quad (2.2d)$$

$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi \quad (2.2e)$$

where m is the vehicle's mass, I is the inertia about the z axis, a is the distance between the CoG and the front axle, b is the distance between the CoG and the rear axle, and c is the distance between Cog and the wheel along y axis of the body frame.

The forces F_x and F_y are the projection of the longitudinal (F_l) and lateral (F_c) tire forces along fixed tire frame, and are computed as follow:

$$F_{x_{fl}} = F_{l_{fl}}\cos\delta - F_{c_{fl}}\sin\delta \quad (2.3a)$$

$$F_{x_{fr}} = F_{l_{fr}}\cos\delta - F_{c_{fr}}\sin\delta \quad (2.3b)$$

$$F_{x_{rl}} = F_{l_{rl}} \quad (2.3c)$$

$$F_{x_{rr}} = F_{l_{rr}} \quad (2.3d)$$

$$F_{y_{fl}} = F_{l_{fl}}\sin\delta + F_{c_{fl}}\cos\delta \quad (2.3e)$$

$$F_{y_{fr}} = F_{l_{fr}}\sin\delta + F_{c_{fr}}\cos\delta \quad (2.3f)$$

$$F_{y_{rl}} = F_{c_{rl}} \quad (2.3g)$$

$$F_{y_{rr}} = F_{c_{rr}} \quad (2.3h)$$

For the rear wheels the steering angle is assumed to be zero, simplifying the equations. The longitudinal and lateral forces will be detailed in the chapter (2.5).

2.3 Nonlinear two-wheel model

The two-wheel model is a vital component of the Model Predictive Control algorithm, designed to simplify the vehicle's dynamics while rationing essential characteristics needed for effective control computation. This model abstracts the complexities of the full four-wheeled system by focusing on the primary dynamics of the vehicle, namely its

longitudinal and lateral behaviors, along with its yaw motion. By employing this simplified representation, the MPC can effectively compute control inputs and manage vehicle responses without the computational burden of a full four-wheeled model.

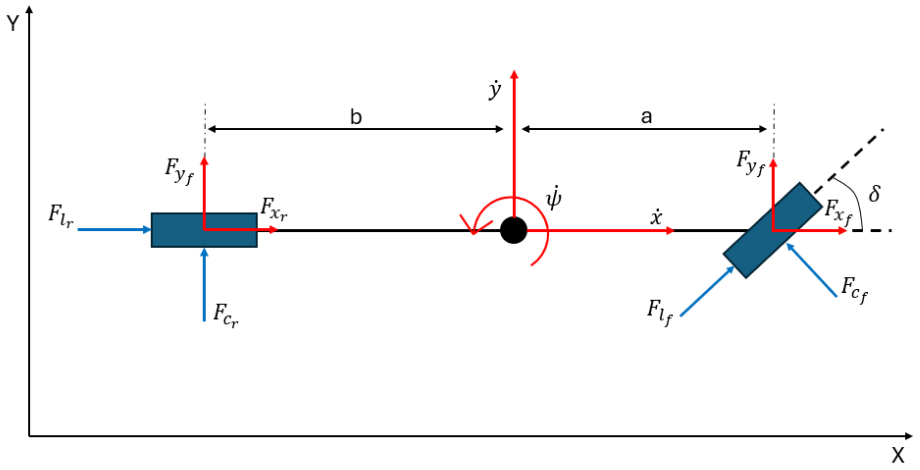

Figure 2.3: Two-wheel forces and reference frames

The dynamic of the system is described in a compact form by the following equation:

$$\dot{\xi}(t) = \mathrm{f}\big(\xi(t), u(t)\big) \tag{2.4}$$

where $\xi(t)$ represent the states vector $\left[\dot{x}, \dot{y}, \dot{\psi}, \psi, X, Y\right]^T$ and $u(t)$ represent the input vector $\left[T_f, T_r, \delta\right]^T$. In this context the state vector includes the longitudinal velocity and the lateral velocity in the body frame, the yaw rate and the yaw angle, and the global frame positions. The input vector comprises the torque applied to the front and rear wheels and the steering angle.

As a consequence of the reduction in the number of wheels the system's equations and the coordinate transformation from the body frame to the global frame can be summarized as:

$$m\ddot{x} = m\dot{y}\dot{\psi} + 2F_{x_f} + 2F_{x_r} \tag{2.5a}$$

$$m\ddot{y} = -m\dot{x}\dot{\psi} + 2F_{y_f} + 2F_{y_r} \tag{2.5b}$$

$$I\ddot{\psi} = 2aF_{y_f} - 2bF_{y_r} \tag{2.5c}$$

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \tag{2.5d}$$

$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi \tag{2.5e}$$

where m is the vehicle's mass, I is the inertia about the z axis, a is the distance between the CoG and the front axle, b is the distance between the CoG and the rear axle and c is the distance, along y axis of the body frame, between Cog and the wheel.

The forces F_x and F_y are the projection of the longitudinal (F_l) and lateral (F_c) tire forces along fixed tire frame, and are computed as follow:

$$F_{x_f} = F_{l_f}\cos\delta - F_{c_f}\sin\delta \tag{2.6a}$$

$$F_{x_r} = F_{l_r} \tag{2.6b}$$

$$F_{y_f} = F_{l_f}\sin\delta + F_{c_f}\cos\delta \tag{2.6c}$$

$$F_{y_r} = F_{c_r} \tag{2.6d}$$

For the rear wheels the steering angle is assumed to be zero, simplifying the equations. The longitudinal and lateral forces will be detailed in the chapter (2.5).

2.4 Point mass model

The point mass model is employed to further simplify the vehicle dynamics, focusing on high-level control tasks such as trajectory generation within the two-layer architecture.

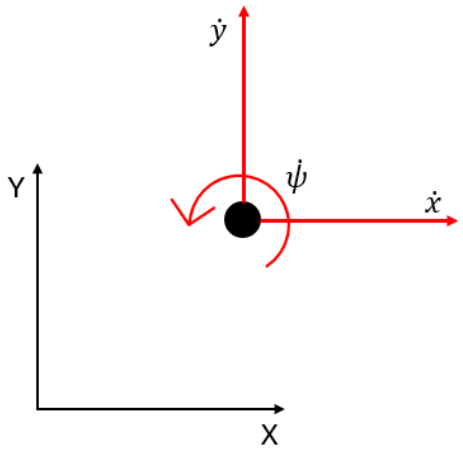

Figure 2.4: Point Mass reference frames

The dynamics of the point mass can be summarized as:

$$\dot{\xi}(t) = \mathrm{f}\big(\xi(t), u(t)\big) \tag{2.7}$$

where $\xi(t)$ represent the states vector $[\dot{x}, \dot{y}, \psi, X, Y]^T$ and $u(t)$ represent the input a_y. In this context the state vector includes the longitudinal velocity and the lateral velocity in the body frame, the yaw angle and the global frame positions. The input vector is defined as a single term and represents the lateral acceleration.

The equations are derived from the other models through the application of appropriate simplifications. The longitudinal velocity is assumed to be constant, and the lateral acceleration is defined solely in terms of the control input. The yaw velocity is then derived as a consequence of the aforementioned choices. The equation (2.7) can be decomposed into:

$$\ddot{x} = 0 \tag{2.7a}$$

$$\ddot{y} = a_y \tag{2.7b}$$

$$\dot{\psi} = \frac{a_y}{\dot{x}} \tag{2.7c}$$

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \tag{2.7d}$$

$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi \tag{2.7e}$$

2.5 Tire model

The tire model represents a crucial element in the simulation of vehicle dynamics as it captures the interaction between the tires and the road surface. At the point of contact between the tire and the road, three primary forces act: the vertical force, the longitudinal force and the lateral force.

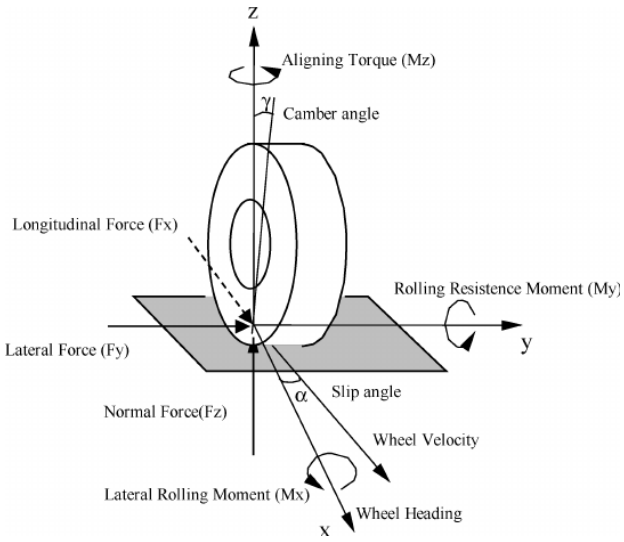

Figure 2.5: Tire model notation

The vertical force (F_z) is computed by assuming a static and uniform distribution of the vehicle's mass. In this context, the force is directly proportional to the mass (m) and the gravitational force (g) with adjustments based on the positioning of the center of gravity with respect to the front and rear axles.

For the four-wheel model we can write:

$$F_{z_{fl}} = F_{z_{fr}} = \frac{bmg}{2(a+b)} \tag{2.8a}$$

$$F_{z_{rl}} = F_{z_{rr}} = \frac{amg}{2(a+b)} \tag{2.8b}$$

While for the two-wheel model the forces are:

$$F_{z_f} = \frac{bmg}{a+b} \tag{2.9a}$$

$$F_{z_r} = \frac{amg}{a+b} \tag{2.9b}$$

The longitudinal force (F_l) represents the force generated by the tire in the direction of the vehicle's movement, primarily responsible for acceleration and braking. In this thesis, a zero-slip condition is assumed, meaning that the tire maintains full traction with the road without any relative motion between the tire and the ground. This assumption simplifies the calculation of longitudinal forces by focusing on the ideal case of perfect grip. The force can be obtained by dividing the traction, $T > 0$, or braking torque, $T < 0$, by the wheel radius r_ω:

$$F_l = \frac{T}{r_\omega} \tag{2.10}$$

The Pacejka formula [14] was initially considered for the lateral force (F_c) due to its proven ability to model tire behavior under a wide range of condition. This model (2.11) expresses the tire force as a nonlinear function of slip angle using a set of coefficients, such as stiffness (B_y), shape (C_y), peak (D_y) and curvature factor (E_y), that are typically derived from real-world tire testing. These coefficients are essential for accurately fitting the model to the specific tires and some studies [15], [16] explore different methodologies for obtaining them through experimental data. Since these coefficients were unavailable for this study, the use of the Pacejka formula was not feasible.

$$F_c = (D_y \sin[C_y \arctan\{B_y\alpha_y - E_y(B_y\alpha_y - \arctan(B_y\alpha_y))\}] + S_{V_y})G_{yk} + S_{V_{yk}} \tag{2.11}$$

Consequently, the so-called Fiala model [17] was identified as potential alternative, it is grounded in theoretical principles and does not depend on empirical coefficients, making it suitable choice for this study.

The following section presents the formula employed in this thesis.

$$F_c = \min\left(\mu F_z, \max\left(-\mu F_z,\ -C\tan(\alpha) + \frac{C^2\,|\tan(\alpha)|\tan(\alpha)}{3\mu F_z} - \frac{C^3\tan(\alpha)^3}{27\mu^2 F_z^2}\right)\right) \tag{2.12}$$

where μ is the road tire friction coefficient, C is the cornering stiffness, α is the slip angle and F_z is the vertical force.

In the linear region, for small values of the slip angle, the lateral force is proportional to α. This behavior is consistent with a tire's initial response, where there is still sufficient

contact between the tire and the road surface, and the tire behaves elastically. The tire is capable of generating increasing lateral forces as long as the slip angle remains within this range. As α increases the lateral force reaches a peak and then saturates. This occurs when the tire's contact patch starts to slide and the maximum lateral force that can be generated by the tire is reached. Beyond this point, the force begins to level off reflecting the tire's limit of adhesion to the road. It is also interesting to observe how this force varies when the road adhesion coefficient changes, as different surfaces (dry, wet, icy) can significantly influence the tire's behavior. In this work an optimal condition is considered with $\mu = 0.9$. A visual representation of this relationship is reported below.

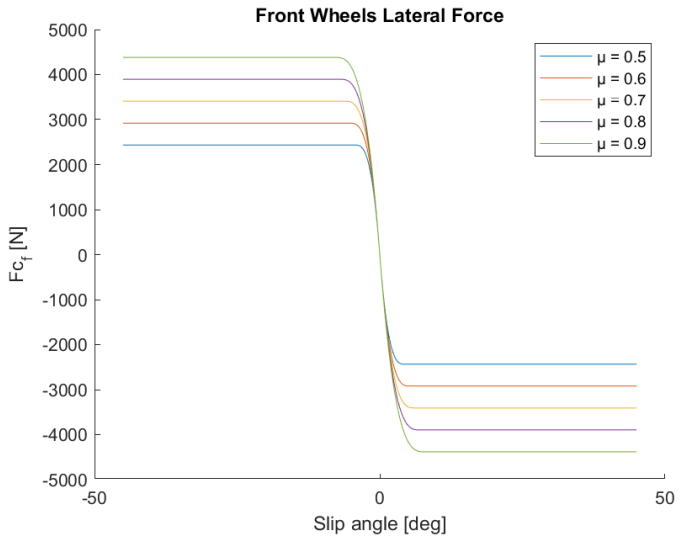

Figure 2.6: Front wheels lateral force with different road adhesion coefficient

The lateral force generated by the tire is directly dependent on the vertical force (F_z), which represents the load on the tire. In general, a higher vertical force allows the tire to generate a greater lateral force, improving the vehicle's ability to handle lateral maneuvers. In this case, since the center of gravity is located near the front axle, the vertical force acting on the front tires is greater compared to the rear ones. As a result, the rear tires can produce a lower lateral force, reducing their contribution to the vehicle's cornering capability. The following figure illustrates the above relationship.

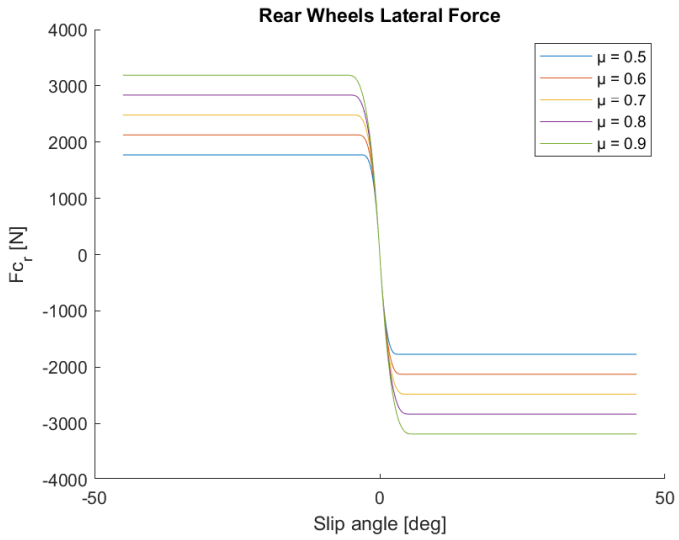

Figure 2.7: Rear wheels lateral force with different road adhesion coefficient

The slip angle α is a key parameter in tire dynamics, representing the angle between the direction in which the wheel is pointing and the actual path the tire follows as it moves along the road. Mathematically it can be expressed as:

$$\alpha = \arctan\frac{v_c}{v_l} \tag{2.13}$$

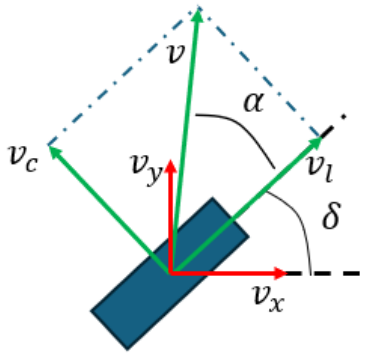

Figure 2.8: Graphical representation of slip angle, velocities and reference frames

The longitudinal and lateral wheel velocities can be computed from:

$$v_c = v_y \cos\delta - v_x \sin\delta \tag{2.14a}$$

$$v_l = v_y \sin\delta + v_x \cos\delta \tag{2.14b}$$

The slip angle and the velocities can be computed for all four wheels or for the two wheels, depending on the model used. The following equations allow for the calculation of the aforementioned values.

For the four-wheel model:

$$v_{x_{fl}} = v_{x_{rl}} = \dot{x} - c\dot{\psi} \tag{2.15a}$$

$$v_{x_{fr}} = v_{x_{rr}} = \dot{x} + c\dot{\psi} \tag{2.15b}$$

$$v_{y_{fl}} = v_{y_{fr}} = \dot{y} + a\dot{\psi} \tag{2.15c}$$

$$v_{y_{rl}} = v_{y_{rr}} = \dot{y} - b\dot{\psi} \tag{2.15d}$$

While for the two-wheel model:

$$v_{x_f} = v_{x_r} = \dot{x} \tag{2.16a}$$

$$v_{y_f} = \dot{y} + a\dot{\psi} \tag{2.16b}$$

$$v_{y_r} = \dot{y} - b\dot{\psi} \tag{2.16c}$$

Let's write the formula for the slip angle of the front left wheel of the four-wheel model substituting (2.15a) and (2.15c) into (2.14a) (2.14b) and the resulting equations into (2.13):

$$\alpha_{fl} = \arctan\frac{\left(\dot{y} + a\dot{\psi}\right)\cos\delta - \left(\dot{x} - c\dot{\psi}\right)\sin\delta}{\left(\dot{y} + a\dot{\psi}\right)\sin\delta + \left(\dot{x} - c\dot{\psi}\right)\cos\delta} \tag{2.17}$$

Assuming a zero-steering angle i.e. real-life practical case

$$\alpha = \arctan\frac{\left(\dot{y} + a\dot{\psi}\right)}{\left(\dot{x} - c\dot{\psi}\right)} \tag{2.18}$$

Analyzing the formula (2.18), in static condition when the longitudinal velocity is close to zero, the argument of the arctan goes to infinity and the slip angle increases reaching a value of 90°. This results in a critical problem resulting in unrealistic dynamics and numerical instability. To avoid this, the longitudinal velocity is kept above zero in the

simulations, ensuring that the slip angle remains within a manageable range, thereby maintaining the accuracy of the vehicle's dynamic response.

