Gerard Ferrer Usieto

Trajectory generation for Autonomous Highway Driving using Model Predictive Control

Institute of Automation and Control

Graz University of Technology

Supervisors:

Astrid Rupp

Martin Steinberger

Graz, September 2017

Abstract

Model Predictive Control (MPC) has had an increasing role in autonomous driving applications over the last decade, enabled by the continuous rising of the computational power in microcontrollers.

In this thesis a collision avoidance trajectory generation algorithm based in MPC formulation is developed. The operating environment consists in a one-way highway with two lanes. The overall system is equipped with a low-level controller capable of tracking the trajectory generated by the MPC planner. In the path towards this goal, a MPC based lane changing application in an obstacle-free highway environment has been developed. A point-mass kinematic vehicle model is used as the MPC plant model for its simplicity and enabled by the usage of a low-level controller.

This thesis studies several obstacle representation approaches and then, explains in detail the development process of the collision avoidance trajectory generation application, defining and discussing simulation results for each intermediate approach obtained.

Both applications have been implemented in a BeagleBone Black online board situated in small-scale trucks (1:12) for testing purpose. The experimental results have been studied and discussed to prove the algorithms functionalities, as well as to check the board capabilities to run online MPC applications in comparison with polynomials based approaches.

Acknowledgement

During the time spent working on this thesis I was fortunate to be able to count on a number of amazing people that supported me in many different ways.

First of all, I would like to thank my supervisors Ms Astrid Rupp and prof. Martin Steinberger for making this thesis possible and advising me during the whole process. Thank you for giving me the opportunity to work on this research. It has been fascinating. Your support in control theory as well as with more specific issues have given me the knowledge that I needed and I learned concepts that I was not aware of when I started the project. Moreover, the regular meetings we had, where you showed your understanding while you were also strict, have shaped my way of working to become more professional. It has been a great pleasure to work at the Institute of Automation and Control of TU Graz and in your amazing Autonomous Driving Lab.

Next, I would like to thank Mr Raffael Wallner for assisting me with any technical difficulties I found and providing me with some awesome scripts to make the testing lot easier. I had an incredible time sharing the lab with you and the rest of the students, carrying out tests together, sometimes even until late night.

I would also like to thank all the friends I made during my Erasmus programme in Graz for cheering me up when things got tougher and making of my stay far away from home such an incredible and unrepeatable experience that I will never forget. Special thanks to Alex, Christina, Clara, Felix, George, Hugo and Robin.

Contents

1 Introduction 5
1.1 Motivation 5
1.2 Objectives 5
1.3 Structure 6

2 Background and Literature Review 8
2.1 Autonomous driving 8
2.2 Model Predictive Control in autonomous driving 10

3 Problem Statement 12
3.1 Model Predictive Control 12
3.2 Low-level Control System 13
3.3 Plant model – Vehicle modelling 14
3.3.1 Four-wheel dynamic vehicle model 14
3.3.2 Bicycle dynamic vehicle model 15
3.3.3 Point-mass kinematic vehicle model 16
3.3.4 Tire models 17
3.3.5 Vehicle model choice 18

4 Lane change approach 19
4.1 Cost function 19
4.2 Constraints 22
4.3 MPC simulation and tuning 24
4.4 Overall system simulation and tuning 36

5 Collision avoidance 46
5.1 Obstacle representation approaches 46
5.1.1 Nilson's approach 46
5.1.2 Eilbrecht's approach 49

5.1.3 Ferrara's approach 50
5.1.4 Obstacle representation choice 52
5.2 Model extension 53
5.3 Cost function 54
5.4 Constraints 55
5.5 Simulation 56
5.5.1 First approach 57
5.5.2 Second approach 65
5.5.3 Third approach 69

6 Testing 84
6.1 Hardware and software 84
6.2 Code generation changes 86
6.3 Lane change test results 87
6.4 Collision avoidance test results 96

7 Conclusions and Future work 109
7.1 Conclusions 109
7.2 Future work 110

8 References 112

1 Introduction

1.1 Motivation

Self-driving cars have always stoked our imaginations. From movies, to advertisements, autonomous vehicles show a life only possible in science fiction or a dream to be accomplished in the distant future. This future is not that distant as one might think. Today, aspects of self-driving cars are a reality shown in new car models, introducing intelligent, driver-assisted features that are slowly pushing semi-autonomous vehicles towards full vehicle autonomy.

Autonomous driving is a field that has always fascinated me. It combines several fields of knowledge such as mechanics, electronics, programming, automation and control. All these are fields that I have been studying during my university degree in industrial engineering specialized in automation and control in one way or another. Hence, it is a perfect field to test the knowledge acquired and deepen them.

The realization of the thesis as an Erasmus exchange programme in TU Graz opened a new range of possibilities in the field of autonomous driving to work with. Its Institute of Automation and Control encourages students and researchers to work in this field, owning an autonomous driving lab equipped with the necessary features to perform experimental tests, as well as the teaching and technical assistance needed.

People in the institute had already developed lane changing algorithms, among others, but always through the usage of polynomial approaches. With recent academic papers proving successful implementations of Model Predictive Control in autonomous driving, came the idea of developing a new lane change algorithm using MPC as planner and then, extend it to generate collision avoidance trajectories.

1.2 Objectives

The main goal of this thesis is to study and develop an algorithm capable of generating trajectories to avoid collision with obstacles in a one-way two-lane highway environment. The trajectory planner must be built using Model Predictive

Control formulation that assure collision avoidance while pursuing comfort and safety for the hypothetical passengers.

In order to achieve the main goal just presented, several intermediate objectives have been posed. The first one is to study and develop a lane changing algorithm in an obstacle-free one-way two-lane highway environment through MPC formulation. The second one is to redesign and tune a low-level controller capable of using the trajectory generated by the MPC planner to send the suitable signals to control the vehicle.

It is also a goal of this thesis to carry out experimental tests for the lane changing and the collision avoidance trajectory generation algorithms and to discuss their results. The tests will be done in the Autonomous Driving Lab of the Institute of Automation and Control using small-scale trucks equipped with a BeagleBone Black Board as online computer.

With the experimental testing it wants to be studied the implementation capabilities of MPC based algorithms in a single-board computer such as the BeagleBone Black. Hence, computation times and CPU usage will be analysed.

1.3 Structure

The academic content of this thesis is divided into seven chapters. Chapter 2 describes the background of the thesis and presents important literature reviewed about Model Predictive Control implemented in the matter of autonomous driving. It also describes autonomous driving state-of-art and its development in the industry.

Chapter 3 presents the thesis problem statement. There, an understanding of MPC theory is developed. Then, the overall control architecture implemented in this thesis is described. Finally, several vehicle models reviewed are discussed and the vehicle model chosen to use as the plant model is presented and argued.

Chapter 4 gives a detailed description of the lane change problem developed as a first contact towards the goals set. This description includes the fundamental aspects of the MPC used in this thesis, like the constraints formulation and the cost function used. This chapter also explains the MPC development, presenting and discussing the results of the simulations carried out, and its implementation together with a low-level controller.

Chapter 5 provides a thorough description of the collision avoidance trajectory generation problem, which is the core study of the thesis. First, several obstacle representation approaches reviewed are presented, together with the final choice taken and the reasons behind it. Second, taking advantage that some of the MPC aspects described in the previous chapter are also used for this problem, the new

MPC planner is explained, including the constraints formulation and the cost function used. Then, the development process of the MPC is described while presenting and discussing the results of the simulations carried out.

Everything related with the experimental procedure is explained in chapter 6. First it describes the testbed, explaining the hardware and software used and the operation of the assembly. Second, several issues suffered regarding the code generation are explained, as well as their resolve. Last, the tests carried out and their results are presented, dividing this section in two: one for the lane change approach and another for the collision avoidance trajectory generation problem.

Finally, chapter 7 provides a conclusion of the thesis and a discussion about the future work directions, including possible improvements and potential scenarios where the system described in this thesis could be implemented after some adaptations.

2 Background and Literature Review

2.1 Autonomous driving

Currently autonomous or self-driving vehicles are in the spotlight of academia and automotive industry research as they are promising evolution of current vehicle technology and advanced driver assistant systems (ADAS). This evolution is previsedto be the way to achieve a sustainable future for improved road safety, reduced congestion, decreased fuel consumption and emissions and greater mobility. Because of that, research on autonomous vehicles has been growing rapidly in the recent years and includes different fields of study as robotics, computer science and engineering. It should also be noted that some scientific advances are made by car manufacturers who do not always publicly reveal the details on their approaches or algorithms.

Semi-autonomous vehicles are the stepping stone to fully autonomous vehicles, offering features such as self-parking, adaptive cruise control (ACC), emergency braking, and semi-hands-off driving within highway conditions.

On highways, a high percentage of traffic accidents and fatalities is caused by human errors in lane change and overtake manoeuvres. Semi-autonomous features or Advanced Driver Assistance Systems (ADAS) as ACC have been shown to have a positive impact on traffic safety and help consumers become more comfortable with the idea of robots taking the wheel. For this reason, the introduction of fully automated systems, capable of safely lane change and overtake manoeuvres on highways, is expected to further contribute to increase traffic safety.

2021 is the year auto manufacturers have promised fully autonomous vehicles on the road. Start-ups and incumbents are racing toward a self-driving future but the race is as much about technology as it is about infrastructure and support. What remains unclear, though is when autonomous vehicles will arrive on the roads and how consumers and commercial industries will adopt progressing levels of autonomy over time.

Critical decision making is the key to autonomy and is realised through planning algorithms, situation understanding and a decision-making module. The main purpose of planning is to provide the vehicle with a safe and collision-free path towards its destination, while taking into account the vehicle dynamics, its manoeuvre capabilities in the presence of obstacles, traffic rules and road boundaries.

Existing planning algorithms originate primarily from the field of mobile robotics and then have been applied to different vehicles and operational environments, such as on-road autonomous car vehicles. Planning for autonomous driving is divided into four hierarchical classes, as suggested by Varaiya [17]: route planning, path planning, manoeuvre choice and trajectory planning or control planning. Route planning is concerned with finding the best global route from a given position to a goal. Path planning is the problem of finding a sequence of feasible configurations and states creating a path from an initial state to an end state. Manoeuvre planning addresses the problem of taking the best high-level decision for the vehicle (such as turning, overtaking, etc.), while taking into account the path that is specified from the path planning. Finally, trajectory planning is concerned with the real-time planning of the actual vehicle's transition from one feasible state to the next, satisfying the vehicle's kinematic limits based on vehicle dynamics and constrained by the navigation comfort, lane boundaries and traffic rules, while avoiding obstacles including other road users.

Highways are structured environments with relatively simple and easily maintainable traffic rules. As such, the driving task is quite straightforward, i.e. maintaining a desired velocity while avoiding collision conflicts with surrounding vehicles, and respecting the traffic rules. For this reason, the problem of determining how a vehicle should behave with respect to surrounding vehicles on highways has been posed as an obstacle avoidance trajectory planning problem, in this thesis.

In many applications, the trajectory generation is desired to be performed in a way that the time for executing a task, or the energy consumed during the motion is minimized. For that reason, motion-planning problems are often formulated as optimal control problems. After finding the best path to follow and the best manoeuvre to run, a trajectory must be generated that satisfies the motion model or state constraints and guarantees comfort for the passengers and smoothness for the trip.

The problem of generating a trajectory, according to the path and the manoeuvre that is chosen, can be solved by selecting a geometric curve to ensure smooth motion through the road network. Then, this trajectory is optimised by using a cost function

according to the dynamic model and the presence of obstacles along that trajectory. The geometric representations of the trajectories include arcs, polynomials of different orders, splines and Bezier curves. In these type of planning problems, the geometric curve used is restricted to a certain type and the path is constructed through solving a two-point boundary value problem at each time moment.

A different approach for the trajectory planning, which combines aspects of control engineering within the planning module is Model Predictive Control (MPC). Within MPC, a dynamic model for the vehicle is used and, through it, inputs for the controller are sampled about the future evolution of the vehicle's motion. From the dynamic model and the controller inputs, the optimisation problem of finding the best trajectory for the vehicle is solved.

Several academic articles have been used for the writing of this chapter such as [5], [8], [13], [16] and [17].

2.2 Model Predictive Control in autonomous driving

Recently, MPC control plays a more and more important role in autonomous driving especially in adaptive cruise control mainly because it solves an optimization problem with prediction states and various constraints can be added to meet design specifications. Several publications showing the increasing role of MPC in the autonomous driving scene over the last decade are described below.

In Falcone [6], from 2007, a model predictive control (MPC) approach to control an active front steering system in an autonomous vehicle is presented. The trajectory is assumed to be known over a finite horizon for each time step. Hence, the MPC is used to control the front steering angle in order to follow the trajectory. This is simulated and tested on slippery roads at the highest speed possible (simulation and experimental tests up to 21 m/s on icy roads are presented). Two approaches are developed in this paper, each one with a different plant model. The first one, uses a nonlinear vehicle model, while the second on is based on successive online linearization of the vehicle model.

In Yoon [19], from 2009, a model predictive approach for obstacle avoidance trajectory generation of unmanned ground vehicles (UGVs) is presented. Information on the obstacles is sensed within a limited range and incorporated online in the nonlinear model predictive that combines a tire model. The overall problem is solved online with nonlinear programming, augmenting the cost function upon detecting new obstacles using the obstacle information with two kinds of potential functions. The first one uses the distance from the vehicle to the nearest obstacle, while the second one uses the parallax information. Simulation results proved that in

complex environments, the parallax method shows clear improvements in terms of tracking and computation time over the distance-based method.

In Emre Sancar [4], from 2014, a model predictive control based approach is applied to better perform the collaborative adaptive cruise control (ACC) schemes. The MPC is able to accommodate actuator limits and parameter estimation. In addition, rear end collisions control to avoid hit with the following car can be avoided while the safe distance with the preceding vehicle can be maintained. Two types of MPC are designed, with and without rear collision detection and a switch is used based on the relative distance between the preceding vehicle and rear vehicle. A comparison between PID controller and MPC controller is given to show the advantage of MPC controller.

In Nilsson [12], from 2013, an obstacle avoidance trajectory planning for autonomous vehicles in a highway environment (one-way two -lane roads) is presented. The problem is formulated in a receding horizon framework (MPC) as a convex optimization problem. A point-mass model is used as vehicle plant model together with linear constraints that allow the problem to be solved as a convex quadratic programming (QP) problem. This approach has been shown to produce paths which can be tracked by a four-wheel vehicle model in Nilsson [13]. There, the trajectories generated by the MPC planner act as reference for a low-level controller, in charge of the front steering control of the vehicle. In Nilsson [11], the results presented in the previous papers are extended with more complex situations involving two surrounding vehicles.

This last academic paper presented [11] will be of a huge relevance in this thesis as it will be detailed explained in the following chapters, specifically at chapter 5.1.1.

3 Problem Statement

3.1 Model Predictive Control

In this chapter, the basic concepts of Model Predictive Control (MPC) are described as well as the theory behind it. This will allow the reader to fully understand the specific parameters and MPC formulation used in this thesis and explained in more detail in a further chapter.

Model predictive control is an optimal control technique mainly used to solve control problems in presence of constraints. These constraints can represent physical capabilities of the controlled system, like the steering angle of a vehicle or its acceleration, or restrictions/requirements of the control problem that is being solved, like the road boundaries that the vehicle is not allowed to cross or its desired maximum orientation angle.

In order to determine the optimal inputs that will be applied to the plant an optimization problem is solved by minimizing or maximizing a cost function over a finite horizon in an iterative manner. Using the figure 3.1 as reference lets explain in more detail this strategy. At discrete time k the current plant state is sampled and the optimizer computes a sequence of control inputs over a control horizon $[k, \dots, k + N_u]$ that minimize/maximize the cost function for a finite time horizon $[k, \dots, k + N_p]$, where N_u and N_p are the number of steps of the control and prediction horizons, respectively. Anyways, only the first step of the control sequence is applied to the plant in order to avoid errors between the predicted output and the actual process output due to the usage of an inaccurate plant model or possible disturbances in the controlled system. Then the plant state is sampled again and the process presented before is repeated starting from the new current state and shifting the prediction horizon one step forward $[k + 1, \dots, k + N_p + 1]$, reason why MPC is also called Receding Horizon Control (RHC).

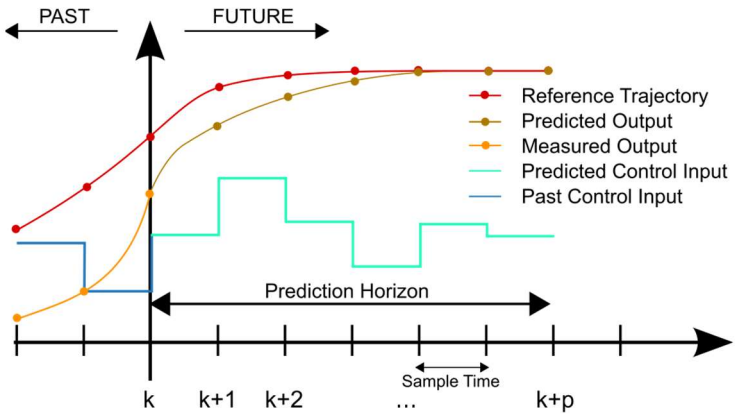


Figure 3.1: Scheme of the receding horizon principle. Source: https://en.wikipedia.org/wiki/Model_predictive_control. Author: M. Behrendt

In conclusion, MPC optimize the current timestep while keeping future steps in account due to optimizing a whole-time horizon but only implementing the current step and then optimizing again. This is the main difference from Linear-quadratic regulator (LQR) which optimize in a fixed horizon and uses that single solution for the whole horizon. Therefore, MPC allows real-time optimization against hard constraints. Also, the predictive ability to anticipate future events and take control actions accordingly differentiates MPC from PID controllers.

3.2 Low-level Control System

The implementation of the Model Predictive Control it is not done by connecting the plant model directly with the vehicle. Instead, a low-level control system capable of tracking the path generated by the MPC has been used, as it is shown in figure 3.2. That control system allows the usage of a plant model that only needs the references of the vehicle position states (x, y, θ) and the longitudinal velocity (v_x) as outputs. The low-level control system consists of different subsystems or blocks. First, there is a subsystem in charge of computing the error between the reference and the real vehicle position states. All these errors are used in the control block together with the longitudinal velocity to calculate the steering angle and the longitudinal velocity needed. Each of these outputs are calculated through their own controller, dividing the control block in two independent controllers. To control the steering angle, gains on the y and θ error signals are used. Regarding the velocity controller, it basically consists of a PI controller as it uses the signals of the vehicle velocity and the longitudinal position error. The steering and velocity control signals obtained from the controller block are then sent to the vehicle. A scheme of the overall system

which consists of the MPC together with the low-level control can be seen in the figure 3.2.

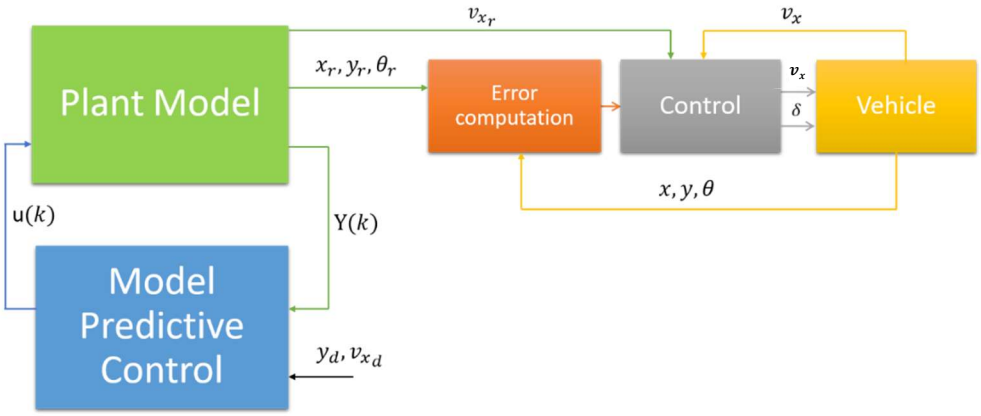


Figure 3.2: Scheme of the overall system.

3.3 Plant model – Vehicle modelling

In this chapter, the definition of a plant model is described. Then, several feasible options for the plant model are presented and finally, the choice made together with the reasons that lead to that decision are explained.

The plant model imitates the vehicle behaviour and it is needed in order to predict the future states of the vehicle in conjunction with the MPC. There are multiple feasible options for the model of a vehicle so we will have to choose the one that fits the most with the problem stated while trying to obtain an accurate representation of the vehicle behaviour.