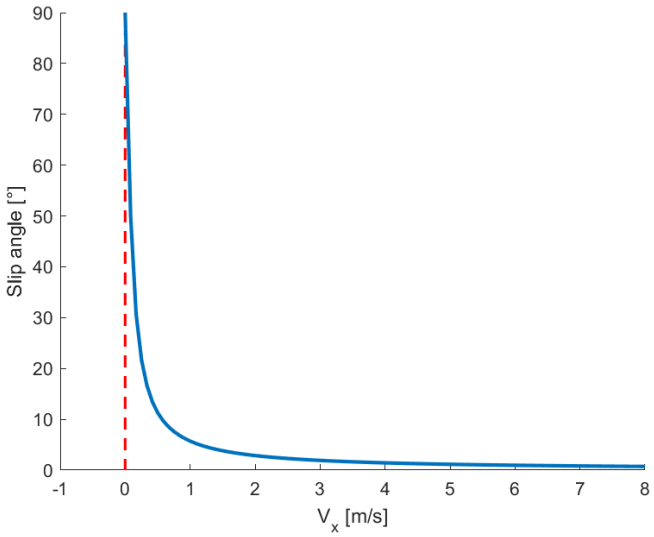

Figure 2.9: Slip angle for low longitudinal velocity

2.6 Vehicle models – Validation

Model validation is crucial to verify the accuracy and reliability of the vehicle models developed in this study. This chapter outlines the approach used to validate the three models through a series of simulations. The time evolution of the model has been simulated using Euler method with discrete time steps, ensuring computational simplicity while maintaining acceptable accuracy. The evolution over time of the state variable is represented thanks to the following formula:

$$\dot{\xi}(k) = f\big(\xi(k), u(k)\big) \tag{2.19a}$$

$$\xi(k+1) = \xi(k) + Ts\,\dot{\xi} \tag{2.19b}$$

The four-wheel wheel and the two-wheel models are tested across four different scenarios, starting from an initial position equal to $(X, Y) = (1,1)$ and with an initial longitudinal velocity of 10m/s.

1. Torque applied to the rear wheels, no steering angle and 0° yaw angle: this simulation assesses the model's performance on a straightforward acceleration.

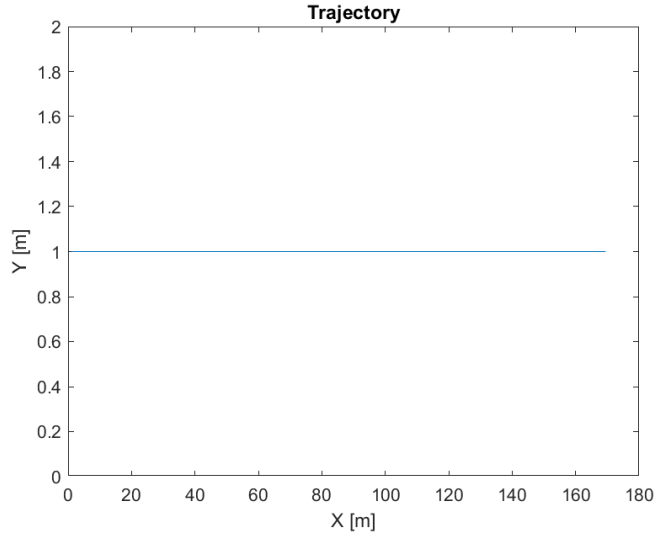

Figure 2.10: Straightforward acceleration four-wheel model

2. Torque applied to the rear wheels, no steering angle and 45° yaw angle: this scenario introduces an initial yaw angle of 45° evaluating the model's response to a different yaw angle while maintaining a straight path

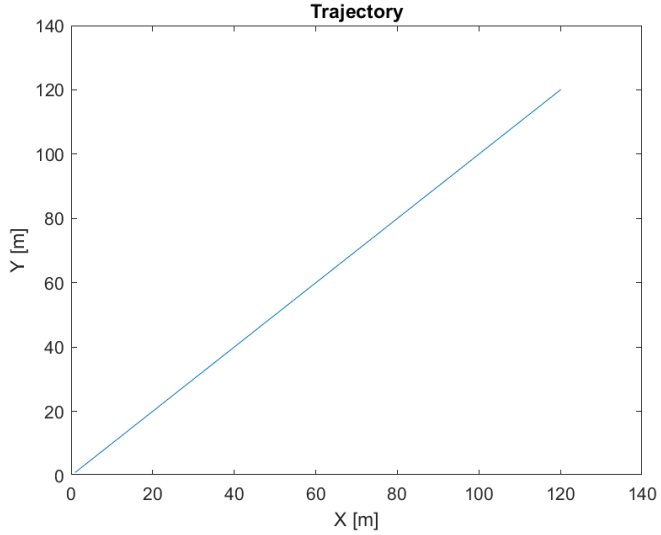

Figure 2.11: Straightforward acceleration four-wheel model with initial yaw position equal to 45°

3. Torque applied to the rear wheels, 3° steering angle and 0° yaw angle: this test examines the model's performance with a small positive steering angle

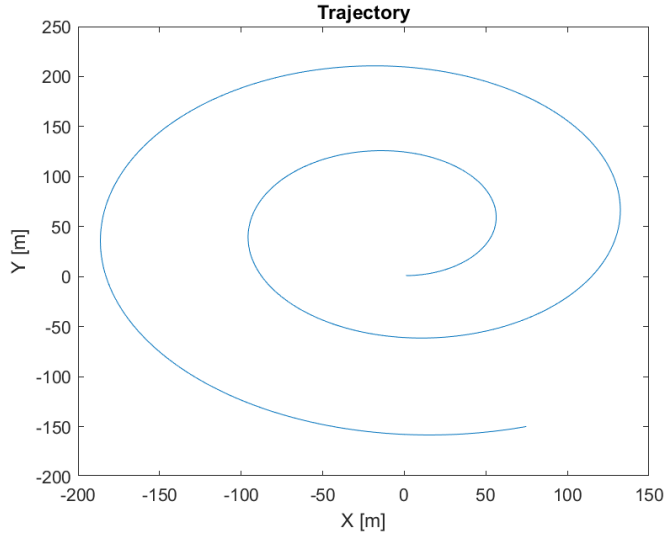

Figure 2.12: Four-wheel model simulation with constant positive steering angle

4. Torque applied to the rear wheels, -3° steering angle and 0° yaw angle: this simulation assesses the model's behavior when steering in the other direction.

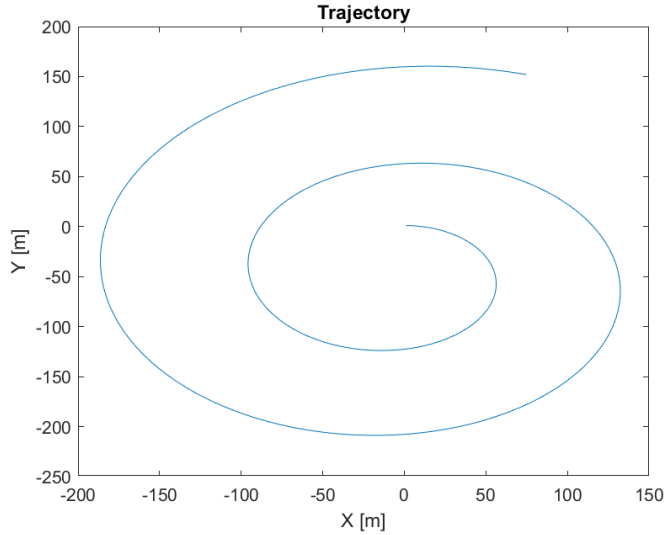

Figure 2.13: Four-wheel model simulation with constant negative steering angle

For simplicity only the pictures of the four-wheel model are shown but the two-wheel model also undergoes the same tests.

In addition to analyzing the vehicle's path, the validation process also includes examining the slip angle and the lateral force.

For the first two scenarios where the steering angle is zero, the slip angle and the lateral force are expected to remain close to zero. Since the vehicle is traveling in a straight line without any steering input, there should be no significant lateral acceleration, resulting in negligible lateral forces and slip angle.

However, in the third and fourth scenarios, where a small steering angle is introduced, the slip angle and the lateral force are expected to show a non-zero trend. Specifically:

- In the third scenario, a negative slip angle and corresponding lateral forces are generated as the vehicle turns slightly to the left.

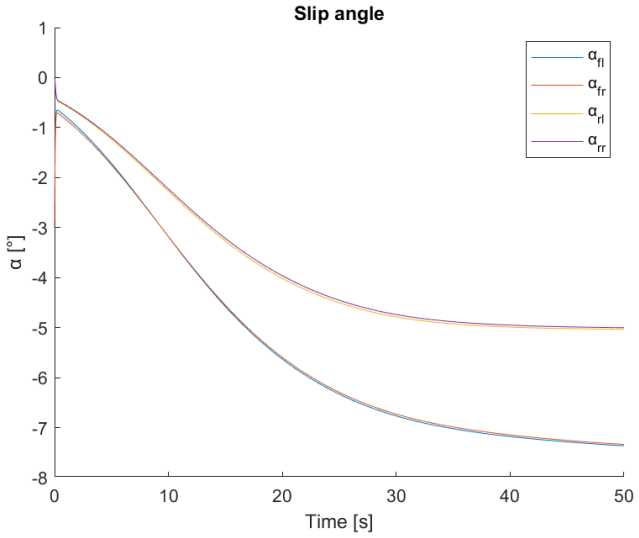

Figure 2.14: Four-wheel slip angle with a constant positive steering angle

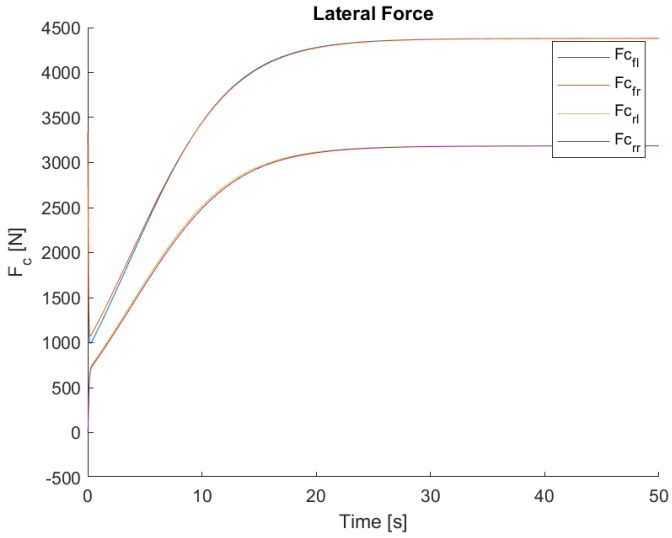

Figure 2.15: Four-wheel lateral force with a constant positive steering angle

- In the fourth scenario, the slip angle and lateral forces should mirror those from the third case but with opposite values, as the vehicle turns to the right

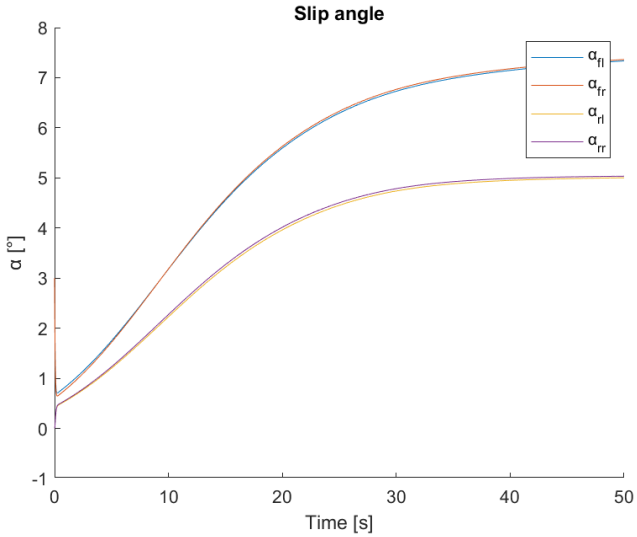

Figure 2.16: Four-wheel slip angle with a constant negative steering angle

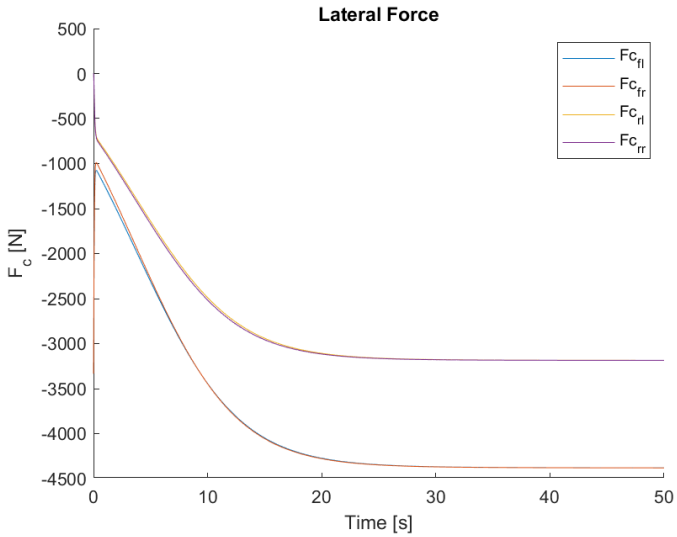

Figure 2.17: Four-wheel lateral force with a constant negative steering angle

For the point mass model, the same four test scenarios are conducted, but instead of manipulating the steering angle and the torque, the lateral acceleration is controlled directly. This model operates under the assumption of constant longitudinal velocity, which is set to 10m/s in all cases.

The tests are structured as follows:

1. Zero lateral acceleration and 0° yaw angle

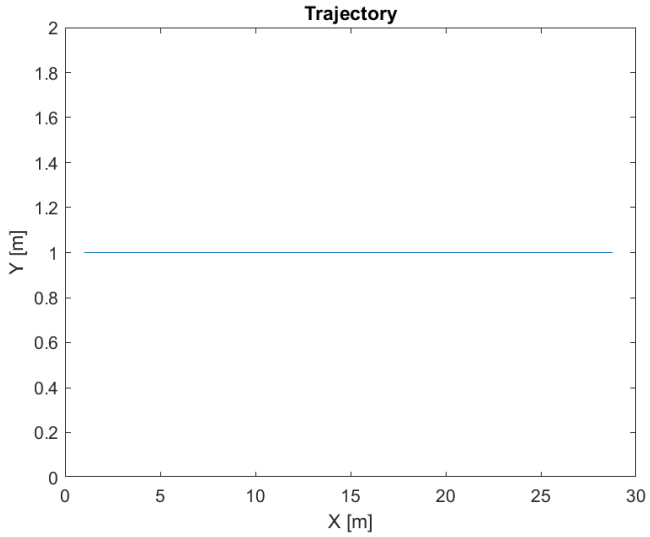

Figure 2.18: Straightforward trajectory with constant velocity

2. Zero lateral acceleration and 45° yaw angle

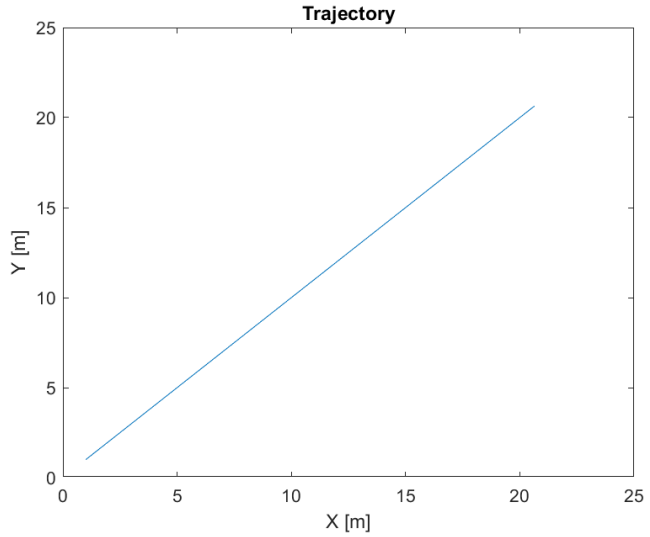

Figure 2.19: Straightforward trajectory with constant velocity and initial yaw angle equal to 45°

3. Positive acceleration and 0° yaw angle

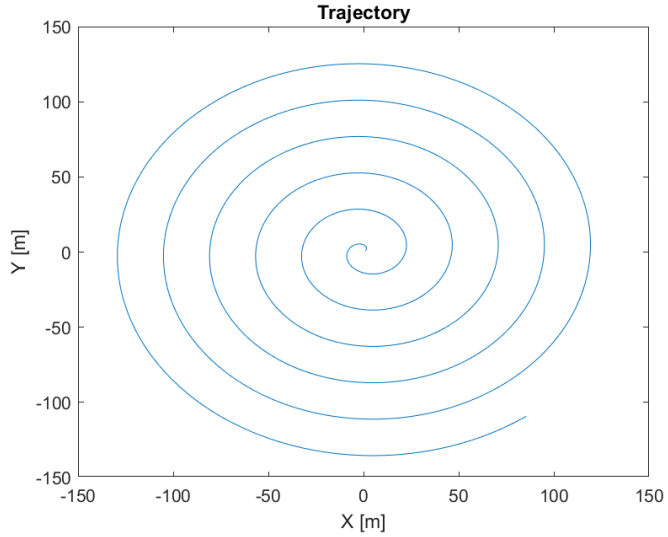

Figure 2.20: Curvilinear trajectory with positive lateral acceleration

4. Negative lateral acceleration and 0° yaw angle

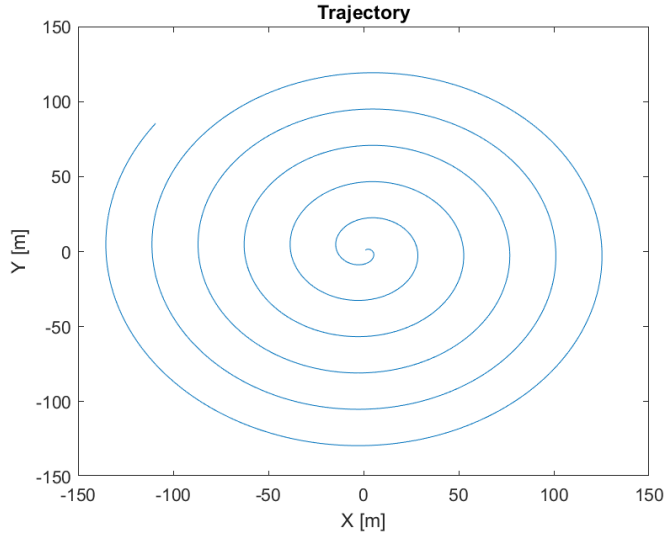

Figure 2.21: Curvilinear trajectory with negative lateral acceleration

The theoretical predictions are well supported by the analysis of the graphs. These results confirm the accuracy, validity and reliability of the models under diverse circumstances. Having validated all the models, it is possible to proceed with the other stages.