3.3.1 Four-wheel dynamic vehicle model

This is a general model to describe the planar motion of a vehicle. It takes into account the forces acting on each of the four wheels of the vehicle. It comes from calculating the force balances for lateral and longitudinal acceleration and moment balance for the yaw rotation. The equations of this model are shown from (3.1) to (3.3).

$$m(\dot{u} - rv) = \cos\delta(F_{x1} + F_{x2}) - \sin\delta(F_{y1} + F_{y2}) + F_{x3} + F_{x4} \quad (3.1)$$

$$m(\dot{v} - ru) = \cos\delta\big(F_{y1} + F_{y2}\big) - \sin\delta(F_{x1} + F_{x2}) + F_{y3} + F_{y4} \quad (3.2)$$

$$I_z\dot{r} = a(\cos\delta\big(F_{y1} + F_{y2}\big) + \sin\delta(F_{x1} + F_{x2})) - b(F_{y3} + F_{y4}) + d(\sin\delta(F_{y1} - F_{y2}) + \cos\delta(F_{x2} - F_{x1}) - F_{x3} + F_{x4}) \quad (3.3)$$

where F_{xi} and F_{yi} are the forces in the longitudinal and lateral direction of the wheel i, respectively, u and v are the longitudinal and lateral velocities of the vehicle, respectively, and r the angular velocity, m is its mass, δ the steering angle of the front wheels and a, b and c are vehicle dimensions represented in the figure 3.3.

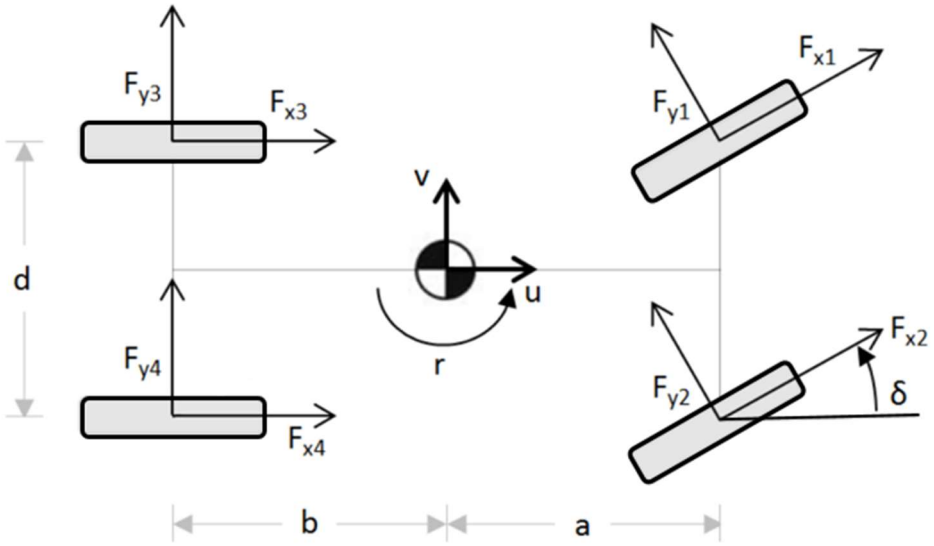


Figure 3.3: Scheme of the four-wheel dynamic vehicle model with the parameters used in its equations.

This model is nonlinear due to the trigonometric functions and the multiplicative relations between the states. While the model is well suited for simulation applications, simpler models are better for control design. This vehicle model is used in MPC autonomous driving papers such as Carvalho [1].

3.3.2 Bicycle dynamic vehicle model

The bicycle model is a well-known and commonly used model for car-like vehicles. This model simplifies the dynamic behaviour of the four wheels in just two, one for the front or steering wheels and the other for the rear ones. These simplified wheels are situated in the longitudinal axis of the vehicle. A general description of this model can be written as shown in the equations (3.4) and (3.5).

$$\dot{v} = \frac{F_{yf}}{m}\cos\delta_f - \frac{F_{xf}}{m}\sin\delta_f + \frac{F_{yr}}{m}\cos\delta_r - \frac{F_{xr}}{m}\sin\delta_r - ru \tag{3.4}$$

$$\dot{r} = \frac{l_f}{I_z}\left(F_{yf}\cos\delta_f - F_{xf}\sin\delta_f\right) - \frac{l_r}{I_z}\left(F_{yr}\cos\delta_r - F_{xr}\sin\delta_r\right) \tag{3.5}$$

where now the forces are indicated with the subscript r or f depending if they are applied to the rear or the front wheel, respectively. I_z is the inertia in z axis and l_f and l_r are dimensional parameters represented in the figure 3.4.

The generic bicycle model is nonlinear due to the trigonometric functions and the lateral velocity depending on the longitudinal one. Despite this, this model can be linearized by assuming constant longitudinal velocity and small angle approximation and neglecting the higher order terms. The linear equations are described in (3.6) and (3.7).

$$\dot{v} = \frac{F_{yf}}{m} + \frac{F_{yr}}{m} - ru \tag{3.6}$$

$$\dot{r} = \frac{F_{yf} \cdot l_f}{I_z} - \frac{F_{yr} \cdot l_r}{I_z} \tag{3.7}$$

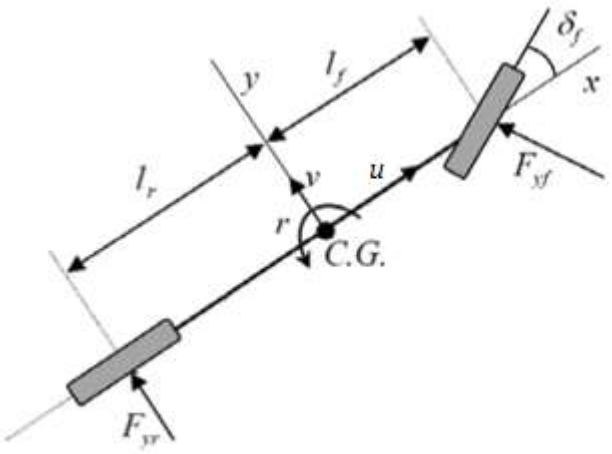


Figure 3.4: Scheme of the Scheme of the bicycle dynamic vehicle model with the parameters used in its equations. Source: modified from Mashadi [9].

3.3.3 Point-mass kinematic vehicle model

One of the simplest but, at the same time, effective vehicle model is the point driving kinematic model. It is a linear model where the longitudinal and the lateral dynamics are modelled as double integrators. The model can be described in a state space formulation with the state vector containing the positions and velocities, $X = [v_x \quad x \quad v_y \quad y]'$, and the input vector containing the accelerations, $u = [a_x \quad a_y]'$, as shown in equation (3.8).

$$\dot{X} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot X + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot u \qquad (3.8)$$

Based on zero-order hold discretization using a sample time T_s, the discrete-time state space system shown in equation (3.9) is obtained.

$$X_{k+1} = A \cdot X_k + B \cdot u_k \qquad (3.9)$$

with $A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ T_s & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & T_s & 1 \end{bmatrix}$ and $B = \begin{bmatrix} T_s & 0 \\ 0.5 \cdot {T_s}^2 & 0 \\ 0 & T_s \\ 0 & 0.5 \cdot {T_s}^2 \end{bmatrix}$.

3.3.4 Tire models

Tire models have also been studied but at the end, for several reasons explained below in chapter 3.3.5, they have not been taken into account. The interaction of the vehicle with the road is based on the distribution of forces through a small tire contact patch on each wheel. The tire models describe the relation between the lateral tire forces and the tire slip angle through an experimental parameter called cornering stiffness. The most common tire models are the Linear tire model, Brush tire model and the so-called Magic tire formula. A comparison between them can be seen in the figure 3.5.

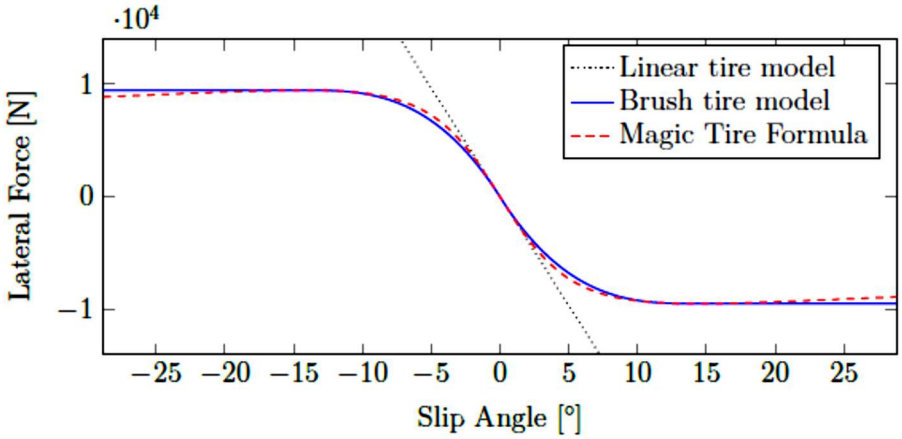


Figure 3.5: Comparison between tire models

3.3.5 Vehicle model choice

First of all, the nonlinear models have been discarded. This decision is motivated by the fact that nonlinear models are computationally less tractable in MPC problems and also, taking into account the existence of a low-level control, it is assumed that a linear model will be sufficient for the trajectory planning.

Secondly, linearization of nonlinear models has been also discarded due to the needed assumption of constant longitudinal velocity. One of the most complex situations in a two-lane highway environment is when the controlled vehicle approaches a slower obstacle located in the same lane and simultaneously, a faster obstacle approaches the vehicle on the other lane. As the faster obstacle prevents the vehicle to overtake the slower one, the vehicle has to adapt his longitudinal velocity to the slower one, making not feasible the assumption previously presented and therefore, the linearization of the model.

To make the choice of the vehicle model that is going to be implemented in our control system, it has obviously been taken into account the existence of the low-level control and the inputs that it needs. These inputs needed are the position coordinates (x, y, θ) and the longitudinal velocity (v_x). The model that is best described by these variables is the Point-mass kinematic vehicle model with the issue that the heading angle (θ) does not appear in the model. This is solved by approximating it through the variation of the longitudinal and lateral coordinates (x and y) as it is shown in equation (3.10) with the assumption that his initial value is zero.

$$\theta_{k+1} = \arctan\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right), \qquad \theta_0 = 0 \tag{3.10}$$

Finally, as this thesis results are only intended to be tested and used in miniature testing trucks, the tires of which behave differently from a real truck tire and in an unknown way, no tire model has been used.

4 Lane change approach

In order to achieve the main goal of this thesis, which is to implement in the testing trucks a highly automated collision avoidance system using MPC formulation on a two-lane highway environment, it has been deemed convenient to firstly implement a lane change system in an obstacle-free environment.

4.1 Cost function

A cost function is a performance index which is minimized or maximized (the first one in our case). There are several types of cost function, the most basic ones are the Time Optimal, which only penalize the time that takes to a system to go from one point to another, the Minimum-Effort, which minimize the effort or cost of a process and the General Optimal Control, which is the most used as it can represent any problem with the suitable functions and it can be defined in a quadratic form as shown in equation (4.1).

$$J = x'(t_f) \cdot F \cdot x(t_f) + \int_{t_0}^{t_f} [x'(t) \cdot Q \cdot x(t) + u'(t) \cdot R \cdot u(t)]dt \tag{4.1}$$

where F, Q and R are tuning matrices that determine the weight of each parameter in the cost function. x and u are the states and control vectors, respectively, and t_0 to t_f is the interval of time used.

This definition of the cost function is called quadratic form because the terms of the states and control are squared and are often written as $\|x\|^2{}_Q$ and $\|u\|^2{}_R$, respectively. For the lane change problem, only the term called running cost is used, which is the integral term, or in other words, the value of the matrix F is zero.

Another way more specific to define the cost function used in this problem, once discretized and taking into account the prediction and control horizons (N_p and N_u, respectively) is the one shown in equation (4.2).

$$V(k) = \sum_{i=1}^{N_p} \|Y_{k+i}(k) - r_{k+i}(k)\|^2{}_{Q_i} + \sum_{i=0}^{N_u-1} \|\Delta U_{k+i}(k)\|^2{}_{R_i} \tag{4.2}$$

where $Y_{k+i}(k)$ is the output vector of the model for the step $k+i$ predicted at time k. This same notation is used for the reference vector r, and the variation of the control vector ΔU. For the lane change approach, the output vector is composed by the lateral position y, and the longitudinal velocity v_x. Q_i and R_i are the weight matrices at discrete time i and, in this problem, their values will remain constant over their whole horizon. Hence, they can be simply referred as matrices Q and R. Having in mind the outputs just defined and the control vector already described in chapter 3.3.3, the weighing matrices can be defined as

$$Q = \begin{bmatrix} Q_y & 0 \\ 0 & Q_{v_x} \end{bmatrix} \qquad R = \begin{bmatrix} R_x & 0 \\ 0 & R_y \end{bmatrix}.$$

In order to solve the optimization problem, is important to rewrite the cost function to only depend on a single vector of variables, called optimization vector. In this case the variation of the control vector, ΔU, has been chosen for this purpose.

First, equation (4.2) is rewritten as shown in (4.3) to simplify it:

$$V(k) = \left\|\hat{Y}(k) - \hat{r}(k)\right\|_Q^2 + \left\|\Delta\hat{U}(k)\right\|_R^2 \tag{4.3}$$

with

$$\hat{Y}(k) = \begin{bmatrix} Y_{k+1}(k) \\ \vdots \\ Y_{k+N_p}(k) \end{bmatrix} \qquad \Delta\hat{U}(k) = \begin{bmatrix} \Delta U_k(k) \\ \vdots \\ \Delta U_{k+N_u-1}(k) \end{bmatrix} \qquad \hat{r}(k) = \begin{bmatrix} r_{k+1}(k) \\ \vdots \\ r_{k+N_p}(k) \end{bmatrix}.$$

Then, it is needed to write the output vector $\hat{Y}(k)$ as a function of the optimization vector $\Delta\hat{U}(k)$. Using equation (4.4) and (4.5), it can be firstly rewritten as a function of the state vector $\hat{X}(k)$.

$$Y_{k+1}(k) = \mathrm{C} \cdot X_{k+1}(\mathrm{k}) \tag{4.4}$$

$$\hat{Y}(k) = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C \end{bmatrix} \cdot \hat{X}(k) \tag{4.5}$$

with

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \hat{X}(k) = \begin{bmatrix} X_{k+1}(k) \\ \vdots \\ X_{k+N_p}(k) \end{bmatrix}.$$

Then, to write the state vector $\hat{X}(k)$ as a function of the optimization vector $\Delta\hat{U}(k)$, equations (4.6), (4.7) and their successive equation development until $(k + N_u - 1)$ are applied in the basic state space equations, (4.8) and their successive equation development until $(k + N_p)$, as it is shown in equations (4.9) and (4.10).

$$u_k(k) = \Delta U_k(k) + u_{k-1} \tag{4.6}$$

$$u_{k+1}(k) = \Delta U_{k+1}(k) + \Delta U_k(k) + u_{k-1} \tag{4.7}$$

$$X_{k+1}(k) = A \cdot X_k + B \cdot u_k(k) \tag{4.8}$$

$$X_{k+1}(k) = A \cdot X_k + B \cdot [\Delta U_k(k) + u_{k-1}] \tag{4.9}$$

$$X_{k+2}(k) = A^2 \cdot X_k + AB \cdot [\Delta U_k(k) + u_{k-1}] + B \cdot \underbrace{[\Delta U_{k+1}(k) + \Delta U_k(k) + u_{k-1}]}_{u_{k+1}(k)} \tag{4.10}$$

where *A* and *B* are matrices defined in chapter 3.3.3 equation (3.9), X_k is the state vector at time k and u_{k-1} is the control vector at time $k - 1$.

With this procedure, equation (4.11) is obtained. Then, the output vector $\hat{Y}(k)$ is written as a function of the optimization vector $\Delta\hat{U}(k)$ in equation (4.12), though the usage of equations (4.11) and (4.5).

$$\hat{X}(k) = A_a \cdot X_k + A_b \cdot u_{k-1} + A_{ba} \cdot \Delta\hat{U}(k) \tag{4.11}$$

$$\hat{Y}(k) = \Psi \cdot X_k + \Upsilon \cdot u_{k-1} + \Theta \cdot \Delta\hat{U}(k) \tag{4.12}$$

with A_a, A_b, A_{ba}, Ψ, Υ and Θ being constant matrices.

Finally, the cost function is rewritten as a function of the optimization vector in equation (4.13).

$$V(k) = const - \Delta\hat{U}(k)' \cdot Gamma + \Delta\hat{U}(k)' \cdot H \cdot \Delta\hat{U}(k) \tag{4.13}$$

with $Gamma = 2 \cdot \Theta' \cdot Q \cdot \hat{\xi}(k)$ where $\hat{\xi}(k) = \hat{r}(k) - \Psi \cdot X_k + \Upsilon \cdot u_{k-1}$ and $H = \Theta' \cdot Q \cdot \Theta + R$.

4.2 Constraints

This section presents the constraints that will act in the control. Some of them are hard restrictions from the definition of the problem and others are included for comfort or security reasons.

One of the obvious constraint to define is the restriction of the vehicle to not pass the road boundaries. With a lane width of 0.6 m and a vehicle dimensions of 0.5 m and 0.2 m for the length and width, respectively, and taking into account that it is desired to set the middle of the right lane as $y = 0$ m, the boundaries are finally defined as $y_{max} = 0.75$ m and $y_{min} = -0.25$ m. Another constraint to define for security reasons and for the capabilities of the vehicle motor is the longitudinal velocity and acceleration. The longitudinal velocity cannot be negative and it is capped at $v_{x\,max} = 1\,{}^{\mathrm{m}}/_{\mathrm{s}}$ and regarding the longitudinal acceleration it is capped at $a_{x\,lim} \pm 0.5\ \mathrm{m/s^2}$. For comfort reasons, the lateral acceleration as well as both accelerations variations have also been constrained with values of $a_{y\,lim} = \pm 0.5\ \mathrm{m/s^2}$ and $\Delta a_{lim} = \pm 0.25\ \mathrm{m/s^2}$.

Taking into account the values just defined, the constraints can be defined as shown in equations from (4.14) to (4.19).

$$\widehat{U}(k): \qquad a_{x_{min}} \leq a_{x_k} \leq a_{x_{min}} \tag{4.14}$$

$$a_{y_{min}} \leq a_{y_k} \leq a_{y_{max}} \tag{4.15}$$

$$\widehat{Y}(k): \qquad y_{min} \leq y_k \leq y_{max} \tag{4.16}$$

$$v_{x_{min}} \leq v_{x_k} \leq v_{x_{max}} \tag{4.17}$$

$$\Delta\widehat{U}(k): \qquad \Delta a_{x_{min}} \leq \Delta a_{x_k} \leq \Delta a_{x_{min}} \tag{4.18}$$

$$\Delta a_{y_{min}} \leq \Delta a_{y_k} \leq \Delta a_{y_{max}} \tag{4.19}$$

In order to solve the optimization problem, is important to rewrite the constraints to set them as a function of the optimization vector. To achieve this, they are first built in a matrix form as shown in equations from (4.20) to (4.22).

$$[F, f] \cdot \begin{bmatrix} \widehat{U}(k) \\ 1 \end{bmatrix} \leq 0 \tag{4.20}$$

with $F = [F_1, \dots, F_{Nu}]$ and f is the last column vector of the matrix, which is the vector that defines the scalar part of the constraints.

$$[G, g] \cdot \begin{bmatrix} \hat{Y}(k) \\ 1 \end{bmatrix} \leq 0 \qquad (4.21)$$

with $G = [G_1, \dots, G_{Np}]$ and g is the last column vector of the matrix, which is the vector that defines the scalar part of the constraints.

$$[E, e] \cdot \begin{bmatrix} \Delta\widehat{U}(k) \\ 1 \end{bmatrix} \leq 0 \qquad (4.22)$$

with $E = [E_1, \dots, E_{Nu}]$ and e is the last column vector of the matrix, which is the vector that defines the scalar part of the constraints.

Then, making use of equation (4.12), the constraints can finally be rewritten as shown in equation (4.23).

$$\Omega \cdot \Delta\widehat{U}(k) \leq \omega \qquad (4.23)$$

with

$$\Omega = \begin{bmatrix} \mathcal{F} \\ G \cdot \Theta \\ \mathrm{E} \end{bmatrix}, \qquad \omega = \begin{bmatrix} -\mathcal{F}_1 \cdot u_{k-1} - f \\ -G \cdot [\Psi \cdot X_k + \Upsilon \cdot u_{k-1}] - g \\ e \end{bmatrix}$$

where $\mathcal{F}_i = \sum_{j=i}^{Nu} F_j$ and $\mathcal{F} = [\mathcal{F}_1, \dots, \mathcal{F}_{Nu}]$.

A severe problem that can occur with the predictive control problem as it has been formulated so far, is that the optimizer may be faced with an infeasible problem. This can happen because of unexpected large disturbance affecting our system, so the control has no way to keep the plant within the specified constraints. This kind of behaviour is unacceptable for an on-line controller so it is essential to deal with the possibility of infeasibility.

One systematic strategy to deal with infeasibility is to soften the constraints. This means to allow some constraints to be violated occasionally, but only if it is strictly necessary. A way to soften the constraints is to add new variables, called “slack variables”, which are non-zero only if the constraints are violated and they are heavily penalized in the cost function, so that the optimizer keep them at zero if possible. The optimization problem can be reformulated using soft constraints as shown in equation (4.24), subject to the constraints (4.25) and (4.26).

$$\min_{\Delta U, \epsilon} \left(V\left(\Delta\widehat{U}\right) + \rho\epsilon \right) \qquad (4.24)$$

subject to

$$\Omega \cdot \Delta\widehat{U} \leq \omega + \epsilon \quad (4.25)$$

$$\epsilon \geq 0 \quad (4.26)$$

where ϵ is the slack variable and ρ is a scalar, which has to be large enough for the constraints to only be violated when strictly necessary.

The only constraints that will not have a slack variable on them, hence, they will remain as hard constraints, are the ones regarding the MPC outputs. These constraints correspond to the road boundaries and the longitudinal velocity limits of the controlled vehicle, equations (4.16) and (4.17).

4.3 MPC simulation and tuning

The MPC structure used in this approach is a typical structure for a controller with constraints, consisting of the reference r, as input for the optimizer together with the feedbacks of the control vector u_{k-1}, and the predicted states obtained through the Discrete-time state-space plant. Other blocks than these mentioned are just additions, like the orientation angle computation, which is needed as input for the low-level control, or the data storage blocks for plotting purpose.

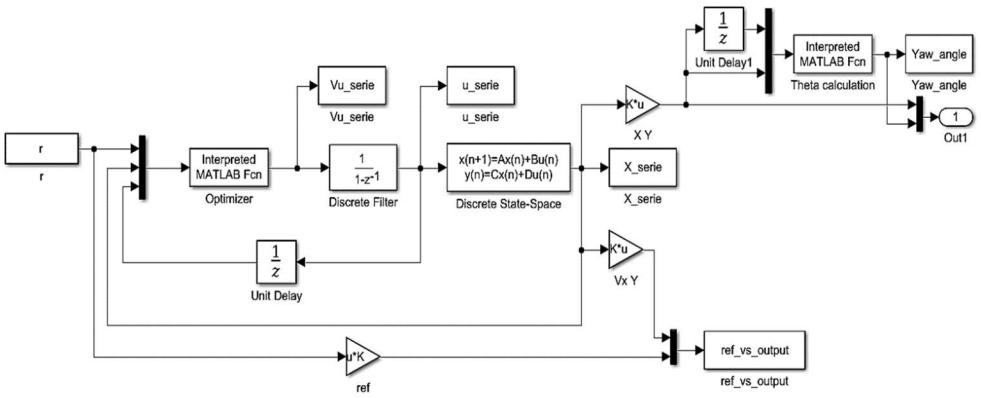
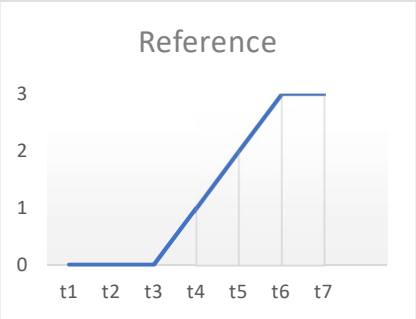


Figure 4.1: Simulink block diagram of the MPC planner.

One important aspect for the MPC to discuss is the reference computation, or in other words, how the controller reads the reference data for the prediction horizon. There are mainly two ways of doing that: with and without preview.



With preview

t1	0	0	0
t2	0	0	1
t3	0	1	2
t4	1	2	3
t5	2	3	3

Without preview

t1	0	0	0
t2	0	0	0
t3	0	0	0
t4	1	1	1
t5	2	2	2

Figure 4.2: In the left, plot of a reference signal used as example and in the right its matrices computed with the two ways.

Let us suppose the reference shown in the figure 4.2 and a prediction horizon of $N_p = 3$. For a discrete time of 5 steps the two reference matrices are obtained using each computation. As it can be seen while the first one is using the prediction horizon to anticipate the reference change, the second one only uses the immediate upcoming value.

After understanding these two definitions, we could fall to the reckless assumption that the reference computation with preview is always better, as it can anticipate the reference change. But this is not always true, as it is proved after calling out some simulations with the two different reference computations and doing a comparison between them.

As it can be seen in the figures 4.4 and 4.6, in the case of using the reference computed without preview, we always have a delayed response in comparison with the other option, figures 4.3 and 4.5. Despite this, the obtained signal reaches the reference value without any overshoot, which cannot be said for the case with the reference computed with preview.

It is important to notice that these plots have been taken with the parameters perfectly tuned and independently for each reference computation, therefore, we are able to compare the best signals achievable for both cases.

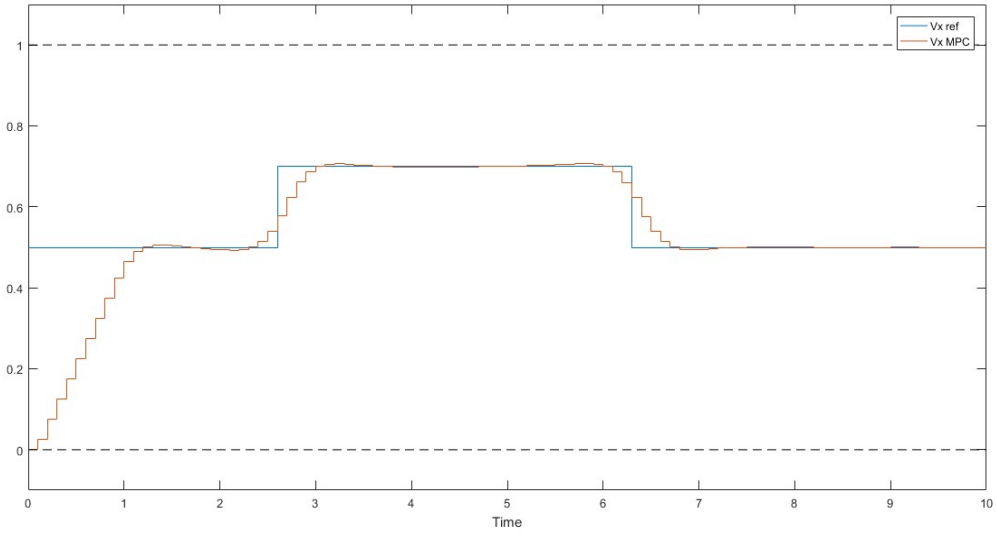


Figure 4.3: Plot over time of the reference and MPC output for the longitudinal velocity signal using reference computation with preview. All units in SI.

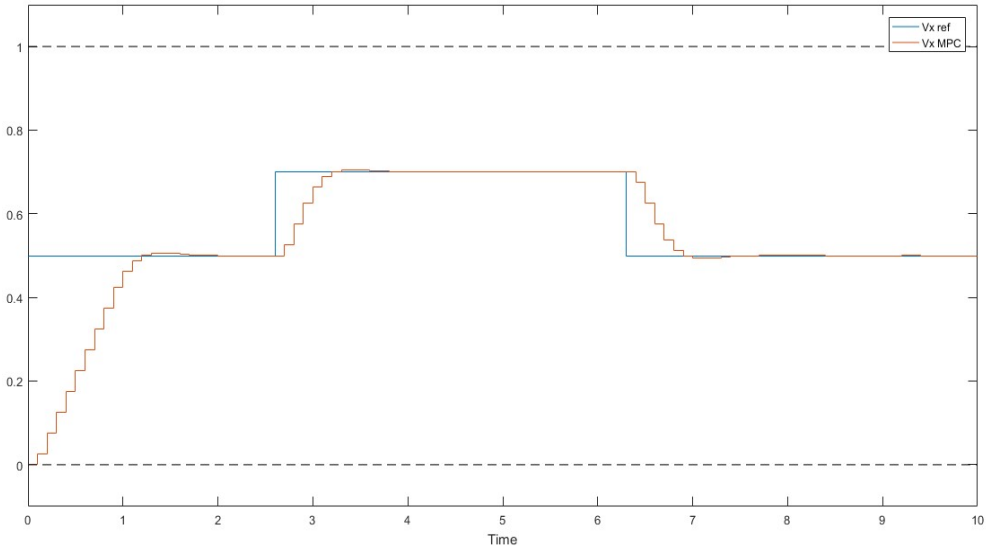


Figure 4.4: Plot over time of the reference and MPC output for the longitudinal velocity signal using reference computation without preview. All units in SI.

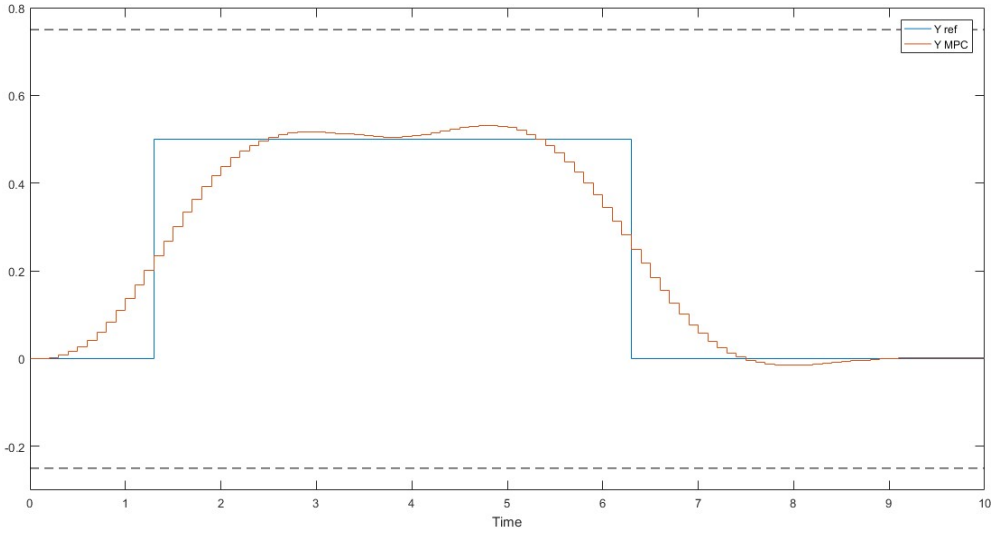


Figure 4.5 Plot over time of the reference and MPC output for the lateral position signal using reference computation with preview. All units in SI.

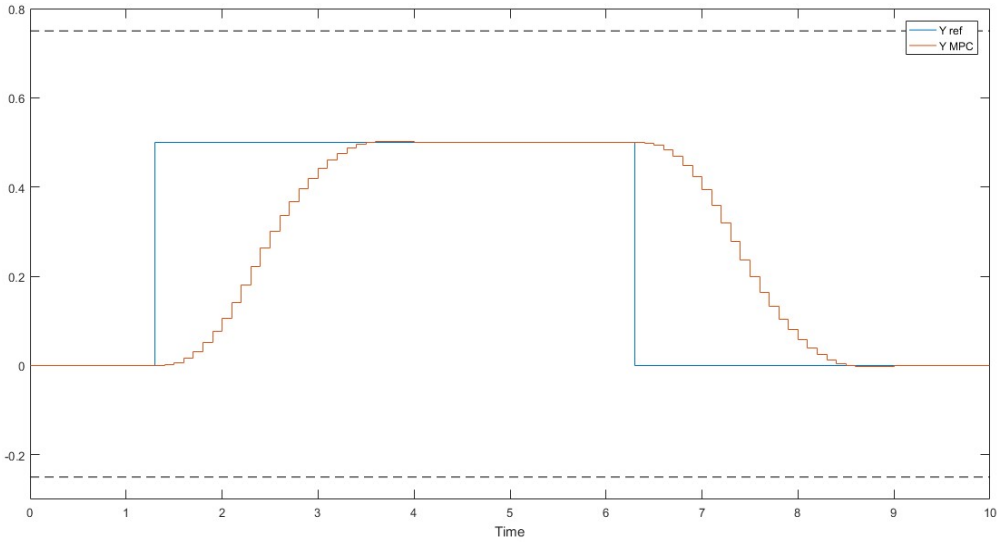


Figure 4.6: Plot over time of the reference and MPC output for the lateral position signal using reference computation without preview. All units in SI.

Taking into account these observations, we now have to decide which reference computation works better in the highway collision avoidance problem that we are facing. Having in mind that we know in every moment where the obstacles are, it is easy to see that the delay response from the reference signal should not suppose much worry as it is always possible to advance the reference change by, for example, incrementing the forward safe distance.

In the other hand, handling overshoots is never a desirable behaviour and less in a highway environment with limited lanes width and vehicles moving at high speeds.

In addition, even if the overshoots seem small now we should have in mind that they can be accentuated in the low-level control.

In conclusion, it has been proved that the behaviour of the signal coming from the reference computed without preview fits better in the collision avoidance problem, contrary to what it could be thought before doing the comparison.

Once it has been decided which reference computation use, we can move to the tuning phase. The tuning parameters for the MPC are the prediction horizon (N_p), the control horizon (N_u) and the weight matrices Q and R from the quadratic cost function. The values of these matrices could change for each discrete time in the prediction and control horizon, respectively, but they will remain constant in our problem. Each of the matrices has two values situated in the diagonal, one for each variable that they weight. We define the weight of the y and v_x errors as Q_y and Q_v, respectively and the weights of the accelerations variations (or input variations Δu) as R_x and R_y, respectively.

$$Q = \begin{bmatrix} Q_y & 0 \\ 0 & Q_{v_x} \end{bmatrix} \qquad R = \begin{bmatrix} R_x & 0 \\ 0 & R_y \end{bmatrix}$$

In order to do the tuning in a structured way, we started with a set of initial values for the tuning parameters that seemed logical and then we proceed changing only one parameter from the initial values per simulation to be able to see how each parameter influences de response. The initial set of tuning parameters is: $Q_v = 0.75$, $Q_y = 1$, $R_x = R_y = 0.5$, $N_p = 8$, $N_u = 5$.

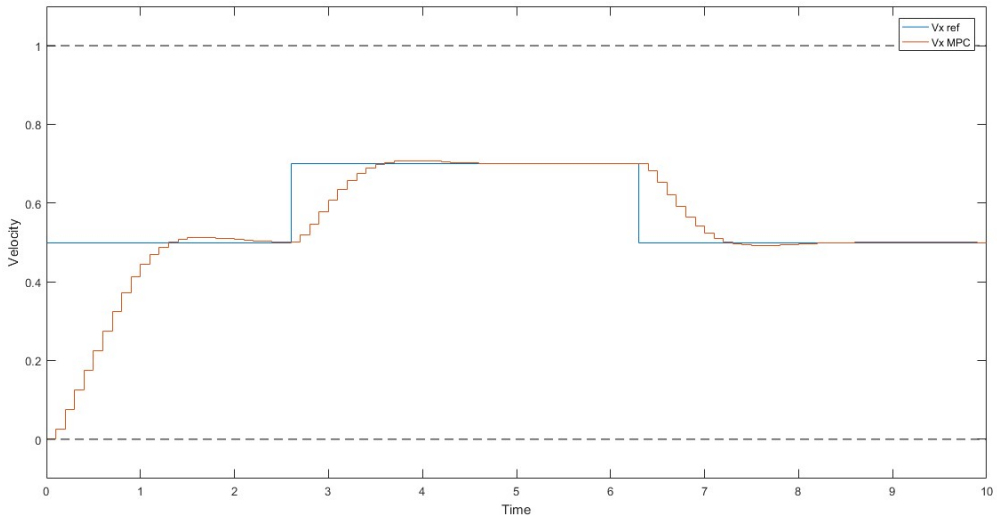


Figure 4.7: Plot over time of the reference and the MPC output signal for the longitudinal velocity with the initial tuning parameters. All units in SI.

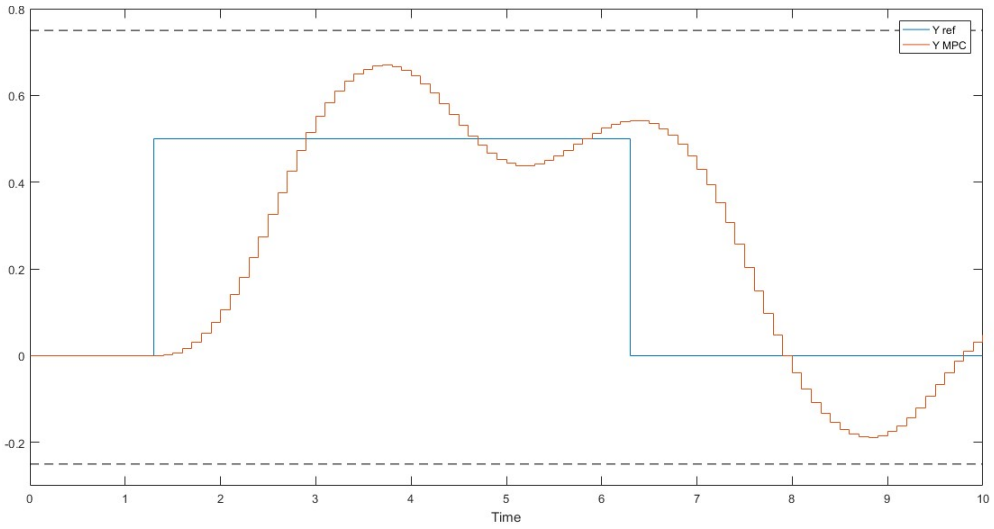


Figure 4.8: Plot over time of the reference and the MPC output signal for the lateral position with the initial tuning parameters. All units in SI.

The first two tuning simulations consist on increasing the weights for the longitudinal velocity and the lateral position to $Q_v = 10$ and $Q_y = 10$, respectively.

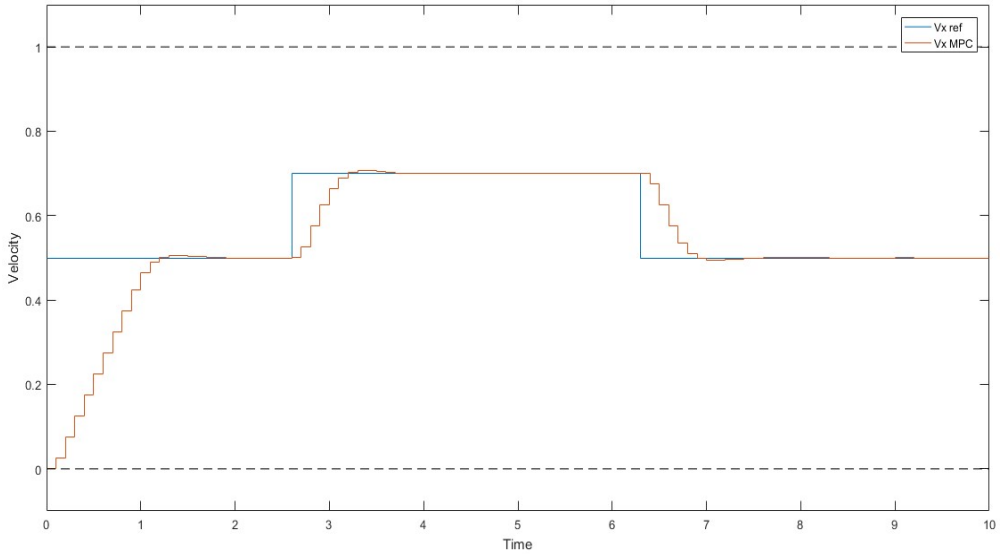


Figure 4.9: Plot over time of the longitudinal velocity with $Q_v = 10$. All units in SI.

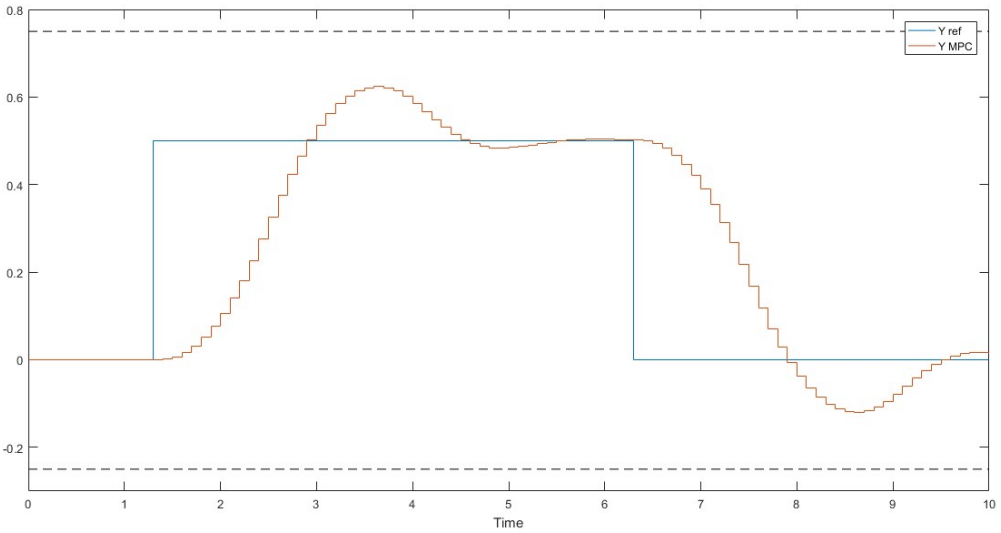


Figure 4.10: Plot over time of the lateral position with $Q_y = 10$. All units in SI.