2.7 Environment models

The environment model is created using data from OpenStreetMap [18] and MATLAB's DrivingScenarioDesigner [19]. OSM allows the selection of specific geographic areas, providing real-world road data such as lane boundaries and road geometry. After importing the chosen area into DSD, individual roads of interest can be specifically selected and refined for use in simulations.

Points representing road boundaries were generated from the selected road using a MATLAB function 'roadBoundaries'[20]. To ensure a smooth and continuous road representation a spline function [21], that returns a piecewise polynomial structure, was applied to these points. This method enables precise evaluation of the vehicle's position relative to the road boundaries at any location, as the spline provides interpolation between the points which are sampled every 2-3 meters.

The chosen road is a double-lane street. The aforementioned process generates points only for the road boundaries. The middle lane is calculated as the mean between the two boundaries.

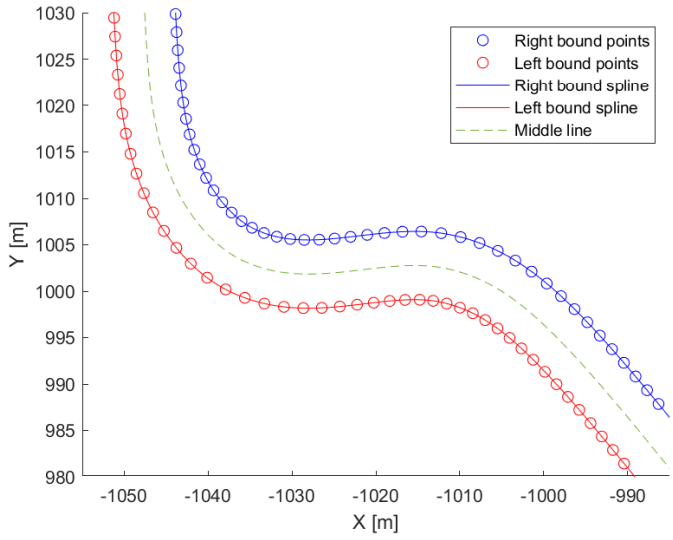

Figure 2.22: Section of the road boundaries points and fitting curves

All the points concerning the road bounds are generated with reference to a global frame. In a real-world scenario, such data would be collected by sensors mounted on the vehicle. To simulate this, a frame transformation is applied to convert the global coordinates into the body's frame. The formulas applied are explained below.

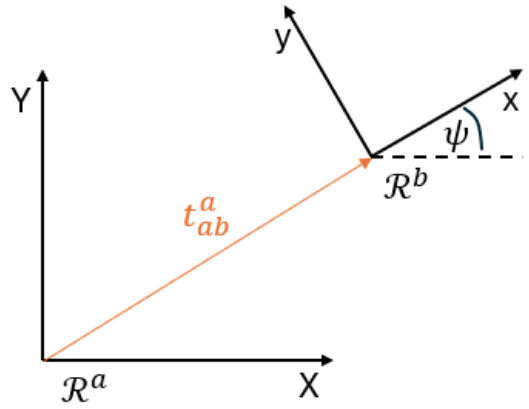

Figure 2.23: Representation of global and body frames

The transformation between the body frame ($\mathcal{R}^b$) and the absolute frame ($\mathcal{R}^a$) can be uniquely characterized by a rotation matrix and a translation vector.

$$R_b^a = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.20a}$$

$$t_{ab}^a = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.20b}$$

where ψ is the yaw angle and X, Y, Z represent the position of the vehicle in the absolute frame.

Let's simplify the computation removing the z coordinate, since it is negligible for our purpose. Now we can write the formula that transforms the coordinate of a point in the body frame into the global frame:

$$\begin{bmatrix} X^a \\ Y^a \end{bmatrix} = t_{ab}^a + R_b^a \begin{bmatrix} X^b \\ Y^b \end{bmatrix} \tag{2.21}$$

For what we want to achieve, the reverse of the previous formula is needed.

$$R_a^b = (R_b^a)^{-1} \tag{2.22a}$$

$$t_{ba}^b = -R_a^b t_{ab}^a \tag{2.22b}$$

Finally:

$$\begin{bmatrix} X^b \\ Y^b \end{bmatrix} = t_{ba}^b + R_a^b \begin{bmatrix} X^a \\ Y^a \end{bmatrix} \tag{2.23}$$

To conclude the environment modelling, the obstacle representation is discussed. The obstacles are represented in simplified forms, such as circles, which makes their integration into the simulation more computationally efficient. This circular approximation allows for easy calculation of distances between the vehicle and the obstacles, a critical element in both static and dynamic obstacle avoidance scenarios. The size, the position and the safe distance are key parameters and their role in the simulation is to create constraints for the NMPC and ensuring safe navigation around them. For dynamic obstacles, their velocities are also incorporated.

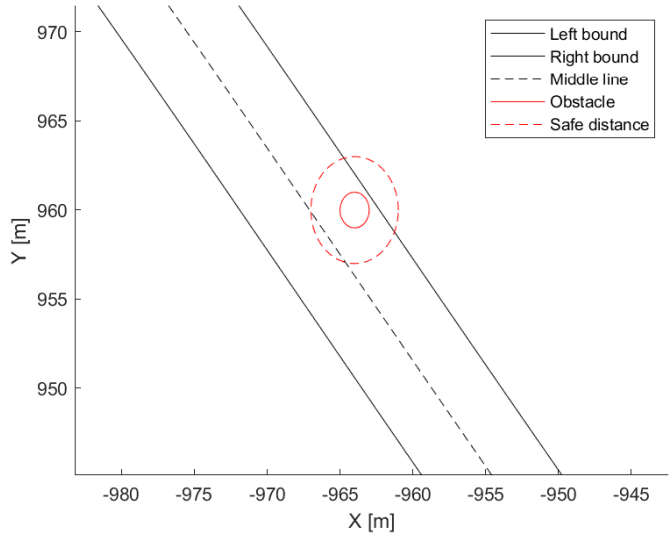

Figure 2.24: Static obstacle and safe distance inside a road section

Chapter 3

NMPC - Single layer

3.1 Introduction

Model Predictive Control operates by predicting the future behavior of the system over a specified time horizon and calculating control inputs to optimize performance criteria while respecting system constraints. The core principle is the iterative optimization process: at each time step, the current state of the plant is used to predict future states, a cost function is minimized to find the optimal control sequence that respect the constraints and only the first control input is applied. This process repeats as new data becomes available.

The idea is to begin with a base NMPC architecture and use it as a benchmark for further comparison. This approach is helpful because the implementation is more straightforward, allowing for a clear starting point and a reference for evaluating more complex architectures. In all the implementations the MATLAB solver 'fmincon' [22] is used to solve the constrained nonlinear multivariable control problem.

3.2 Single layer NMPC architecture

The base nonlinear Model Predictive Control architecture is designed to take specific inputs. The references are the desired vehicle velocity, the position of potential obstacle and the road boundaries. These inputs guide the control strategy, ensuring the vehicle maintains lane centering while avoiding obstacles.

In this setup, the plant is simulated running the nonlinear four-wheel model. The NMPC controller on the other hand operates based on a simplified two-wheel model.

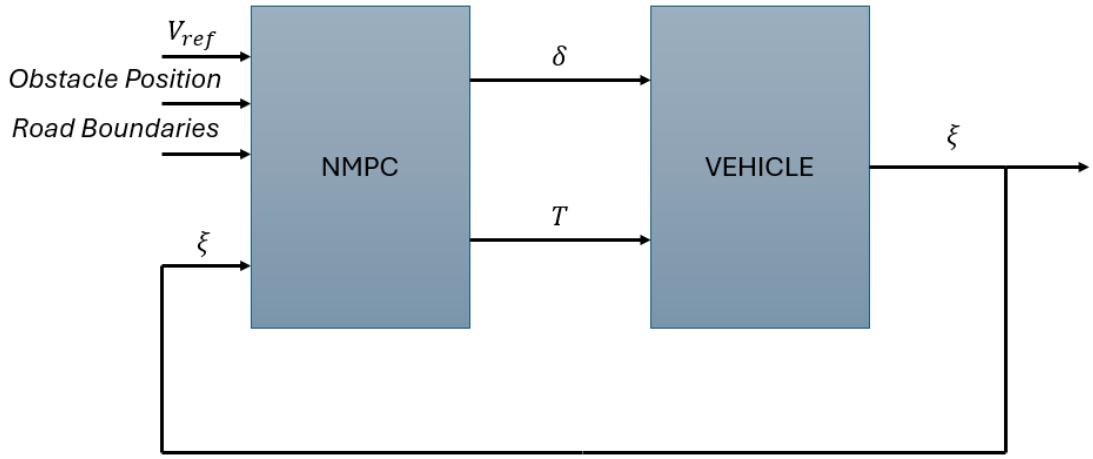

Figure 3.1: Single layer NMPC architecture

3.3 Problem description

The general formulation of a model predictive control problem can be expressed as an optimization problem, where the goal is to minimize a cost function over a prediction horizon, subject to system dynamics and constraints. The MPC problem is typically formulated as follows:

$$\min J = \sum_{i=0}^{N_p-1} (f(\xi_k, u_k) + f_f(\xi_{Np}) \tag{3.1}$$

Where:

- J is the total cost
- $f(\xi_k, u_k)$ is the running cost function that penalizes deviation from the reference trajectory and control effort at each time step k
- $f_f(\xi_{Np})$ is the terminal cost that penalizes the final state ξ_{N_p} at the end of the prediction horizon

Subject to:

1. System dynamics: the future states of the system are governed by the discrete vehicle dynamics model

$$\xi_{k+1} = f(\xi_k, u_k) \tag{3.2}$$

Where:

- ξ_k is the state vector at time step k
- u_k is the control input vector at time step k
- $f(\xi_k, u_k)$ represent the system dynamics, in this specific case the two-wheel dynamics

2. Initial conditions:

$$\xi_0 = \xi_k \tag{3.3}$$

Where ξ_k is the current state of the vehicle at the start of the optimization.

3. Control constraints

$$u_{min} \leq u_k \leq u_{max} \tag{3.4}$$

The constraint ensure that the control input remain within allowable bounds.

At each time step, the optimization problem is solved using an appropriate numerical solver, such as MATLAB fmincon. The basic syntax of the function is:

$$x = fmincon(fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options) \tag{3.5}$$

Where fun represents the objective function, the variable $x0$ denotes the initial guess for the optimization process, the parameters A and b define linear inequality constraint in the form $Ax \leq b$, similarly Aeq and beq define linear equality constraints $Aeqx = b$. To further constrain the solution, lb and ub define the lower and the upper bounds for the optimized variables, $nonlcon$ is used for nonlinear constraints and finally $options$ provides solver settings such as tolerance, step size and algorithm type allowing the user to fine-tune the optimization process to balance computational efficiency with solution accuracy. In this implementation, the standard options for the MATLAB solver were adjusted to optimize performance. Specifically, the maximum number of function evaluations was increased to 5000 to ensure the solver had enough iterations to find accurate solution within the system's constraints. Additionally, the step tolerance was increased from 10^{-10} to 10^{-4}. The reasoning behind this adjustment was that the default value led to unnecessarily long computation times without a corresponding improvement in the solution quality. By setting the tolerance to 10^{-4}, the balance between solution accuracy and computational efficiency was better aligned with the objectives of the control system, ensuring faster convergence while maintaining the necessary performance standards.

Once the solver finds the minimum the optimal values are returned inside the vector x , in our specific case the values are the control inputs $u_1, u_2, \cdots, u_{N_p}$. Exclusively the first element u_1 is applied to the system, as the receding horizon logic sudgests [23].

In this implementation, the receding horizon approach is simulated since there is no real vehicle to sample over time. Instead, the plant model is used to simulate the system. The process involves passing the initial state and the first optimal control to a function that, using the four-wheel model, compute the next state vector, which will then be imposed as the new starting point for the successive optimization. Additionally, the optimal control vector is updated by shifting it to the left. The first input of the new initial condition for the optimal control sequence becomes u_2, while the last element is set equal to u_{N_p}. This ensures continuity in the control actions over the horizon.

The selected cost function is a quadratic cost function. It minimizes the weighted sum of the tracking error and the control effort.

$$J = \sum_{i=0}^{N_p} (\xi^{ref} - \xi(i|t))^T Q(\xi^{ref} - \xi(i|t)) + u(i|t)^T R\, u(i|t) \tag{3.6}$$

Note that the penalization factor of the prediction horizon has not been considered.

The first tests were done using only a single prediction horizon. This allowed for a simpler setup, as both the prediction of the future states and the computation of the control inputs were done over the same horizon. However, after running initial tests, it became clear that using a single horizon increases computational complexity without significant improvement. The choice of implementing both prediction and control horizon ensures a balance between performance and computational efficiency. The control horizon determines how many control inputs are optimized. A shorter control horizon reduces the computational load without affecting optimal control. This is based on the principle that the most effective control actions are determined in the initial steps, while subsequent steps are less impactful. To correctly simulate the dynamics over the entire prediction horizon a new input vector will be constructed as follow:

$$u = \left[u_1, u_2, \cdots, u_{N_c}, u_{N_c+1}, \cdots, u_{N_p}\right] \tag{3.7}$$

where the first N_c values are those computed by the optimizer while the remaining values are set equal to u_{N_c}.

3.3.1 Static obstacle avoidance

The obstacle is simulated with a circle, and to ensure avoidance, the distance vector that connect the vehicle to the center of the obstacle is continuously computed. Since the objective is not only to bypass the obstacle but also to maintain a safe distance, the

constraint formulation incorporates this additional safety parameter. This ensures that the vehicle maintains a predefined buffer zone around the obstacle, rather than simply avoiding a collision. The obstacle is treated as a hard constraint, so the solver only considers solution that respect this condition.

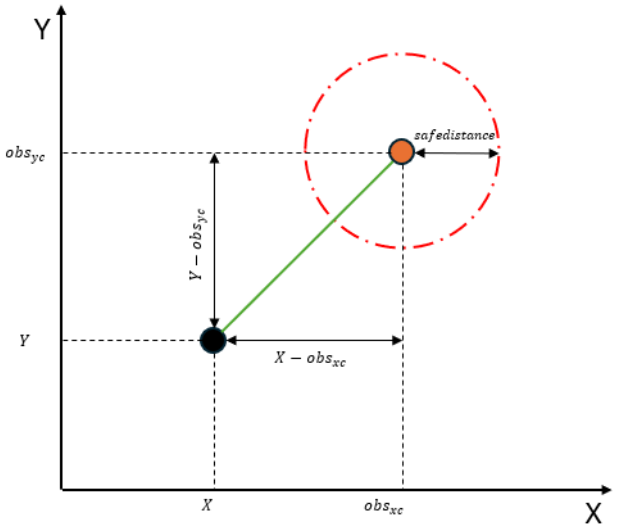

Figure 3.2: Distance computation between vehicle and obstacle

$$(X - obs_{xc})^2 + \left(Y - obs_{yc}\right)^2 \geq (obs_r + safedistance)^2 \tag{3.8}$$

Within the 'nonlcon' function of the 'fmincon' solver, the constraint has been developed in the following way: since $c(x)$ is the vector of nonlinear inequality constraints evaluated at x and fmincon attempts to satisfy $c(x) \leq 0$ for all entries, the inequality (3.8) must be rephrased:

$$c = \left[-(X - obs_{xc}).^2 - \left(Y - obs_{yc}\right).^2 + (obs_r + safedistance)^2\right]^T \tag{3.9}$$

where $X \in \mathbb{R}^{Np}$ and $Y \in \mathbb{R}^{Np}$ are the vectors that contain the longitudinal and lateral coordinates of the vehicle in the global frame, obs_{xc} and obs_{yc} are the coordinates of the obstacle position, obs_r is the obstacle radius and $safedistance$ is the value that defines how much distance you want to keep from the obstacle.

3.3.2 Dynamic obstacle avoidance

Dynamic obstacle avoidance requires a more sophisticated approach compared to static obstacles, as the vehicle must anticipate the obstacle's future positions in addition to its current location. In this chapter dynamic obstacles are considered under the assumption

that the speed of the obstacle remains constant throughout the prediction horizon. To handle this, the future positions of the obstacle are simulated within the 'nonlcon' function of the solver, taking into account its motion over time. This allows the controller to predict potential collisions and adjust the vehicle's path accordingly. The computation of distances between the vehicle and the obstacle becomes more complex due to the need to account both predictions. Unlike static obstacles, where the obstacle is considered as single and fixed point, here, both obs_{xc} and obs_{yc} are vectors.