As it can be seen in the figures 4.9 and 4.10, as it was expected, incrementing the value of the output weights improve the respective output signal in order to follow better his reference. While the longitudinal velocity signal has an acceptable shape, it cannot be said for the lateral position one.

Let us move now to the weights of the input variation R_x and R_y, which will be tuned together increasing their value to $R_x = R_y = 5$.

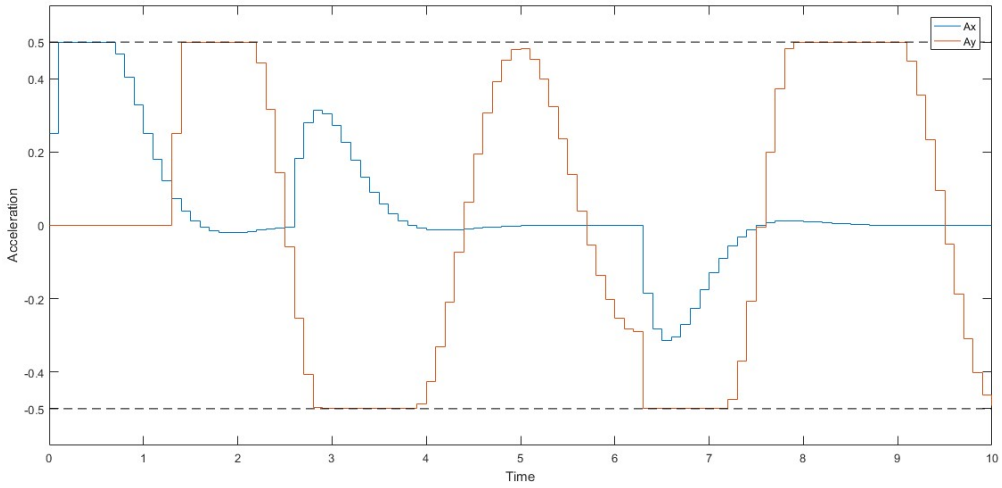


Figure 4.11: Plot over time of the acceleration with the initial tuning parameters. All units in SI.

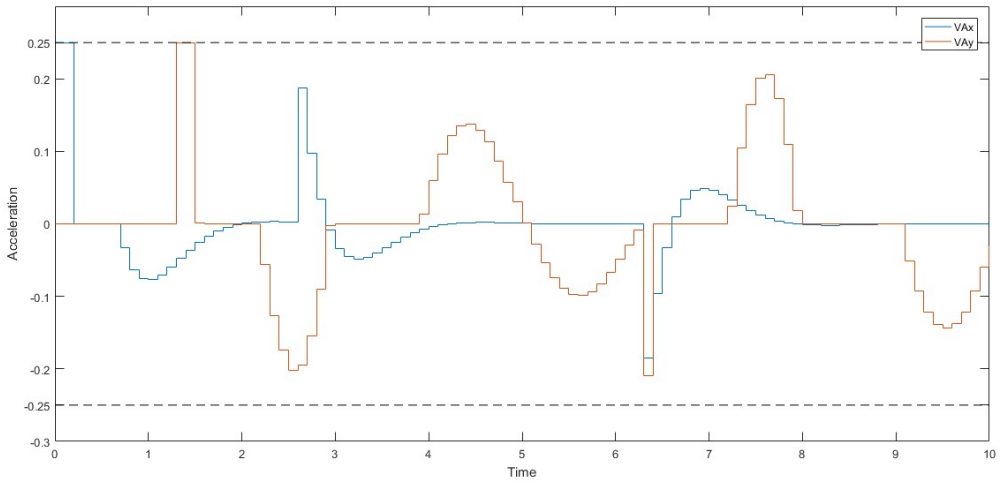


Figure 4.12: Plot over time of the acceleration variation with the initial tuning parameters. All units in SI.

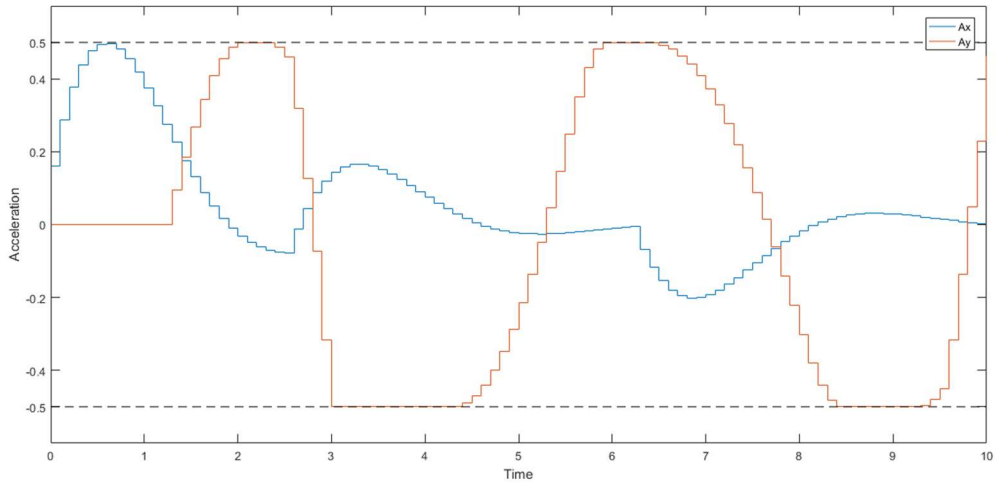


Figure 4.13: Plot over time of the acceleration with $R_x = R_y = 5$. All units in SI.

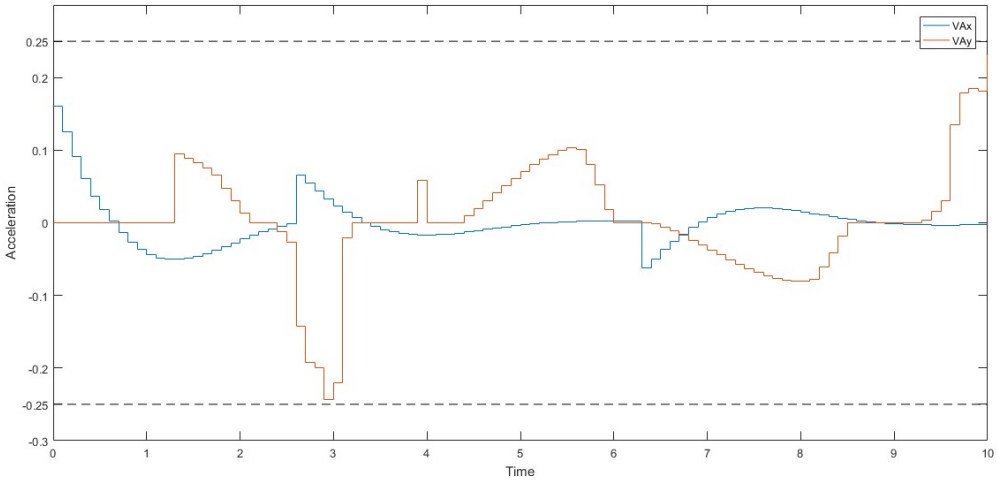


Figure 4.14: Plot of the acceleration variation with $R_x = R_y = 5$. All units in SI.

As it can be seen in the figures from 4.11 to 4.14, increasing the weight of the input variation smooths the input signals as well as their variation. We should also keep in mind that increasing these weights while keeping the others low will, of course, decrease the output signals performance.

The last tuning parameters to study are the prediction and control horizons. As the control horizon can only be inferior or equal to the prediction one, firstly, only the prediction horizon will be increased to $N_p = 30$ and then, in the second simulation both parameters will be increased to $N_p = 30$ and $N_u = 15$.

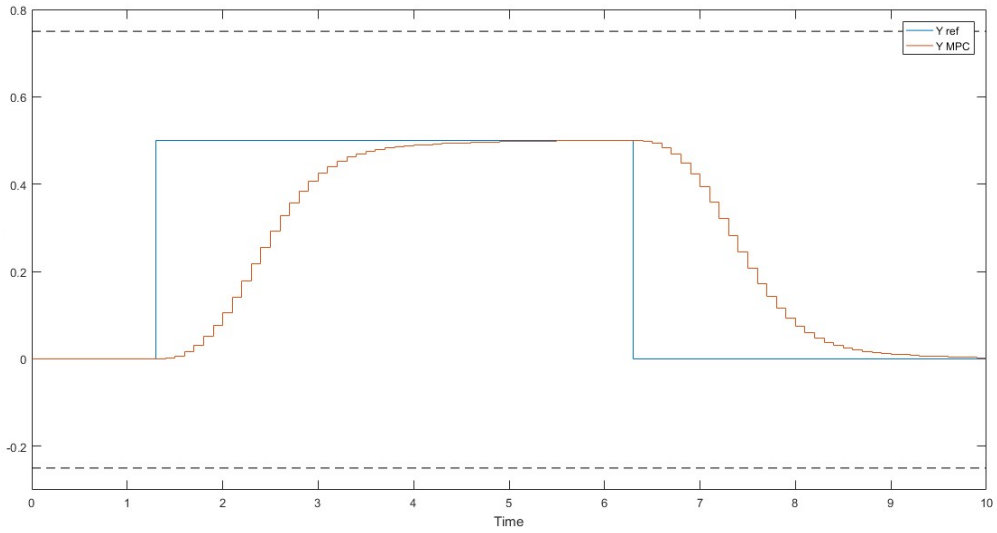


Figure 4.15: Plot of the lateral position signal with $N_p = 30$. All units in SI.

The increase on the prediction horizon has greatly increase the performance of the overall signals. As it can be seen in the figure 4.15, this improvement is more

noticeable in the lateral position signal as it was initially the worst. In the figure, it can also be seen that the overshoots have disappeared.

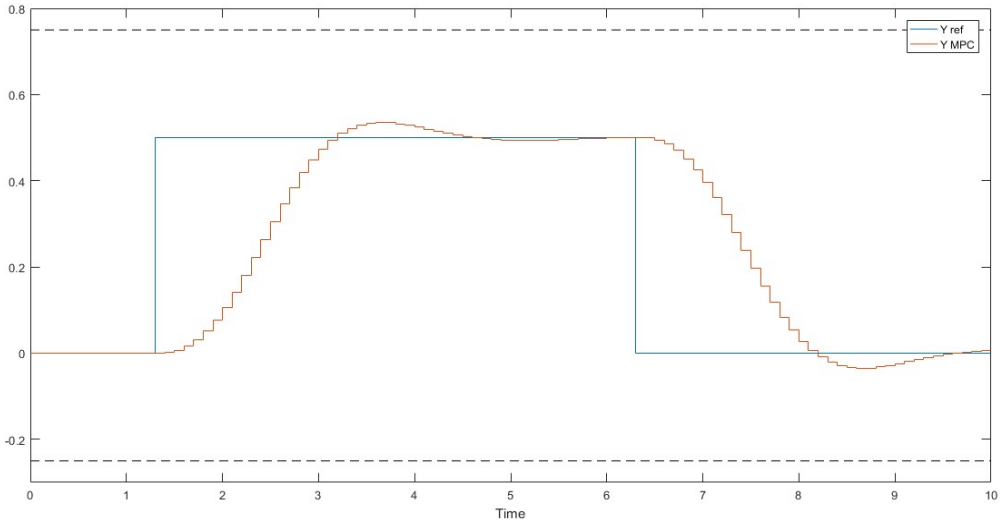


Figure 4.16: Plot over time of the lateral position signal with $N_p = 30$ and $N_u = 15$. All units in SI.

With the second simulation, we can see in the figure 4.16 that when increasing the control horizon, the signal is able to perform the reference change faster but with overshoot. If we keep increasing this parameter we can barely see any change on any of the signals and if we do, it is slightly worse.

Keeping all this observations in mind it is possible to tune the controller in order to reach the best signals performance, concluding with the following set of parameters: $Q_v = 15,\ Q_y = 200,\ R_x = R_y = 1,\ N_p = 30,\ N_u = 6.$ In the following figures (from 4.17 to 4.20), it can be seen the resulting signals plotted.

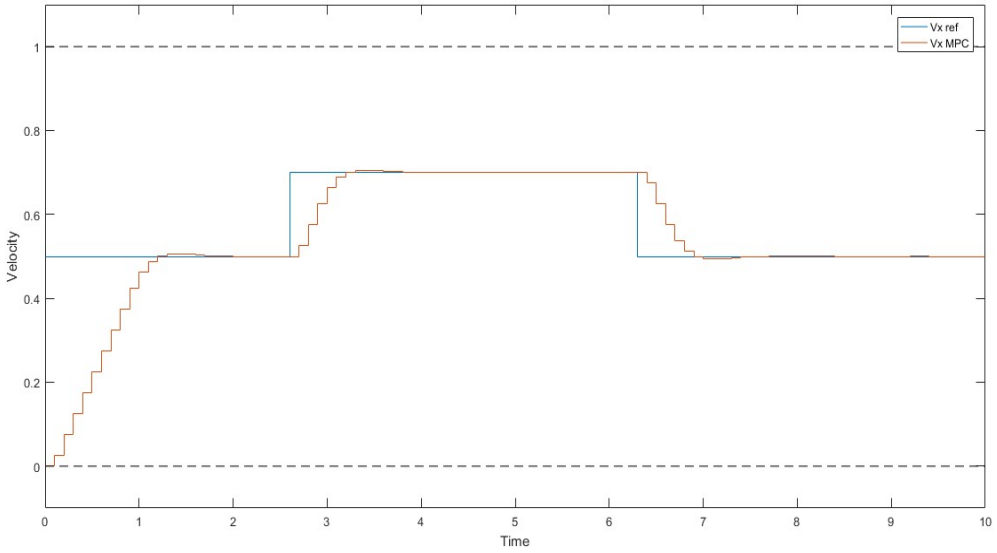


Figure 4.17: Plot over time of the longitudinal velocity signal with the final tuning values. All units in SI.

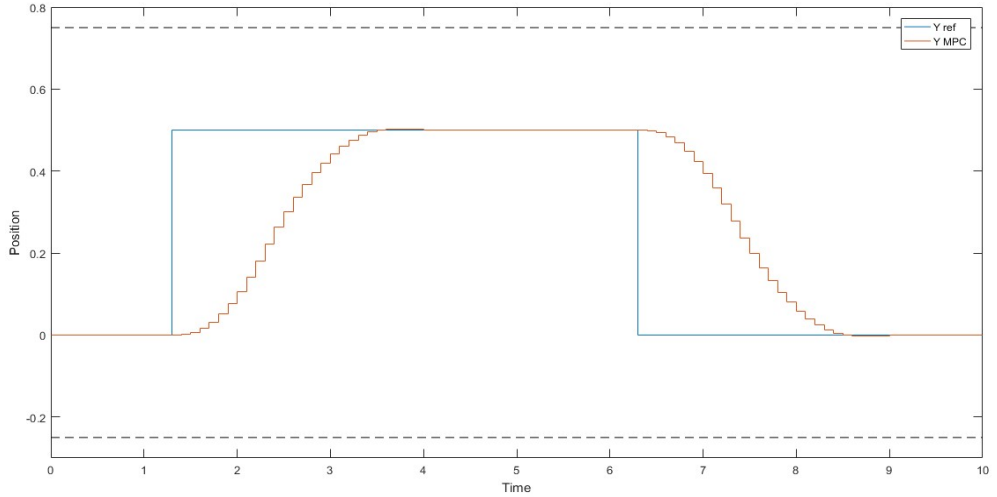


Figure 4.18: Plot over time of the lateral position signal with the final tuning values. All units in SI.

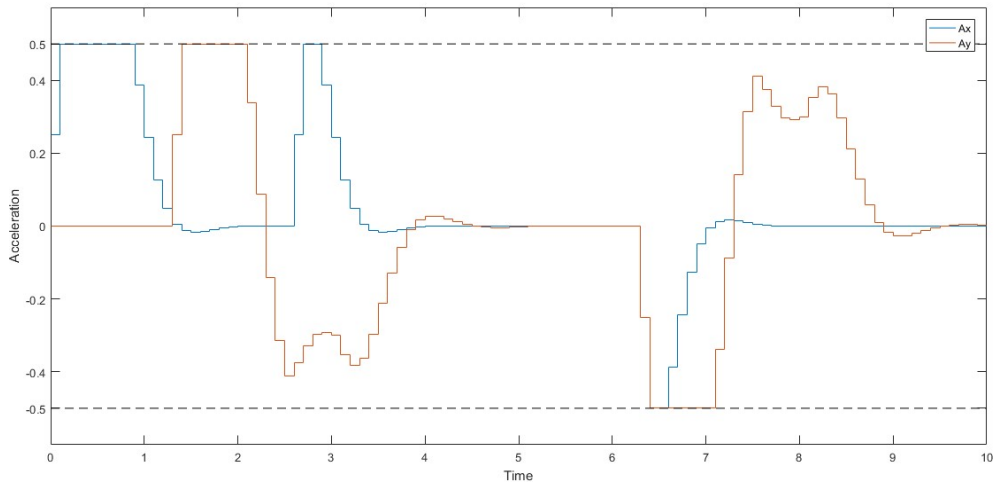


Figure 4.19: Plot over time of the acceleration signals with the final tuning values. All units in SI.

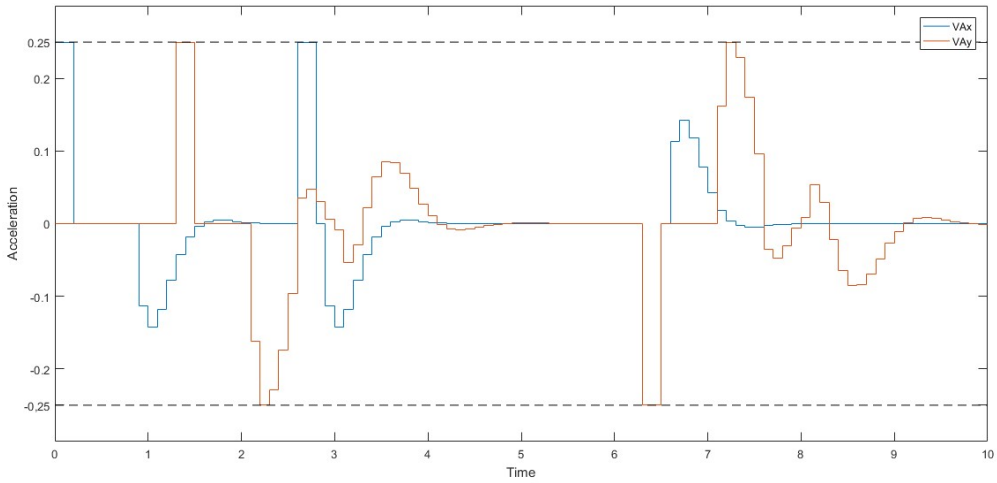


Figure 4.20: Plot over time of the acceleration variation signals with the final tuning values. All units in SI.

It should be kept in mind that this tuning only concerns the MPC controller parameters and as there is no output for some important signals as is the steering angle, further tuning will have to be made after the link with the low-level control.

Once the MPC controller is tuned, we can check his robustness and if the softening of the constraints explained in the last chapter is working or not. To test this, a pulse disturbance with an amplitude of 0.5 is added in both control variables (a_x and a_y). As the figures 4.21 and 4.22 show, the controller successfully passes the test as no unfeasibility situation appears even when the constraints are violated. In the figure 4.22 it can also be seen that, as natural, some overshoots appear in the signal due to the disturbance but it can still follow successfully the given reference.

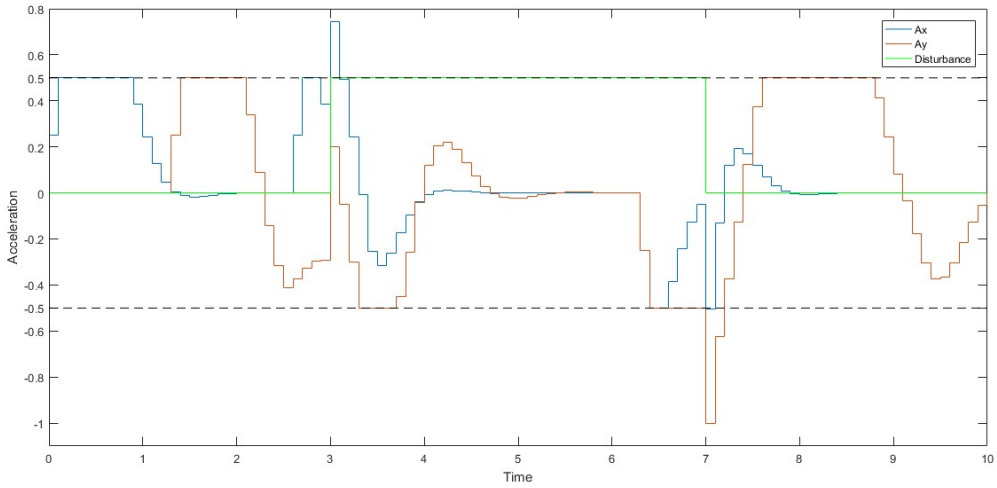


Figure 4.21: Plot over time of the acceleration signals and the disturbance applied. All units in SI.

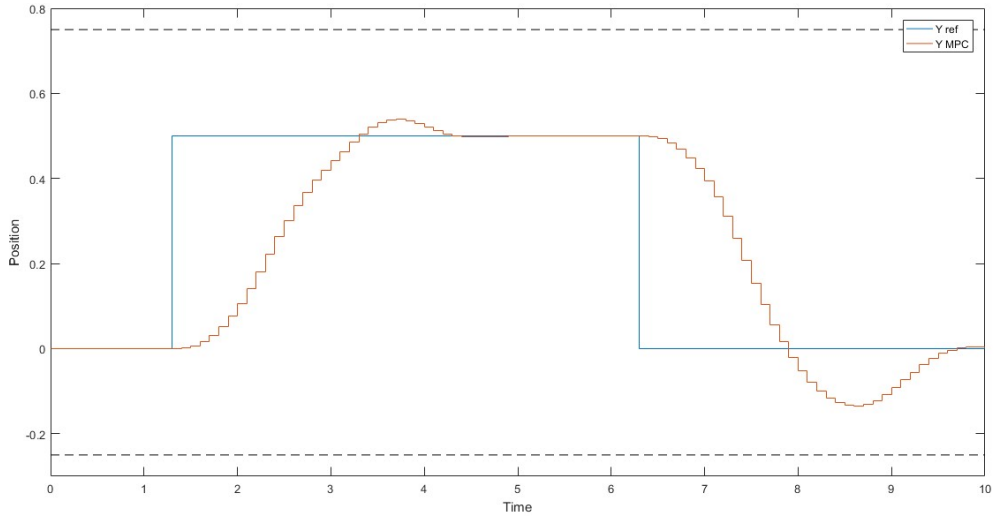


Figure 4.22: Plot over time of the lateral position signal in presence of disturbance. All units in SI.

4.4 Overall system simulation and tuning

Now it is time to link the MPC controller with the low-level control. The first thing to notice and to take into account is that the low-level control works with a sample time of 0.01 s while the MPC does it at 0.1 s. In order to be able to simulate, some rate transitions will be needed. Soon, after the very first simulation results, it will be notice that some more in deep signal treatment is needed and that it is not enough to just add rate transitions to link the controllers successfully.

This can be perceived in the figure 4.23, which shows the error signals. These errors are computed from the trajectory given by the MPC controller and the vehicle

states calculated in the kinematic model which simulates the vehicle behaviour. As it can be seen, there is a spiky behaviour in the signals caused for the rate difference of the inputs used to compute them.

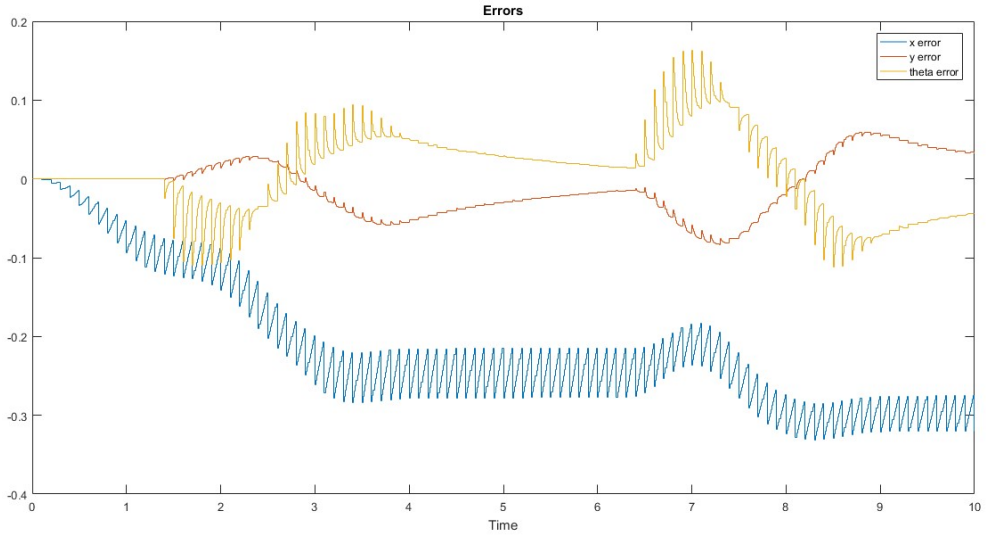


Figure 4.23: Plot over time of the error signals when no conditioning is done on the output signals of the MPC. All units in SI.

In order to solve this issue, we came up with several ideas for the conditioning of the MPC output signals: interpolating manually these signals, using a first order hold followed by a discrete lowpass filter or just using the discrete lowpass filter.

To decide which is the best option, firstly, a set of lowpass filters with different parameters are compared to see which ones adapt better to the signal. The filters 1 and 5 that are shown in the figure 4.24, were selected as the best (from now on they will be referred as first and second filter). The first, due to following the signal more precisely and the second one, for his smoothness. In this figure, it can also be seen that the signal processed with filter 4 shows really bad results.

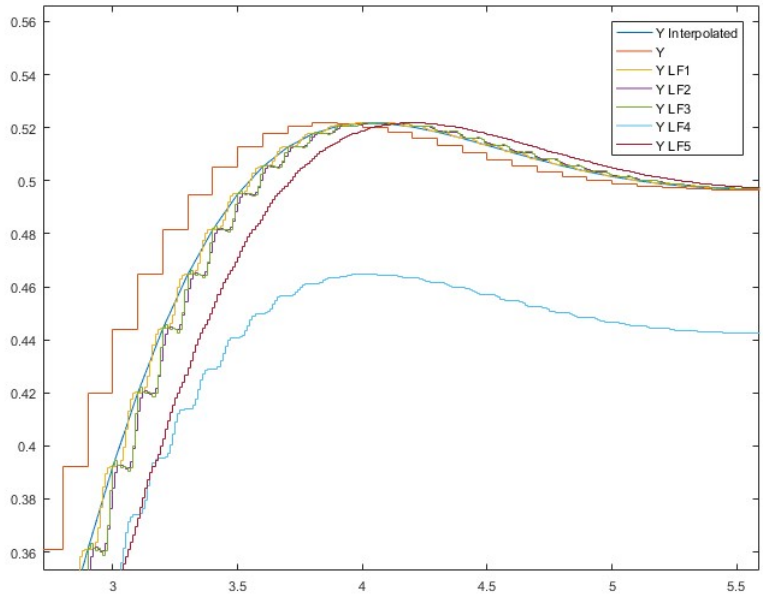


Figure 4.24: Comparison plot between filters for the lateral position signal over time. All units in SI.

The next step will be to do a comparison between all the signal processing methods proposed. In one hand, if we have a look at the figure 4.25 that corresponds to the lateral position signal, it can be seen that all the methods deliver a similar signal but with a different delay respect to the original, so all of them would be capable of doing the job. In the other hand, in the figure 4.26 corresponding to the steering angle it is shown that the signals processed with the first order hold element plus the first filter and the one processed just with the second filter have an undesired oscillating behaviour. Hence, they are discarded and the choice is between the interpolated signal and the one processed with the first order holder plus the second filter. An optimization test was held to see which processing method takes less time but the results were not decisive enough to make a solid choice so both methods were considered equally valid.

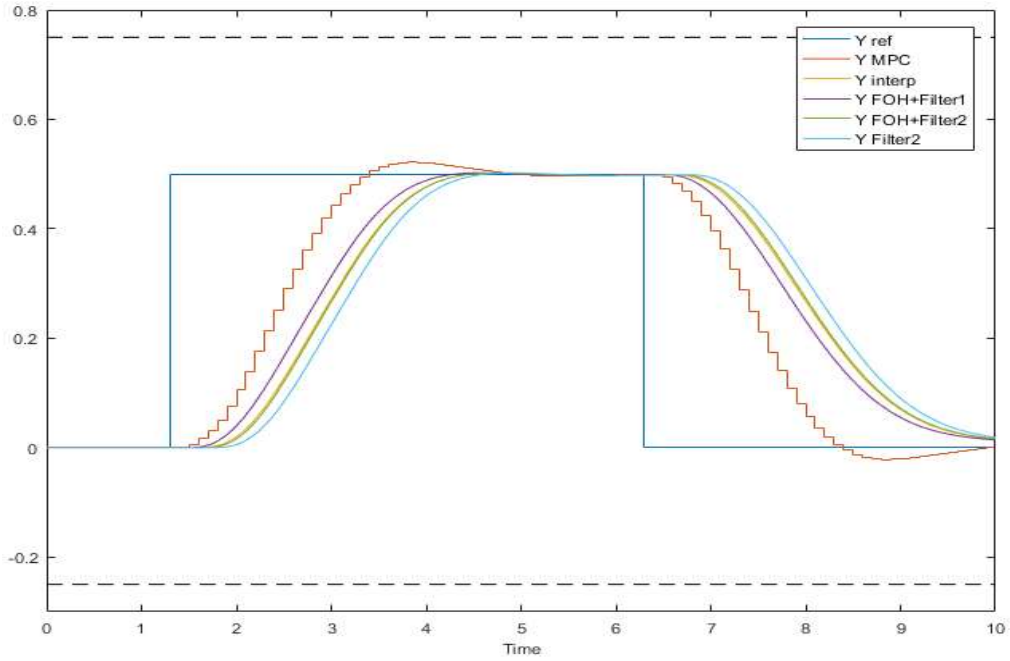


Figure 4.25: Comparison plot between the signal conditioning methods for the lateral position signal over time. All units in SI.

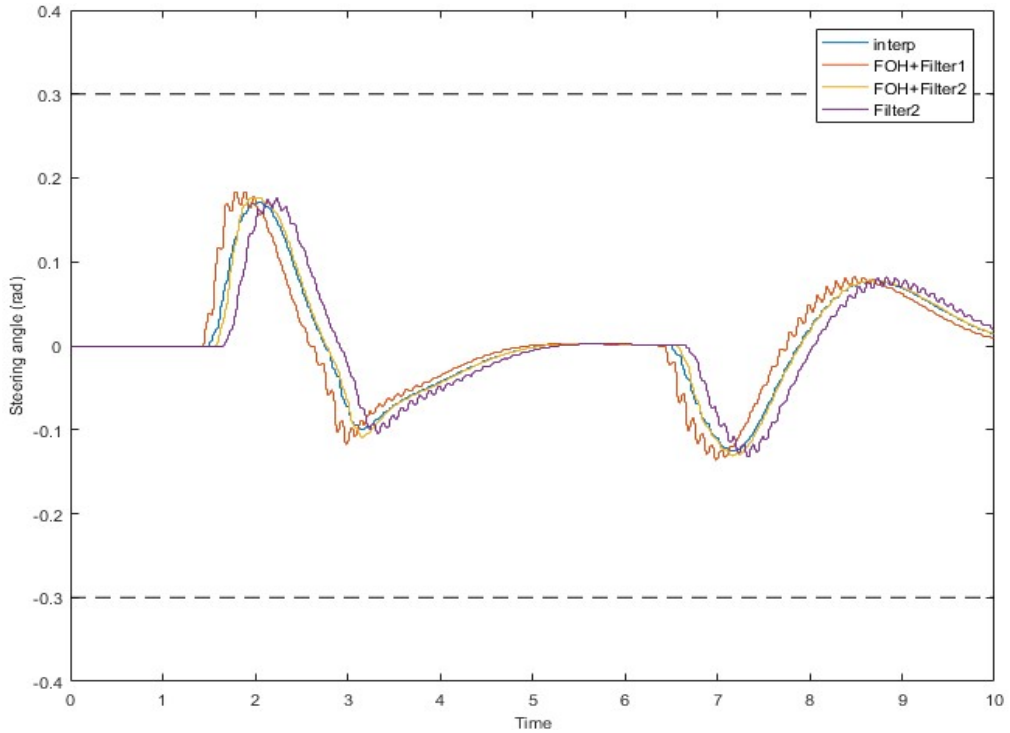


Figure 4.26: Comparison plot between the signal conditioning methods for the steering angle signal over time. All units in SI.

The interpolation processing was applied to each of the MPC output signals allowing the rest of the control to obtain a lot more valuable reference input. A clear display of this can be seen comparing the errors plot shown previously in the figure 4.23 with the one shown below, in figure 4.27.

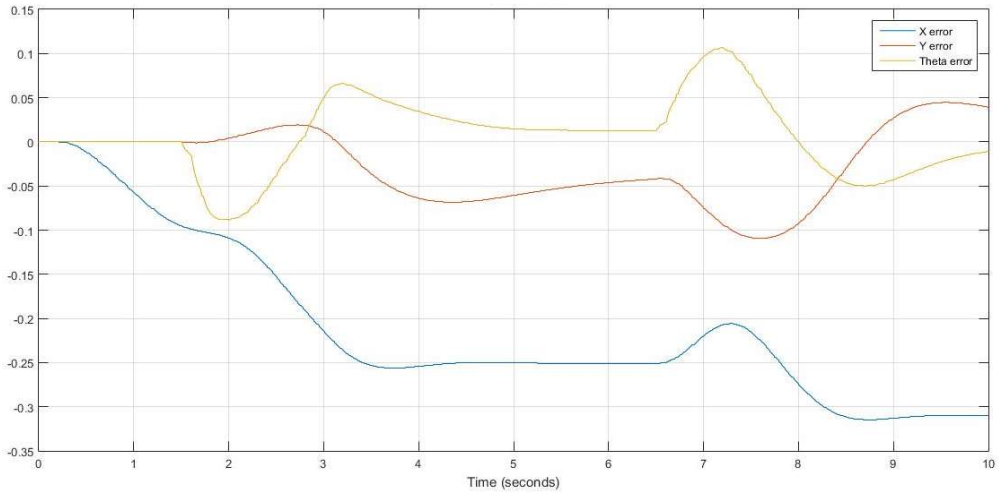


Figure 4.27: Plot of the error signals over time when an interpolation conditioning is done on the output signals of the MPC. All units in SI.

Now that both controllers are linked and the outputs of the MPC have been adapted successfully to the low-level control, we are able to proceed to the second tuning phase. In this stage, a kinematic model is used in simulation to try to mimic a real vehicle behaviour. Thanks to this model we can have a look at the simulated vehicle position, velocity and steering angle, also called vehicle states, and tune the controllers accordingly. The low-level controller is formed by a speed control, with its main tuning parameters being a proportional coefficient (K_p), and a steering control, with a gain for the lateral position error (Kstan) and another for the heading error (Ktheta) being its main tuning parameters.

The first thing to notice after running the first simulation is that the steering angle is not in between the saturation boundaries, which are set at $\pm$ 0.3 radians due to the capabilities of the testing trucks. Furthermore, the lateral position generated by the kinematic model, which represents the real truck behaviour, doesn't reach the step value of the reference (shown in figure 4.28). Hence, it will be needed to tune again the MPC parameters, this time, together with the low-level control.

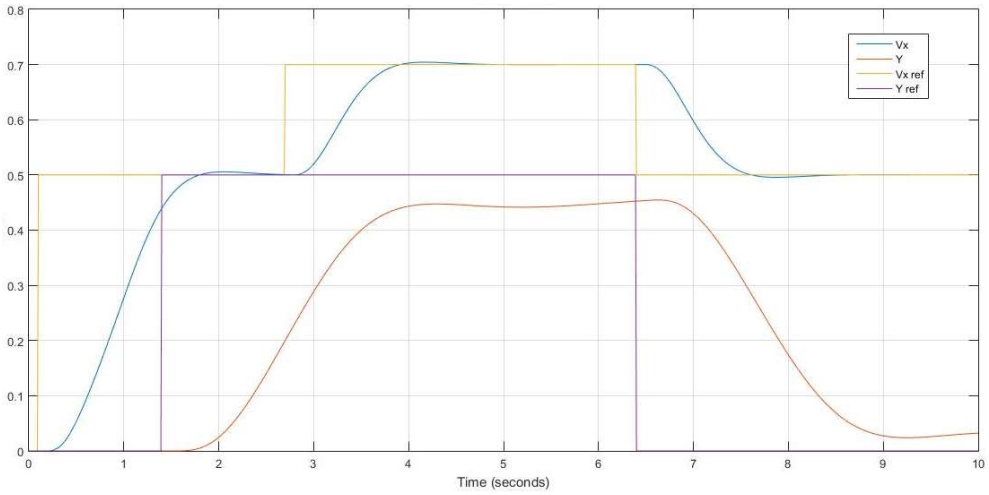


Figure 4.28: Plot of the lateral position (red) and longitudinal velocity (blue) signals generated by the kinematic model with the initial tuning values. All units in SI.

Firstly, the low-level control parameters presented previously are tuned, setting them at K_p = 5, Kstan = 4 and Ktheta = 1. With these modifications, the goal of lowering the steering to fit in between the limits is fulfilled and the lateral position reach the step value. Anyway, as it can be seen in figure 4.29, it looks like there is still some room for improvement.

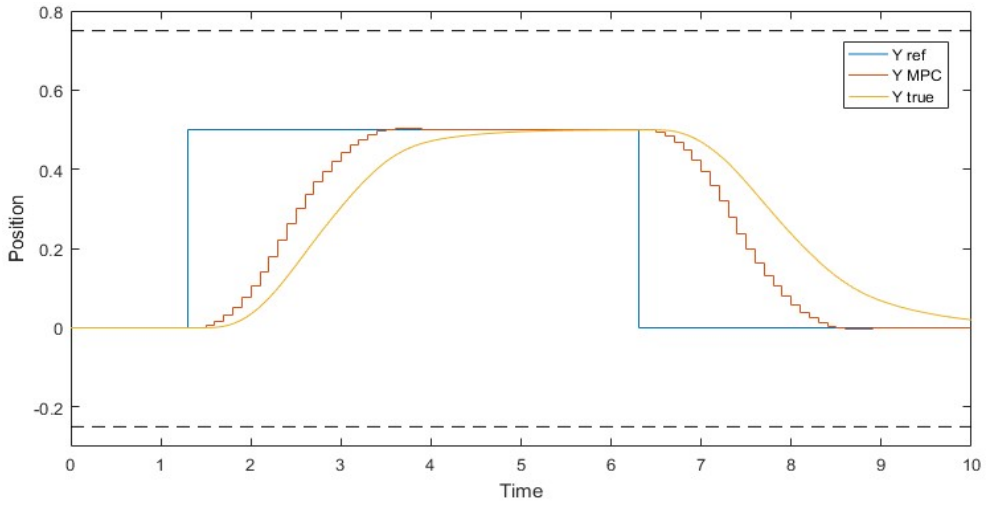


Figure 4.29: Lateral position signals corresponding to the reference (blue) and the generated by the MPC (red) and by the kinematic model (yellow). All units in SI.