Hence, computing:

$$c = \left[-(X - obs_{xc}).^2 - \left(Y - obs_{yc}\right).^2 + (obs_r + safedistance)^2\right]^T \qquad (3.10)$$

would not fully capture the interactions between the vehicle and the obstacle since the result is a vector $c \in \mathbb{R}^{Np}$ of distances between the i-th prediction of the vehicle and the i-th prediction of the obstacle. Applying this constraint the vehicle will start the path replanning only when the vehicle reaches the actual obstacle, losing the ability to act in advance.

The initial solution involved reversing the vector of predicted obstacle positions and calculating the difference between the vehicle's last prediction and the obstacle's first position, continuing in this manner for all subsequent predictions. This method provided a simple yet effective solution for managing the first part of the overtaking maneuver. However, while this approach performed well at the start, the vehicle's trajectory exhibited noticeable oscillations during the overtaking, leading to a less smooth path.

To provide an accurate representation of the relationship between the predicted positions of the vehicle and the obstacle, a distance matrix $c \in \mathbb{R}^{N_p \times N_p}$ is computed. This matrix compares each predicted position of the vehicle with each predicted position of the obstacle across the entire prediction horizon, ensuring that the optimization process considers all possible interactions over time to effectively avoid the obstacle.

The resulting inequality constraint, in a matrix form can be expressed as follows:

$$c_{matrix} = \left[-(X - obs_{xc}^T).^2 - \left(Y - obs_{yc}^T\right).^2 + (obs_r + safedistance)^2\right] \qquad (3.11)$$

Where $X \epsilon \mathbb{R}^{N_p}$ and $Y \epsilon \mathbb{R}^{N_p}$ are the vectors that contain the longitudinal and lateral coordinates of the vehicle in the global frame, $obs_{xc} \epsilon \mathbb{R}^{N_p}$ and $obs_{yc} \epsilon \mathbb{R}^{N_p}$ are the vectors that contain the coordinates of the obstacle position, obs_r is the obstacle radius and $safedistance$ is the value that defines how much distance you want to keep from the obstacle.

3.3.3 Lane centering

Lane centering is formulated as a soft constraint within the Model Predictive Control framework. Instead of being strictly enforced, it is included in the cost function, allowing the solver to minimize the deviation from the lane center. This approach provides flexibility, allowing the vehicle to prioritize other objectives, such as obstacle avoidance, without always requiring perfect adherence to the lane center.

As already explained in chapter (2.7), the data regarding the right bound and the central line are provided in the body frame for each vehicle position. Within the cost function, the difference between the absolute values is applied so that minimizing it the vehicle is able to maintain the lane center.

$$\min\left(\left|y^b_{middle}\right| - \left|y^b_{right}\right|\right) \tag{3.12}$$

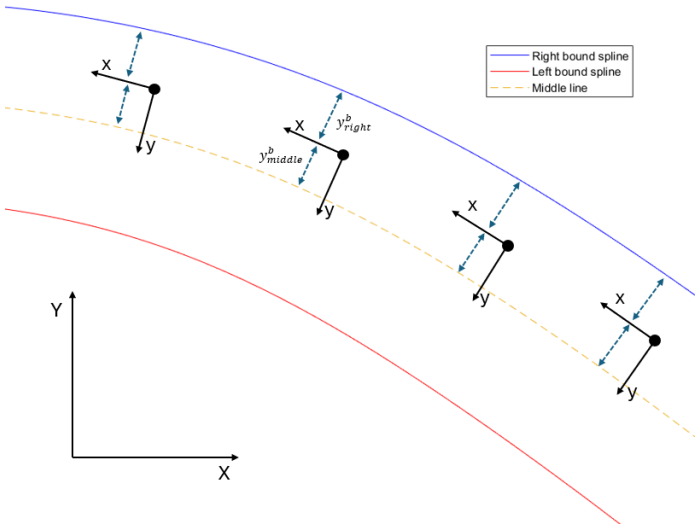

Figure 3.3: Representation of distances between body frame and lane boundaries

3.4 Single layer NMPC Controller – Formulation

In the actual MATLAB implementation, the setup begins with defining the x_0 vector, which is required by the solver and represents the initial guess for the control inputs. This vector is initially set to zero and serves as the starting point for the optimization process.

$$x_0 = \left[T_{0,1}, \cdots, T_{0,N_c}, \delta_{0,1}, \cdots, \delta_{0,N_c}\right]^T \tag{3.13}$$

Additionally, the initial conditions for the simulation are specified, in the form $[\dot{x}, \dot{y}, \dot{\psi}, \psi, X, Y]^T$, representing the system's starting state. Both vectors are updated during the simulation by the function responsible for managing the receding horizon approach, ensuring that the control inputs and states evolve dynamically as the simulation progresses.

To handle control input constraints, lower and upper bound vectors are created, defining the permissible physical range for the control actions. These bounds ensure that the control inputs remain within feasible limits during the optimization. The actual implementation is reported below.

$$lb = [T_{min,1}, \cdots, T_{min,N_c}, \delta_{min,1}, \cdots, \delta_{min,N_c}]^T \quad (3.14a)$$

$$ub = [T_{max,1}, \cdots, T_{max,N_c}, \delta_{max,1}, \cdots, \delta_{max,N_c}]^T \quad (3.14b)$$

Moreover, the solver requires the implementation of two functions, namely fun and $nonlcon$. The model is evaluated within these functions using the current value of the x vector in order to compute the value of the cost function and the constraint vector. The specific formulations of the cost function and nonlinear constraints will be provided in the following sections.

Cost function:

$$f = Q_1(y_b * y_b^T) + Q_2(V * V^T) + R_1(\delta * \delta^T) + R_2(T_r * T_r^T) \quad (3.15a)$$

$$y_b = |y^b_{middle}| - |y^b_{right}| \quad (3.15b)$$

$$V = V_{ref} - \sqrt{\dot{x}^2 + \dot{y}^2} \quad (3.15c)$$

Where Q_1, Q_2, R_1, R_2 are the weights that must be tuned, δ is the steering angle, T_r is the torque applied to the rear wheels, y^b_{middle} and y^b_{right} are explained in picture (3.3), $\dot{x}$ is the longitudinal velocity and $\dot{y}$ is the lateral velocity.

Nonlinear constraint:

$$c = [-(X - obs_{xc}).^2 - (Y - obs_{yc}).^2 + (obs_r + safe)^2, yfit_{left} - Y, -yfit_{right} + Y\ ]^T \quad (3.16)$$

The first element of the vector has been already explained in the chapter (3.3.1). The other two elements are essential for imposing the road boundaries as hard constraint for the solver. $yfit_{left}$ and $yfit_{right}$ are two vectors that contain the values of the road boundaries evaluated for each x-position assumed by the vehicle. Y, as previously stated, contains the lateral position of the vehicle.

In the dynamic obstacle avoidance, certain modifications were made to both the cost function and the nonlinear constraints to accommodate the different environment.

The lane centering inside the cost function were simplified, its aim is maintaining a constant Y-value in the global frame namely Y_{ref}. The dynamic cost function is presented below.

$$f = Q_1(y * y^T) + Q_2(V * V^T) + R_1(\delta * \delta^T) + R_2(T_r * T_r^T) \quad (3.17a)$$

$$y = Y_{ref} - Y \quad (3.17b)$$

$$V = V_{ref} - \sqrt{\dot{x}^2 + \dot{y}^2} \quad (3.17c)$$

While inside the nonlinear constraint, instead of the fit values relative to the lane boundaries there are simplified constraints on the longitudinal position of the vehicle in the global frame since it is moving on a straight road.

$$c = [c_{matrix}(:), Y - leftbound, Y - rightbound] \quad (3.18)$$

Where the first element is the matrix of distances transformed into a vector, Y contains the lateral positions of the vehicle and $leftbound$ and $rightbound$ are constant values representing the road boundaries.

Chapter 4

NMPC – Double layer

4.1 Introduction

The double layer architecture is designed to improve both computational efficiency and control performance in complex driving scenarios. In this structure, the control problem is split into two distinct layers: a higher-level planner and a lower-level controller. This division allows for more efficient handling of complex scenarios, ensuring better scalability and faster computational time.

4.2 Double layer NMPC architecture

The higher layer generates a feasible path based on the vehicle's state and on the environment analysis. It uses a simplified point mass model, which assumes a constant longitudinal velocity and optimize the lateral acceleration required to keep the desired trajectory. This layer processes inputs such as the position of the obstacles and the road boundaries. It outputs a reference trajectory that ensures the avoidance of the obstacles and stay centered within the lane.

The lower layer takes the reference trajectory from the path planner and computes the necessary control actions using a more detailed two-wheel model. It ensures that the vehicle follows the reference path while adjusting the velocity as needed to adapt to the road conditions.

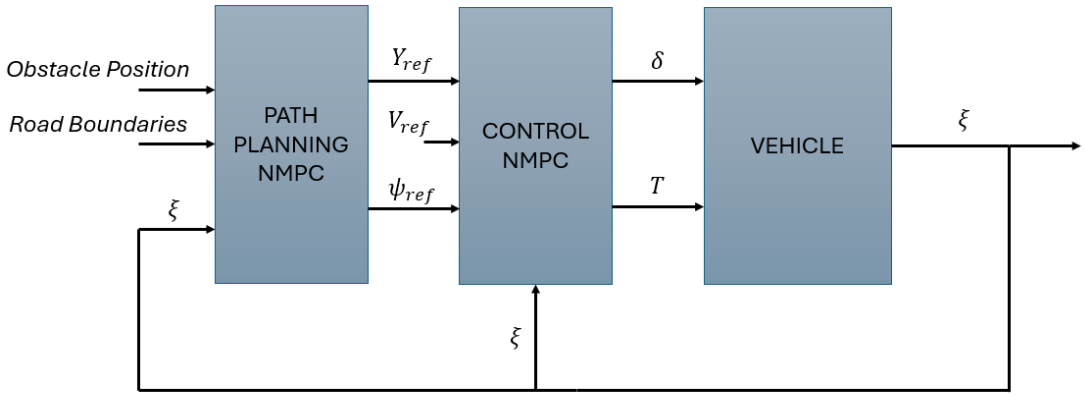

Figure 4.1: Double layer NMPC architecture

4.3 High level MPC – Formulation

In the actual MATLAB implementation of the high-level path planning algorithm, the initial condition vector $x_0$is initialize to zero, this time it has a different size since the input is only the lateral acceleration:

$$x_{0,hl} = \left[a_{y0,1}, \cdots, a_{y0,N_c}\right]^T \tag{4.1}$$

The initial state vector is shared between the two layers. However, since the state vector has different sizes between the point mass model used in the high-level path planner and the two-wheel model used in the low-level controller, an adaptation was necessary at the code level. The vector form remains unchanged, but the point mass excludes the evaluation of $\dot{\psi}$ dynamics.

Lower and upper bound vectors are then defined to ensure that control inputs constraints, such as acceleration limits, are respected during the optimization.

$$lb_{hl} = \left[a_{y_{min},1}, \cdots, a_{y_{min},N_c}\right] \tag{4.2a}$$

$$ub_{hl} = \left[a_{y_{max},1}, \cdots, a_{y_{max},N_c}\right] \tag{4.2b}$$

The values of maximum lateral acceleration are computed as follows:

$$a_y \leq |g\mu| \tag{4.3}$$

The specific formulations of the high-level cost function and nonlinear constraints will be provided in the following sections.

Cost function:

$$f = Q_1(y_b * y_b^T) + R_1(u * u^T) \quad (4.4a)$$

$$y_b = \left|y_{middle}^b\right| - \left|y_{right}^b\right| \quad (4.4b)$$

$$u = \left[a_{y,1}, \cdots, a_{y,Np}\right] \quad (4.4c)$$

Where Q_1, R_1 are the weights that must be tuned, y_{middle}^b and y_{right}^b are explained in picture (3.3) and $a_y$is the lateral acceleration.

The nonlinear constraint vectors are exactly the same as the ones implemented in section (3.4).

The output of this optimization process is the lateral acceleration. However, in order to generate the trajectory references for the low-level an additional step is required. A dedicated function, that takes as input the optimized lateral accelerations and computes the corresponding point-mass trajectory, has been developed. It returns two vectors of length N_p, representing the lateral position and yaw angle of the point mass over the entire prediction horizon.

4.4 Low level MPC – Formulation

The steps for the low-level controller are the same of the previous implementations.

The initial conditions vector is in the form:

$$x_{0,ll} = \left[T_{0,1}, \cdots, T_{0,N_c}, \delta_{0,1}, \cdots, \delta_{0,N_c}\right] \quad (4.5)$$

The initial state vector is the same as the high-level, but in this case the angular rate is taken into account for the two-wheel dynamics.

Control input constraint are the ones written in formula number (4.2a) and (4.2b).

The specific formulations of the low-level cost function will be provided in the following sections.

$$f = Q_1\left(Y_f * Y_f^T\right) + Q_2\left(\psi_f * \psi_f^T\right) + Q_3(V * V^T) + R_1(\delta * \delta^T) + R_2(T_r * T_r^T) \quad (4.6a)$$

$$Y_f = Y_{ref} - Y \tag{4.6b}$$

$$\psi_f = \psi_{ref} - \psi \tag{4.6c}$$

$$V = V_{ref} - \dot{x} \tag{4.6d}$$

Where Q_1, Q_2, Q_3, R_1, R_2 are the weights, Y_{ref} and ψ_{ref} are the reference vectors generated by the high-level.

Chapter 5

Control schemes simulation

5.1 Introduction

The control scheme comparison was conducted using two distinct simulation environments to evaluate the performance of both NMPC architecture in different scenarios.

The first environment was designed to test lane centering and static obstacle avoidance. For this purpose, a model of an actual road was developed, incorporating realistic road boundaries and lane markings to reflect common driving conditions. This environment allowed for a comprehensive assessment of how well each control architecture managed to maintain the vehicle within its lane while avoiding static obstacles.

The second simulation environment was designed to facilitate the study of dynamic obstacle avoidance. In this case, a straight road was simulated to simplify the evaluation of the vehicle's behavior while avoiding moving objects. The road was dimensioned using the average width of typical lanes and streets, ensuring a realistic but simplified testing scenario. This environment allowed for testing of how each control scheme handled dynamic obstacles.

The simulation setup involves defining key parameters such as the prediction horizon (N_p) and the control horizon (N_c), both of which influence the controller's performance. The sampling time of the controller dictates how frequently control inputs are updated. Additionally, the vehicle's initial state, such as its starting position, velocity and yaw angle, is specified.

The weights employed in the cost function are of critical importance for the tuning of the controller's behavior. These weights serve to balance between the minimization of control effort, the maintenance of lane centering and the avoidance of obstacles.

Along with these parameters, the characteristics of the obstacle, including its dimension, the required safe distance and its velocity, are configured to simulate realistic driving scenario.

5.2 Single layer – Results

5.2.1 Lane centering and static obstacle avoidance

To conduct the simulation, the following values were assigned to the key parameters:

Table 5.1: Single layer parameters for LC and SOA

	Value	*Unit*
Prediction horizon	20	-
Control horizon	2	-
Sampling time	0.1	s
Initial state $[\dot{x}, \dot{y}, \dot{\psi}, \psi, X, Y]$	$[2.\bar{7}, 0, 0, -\pi, -830, 893]$	$[m/s, m/s, rad/s, rad, m, m]$
Upper bound $[T, \delta]$	[890,0.61]	$[Nm, rad]$
Lower bound $[T, \delta]$	$[-5100, -0.61]$	$[Nm, rad]$
Velocity reference [m/s]	35/3.6	m/s
Obstacle radius [m]	1	m
Safe distance [m]	2	m

The images below display the simulation results.