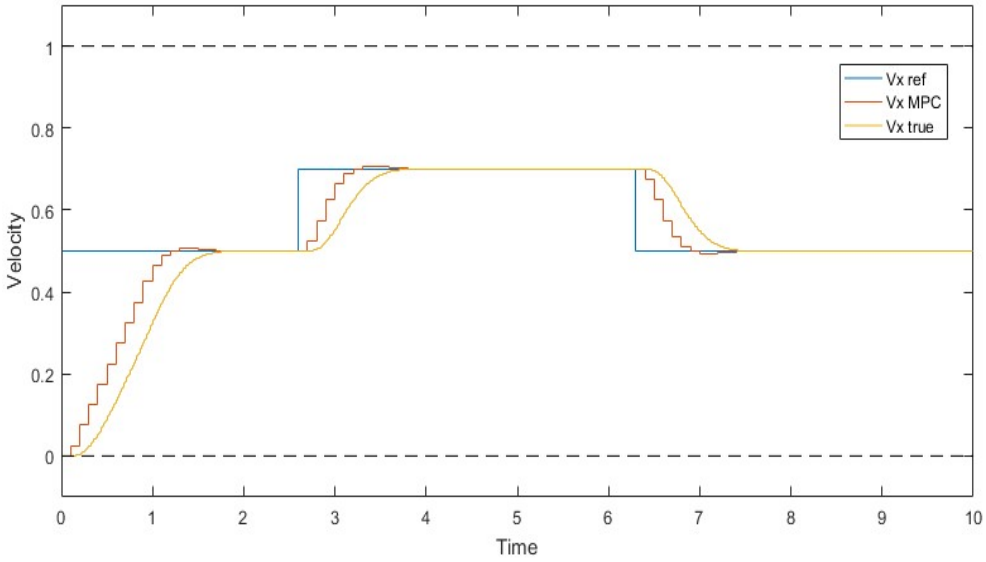


Figure 4.30: Longitudinal velocity signals corresponding to the reference (blue) and the generated by the MPC (red) and by the kinematic model (yellow). All units in SI.

This time the parameters of the MPC will be slightly tuned in order to improve the vehicle response. Specifically, the lateral position weight, Q_y, will be lowered from 200 to 60 and the weight of the lateral acceleration variation, R_y, increased from 1 to 60. The control horizon, N_u, will also be increased from 6 to 12. With this last tuning the set of parameters remains as follows: $K_p = 5$, $Kstan = 4$, $Ktheta = 1$, $Q_v = 15$, $Q_y = 60$, $R_x = 1$, $R_y = 60$, $N_p = 30$, $N_u = 12$. In the following figures (from 4.31 to 4.36), which correspond to the resulting signals with the new tuning parameters, it can be seen that the vehicle states generated by the kinematic model coincide quite well with the reference change, the steering angle remains in between the boundaries and the lateral acceleration signal has been smoothed.

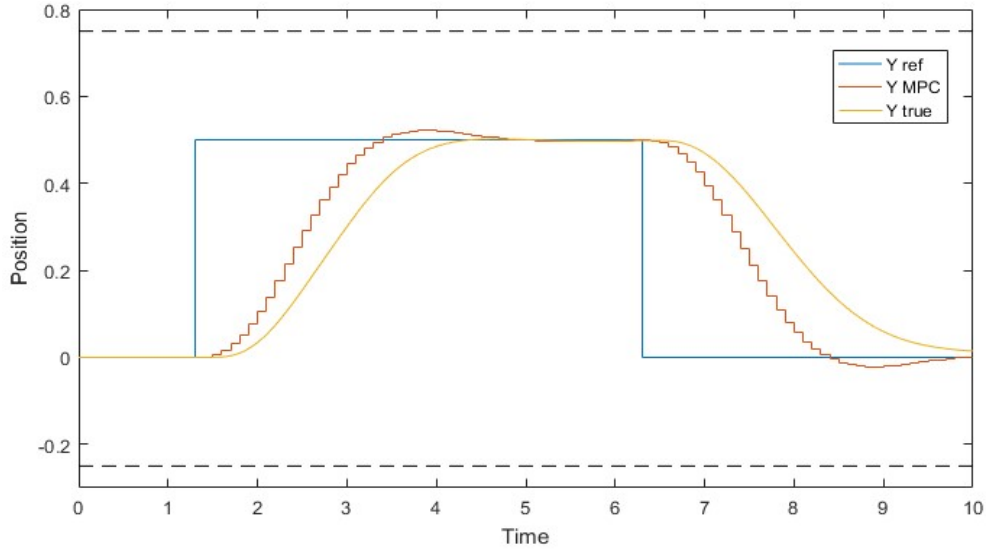


Figure 4.31: Lateral position signals corresponding to the reference (blue) and the generated by the MPC (red) and by the kinematic model (yellow). All units in SI.

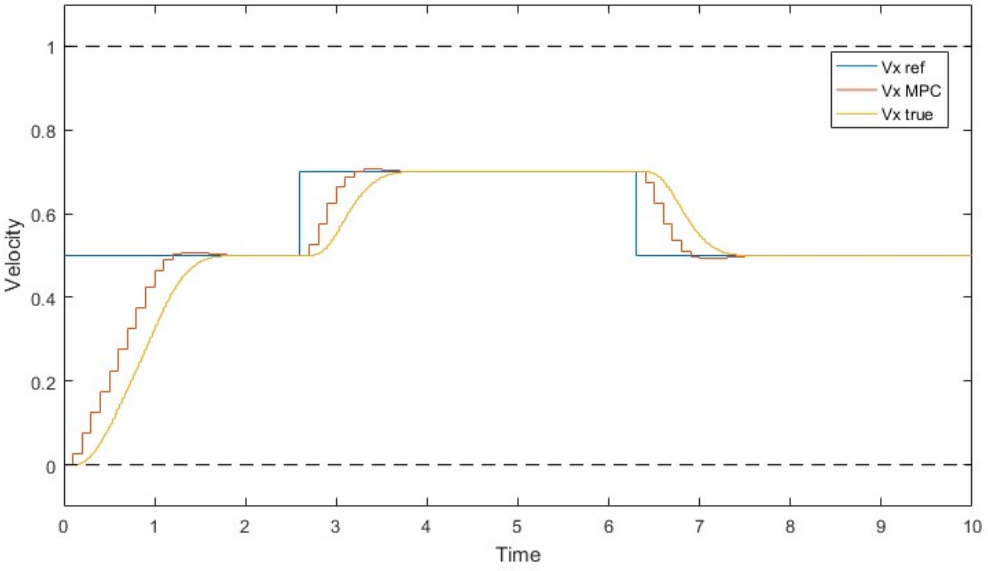


Figure 4.32: Longitudinal velocity signals corresponding to the reference (blue) and the generated by the MPC (red) and by the kinematic model (yellow). All units in SI.

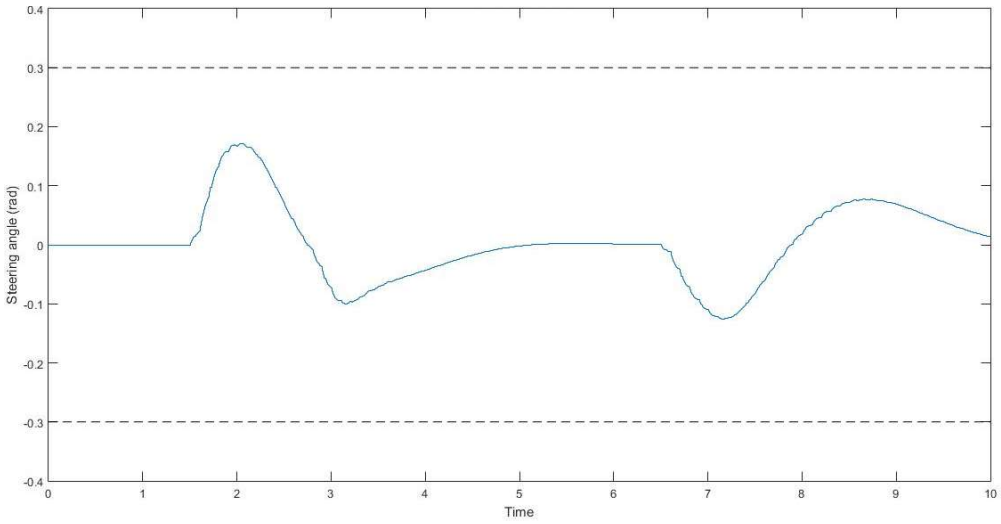


Figure 4.33: Plot of the steering angle signal with the final tuning values. All units in SI.

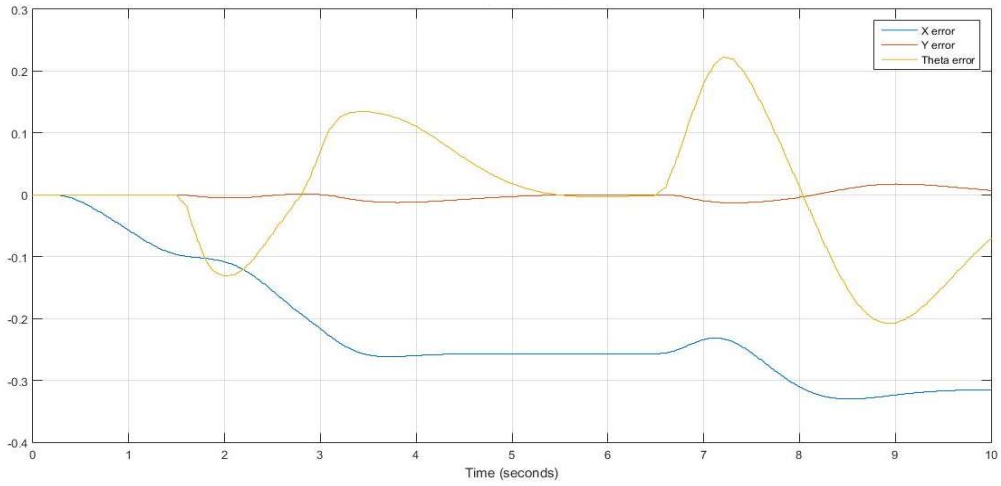


Figure 4.34: Plot of the error signals with the final tuning values. All units in SI.

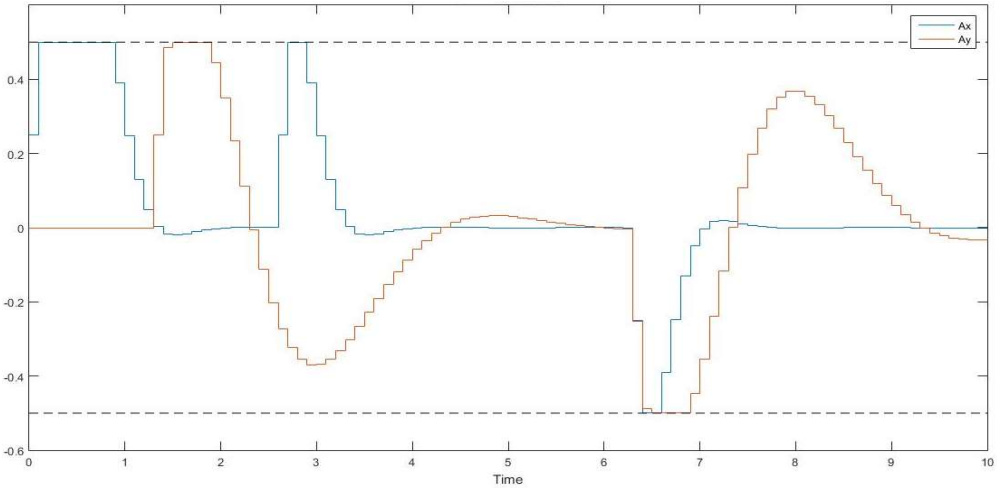


Figure 4.35: Plot of the acceleration signals over time with the final tuning values. All units in SI.

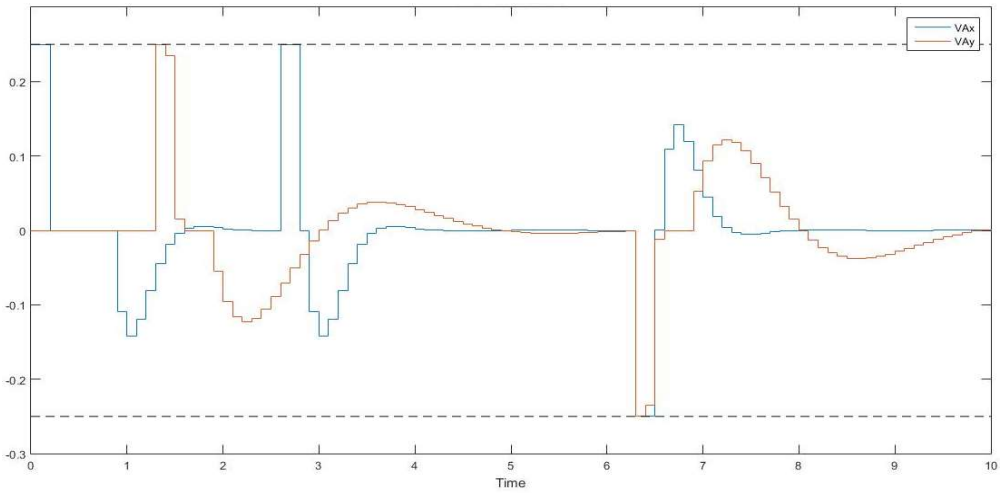


Figure 4.36: Plot over time of the acceleration variation signals with the final tuning values. All units in SI.

The tuning parameters of the low-level control haven't been deeply tuned in this stage because they only act as initial values for the testing stage, where they will have to be tuned again using the real testing vehicle feedback instead of the kinematic model.

5 Collision avoidance

5.1 Obstacle representation approaches

The first thing to do in order to implement a collision avoidance algorithm is to decide how to represent the obstacle. To choose the approach that fits the best our problem and the easiest to implement in our model, different approaches have been taken into account. The obstacle representation finally chosen will only be used as an initial step as it will be improved during the algorithm development and simulation. The most significant representations are explained below.

5.1.1 Nilson's approach

The approach that is explained firstly is the one developed in Nilson [11], [12] and [13]. This approach is thought to be used on highway environment and easily implemented in MPC formulation with constraints as each obstacle is represented with two lines that bound the feasible area. Those boundaries are called Forward Collision Avoidance Constraint (FCC) and Rear Collision Avoidance Constraint (RCC) and are shown in the figures 5.1 and 5.2 and formulated in the equations (5.1) and (5.2), respectively (forward and rear relative to the controlled vehicle, which is represented with the letter *E* and colour blue in the figures).

$$\text{FCC:} \quad \frac{\Delta x_j}{L_f} \pm \frac{\Delta y_j}{W} + \varepsilon_{jf} \geq 1, \quad j = 1, \dots, q \tag{5.1}$$

$$\text{RCC:} \quad \frac{\Delta x_j}{L_r} \pm \frac{\Delta y_j}{W} + \varepsilon_{jr} \leq -1, \quad j = 1, \dots, q \tag{5.2}$$

where the subscript *j* stands for the obstacle, hence, *q* is the number of obstacles. ε_{jf} and ε_{jr} are the slack variables for the front and rear collision avoidance constraint, respectively. The rest of the collision constraints parameters are defined as follows:

$$\Delta x_j = x_j - x \qquad \Delta y_j = y_j - y$$

$$L_f = v_x \cdot \theta_f + L_c \qquad L_r = v_x \cdot \theta_r + L_c \qquad W = \frac{1}{2} \cdot W_L + W_c$$

with θ_f and θ_r being the desired time gap between vehicles for the front and rear constraints, respectively, W_L stands for the lane width and the length and width of the obstacle are respectively denoted by L_c and W_c.

If both collision avoidance constraints (CC) are simultaneously enforced, the feasible area is reduced to a small triangle adjacent to the obstacle. Hence, the slack variable $\varepsilon_{jf} \geq 0$ is included to relax the FCC if and only if $\Delta x_j \leq 0$ and $\varepsilon_{jr} \leq$ relax the RCC if and only if and $\Delta x_j \geq 0$.

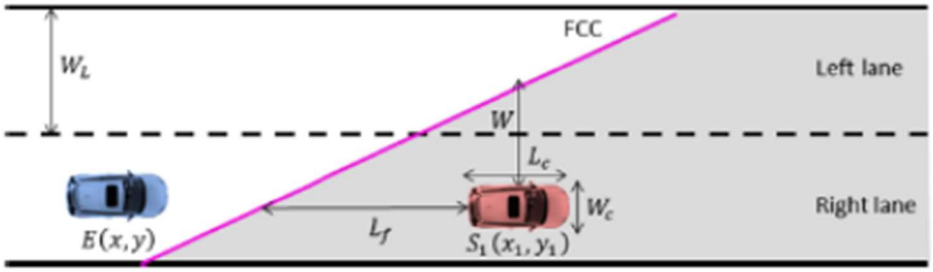


Figure 5.1: FCC schematic representation of the obstacle S_1. The unfeasible area is displayed in colour gray. Figure acquired from [11].

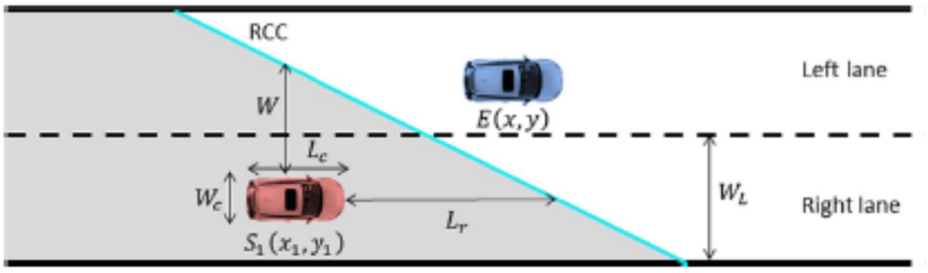


Figure 5.2: RCC schematic representation of the obstacle S_1. The unfeasible area is displayed in colour gray. Figure acquired from [11].

In order to implement this obstacle representation into the MPC formulation, two different approaches are presented:

1) By modifying the cost function
2) By only adding new constraints

1) In this approach, the enforcement of the FCC or RCC is achieved by adding weights for the slack variables in the cost function, as shown in equation (5.3) and introducing a pair of additional constraints which defines the relation between the

slack variables ε_{jf}, ε_{jr} and the relative distance Δx_j, shown in equations (5.4) and (5.5).

$$J = J + \sum_{k=0}^{N-1} \sum_{j=1}^{q} \eta \cdot {\varepsilon_{jf}}_k^2 + \phi \cdot {\varepsilon_{jr}}_k^2 \qquad j = 1, \dots, q \tag{5.3}$$

$$\Delta x_j + \varsigma \cdot \varepsilon_{jf} \geq 0 \qquad j = 1, \dots, q \tag{5.4}$$

$$\Delta x_j + \varsigma \cdot \varepsilon_{jr} \leq 0 \qquad j = 1, \dots, q \tag{5.5}$$

where η and ϕ are weights for the CC slack variables and ς is just a tuning parameter.

Alternatively, we can also formulate this approach by only modifying the cost function as shown in equation (5.6).

$$J_1 = J + \sum_{k=0}^{N-1} \sum_{j=1}^{q} \eta \cdot {\varepsilon_{jf}}_k^2 + \phi \cdot {\varepsilon_{jr}}_k^2 + \varpi \cdot \Delta {x_j}_k \cdot {\varepsilon_{jf}}_k + \Omega \cdot \Delta {x_j}_k \cdot {\varepsilon_{jr}}_k + \Lambda \cdot \Delta {x_j}_k^2 \tag{5.6}$$

where in this case we also have ϖ, Ω and Λ as weighting matrices.

The addition of the term $\Delta {x_j}_k^2$ is needed in order to make the Hessian matrix of the cost function positive semi-definite, in the case the function shown in equation (5.6) is used. In the other hand, this encourages the vehicle to stay close to the obstacle, since the relative position is penalized, and it can cause an undesirable behaviour where the vehicle adapts to the velocity of the obstacle instead of maintaining its desired velocity. For this reason, it is better to not use this alternative where we solely modify the cost function.

It is important to notice that the paper [11] remarks that this approach only works properly when the parameter tuning is well done as it cannot be guaranteed that the slack variables will not assume non-zero values when it is not desirable.

2) In the second approach, the decision to enforce the FCC or RCC is solely done through a set of constraints. This is done considering that the FCC is only allowed to be relaxed if the vehicle has either changed lane or passed the obstacle. This leads to redefine the FCC and RCC as it is shown in the equations (5.7) and (5.8), respectively.

$$\frac{\Delta x_j}{L_f} \pm \frac{\Delta y_j}{W} + \vartheta_j \, \varepsilon_{xjf} + \frac{\varepsilon_{yj}}{\varphi_j} + \varepsilon_{jf} \geq 1, \quad j = 1, \dots, q \tag{5.7}$$

$$\frac{\Delta x_j}{L_r} \pm \frac{\Delta y_j}{W} + \vartheta_j \, \varepsilon_{xjr} + \frac{\varepsilon_{yj}}{\varphi_j} + \varepsilon_{jr} \leq -1, \quad j = 1, \dots, q \tag{5.8}$$

The purpose of the $\vartheta_j \, \varepsilon_{xjf}$ term is to relax the FCC if the vehicle has passed the obstacle, the purpose of the $\varepsilon_{yj}/\varphi_j$ term is to relax the FCC if the vehicle has changed lanes and the term ε_{jf} should only relax the FCC if no other feasible options exist. The same idea can be applied for the RCC.