- Vehicle trajectory

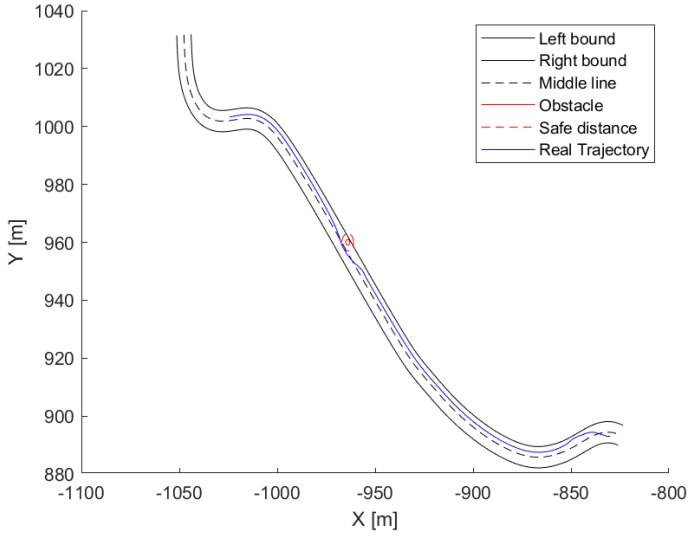

Figure 5.1: Single layer vehicle trajectory in SOA and LC simulation

- Velocity

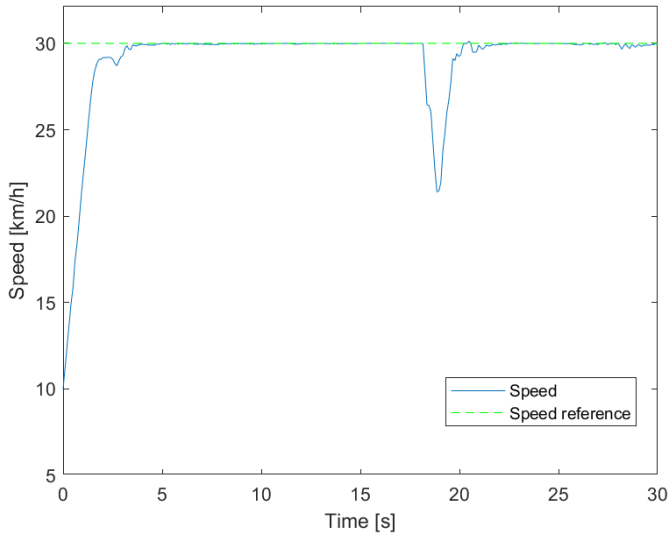

Figure 5.2: Single layer velocity in SOA and LC simulation

- Torque

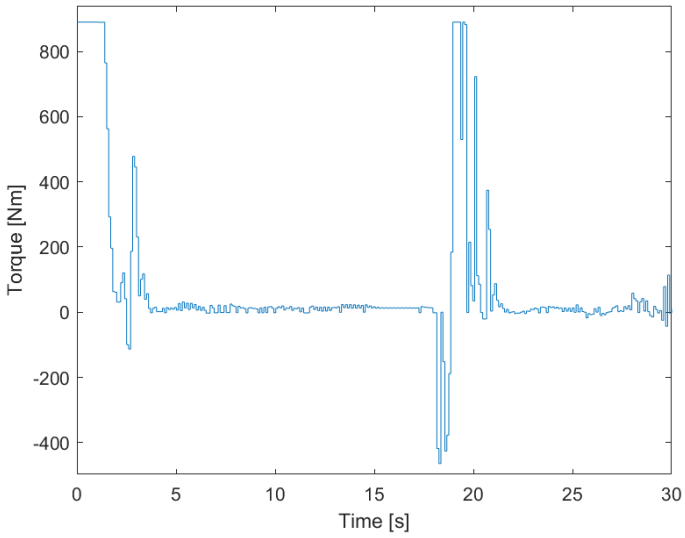

Figure 5.3: Single layer torque in SOA and LC simulation

- Steering angle

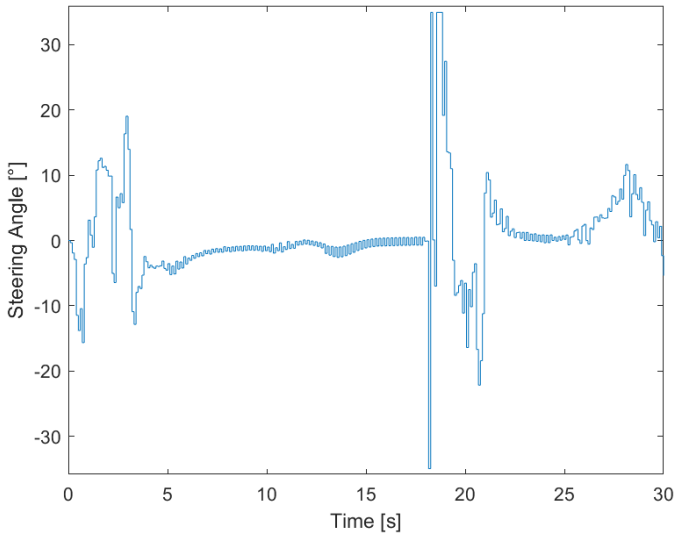

Figure 5.4: Single layer steering angle in SOA and LC simulation

5.2.2 Dynamic obstacle avoidance

Representing the trajectories of both the vehicle and the obstacle simultaneously, inside the same picture represents a significant challenge since both are moving over time. The most effective way to visualize this is by displaying the complete trajectory of the vehicle, while showing only two key positions of the obstacle. The first obstacle position starting from the left represent the moment when the vehicle starts the overtaking maneuver, and the second represents the obstacle's location when the vehicle begins to shift back to the right lane.

Additionally, to further clarify the dynamic behavior, another plot has been included to depict the distance between the vehicle and the obstacle over time. In this plot, the dashed line represents the minimum allowed distance, which is the sum of the obstacle's radius and the safe distance specified by the user. This visualization ensures a clear understanding of how the vehicle maintains the necessary safety margin throughout the maneuver.

In order to conduct the simulation, the following values were assigned to the key parameters:

Table 5.2: Single layer parameters for DOA

	Value	*Unit*
Prediction horizon	20	-
Control horizon	2	-
Sampling time	0.1	s
Initial state $[\dot{x}, \dot{y}, \dot{\psi}, \psi, X, Y]$	$[6.9\overline{4}, 0, 0, 0, 0, 3\ ]$	$[m/s, m/s, rad/s, rad, m, m]$
Upper bound $[\ T, \delta]$	[890,0.61]	$[Nm, rad]$
Lower bound $[\ T, \delta]$	$[-5100, -0.61]$	$[Nm, rad]$
Velocity reference [m/s]	35/3.6	m/s
Obstacle radius [m]	1	m
Obstacle velocity [m/s]	25/3.6	m/s
Safe distance [m]	2	m

The images below display the simulation results

- Vehicle's trajectory

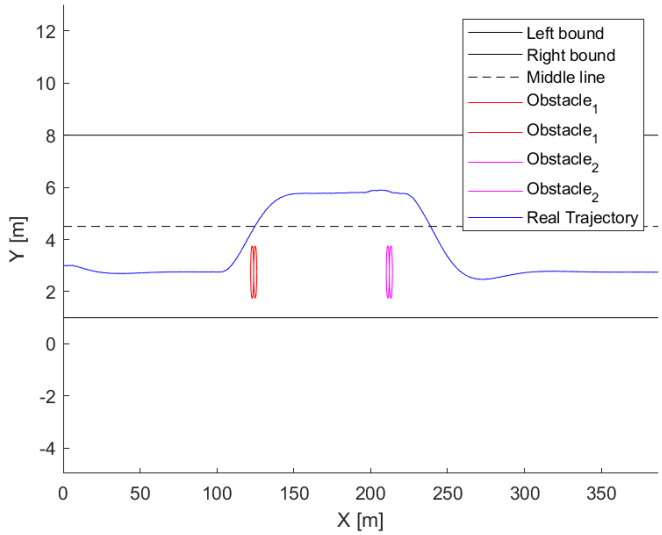

Figure 5.5: Single layer vehicle trajectory in DOA simulation

- Velocity

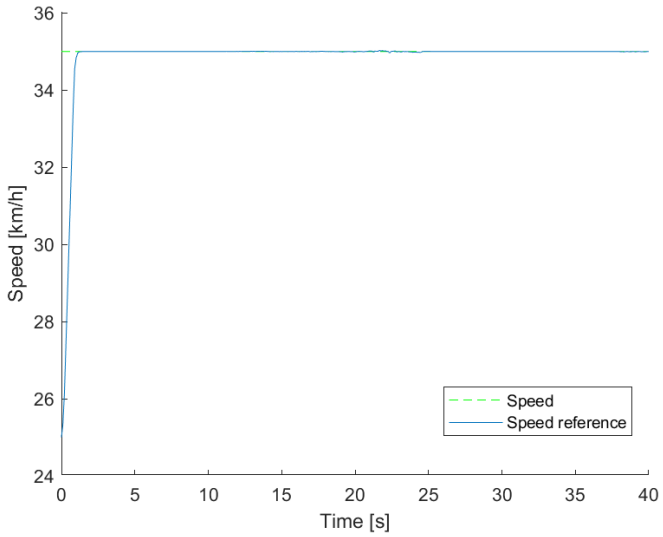

Figure 5.6: Single layer velocity in DOA simulation

- Torque

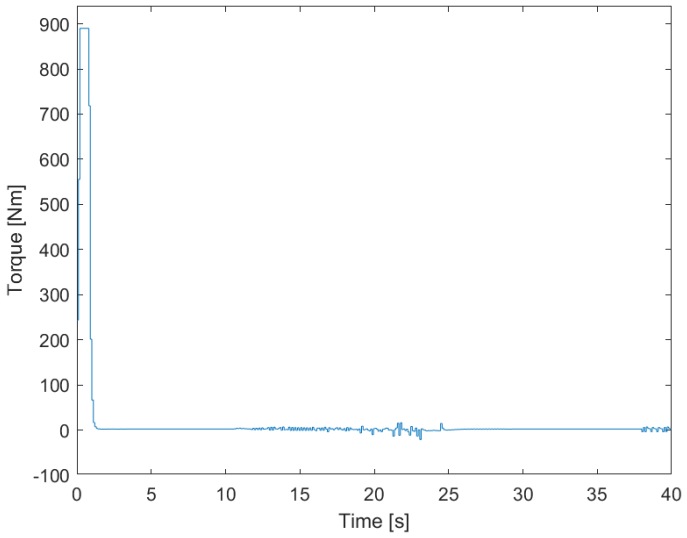

Figure 5.7: Single layer torque in DOA simulation

- Steering angle

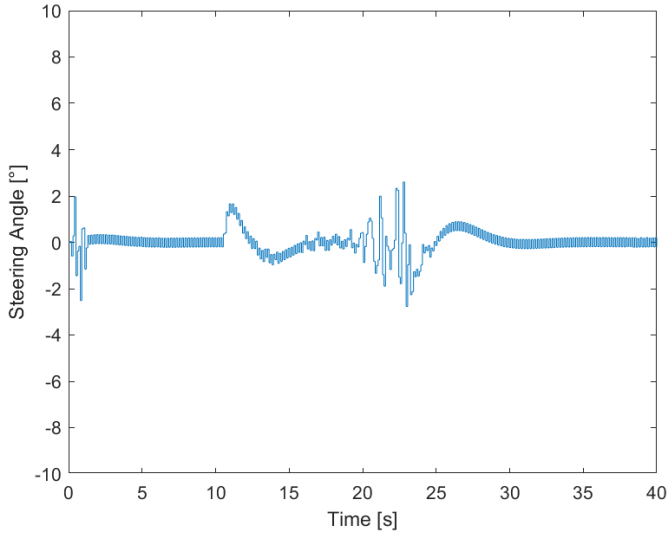

Figure 5.8: Single layer steering angle in DOA simulation

- Distance between the vehicle and the center of the obstacle

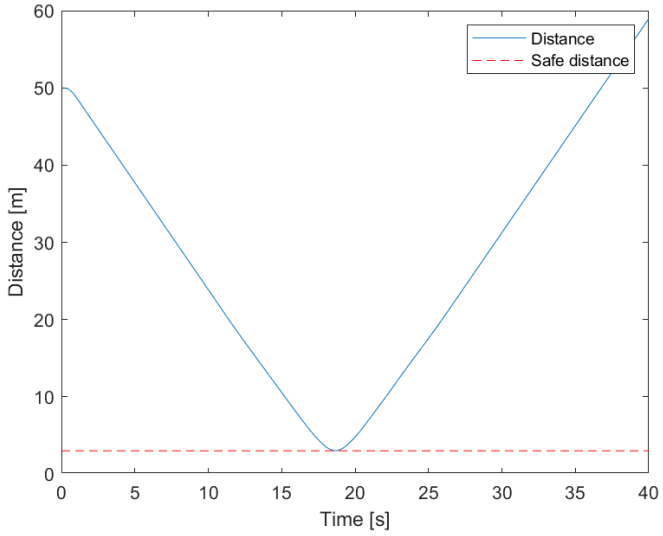

Figure 5.9: Single layer distance between vehicle and obstacle in DOA simulation

5.3 Double layer – Results

5.3.1 Lane centering and static obstacle avoidance

In order to conduct the simulation, the following values were assigned to the key parameters:

Table 5.3: Double layer parameters for LC and SOA

	Value	*Unit*
Prediction horizon	20	-
Control horizon	2	-
Sampling time	0.1	s
Initial state $[\dot{x}, \dot{y}, \dot{\psi}, \psi, X, Y]$	$[2.\bar{7}, 0, 0, -\pi, -830, 893]$	$[m/s, m/s, rad/s, rad, m, m]$
Upper bound high level $[a_y]$	$[0.9 \cdot 9.81]$	$[m/s^2]$
Lower bound high level $[a_y]$	$[-0.9 \cdot 9.81]$	$[m/s^2]$
Upper bound low level $[T, \delta]$	[890,0.61]	$[Nm, rad]$
Lower bound low level $[T, \delta]$	$[-5100, -0.61]$	$[Nm, rad]$
Velocity reference [m/s]	35/3.6	m/s
Obstacle radius [m]	1	m
Safe distance [m]	2	m

The images below display the simulation results

- Vehicle's trajectory

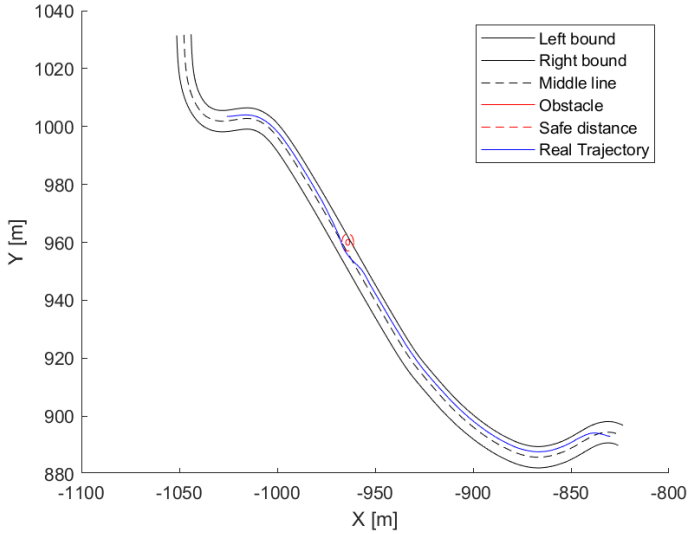

Figure 5.10: Double layer vehicle trajectory in SOA and LC simulation

- Velocity

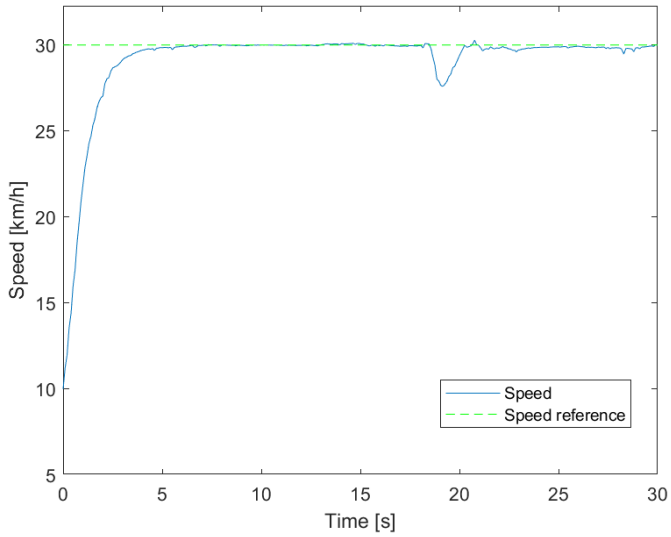

Figure 5.11: double layer velocity in SOA and LC simulation

- Torque

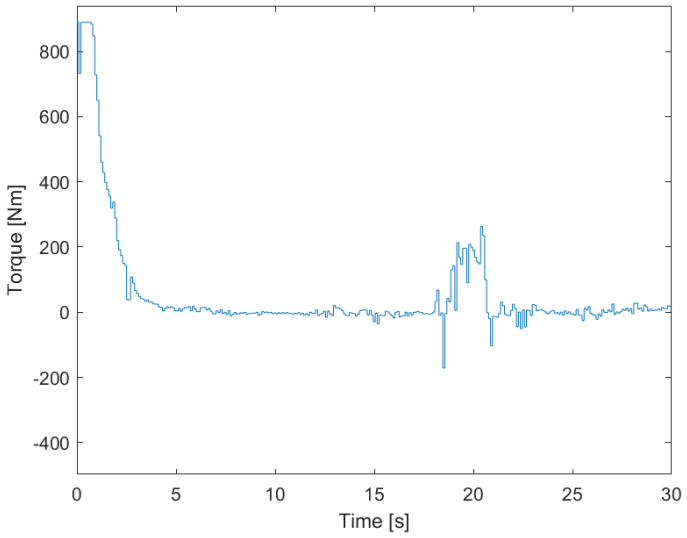

Figure 5.12: Double layer torque in SOA and LC simulation

- Steering angle

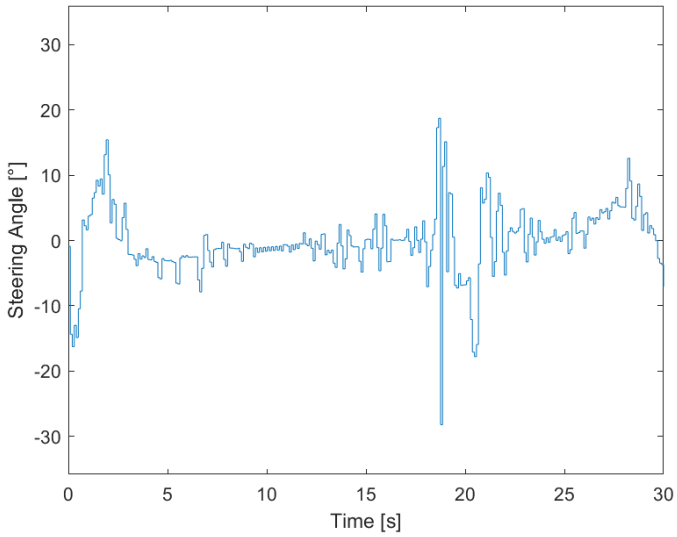

Figure 5.13: Double layer steering angle in SOA and LC simulation

5.3.2 Dynamic obstacle avoidance

The same considerations outlined in the paragraph (5.2.2) have been applied to the subsequent simulations.