The cost function is now set as shown in equation (5.9).

$$J = J + \sum_{k=0}^{N-1} \sum_{j=1}^{q} \chi \cdot {\varepsilon_{jf}}_k^2 + \Xi \cdot {\varepsilon_{jr}}_k^2 \qquad j = 1, \dots, q \tag{5.9}$$

with χ and Ξ being weighting matrices for the CC slack variables.

The main drawback of this approach is that during an optimization cycle a full overtake manoeuvre can never be planned as the vehicle is only allowed to change lane when it has passed the obstacle.

5.1.2 Eilbrecht's approach

The second approach to analyse is the one presented in Eilbrecht [3], where the controlled vehicle and the obstacles are overapproximated by rectangles, making the problem of obstacle avoidance become a problem of avoiding the intersection of rectangular regions. In order to determine if a point lies inside a certain rectangle, one has to check on which side of the four rectangle boundaries the point is located. To model it, a binary variable $\delta_{j,L}$ is used, indicating whether a point belonging to the vehicle has crossed the L boundary of the obstacle j bounding rectangle. Then, by using the "Big M" method explained in Williams [18], the logical expressions can be reformulated as linear inequalities containing binary variables in a mixed-integer formulation, as shown in equations (5.10) and (5.11).

$$\epsilon + (m - \epsilon) \cdot \delta_{j,L} \leq M\left(1 - \delta_{j,L}\right) \tag{5.10}$$

$$\sum_{l=1}^{4} \delta_{j,L} \geq 1 \tag{5.11}$$

where ϵ is a small tolerance, while m and M are large constants, in comparison.

Depending on the vehicle driving orientation, either the lateral or the longitudinal margin needs to be added. Exactly covering arbitrary orientations leads to a nonlinear problem due to trigonometric functions. In order to still account for all cases and retaining a linear formulation, it would be required to enlarge obstacles by the longer side of the vehicle, introducing a high degree of conservatism that may not be tolerable due to the limited size of the road. This is solved following the approach presented by Schouwenaars [14], where the size of the obstacles is increased by either the length or the width of the vehicle bounding rectangle depending on its position relative to the obstacle. Also, as the orientation of the vehicle will never deviate by large angles from the orientation of the road, the vehicle can be represented as an axis-aligned rectangle without introducing too much conservatism as it is shown in the figure 5.3.

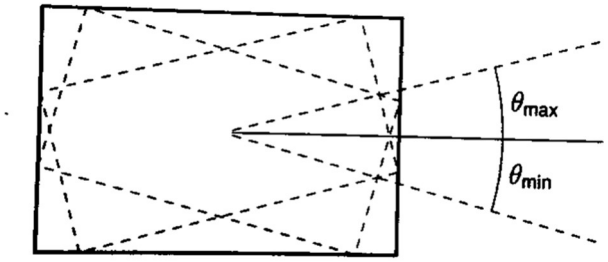


Figure 5.3: Obstacle representation as axis-aligned rectangle. Figure acquired from [3].

Therefore, this allows to reduce the problem to four cases. The vehicle is either in front of an obstacle or behind it with respect to its traveling direction. Otherwise, it is located to the left or right of the obstacle.

Finally, another reasonable extension is to add a velocity-depending safety margin to the obstacle when driving behind another vehicle as shown in the equation (5.12).

$$l_{safe} = \frac{3.6}{2}|v_x| \qquad (5.12)$$

5.1.3 Ferrara's approach

The last approach described in this section is the one presented in Ferrara [7]. In this approach, the collision detection task is performed relying on a collision cone. The theory underlying the construction of this cone has been developed in Chakravarthy [2]. To explain it, let us represent the vehicle as the point O and the obstacle as the circle with centre point F and radius R as shown in the figure 5.4.

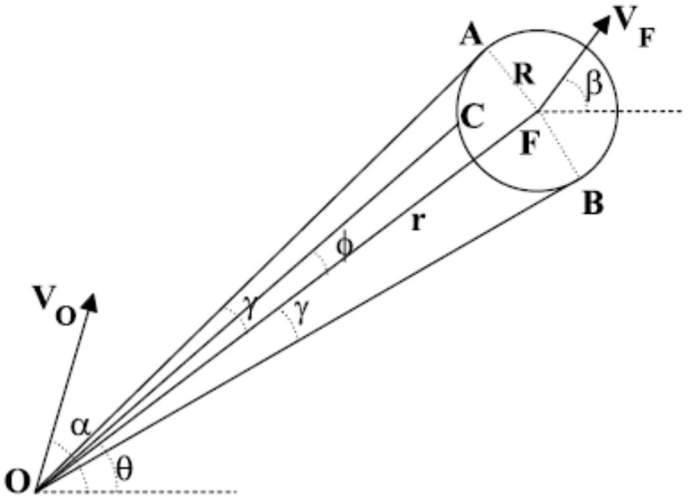


Figure 5.4: Collision cone representation. Figure acquired from [6].

In a polar-coordinates reference centred in the vehicle O, the motion of the obstacle F with respect to the vehicle is described by the two speed components V_r and V_θ as shown in equations (5.13) and (5.14), respectively.

$$V_r = V_F \cdot \cos(\beta - \theta) - V_O \cdot \cos(\alpha - \theta) \qquad (5.13)$$

$$V_\theta = V_F \cdot \sin(\beta - \theta) - V_O \cdot \sin(\alpha - \theta) \qquad (5.14)$$

Relying on the assumption of constant speed for both points, it can be proved that the possibility that a collision occurs depends only on the initial conditions. The range of values of α which verify the condition of future collision, takes the name of "collision cone". To determine the collision cone, it is necessary to consider the extreme points A and B, which leads to the collision condition $(V_\theta)_{OA} \cdot (V_\theta)_{OB} \leq 0$. This can be rewritten as shown in equation (5.15).

$$r^2 \cdot {V_\theta}^2 \leq R^2\{{V_r}^2 + {V_\theta}^2\} \qquad (5.15)$$

where r is the distance between the points O and F, as shown in the figure 5.4.

Then, it can be proved that a point and a circle moving with constant velocities, will collide if, and only if, their initial conditions satisfy equations (5.16) and (5.17).

$${V_{\theta 0}}^2 \leq p^2 \cdot {V_{r0}}^2 \qquad (5.16)$$

$$V_{r0} < 0 \qquad (5.17)$$

with $p = R/\sqrt{r_0^2 - R^2}$.

A further step is to analyse the collision problem when both O and F are circular objects. In this case, the collision cone is simply defined with p replaced by a value computed as if O was a single point and F was enlarged to account $R_F + R_O$.

This approach needs to compute the obstacles and vehicles as circles which can lead to a high degree of conservatism that may not be tolerable due to the limited size of the road. This can be solved by representing the objects by a suitable series of small circles instead of a large one as is it shown in the figure 5.5. Anyway, this representation can lead to a high number of restrictions in an environment with multiple obstacles.

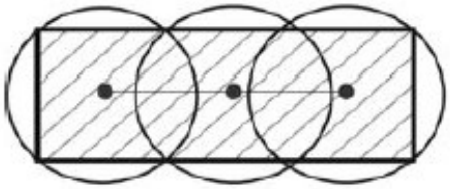


Figure 5.5: Representation of the vehicle or obstacles through a series of small circles. Figure acquired from [6].

5.1.4 Obstacle representation choice

To represent the obstacles that the controlled vehicle has to avoid, it has been thought that the best approach to use as starting point for our problem is through the collision avoidance constraints (CC). This approach corresponds to the first one presented on this chapter and described in [11], [12] and [13]. The main reason behind this choice, having in mind the plant model and the overall MPC formulation already used for the lane change problem, is that it does not need a lot of changes on those in order to achieve a first working approach that can be used as starting point. As explained earlier in this chapter, there are two ways of implementing this approach through MPC formulation. The one firstly used, is through modifications in the cost function, even if it is already known that this implementation has some issues with his performance, as it requires an accurate parameter tuning to work properly.

Once this first approach is implemented and after carrying out several simulations in different scenarios, it will be seen that the high parameter tuning dependence is not the only issue that this approach has. It will also be seen that, with the way the CC are implemented, the approach only works in a certain series of scenarios and when the controlled vehicle is close to the obstacles.

For these reasons, several modifications and improvements will be implemented, leading to new working approaches and ending with a completely different implementation compared with the one used as starting point.

All the different implementations carried out, together with the modifications and the reasons that leaded to them, will be explained in detail in the Simulation chapter.

5.2 Model extension

In this chapter, the general plant model used for the collision avoidance trajectory generation problem will be defined.

First, there are several assumptions made on the model that must be presented. The first one is that it is considered that there are only two obstacles in the proximity of the controlled vehicle, hence, only these two obstacles are described in the model. The second assumption, is that the obstacle with the subscript 1, is supposed to always drive in the middle of the first lane, $y_1 = 0$, and the one with the subscript 2, to always drive in the middle of the second lane, $y_2 = 0.5$. Lastly, it is also assumed that the obstacles move in a constant velocity, hence, the corresponding accelerations are: $a_1 = 0$ and $a_2 = 0$.

The plant used for the collision avoidance trajectory generation problem, is basically an extension of the plant model used for the lane change approach, the point-mass kinematic vehicle model that has already been presented in chapter 3.3.3. In this extension, the longitudinal position and velocity of the obstacles are added as shown in equation (5.18).

$$X_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & T_s & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & T_s \\ 0 & 0 & 1 & 0 & T_s & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & T_s & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot X_k + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.5 \cdot {T_s}^2 & 0 \\ 0 & 0.5 \cdot {T_s}^2 \\ T_s & 0 \\ 0 & T_s \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot u_k \tag{5.18}$$

with

$$X = [x_1 \quad x_2 \quad x \quad y \quad v_x \quad v_y \quad v_1 \quad v_2]' \qquad u = [a_x \quad a_y]'.$$

The output vector of the model is still composed by the lateral position and the longitudinal velocity of the controlled vehicle, $Y = [y \quad v_x]'$, described by equation (5.19).

$$Y_k = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \cdot X_k \tag{5.19}$$

One last important thing to mention, is that in reality and, therefore, in the test bench the assumptions explained in the beginning of this chapter are not always fulfilled. To solve that, the whole plant model described before, is only used for the prediction. Then, only the next vehicle states are calculated through the plant, while the obstacle states are measured in real time.

5.3 Cost function

Regarding the cost function used in the collision avoidance trajectory generation problem, it has the same general structure as the one used for the lane change approach. The only addition is the weights of the collision avoidance slack variables, used on the obstacle representation implemented as initial approach. Anyway, they will not be used in the definitive version, obtained after applying all the improvements.

$$V(k) = \sum_{i=1}^{N_p} \|Y_{k+i}(k) - r_{k+i}(k)\|^2{}_{Q_i} + \sum_{i=0}^{N_u-1} \|\Delta U_{k+i}(k)\|^2{}_{R_i} + R_\varepsilon \varepsilon + \rho \epsilon \tag{5.20}$$

All the parameters used in equation (5.20) are the same described in the lane change approach, chapter 4.1 and 4.2, with the addition of the factor $R_\varepsilon \varepsilon$ that corresponds to the weighting matrix of the CC slack variables multiplied by them. The parameters of this factor change with each approach and it does not even appear in the cost function for the definitive version of the collision avoidance trajectory generation problem. For this reason, it will be described in detail later, in the chapter 5.5, where each approach will be explained.

As done in chapter 4.1, the cost function can be expressed as a function of the optimization vector. This vector is now extended with the slack variables and expressed in matrix form, as it is shown in equation (5.21).

$$V(k) = -\begin{bmatrix} \Delta\hat{U}(k) \\ \varepsilon \\ \epsilon \end{bmatrix}^T \cdot \begin{bmatrix} Gamma \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \Delta\hat{U}(k) \\ \varepsilon \\ \epsilon \end{bmatrix}^T \cdot \begin{bmatrix} H & 0 & 0 \\ 0 & R_\varepsilon & 0 \\ 0 & 0 & \rho \end{bmatrix} \cdot \begin{bmatrix} \Delta\hat{U}(k) \\ \varepsilon \\ \epsilon \end{bmatrix} \tag{5.21}$$

where all the parameters shown in the equation, have already been defined before in this chapter or in chapter 4.1, as most of them remain the same as in the lane change approach, like it has already been said.

5.4 Constraints

In this chapter, the global constraints used for the collision avoidance trajectory generation problem will be presented. For this problem, all the constraints and their limitation parameters already implemented in the lane change approach will be used, but also adding new ones. Hence, the reader is referred to chapter 4.2, where the lane change constraints are described in detail.

A zero-longitudinal velocity and a non-zero lateral velocity generates a lateral movement, which is unfeasible for a vehicle. Nevertheless, it can still generate a trajectory that can be followed by the vehicle but that leads it to an undesirable position, as a vehicle on a highway environment should never be oriented perpendicular to the road direction. For this reason, in addition to the constraints shown in equations from (4.14) to (4.19), it has been included a restriction of the side slip angle defined as $\beta = \arctan\left({}^{v_y}/_{v_x}\right) = \pm 20°(\approx 0.35 \text{ rad})$. Making use of a small angle approximation, equation (5.22) is obtained.

$$\hat{X}(k): \qquad -0.35 \cdot v_x \leq v_y \leq 0.35 \cdot v_x \tag{5.22}$$

The last constraints to be added are the ones corresponding to the collision avoidance. As these constraints change for each approach, they will be presented and described in detail in the next chapter.

Last thing to do, as it was also done for the lane change approach, is to set the constraints as a function of the optimization vector. The lane change constraints are built as shown in equations from (4.20) to (4.22). As states variables appear in the collision avoidance constraints as well as in constraint (5.22), they are built in matrix form together as shown in equation (5.23).

$$[J, j] \cdot \begin{bmatrix} \hat{X}(k) \\ 1 \end{bmatrix} \leq 0 \tag{5.23}$$

where $J = [J_1, \dots, J_{Np}]$ and j is the last column vector of the matrix, which is the vector that defines the scalar part of the constraints.

Finally, making use of equation (5.24), equation (5.23) is expressed together with all the other constraints as a function of the optimization vector, as shown in equation (5.25). Parameters from equations (5.24) and (5.25) were already described in chapter 4.1.

$$\hat{X}(k) = A_a \cdot X_k + A_b \cdot u_{k-1} + A_{ba} \cdot \Delta\hat{U}(k) \tag{5.24}$$

$$\begin{bmatrix} \Omega & 0 \\ J \cdot A_{ba} & \mathcal{E} \end{bmatrix} \cdot \begin{bmatrix} \Delta\hat{U}(k) \\ \varepsilon \end{bmatrix} \leq \omega + \tau\epsilon \tag{5.25}$$

with

$$\Omega = \begin{bmatrix} \mathcal{F} \\ G \cdot \Theta \\ E \end{bmatrix}, \qquad \omega = \begin{bmatrix} -\mathcal{F}_1 \cdot u_{k-1} - f \\ -G \cdot [\Psi \cdot X_k + \Upsilon \cdot u_{k-1}] - g \\ e \\ -J \cdot [A_a \cdot X_k + A_b \cdot u_{k-1}] - j \end{bmatrix}$$

where $\mathcal{E}$ selects the collision avoidance slack variables used in the CC, τ represents a weighting vector of the global slack variable to allow several hardness levels in the constraints ($\tau = 0$ corresponds to a hard constraint as the global slack variable is not used) and the rest of the parameters are already described in chapter 4.2.

5.5 Simulation

To explain the collision avoidance trajectory generation development and simulation process, as several approaches have been implemented on the way to achieve the definitive version and each of them has its specific cost function and collision avoidance constraints formulation, it has been thought convenient to explain first the simulation process and after, describe in detail the specific formulations used on each approach.

Several aspects of the MPC structure and formulation described in the lane change approach are equally valid for the collision avoidance trajectory generation. One of these aspects is the reference computation discussion, where it was previously proved that the computation without preview is more beneficial in this problem than using preview.

Another aspect that remains unchanged, is the softening strategy of the constraints. All the constraints used in the collision avoidance trajectory generation problem are softened with the same strategy implemented in the lane change approach and described in chapter 4.2. Hence, the only constraints that do not have

slack variables and, therefore, that remain as hard constraints are the ones regarding the MPC outputs, which are the same variables. These constraints correspond to the road boundaries and the longitudinal velocity limits of the controlled vehicle. The only aspect that changes regarding the constraint softening, is how the slack variables are implemented for the collision avoidance constraints. In the first approaches, independent slack variables are used for each collision avoidance constraint, having their own weighting in the cost function. This changes in in the definitive version of the problem, where only one slack variable is used for all the constraints but with a different weighting parameter for each of them. This softening strategy used for the definitive version, allows various levels of softening in the constraints, leading the optimizer to prefer violating constraints like the variation of the acceleration limits or the acceleration limits themselves than violating the collision avoidance constraints, when an unfeasible situation is faced. All this, while using a single slack variable and therefore, requiring less computation effort.

One last aspect that remains almost unchanged, is the one regarding the tuning variables. These variables are the same for the MPC as for the low-level control but now, in the MPC there are also the collision avoidance constraints that incorporate several variables that need specific tuning. Some of these variables change for each collision avoidance approach.

Let us move now to the development and simulation of the collision avoidance trajectory generation problem, with the collision approach described in chapter 5.1 as starting point.

5.5.1 First approach

As already explained before, the implementation of this approach is done by first adding to the lane change constraints the collision avoidance ones shown in equations from (5.26) to (5.29). These constraints represent line boundaries that move with the obstacle and that can be relaxed thanks to the four slack variables ε_{f_1}, ε_{r_1}, ε_{f_2} and ε_{r_2}. One difference between the model described in [11] and this one, is that the slack variables are all thought to be positive, therefore, the collision avoidance constraints are changed accordingly.

$$-\frac{x_1}{L_f}+\frac{x}{L_f}-\frac{y}{W}-\varepsilon_{f_1}+1+\frac{y_1}{W}\leq 0 \tag{5.26}$$

$$\frac{x_1}{L_r}-\frac{x}{L_r}-\frac{y}{W}-\varepsilon_{r_1}+1+\frac{y_1}{W}\leq 0 \tag{5.27}$$

$$-\frac{x_2}{L_f}+\frac{x}{L_f}+\frac{y}{W}-\varepsilon_{f_2}+1-\frac{y_2}{W}\leq 0 \tag{5.28}$$

$$\frac{x_2}{L_r} - \frac{x}{L_r} + \frac{y}{W} - \varepsilon_{r_2} + 1 - \frac{y_2}{W} \leq 0 \qquad (5.29)$$

with

$$L_f = L_r = S_d + L_c \qquad W = \frac{1}{2} \cdot W_L + W_c$$

With L_c and W_c being the length and width of the obstacle, respectively, and S_d and W_L as tuning parameters of the collision avoidance constraints that represent the longitudinal and lateral security distance, respectively. The parameters are represented in chapter 5.1, figures 5.1 and 5.2.

For the whole problem, the obstacle dimensions are set as $L_c = 0.5\ m$ and $W_c = 0.25\ m$.

In order to help the collision avoidance slack variables to reach a non-zero value when it is desirable, constraints from (5.30) to (5.34) are also introduced.

$$-x_1 + x - \varsigma \cdot \varepsilon_{f_1} \leq 0 \qquad (5.30)$$

$$x_1 - x - \varsigma \cdot \varepsilon_{r_1} \leq 0 \qquad (5.31)$$

$$-x_2 + x - \varsigma \cdot \varepsilon_{f_2} \leq 0 \qquad (5.32)$$

$$x_2 - x - \varsigma \cdot \varepsilon_{r_2} \leq 0 \qquad (5.33)$$

$$-\varepsilon_{f_1}, -\varepsilon_{r_1}, -\varepsilon_{f_2}, -\varepsilon_{r_2} \leq 0 \qquad (5.34)$$

where ς is another tuning parameter of the collision avoidance constraints.

Note that these constraints do not forbid the slack variables to have a non-zero value when it is not desirable. Hence, only the collision avoidance slack variables weights on the cost function, shown in equation (5.20), are used to keep a zero value for these slack variables when possible. For this first approach, the slack variable vector and its weighting matrix are defined as follow.

$$\varepsilon = \begin{bmatrix} \varepsilon_{f_1} \\ \varepsilon_{r_1} \\ \varepsilon_{f_2} \\ \varepsilon_{r_2} \end{bmatrix} \qquad R_\epsilon = \begin{bmatrix} \Upsilon_1 & 0 & 0 & 0 \\ 0 & \Phi_1 & 0 & 0 \\ 0 & 0 & \Upsilon_2 & 0 \\ 0 & 0 & 0 & \Phi_2 \end{bmatrix}$$

Now that the starting approach model is fully described, let us move to the simulation and tuning phase, starting with the MPC trajectory generation. The simulations done with this approach can be divided in two different scenarios.