In order to conduct the simulation, the following values were assigned to the key parameters:

Table 5.4: Double layer parameters in DOA

	Value	*Unit*
Prediction horizon	20	-
Control horizon	2	-
Sampling time	0.1	s
Initial state $[\dot{x}, \dot{y}, \dot{\psi}, \psi, X, Y]$	$[6.9\bar{4}, 0, 0, 0, 0, 3\ ]$	$[m/s, m/s, rad/s, rad, m, m]$
Upper bound high level $[a_y]$	$[0.9 \cdot 9.81]$	$[m/s^2]$
Lower bound high level $[\ a_y]$	$[-0.9 \cdot 9.81]$	$[m/s^2]$
Upper bound low level $[\ T, \delta]$	[890,0.61]	$[Nm, rad]$
Lower bound low level $[\ T, \delta]$	$[-5100, -0.61]$	$[Nm, rad]$
Velocity reference [m/s]	35/3.6	m/s
Obstacle radius [m]	1	m
Obstacle velocity [m/s]	25/3.6	m/s
Safe distance [m]	2	m

The images below display the simulation results

- Vehicle's trajectory

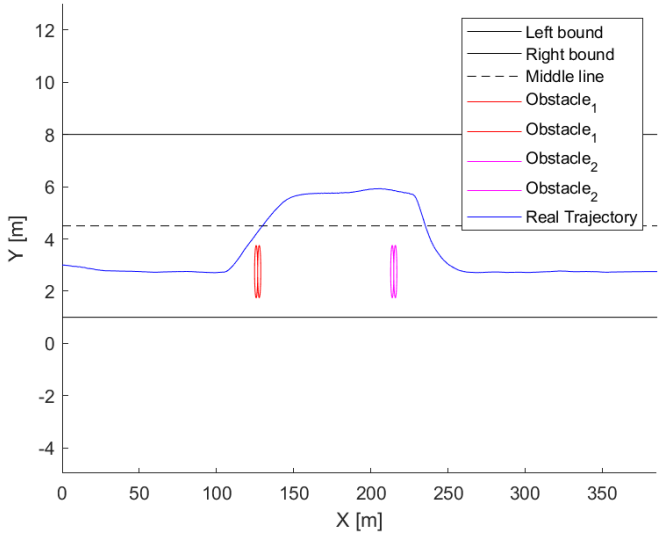

Figure 5.14: Double layer vehicle trajectory in DOA simulation

- Velocity

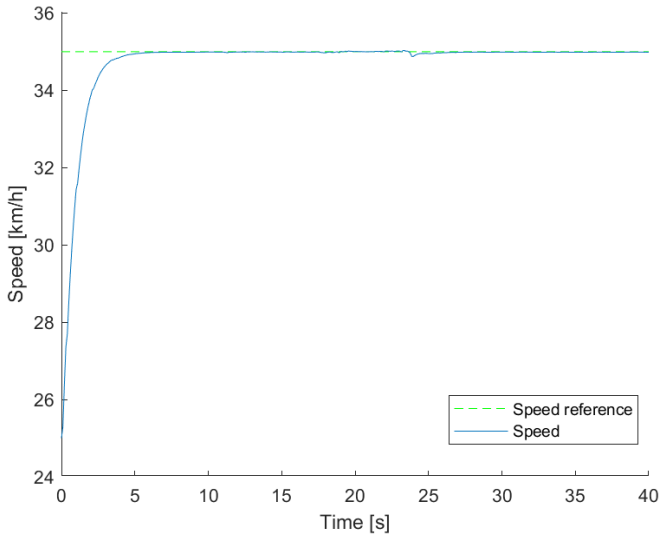

Figure 5.15: Double layer velocity in DOA simulation

- Torque

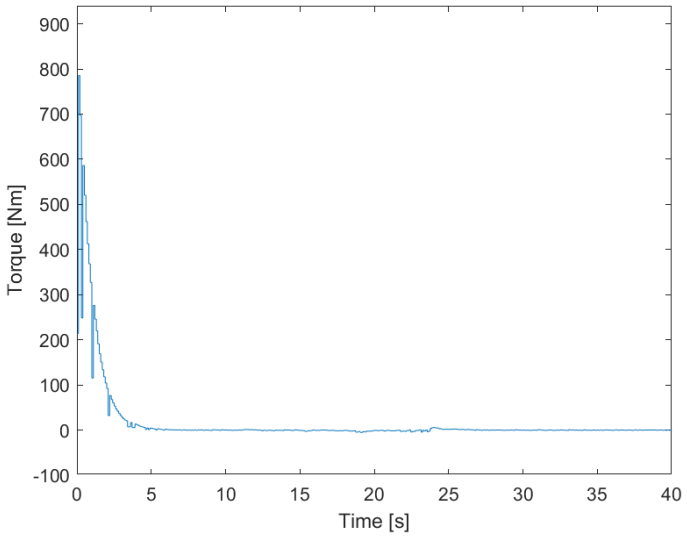

Figure 5.16: Double layer torque in DOA simulation

- Steering angle

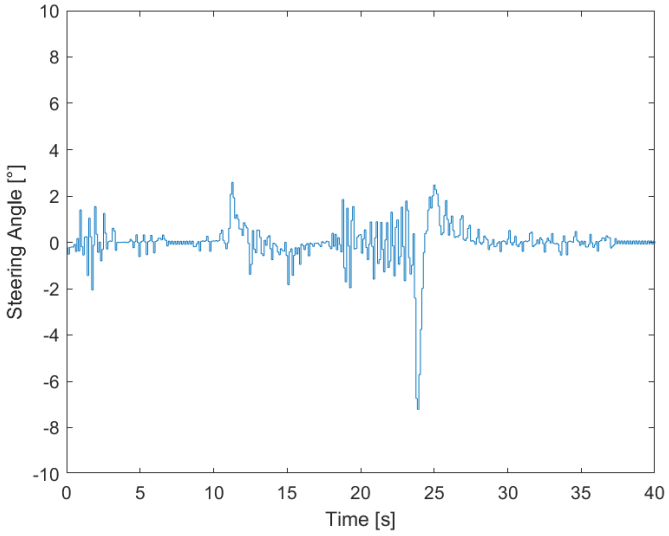

Figure 5.17: Double layer steering angle in DOA simulation

- Distance between the vehicle and the center of the obstacle

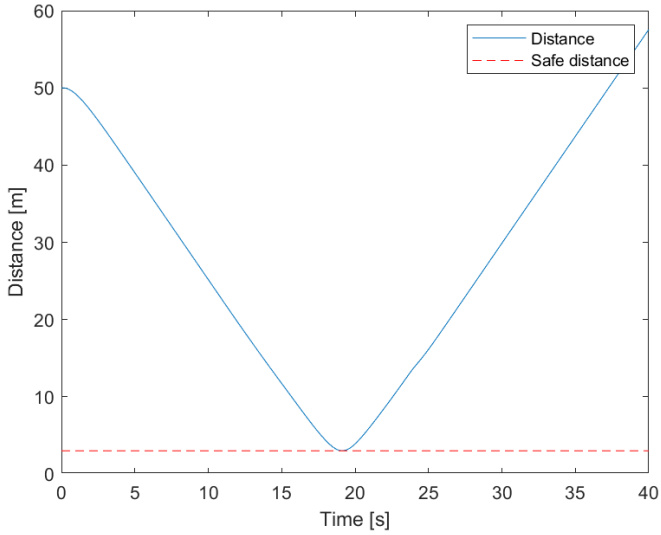

Figure 5.18: Double layer distance between vehicle and obstacle in DOA simulation

Chapter 6

Performance comparison

6.1 Introduction

In this chapter, a detailed comparison of the control schemes is provided, focusing on two main aspects: the execution time and driving comfort. The execution time analysis includes both the mean and maximum execution times, as these are crucial for evaluating the computational efficiency and ensuring real-time feasibility of each control architecture. The mean execution time offers insight into the average computational load, while the maximum execution time helps identify the worst-case scenarios that could affect performance. Additionally, the variance in execution times is also considered, as it highlights the consistency and predictability of the system. A lower variance indicates more stable computational performance, reducing the likelihood of unexpected delays during operation. Driving comfort is assessed through the smoothness of the generated trajectory and more specifically computing the lateral and longitudinal jerk. There are numerous studies that emphasize the importance of jerk in various fields, such as entertainment[24],[25], automotive[26] and elevator[27]. In the entertainment industry, particularly in amusement parks rides, controlling jerk is crucial to balance thrill and comfort for the passengers. Similarly, in the automotive sector, jerk is a critical parameter in the design of advanced driver-assistant system and autonomous driving, where passenger comfort is a key factor. Even in elevator design, jerk is carefully controlled to provide smoot acceleration and deceleration.

6.2 Execution time

The execution time for each simulation was calculated using a MATLAB timer. The time measurement starts as soon as the solver begins the optimization procedure and ends when the optimal control solution is obtained. This time is recorded for each step of the simulation, providing a precise evaluation of how efficiently the solver computes the control inputs. The average, the mean and the variance of these values are then used to conduct the comparison of the computational efficiency.

Static obstacle avoidance

The analysis of execution times for the static obstacle avoidance reveals distinct differences between the two architectures, both in terms of speed and consistency. The mean execution time for the base architecture is 0.61s, whereas the double-layer system reduces this to 0.4s. The reduction offered by the double-layer approach signifies a more efficient handling of the computational demand.

Looking at the maximum execution times provides further insight into the performance of each architecture. In the single layer architecture, the maximum execution time reaches 1.77s, significantly higher than the 1.27s observed in the double-layer system. Excluding the initial optimization step, which tends to be more computationally intensive, the maximum execution time drops significantly to 0.68s. This difference suggests that, in critical situations, the double-layer architecture is more reliable, as it minimizes the potential for the delayed control actions, particularly during complex scenarios like obstacle avoidance.

The variance in execution time is another crucial factor in assessing performance. The base architecture exhibits a variance of 0.0526, indicating a larger fluctuation in computational time. In contrast, the double-layer architecture shows a much smaller variance of 0.0132. This smaller value reflects the ability of the double-layer system to maintain consistent performance across multiple iterations, ensuring that even during complex computations, the control system operates within a predictable time frame.

The following graph illustrate the evolution of the base architecture execution time:

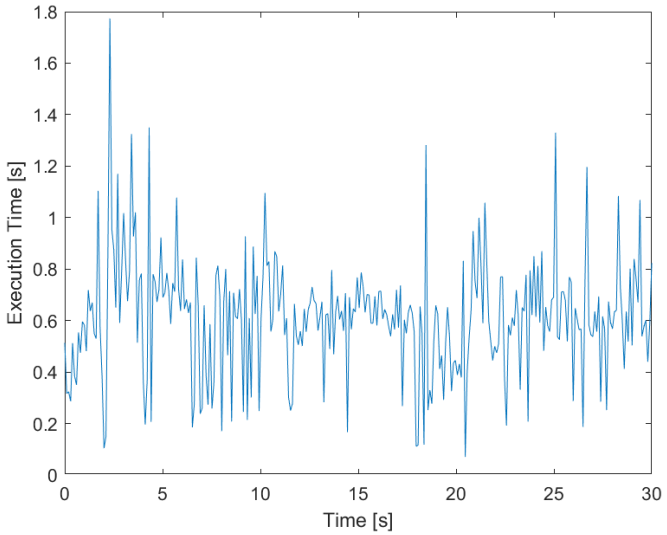

Figure 6.1: Execution time of the single layer architecture in SOA and LC simulation

The following graph illustrate the evolution of the double-layer architecture execution time:

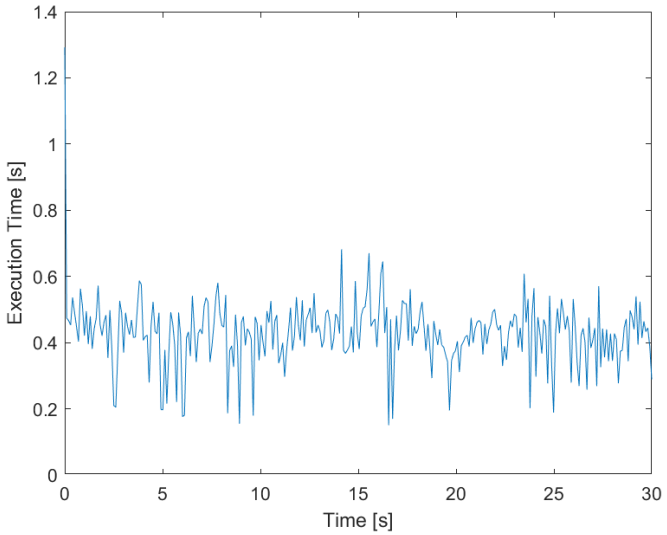

Figure 6.2: Execution time of the double layer architecture in SOA and LC simulation

Table 6.1: Summary of execution time values in SOA and LC simulation

	Single layer	*Double layer*	*Unit*
Mean execution time	0,61	0,4	s
Max execution time	1,77	1,27-0,68	s
Variance	0,0526	0,0132	-

Dynamic obstacle avoidance

In the dynamic obstacle avoidance scenario, the comparison between the single and double-layer architectures again reveals significant differences in computational performance. For the base architecture, the mean execution time is 0,13s, whereas the double-layer approach achieves a reduced mean time of 0,067s. This improvement in speed is essential when considering the real-time nature of dynamic obstacle avoidance, where rapid response is essential to avoid collisions.

The maximum execution time in the base architecture, however, is a major concern, reaching 2,63s. Such a delay could severely impact the system's ability to react in a dynamic environment, particularly when the obstacle's position changes rapidly. On the other hand, the double- layer system shows a much lower maximum execution time of 0,81s. It is worth noting that this maximum time is further reduced to 0,18s when the first optimization step is excluded, highlighting the efficiency of the double-layer system in handling the majority of optimization cycles after the first step. This drop indicates that the double-layer architecture is more adept at maintaining responsiveness once the initial conditions have been optimized.

The variance in execution times also support this observation. The base architecture has a variance $\sigma = 0{,}032$, indicating a wider fluctuation in computational time, which could result in inconsistent performance during critical avoidance maneuvers. By contrast, the double-layer system exhibits a variance σ of just 0,0018, demonstrating a much more consistent and reliable performance throughout the simulation.

The following graph illustrate the evolution of the base architecture execution time:

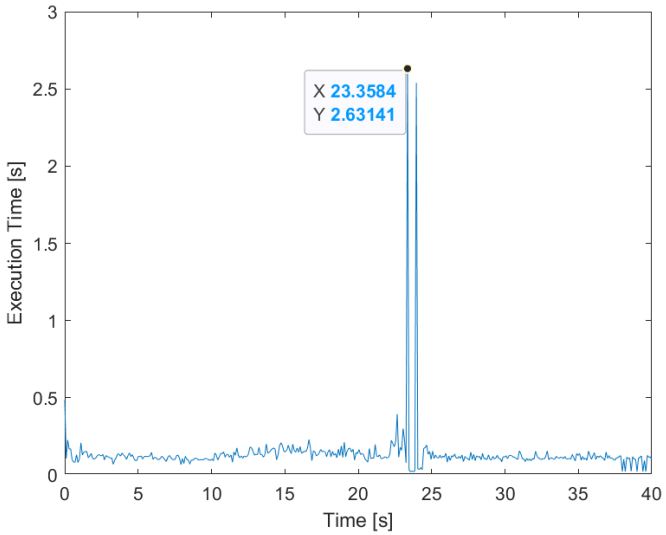

Figure 6.3: Execution time of the double layer architecture in DOA simulation

The following graph illustrate the evolution of the double-layer architecture execution time:

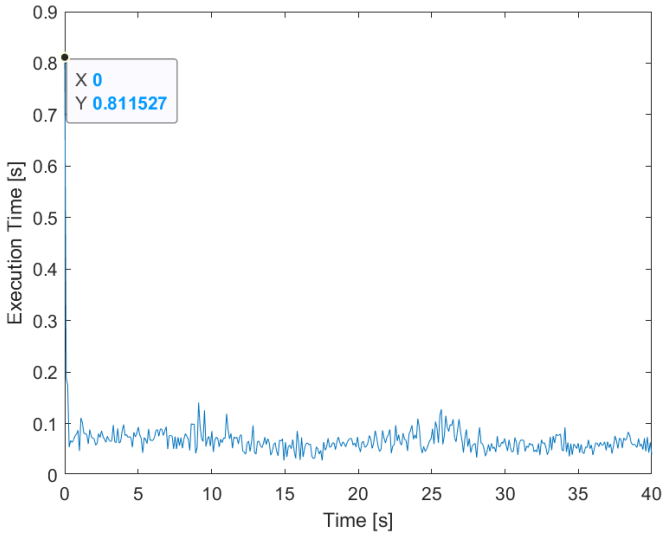

Figure 6.4: Execution time of the double layer architecture in DOA simulation

Table 6.2: Summary of time values in DOA simulation

	Single layer	*Double layer*	*Unit*
Mean execution time	0,13	0,067	s
Max execution time	2,63	0,81-0,18	s
Variance	0,032	0,0018	-

6.3 Driving comfort

Driving comfort can initially be assessed by observing the generated vehicle trajectory. A smooth and stable path generally indicates a comfortable driving experience, whereas irregular or sudden changes in direction may suggest discomfort for passengers.

The first image, depicting the zoomed trajectory generated by the base architecture, highlights some noticeable oscillations in the vehicle's path. These oscillations suggest that the control system is struggling to maintain a smooth and consistent trajectory, likely due to the simultaneous handling of path planning and control input optimization. The result is a trajectory that, while functional, lacks the precision and smoothness desirable for a comfortable driving experience.

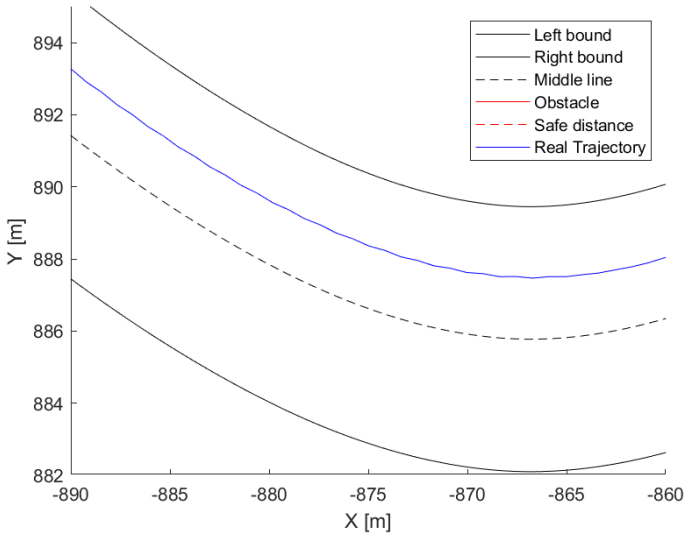

Figure 6.5: Section of the vehicle trajectory generated by the single layer architecture

In contrast, the second image, showcasing the zoomed trajectory generated by the double-layer architecture, presents a far smoother trajectory. By decoupling the tasks of path planning and control input computation, the double-layer approach enables better handling of the vehicle's dynamics, resulting in a more refined trajectory. This improvement not only enhances driving comfort but also reflects the efficiency of the architecture.

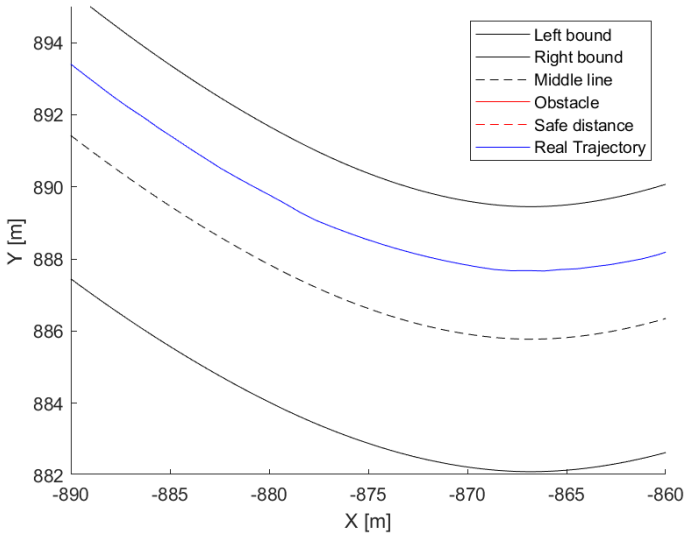

Figure 6.6: Section of the vehicle trajectory generated by the double layer architecture

When comparing the two architectures' behavior in static obstacle avoidance, notable differences emerge, particularly near the obstacle.

For the base architecture, as the vehicle approaches the obstacle, the steering angle starts oscillating significantly, with values bouncing between the maximum and minimum bounds, indicating that the system is struggling to maintain stable control. This instability is also reflected in the torque, where sudden changes are observed, further emphasizing the high control effort required by the base architecture to navigate around the obstacle. Additionally, a small ripple is visible in the image (6.7) of the steering angle which is likely the cause of the oscillation in the generated trajectory. These fluctuations not only contribute to an uncomfortable driving experience, but also highlight inefficiencies in control execution.

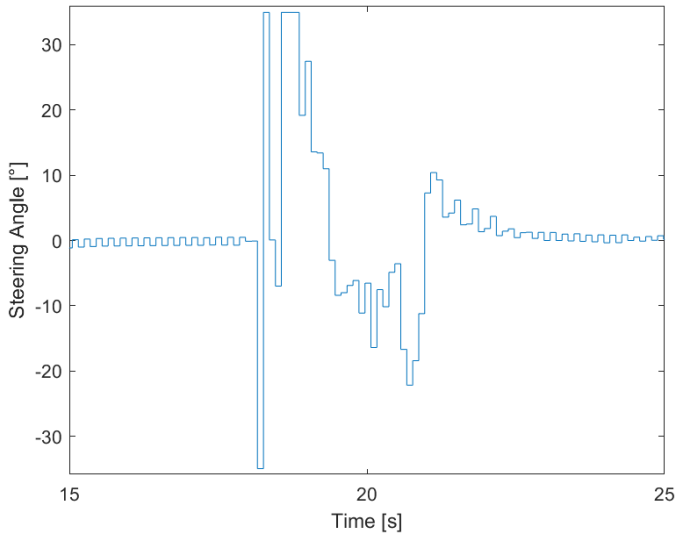

Figure 6.7: Section of the steering angle generated by the single layer architecture while approaching the obstacle

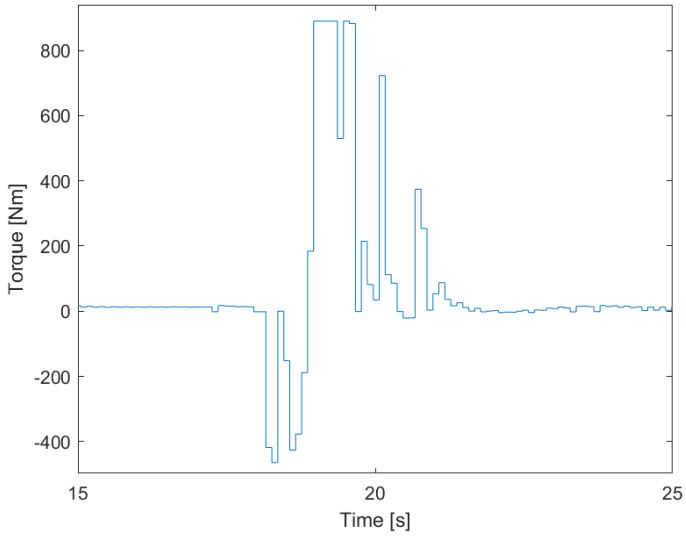

Figure 6.8: Section of the torque generated by the single layer architecture while approaching the obstacle

In contrast, the double-layer architecture demonstrates far more consistent and refined control. The steering angle remains within a lower range, suggesting smoother handling. Similarly, the torque is much more controlled, with fewer sharp variations. The lower control effort indicates that the double-layer architecture is more effective in managing the vehicle's trajectory and dynamics, minimizing sharp corrections and improving driving comfort.

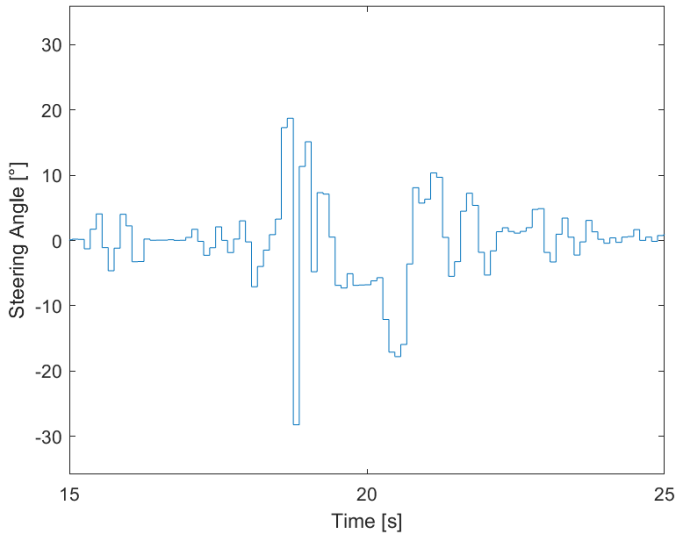

Figure 6.9: Section of the steering angle generated by the double layer architecture while approaching the obstacle

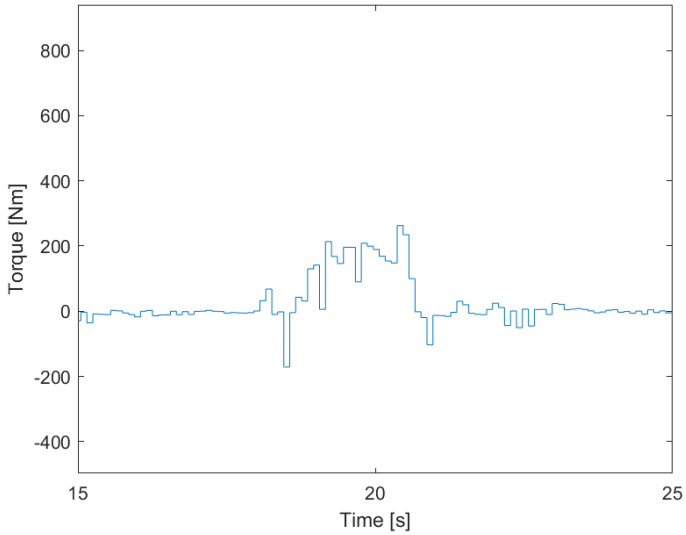

Figure 6.10: Section of the torque generated by the double layer architecture while approaching the obstacle

However, a more precise evaluation of the driving comfort can be conducted by analyzing the jerk, which represent the rate of change of acceleration over time. Jerk is a critical measure in assessing the smoothness of a vehicle's motion, as it captures sudden shifts in force that are often perceptible to passengers. High jerk values indicate rapid changes in acceleration, which can cause discomfort while driving, particularly in maneuvers like lane changes or obstacle avoidance.

By quantifying both lateral and longitudinal jerk, a more detailed understanding of the vehicle's dynamics can be obtained. Lateral jerk affects the sideways forces experienced by passengers, while longitudinal jerk influences the forward and backward forces during acceleration and braking. A lower jerk is correlated with smoother, more comfortable driving, as the forces are applied in a more gradual and controlled manner. Through this detailed analysis, the overall performance of the control scheme can be more comprehensively evaluated.

Static obstacle avoidance

In comparing the lateral jerk between the two architectures for the static obstacle avoidance and lane centering, the base architecture shows a maximum value of 176,73 [m/s^3] and a variance $\sigma = 11760$, indicating significant fluctuations in the vehicle's lateral motion. This higher jerk value reflects the system's struggle to maintain a smooth trajectory, as observed in previous analyses. On the other hand, the double-layer architecture demonstrates a lower maximum lateral jerk of 163,75 [m/s^3] and a much smaller variance $\sigma = 2011$. The reduced jerk and variance in the double-layer system indicate a smother control response, resulting in more stable and comfortable lateral movement during maneuvers.

Below the two plots of the lateral jerk.

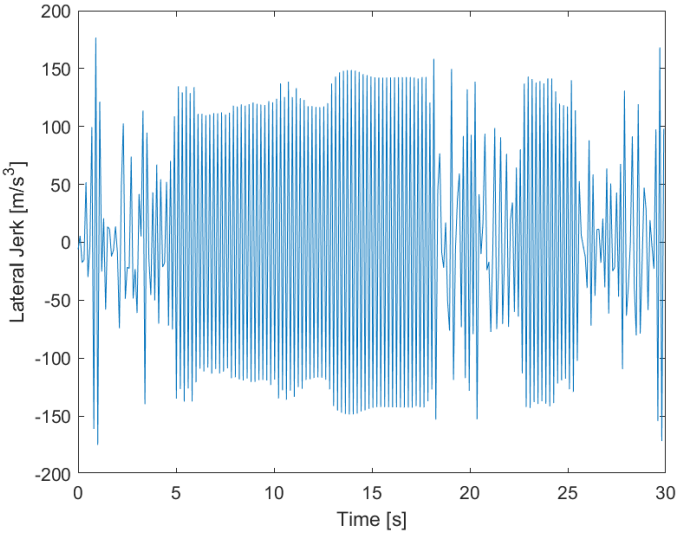

Figure 6.11: Single layer lateral jerk in SOA and LC simulation

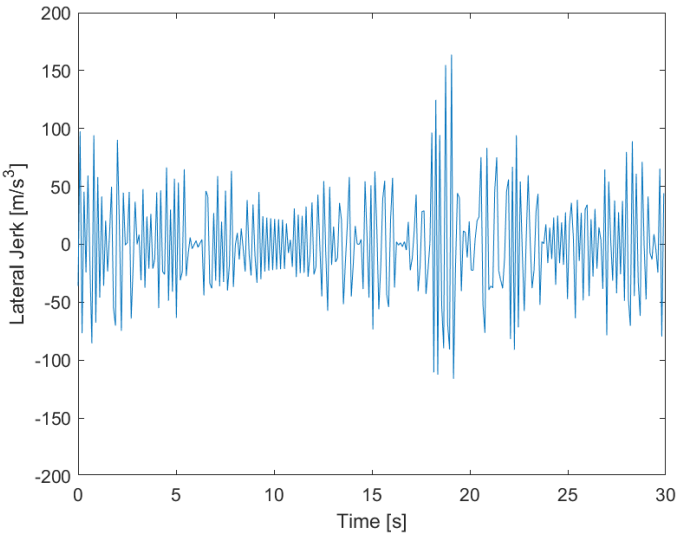

Figure 6.12: Double layer lateral jerk in SOA and LC simulation

Turning to the longitudinal jerk, the base architecture exhibits a maximum value of 49,67 [m/s^3] and a variance $\sigma = 73{,}19$, again highlighting the more aggressive control efforts in acceleration and braking. The double-layer architecture, however, significantly outperforms the base system in this area, with a lower maximum longitudinal jerk of 35,14 [m/s^3] and a variance σ of just 32,81. This suggests that the double-layer architecture not only improves lateral smoothness but also ensures more controlled and comfortable longitudinal behavior, further contributing to a better overall driving experience.

Below the two plots of the longitudinal jerk

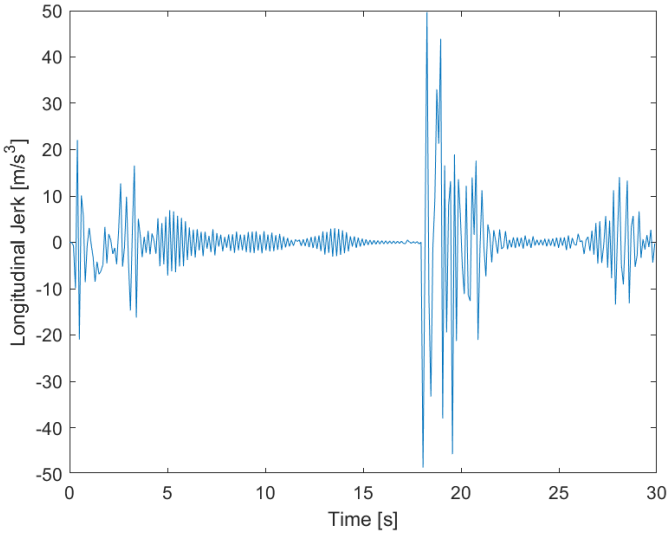

Figure 6.13: Single layer longitudinal jerk in SOA and LC simulation

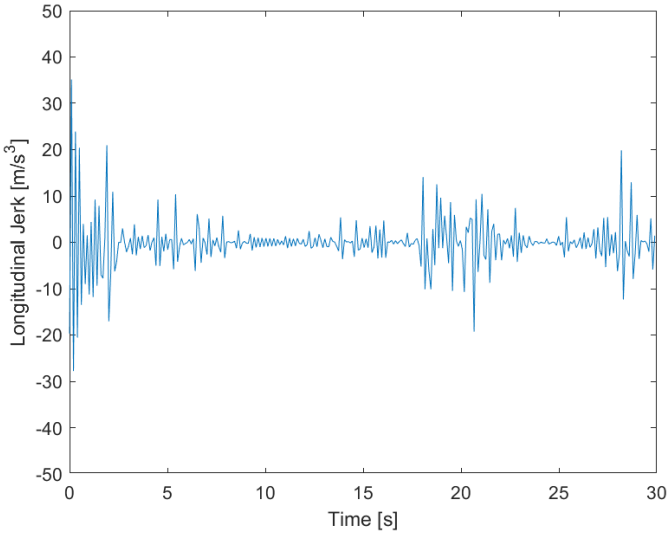

Figure 6.14: Double layer longitudinal jerk in SOA and LC simulation

Table 6.3:Summary of jerk values in SOA and LC simulation

	Single layer	*Double layer*	*Unit*
Lateral jerk	176,73	163,75	m/s^3
Variance	11760	2011	-
Longitudinal jerk	49,67	35,14	m/s^3
Variance	73,19	32,81	-

Dynamic obstacle avoidance

In the dynamic obstacle scenario, comparing the lateral jerk between the two architectures shows slight differences. The single layer architecture presents an absolute maximum lateral jerk of 100,23 [m/s^3] with a variance $\sigma = 1641$, reflecting noticeable variations in the vehicle's lateral movements. The double-layer system, while having a slightly higher absolute maximum of 102,21 [m/s^3], shows a much lower variance $\sigma = 370{,}35$. This suggests that, although both systems reach similar peak of lateral jerk, the double-layer architecture provides a more consistent and controlled response with reduced oscillations.

Below the two plots of the lateral jerk.

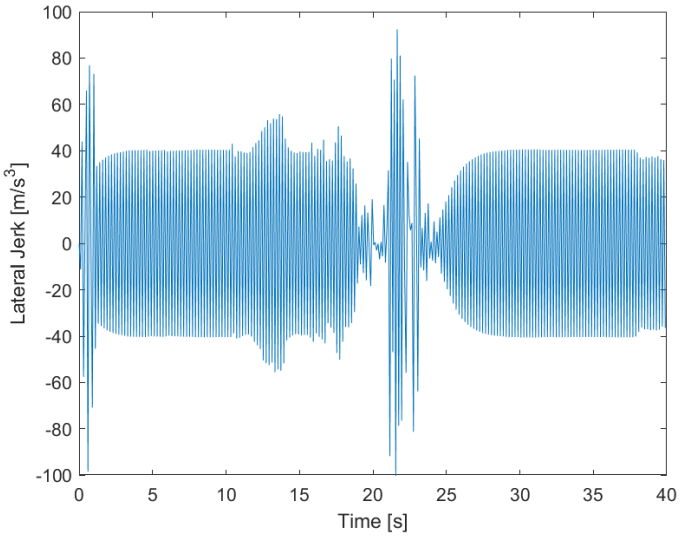

Figure 6.15: Single layer lateral jerk in DOA simulation

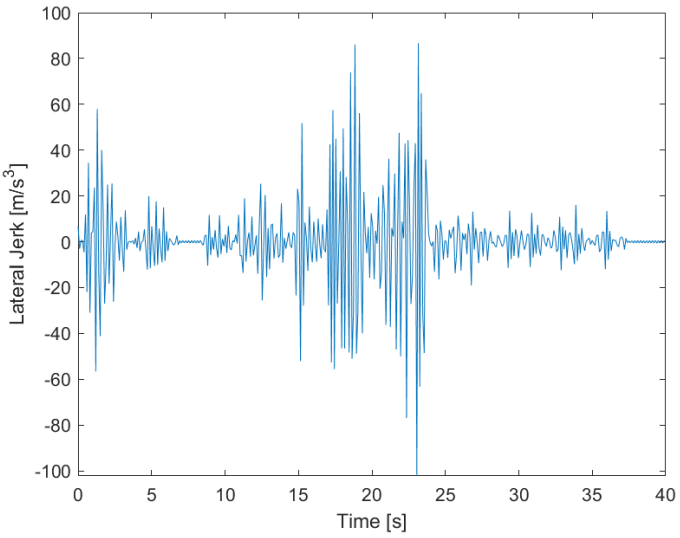

Figure 6.16: Double layer lateral jerk in DOA simulation

For the longitudinal jerk, the base architecture records an absolute maximum value of 20 [m/s^3] with a variance of 2,12, indicating a relatively controlled longitudinal movement. The double-layer implementation shows marginally higher maximum jerk at 22,05 [m/s^3], with a variance $\sigma = 2{,}92$. Although the double-layer system exhibits slightly larger peak values in the longitudinal jerk, the increase is minimal, and the system still manages to maintain reasonable control and smoothness in both longitudinal and lateral directions.

Below the two plots of the longitudinal jerk.

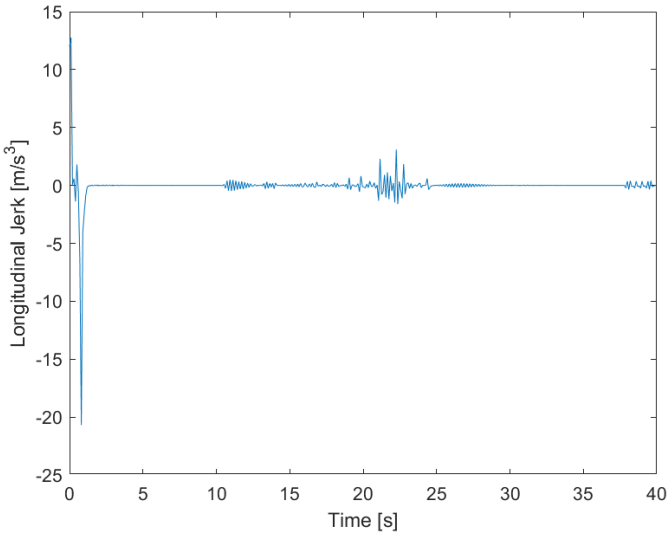

Figure 6.17: Single layer longitudinal jerk in DOA simulation

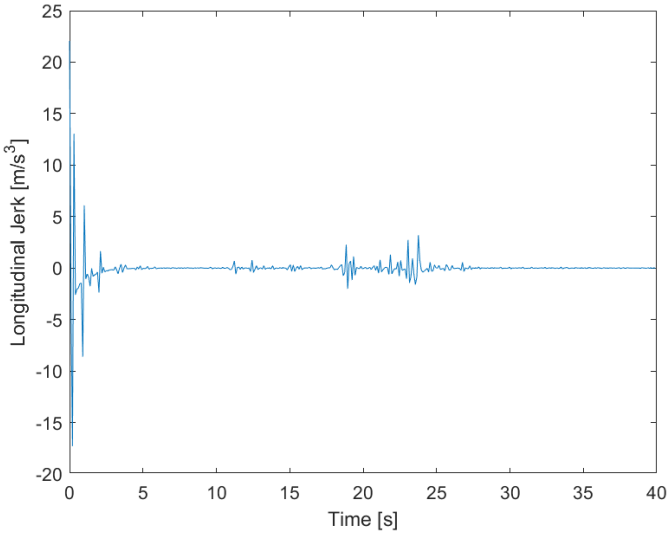

Figure 6.18: Double layer longitudinal jerk in DOA simulation

Table 6.4: Summary of jerk values in DOA simulation

	Single layer	*Double layer*	*Unit*
Lateral jerk	100,23	102,21	m/s^3
Variance	1641	370,35	-
Longitudinal jerk	20	22,05	m/s^3
Variance	2,12	2,92	-

Chapter 7

Conclusions and future development

This thesis has presented a comprehensive comparison of two nonlinear Model Predictive Control architecture, a single-layer and a double-layer system, with the aim of improving lane centering, obstacle avoidance and driving comfort in autonomous driving scenarios. Throughout the study, the focus was placed on analyzing both computational efficiency and driving comfort, which are critical aspects of control systems for advanced driver assistance systems (ADAS).

In early stages of the thesis, the vehicle model was initially developed using Simulink[28], with the solver implemented separately as a MATLAB script. To run the models, the 'sim' function[29] was used to call the Simulink file each time during the simulation process. However, this method proved to be highly inefficient, as Simulink had to be executed thousands of times within a single optimization cycle, significantly slowing down the simulation process. The inefficiency of this setup made it impractical to achieve the required performance. As a result, the entire simulation was moved into a MATLAB script, implementing the state evolution using a for loop and employing the Euler method for numerical integration. This approach bypassed Simulink entirely, optimizing the process and reducing computational overhead, allowing for faster simulation during the optimization.

One of primary objectives was to evaluate the computational performance of the two architectures in various driving scenarios. The results showed that the double-layer architecture significantly outperformed the base one in terms of mean execution time. For static obstacle avoidance, the single-layer architecture had a mean execution time of 0,61s while the enhanced architecture reduced this to 0,4s. Similarly, for dynamic obstacle avoidance, the mean time of the first architecture was 0,13s compared to just 0,067s for the second architecture. On average, this represents a mean percentage of improvement of approximately 37% in the mean execution time.

A crucial difference was also observed in the maximum execution times. In the static obstacle avoidance case, the base architecture experienced a maximum time of 1,77s, while the double layer system reached a maximum of 0,68s excluding the first execution.

The dynamic obstacle avoidance scenario also showed a drastic reduction in the maximum time, with the double-layer architecture decreasing from 2,63s to 0,81s, and even further to 0,18s when ignoring the first step. On average, this represents a mean percentage of improvement of approximately 77% in the max execution time. These results indicate that the double-layer architecture provides not only faster but more consistent performance, avoiding the larger fluctuations seen in the base system.

The variance of execution time is another important factor, as it demonstrates the consistency of the system over multiple iterations. The variance in terms of static obstacle avoidance has been enhanced by a significant margin, namely 74%. Furthermore, the results of the dynamic obstacle avoidance simulation indicate an improvement of 94%. The lower variance in the double-layer system indicates that it is more precise and stable under varying conditions.

In summary, the double-layer architecture showed superior computational performance, reducing both the mean and maximum execution times while maintaining a lower variance. This makes it a more suitable option for complex driving scenarios, especially when fast reactions and computational efficiency are needed.

The second key aspect of the comparison focused on driving comfort, which was evaluated by examining the smoothness of the generated trajectories and the control efforts generated by the optimization process. Lateral and longitudinal jerk was then used to deepen the analysis.

In the case of static obstacle avoidance, the base architecture exhibited noticeable oscillations in the generated trajectory. When analyzing the steering angle and torque near the obstacle, significant oscillations were observed as the control efforts bounced between maximum and minimum values. These oscillations not only indicate that the control system struggled to maintain a smooth trajectory but also resulted in more aggressive and less comfortable maneuvers for the vehicle. In contrast, the double-layer architecture produced a much smoother trajectory, with lower efforts and more consistent steering and torque values, minimizing the oscillations and providing a more stable and comfortable driving experience.

The analysis was deepened by evaluating the lateral and longitudinal jerk, which represents the rate of change of acceleration and is a direct indicator of driving comfort. In the static obstacle avoidance scenario, the base architecture exhibited significant lateral jerk with notable variance, indicating sharp and rapid changes in lateral forces, which translate into uncomfortable movements for passengers. The double-layer system, on the other hand, reduced the maximum lateral jerk and had a significantly lower variance, showing that it handled the vehicle's lateral dynamics much more smoothly. Similarly for

longitudinal jerk, the base system showed higher values, whereas the double layer system reduced these, again demonstrating smoother control inputs and a more comfortable ride.

In the dynamic obstacle avoidance scenario, a similar pattern was observed. While the double-layer system showed a slight increase in maximum lateral jerk, it significantly reduced the variance by around 77%, indicating a much smoother and more consistent handling during the maneuver. For longitudinal jerk, although the double-layer architecture had a marginally higher peak value, the variance remained comparable, reflecting better control over longitudinal forces without negatively affecting driving comfort.

The findings of this study clearly demonstrate the superiority of the double-layer NMPC architecture over the single-layer system, particularly in terms of execution time and driving comfort. The double-layer system not only reduced computation times but also delivered smoother and more stable control actions, making it an ideal solution for advanced autonomous driving systems where efficiency and passenger comfort are critical.

While the results are promising, there is significant potential for further development and optimization of the control architecture. Future work could focus on the following areas:

- Model in the loop simulation with enhanced plant dynamics: future studies could focus on developing a more sophisticated plant model that incorporates additional vehicle dynamics, such as pitch and suspension effects. This would provide a more accurate representation of the vehicle's behavior, especially under dynamic driving conditions, enhancing the control system's precision and increasing the validation value of this step. The model could also account for the nonlinear trend of the longitudinal force, as well as a realistic distribution of forces across all tires, ensuring that the control algorithms respond more effectively to varying road and driving conditions.
- Real-time implementation: It is necessary to modify the control algorithm in order to facilitate real-time deployment. This would require integrating the system into a real-time framework, potentially using HIL simulations to ensure that the control architecture meets strict timing and computational constraints when running on actual embedded systems.
- Embedded system coding: to make the control system practical for real-world applications, it must be implemented on an embedded system designed for automotive use. This would involve optimizing the control code to run efficiently on low-power, high-performance automotive processors, ensuring the system can handle real-time constraints while maintaining computational efficiency.

- Sensor integration: for real-world autonomous driving applications, the system must interact with sensors such as LIDAR, RADAR and GPS. Future work could focus on implementing the sensor fusion algorithms to provide more precise and reliable states estimation, which would enhance both the control performance and the system's ability to respond to dynamic obstacles in real-time.
- Validation with real-world data: while the simulations provide valuable insights, validation with real-world driving data and physical testing will be essential to ensure the robustness and reliability of the control system un diverse environments. Testing in real-world scenarios will provide important information on how the system handles different road conditions, vehicle loads and traffic situations.
- Adaptive control strategies: to further enhance the control architecture, future research could explore adaptive control strategies that dynamically adjust to changing driving conditions. This could involve adjusting control horizons, weights, or even switching between control modes depending on the driving scenarios

Bibliography

[1] J. B. Cicchino, “Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates,” *Accid Anal Prev*, vol. 99, pp. 142–152, 2017, doi: https://doi.org/10.1016/j.aap.2016.11.009.

[2] I. Isaksson Hellman and M. Lindman, “Estimating the crash reducing effect of Advanced Driver Assistance Systems (ADAS) for vulnerable road users,” *Traffic Safety Research*, vol. 4, p. 000036, Nov. 2023, doi: 10.55329/blzz2682.

[3] L. Yue, M. A. Abdel-Aty, Y. Wu, and A. Farid, “The Practical Effectiveness of Advanced Driver Assistance Systems at Different Roadway Facilities: System Limitation, Adoption, and Usage,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3859–3870, 2020, doi: 10.1109/TITS.2019.2935195.

[4] M. Aleksa, A. Schaub, I. Erdelean, S. Wittmann, A. Soteropoulos, and A. Fürdös, “Impact analysis of Advanced Driver Assistance Systems (ADAS) regarding road safety – computing reduction potentials,” *European Transport Research Review*, vol. 16, no. 1, p. 39, 2024, doi: 10.1186/s12544-024-00654-0.

[5] Insurance Institute for Highway Safety and Highway Loss Data Institute, “Real-world benefits of crash avoidance technologies,” 2023.

[6] A. Swief, A. El-Zawawi, and M. El-Habrouk, “A Survey of Model Predictive Control Development in Automotive Industries,” in *2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID)*, 2019, pp. 1–7. doi: 10.1109/ICAAID.2019.8934974.

[7] I. V Di Cairano Stefano and Kolmanovsky, “Automotive Applications of Model Predictive Control,” in *Handbook of Model Predictive Control*, W. S. Raković Saša V. and Levine, Ed., Cham: Springer International Publishing, 2019, pp. 493–527. doi: 10.1007/978-3-319-77489-3_21.

[8] D. Hrovat, S. Di Cairano, H. E. Tseng, and I. V Kolmanovsky, “The development of Model Predictive Control in automotive industry: A survey,” in *2012 IEEE International Conference on Control Applications*, 2012, pp. 295–302. doi: 10.1109/CCA.2012.6402735.

[9] J. H. Lee, “Model predictive control: Review of the three decades of development,” *Int J Control Autom Syst*, vol. 9, no. 3, pp. 415–424, 2011, doi: 10.1007/s12555-011-0300-6.

[10] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, “Path Planning for Autonomous Mobile Robots: A Review,” *Sensors*, vol. 21, no. 23, 2021, doi: 10.3390/s21237898.

[11] F. Micheli, M. Bersani, S. Arrigoni, F. Braghin, and F. Cheli, “NMPC trajectory planner for urban autonomous driving,” *CoRR*, vol. abs/2105.04034, 2021, [Online]. Available: https://arxiv.org/abs/2105.04034

[12] Y. Gao, F. Borrelli, J. K. Hedrick, and L. El Ghaoui, “Model Predictive Control for Autonomous and Semiautonomous Vehicles,” 2014.

[13] T. Gillespie, *Fundamentals of Vehicle Dynamics*. in Electronic publications. SAE International, 2021. [Online]. Available: https://books.google.it/books?id=LeybEAAAQBAJ

[14] A. Mater, S. Sorrentino, F. Zuccari, and F. Gerbino, “PACEJKA MAGIC FORMULA TIRE MODELING AND VALIDATION ON LOW ADHERENCE SURFACES,” 2019.

[15] K. V. N. R. A. Vijay Alagappan and R. K. Kumar, “A comparison of various algorithms to extract Magic Formula tyre model coefficients for vehicle dynamics simulations,” *Vehicle System Dynamics*, vol. 53, no. 2, pp. 154–178, 2015, doi: 10.1080/00423114.2014.984727.

[16] J. L. Olazagoitia, J. A. Perez, and F. Badea, “Identification of Tire Model Parameters with Artificial Neural Networks,” *Applied Sciences*, vol. 10, no. 24, 2020, doi: 10.3390/app10249110.

[17] M. Canale, L. Fagiano, M. Milanese, and P. Borodani, “Robust vehicle yaw control using an active differential and IMC techniques,” *Control Eng Pract*, vol. 15, no. 8, pp. 923–941, Aug. 2007, doi: 10.1016/j.conengprac.2006.11.012.

[18] OpenStreetMap contributors, “OpenStreetMap,” https://www.openstreetmap.org. [Online]. Available: https://www.openstreetmap.org

[19] Inc. The MathWorks, “MATLAB Driving Scenario Designer,” 2022, *The MathWorks, Inc.*: R2022b. Accessed: Oct. 06, 2024. [Online]. Available: https://www.mathworks.com/products/driving-scenario-designer.html

[20] Inc. The MathWorks, “roadBoundaries (MATLAB function).” [Online]. Available: https://it.mathworks.com/help/driving/ref/drivingscenario.roadboundaries.html

[21] Inc. The MathWorks, “spline (MATLAB function).” [Online]. Available: https://it.mathworks.com/help/matlab/ref/spline.html

[22] Inc. The MathWorks, “fmincon (MATLAB function).” [Online]. Available: https://it.mathworks.com/help/optim/ug/fmincon.html

[23] W. H. Kwon and S. H. Han, *Receding Horizon Control: Model Predictive Control for State Models*. in Advanced Textbooks in Control and Signal Processing. Springer London, 2005. [Online]. Available: https://books.google.it/books?id=ITSKhlKy2u8C

[24] A.-M. Pendrill and D. Eager, “Velocity, acceleration, jerk, snap and vibration: forces in our bodies during a roller coaster ride,” 2020.

[25] D. Eager, A.-M. Pendrill, and N. Reistad, “Beyond velocity and acceleration: jerk, snap and higher derivatives,” *Eur J Phys*, vol. 37, no. 6, p. 65008, Oct. 2016, doi: 10.1088/0143-0807/37/6/065008.

[26] I. Bae, J. Moon, and J. Seo, “Toward a Comfortable Driving Experience for a Self-Driving Shuttle Bus,” *Electronics (Basel)*, vol. 8, no. 9, 2019, doi: 10.3390/electronics8090943.

[27] N. Attila and M. Gyula, “ENGINEERING ENERGY EFFICIENT FEASIBLE AUTONOMOUS MULTI-ROTOR UNMANNED AERIAL VEHICLES TRAJECTORIES,” 2016. [Online]. Available: https://www.researchgate.net/publication/320057381

[28] Inc. The MathWorks, “Simulink,” 2024, *The MathWorks, Inc.*: 2022b. [Online]. Available: https://www.mathworks.com/products/simulink.html

[29] Inc. The MathWorks, “sim (MATLAB function).” [Online]. Available: https://it.mathworks.com/help/simulink/slref/sim.html