In the first scenario, shown in figure 5.6, there is only one obstacle in the proximity of the controlled vehicle. This obstacle is driving in the first lane, which is also the desired driving lane for the controlled vehicle, and in a lower speed than

it. Therefore, the controlled vehicle is forced to overtake the obstacle in order to maintain its desired reference. To represent this in the two obstacles model used, the second obstacle is situated in a far enough position with the same speed as the controlled vehicle and with its respective slack variables weights set at almost zero to not interfere in the results (weights at zero causes the optimization problem to be semidefinite positive and it cannot be solved with the optimization function used, explained in detail at chapter 6.2).

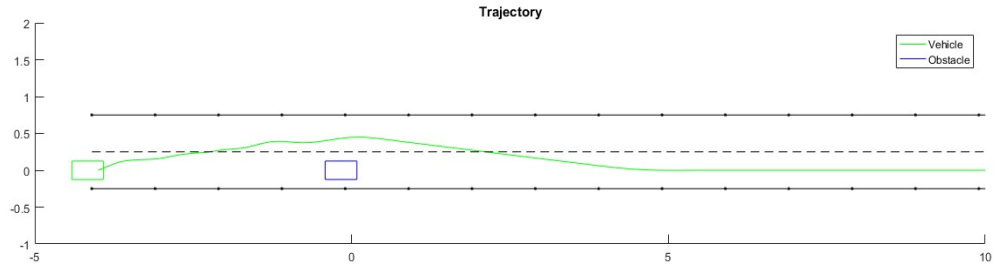


Figure 5.6: Trajectory (lateral vs longitudinal position) of the controlled vehicle in green overtaking an obstacle in blue. All units in SI.

After several simulations, the MPC tuning parameters $Q_v = 100$, $Q_y = 1$, $R_x = 0.5$, $R_y = 0.5$, $\Upsilon_1 = 150$, $\Phi_1 = 150$, $\Upsilon_2 = 0.001$, $\Phi_2 = 0.001$, $N_p = 30$, $N_u = 12$ are considered the ones that fit the best for this scenario. Regarding the collision avoidance constraints parameters, they are set to $S_d = 4\ m$, $W = 0.5\ m$ and $\varsigma = L_f/2 = 2.25\ m$.

Two simulations in this scenario will be studied. The first with the obstacle velocity set to zero (figures from 5.7 to 5.9) and the second one to $0.3\ m/s$ (figures from 5.10 to 5.12). In both simulations, the controlled vehicle velocity reference is $0.8\ m/s$ and the initial longitudinal positions are -4 and 0 for the vehicle and the obstacle, respectively.

As it can be seen in the following figures, in both cases the overtaking manoeuvre is done successfully with the slack variables having an overall desired behaviour. The most interesting thing to notice in these simulations, when having a look at the velocity plots (figures 5.9 and 5.12), it can be seen that while the velocity reference (blue signal) is kept at $0.8\ m/s$, the MPC output (red signal) goes down since the moment that the controlled vehicle is ahead of the obstacle. The deacceleration is bigger for the first case, where the obstacle speed is zero.

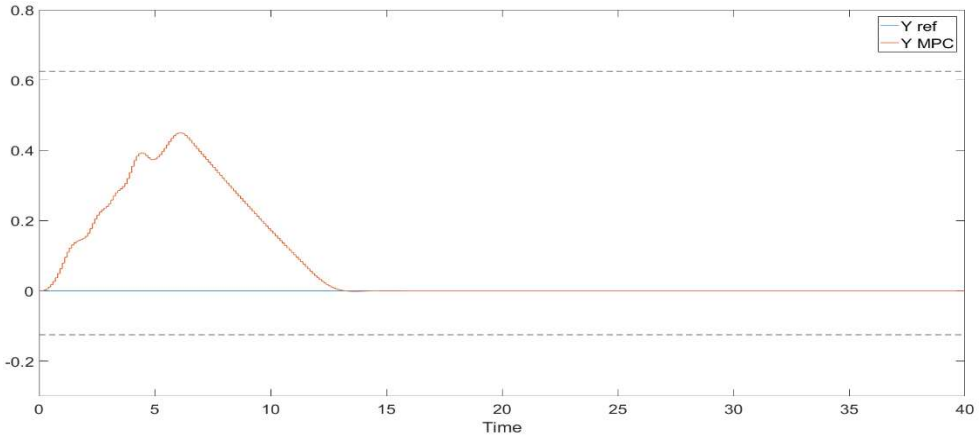


Figure 5.7: MPC output signal of the controlled vehicle lateral position (red) and the reference (blue), when overtaking the obstacle with velocity zero. All units are in SI.

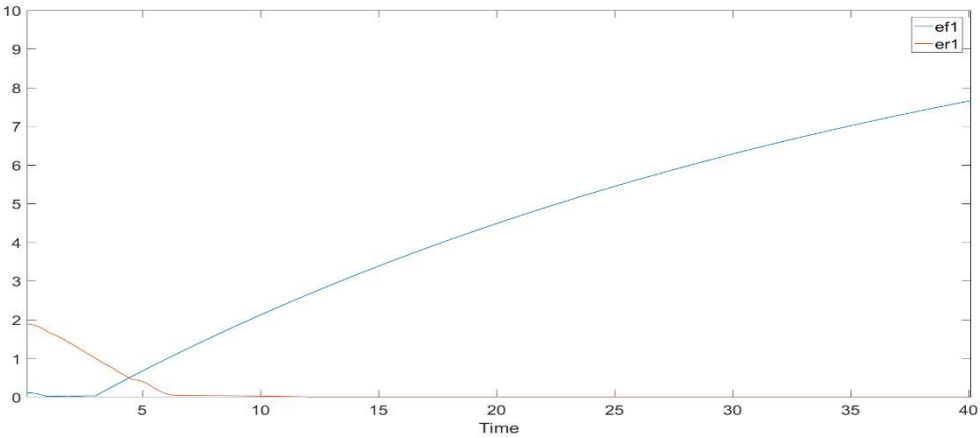


Figure 5.8: Front (blue) and rear (red) collision avoidance constraints slack variables for obstacle 1 with velocity zero.

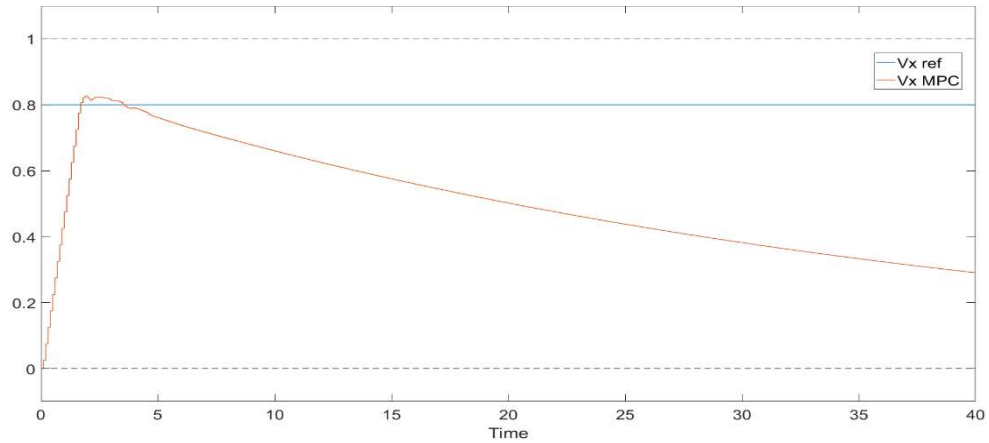


Figure 5.9: MPC output signal of the controlled vehicle longitudinal velocity (red) and the reference (blue), when overtaking the obstacle with velocity zero.

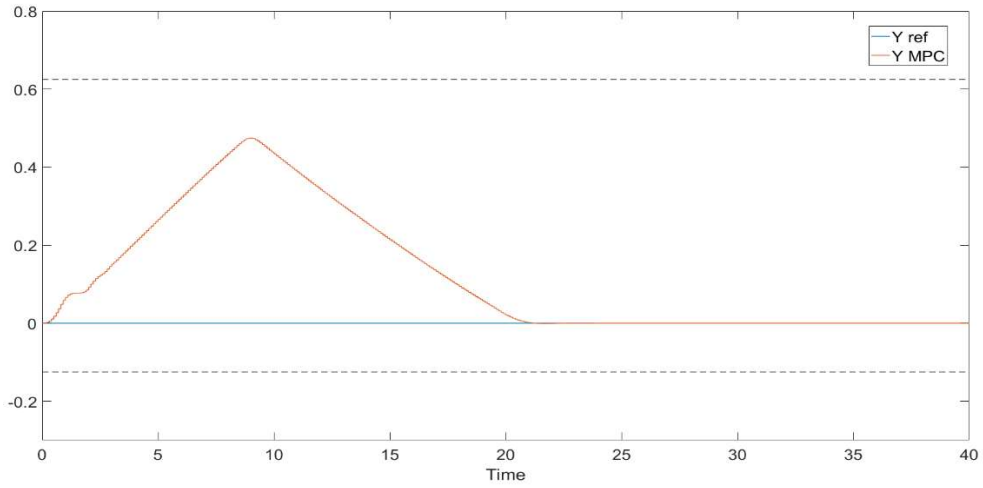


Figure 5.10: MPC output signal over time of the controlled vehicle lateral position (red) and the reference (blue), when overtaking the obstacle with velocity 0.3 m/s.

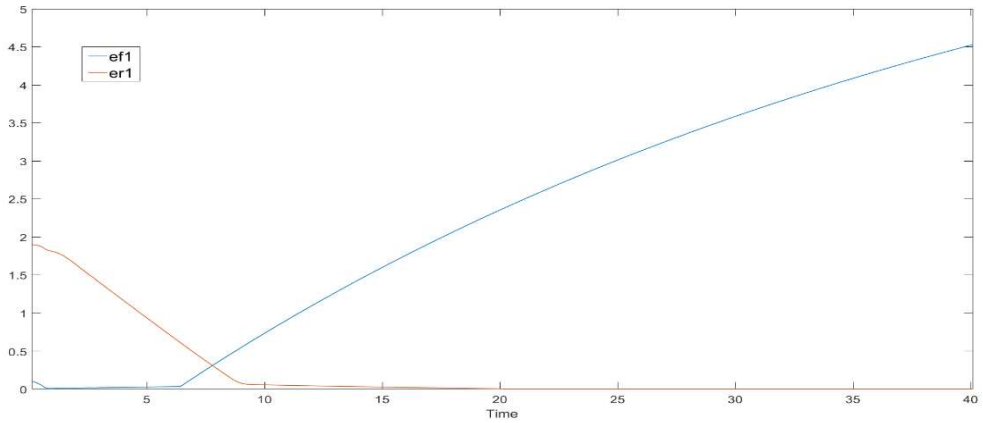


Figure 5.11: Front (blue) and rear (red) collision avoidance slack variables over time for obstacle 1 with velocity 0.3 m/s.

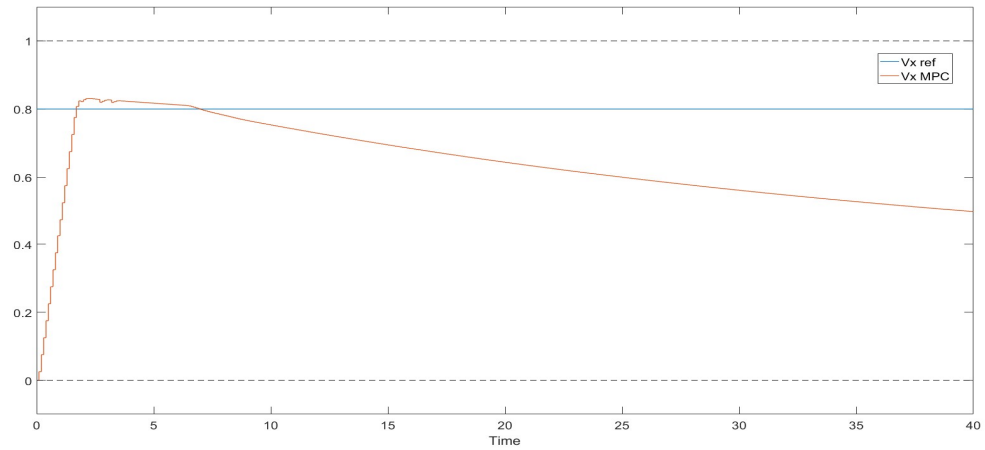


Figure 5.12: MPC output signal of the controlled vehicle longitudinal velocity (red) and the reference (blue), when overtaking the obstacle with velocity 0.3 m/s.

This observation leads to analyse in deep the problem statement to seek for the cause of the issue. For an easier understanding of the explanation of the cause, figure 5.13 is used as visual support. First, it has to be kept in mind that in this approach, the collision avoidance constraints are always active. This means that, as it can be seen in figure 5.13, even when the controlled vehicle has already overtaken the obstacle, the front collision avoidance constraint (FCC) is still active in the formulation but it keeps being displaced to the right thanks to its respective slack variable. Hence, the further the controlled vehicle is from the obstacle the higher the slack variable value is. As the slack variables have their weights in the cost function, the optimizer looks for a balance point, taking into account these weights and the output ones, specifically the longitudinal velocity. Consequently, it can be chosen, through the cost function weights, to give more importance to the longitudinal velocity signal or to respect the collision avoidance, but in the long term, when the vehicle is far from the obstacle, this issue is inevitable with this formulation.

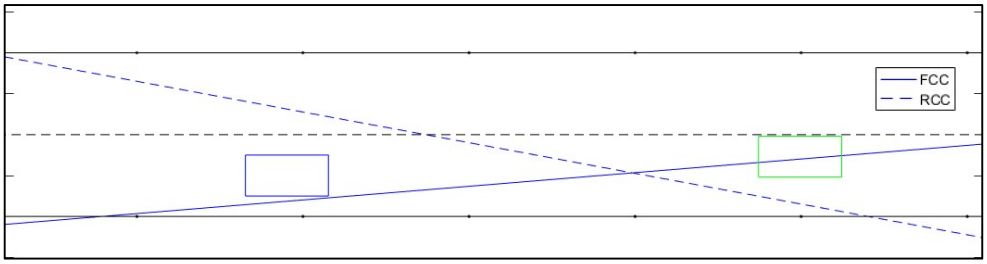


Figure 5.13: Representation of the collision avoidance constraints for obstacle 1 (blue rectangle) after the controlled vehicle (green) has already overtaken it.

Now that it has been shown the behaviour of this approach in presence of one obstacle, let us see what happens when another obstacle is added. In this second scenario, the controlled vehicle desired lane is the first one and its desired speed is $0.7\ m/s$, the obstacle 1 also drives in the first lane with a speed of $0.3\ m/s$ and the obstacle 2 drives in the second lane with a speed of $1.0\ m/s$. The initial longitudinal positions for the vehicle and the obstacles are $x_0 = 0, x_{1_0} = 5$ and $x_{2_0} = -5$, respectively

After several simulations, the MPC tuning parameters $Q_v = 0.5,\ Q_y = 0.01,$ $R_x = 0.5,\ \ R_y = 0.5,\ \ Y_1 = 150,\ \Phi_1 = 1,\ \ Y_2 = 75,\ \ \Phi_2 = 50,\ \ N_p = 30,\ N_u = 12$ are considered the ones that fit the best for this scenario. The collision avoidance constraints parameters remain unchanged.

In the figure 5.14, it can be seen together the trajectories of the vehicle and the obstacles. It has been represented with rectangles three different time instant

positions, to make it easier for the reader to imagine their real-time motion. These time instants correspond to the simulation initial time, $t = 8$ s and $t = 20$ s.

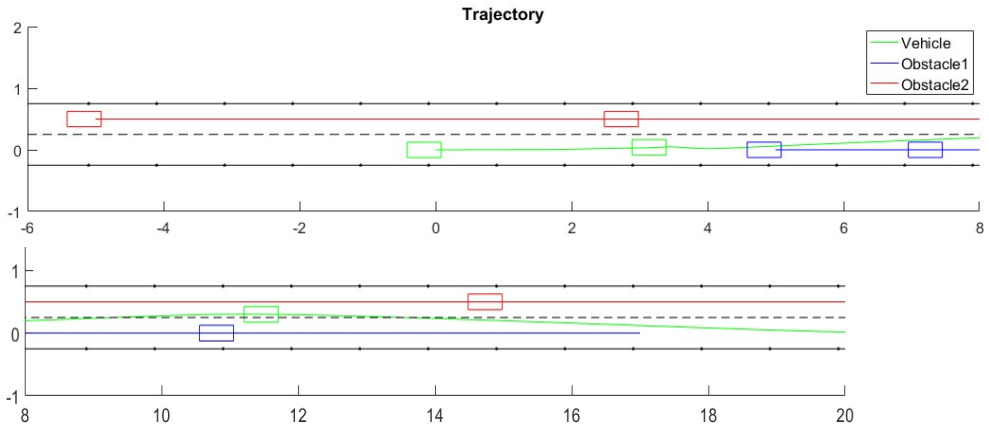


Figure 5.14: Trajectory (lateral vs longitudinal position) of the controlled vehicle (green), the obstacle 1 (blue) and the obstacle 2 (red) for the second scenario. Both axes expressed in metres. All units are in SI.

As it can be seen in figure 5.14 together with figures from 5.15 to 5.17, the collision avoidance manoeuvre has been executed successfully. The controlled vehicle, let the obstacle 2 overtake the obstacle 1 and after, it makes the overtaking manoeuvre. Nonetheless, in the figure 5.15 it can be seen that the overtaking manoeuvre is not done through the middle of the second lane, $y = 0.5\ m$, but at $y = 0.3\ m$ instead. Also, as it is shown in figure 5.17, the vehicle longitudinal velocity stabilizes in an undesired value. This happens again, due to the formulation used in this approach, specifically due to the hidden relation between the vehicle relative speed with the obstacles and the collision avoidance slack variables weights.

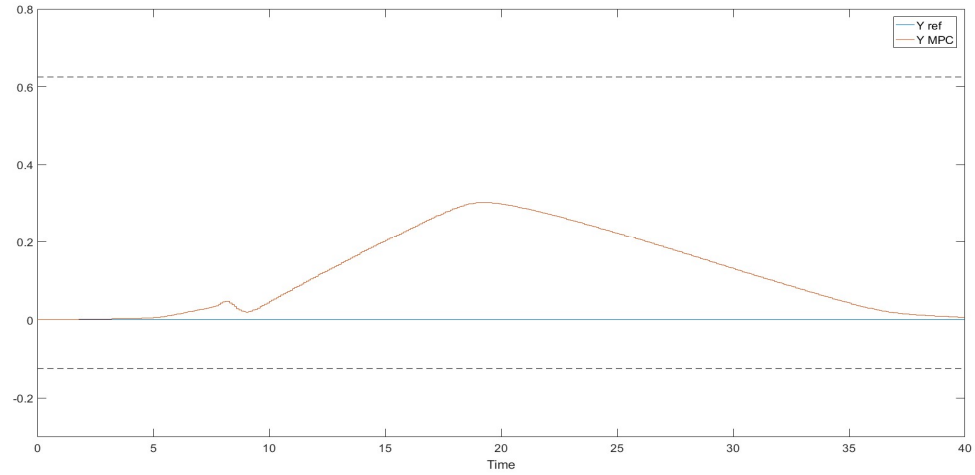


Figure 5.15: MPC output signal over time of the controlled vehicle lateral position (red) together with the reference (blue). All units are in SI.

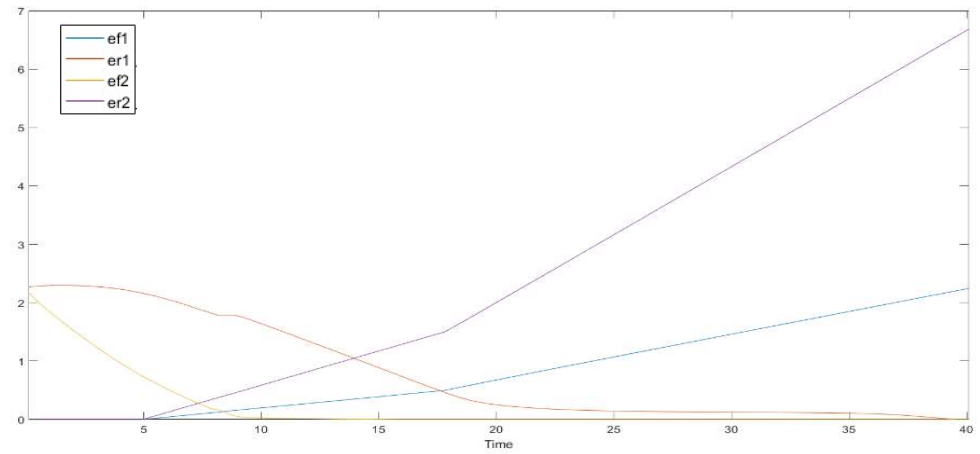


Figure 5.16: Front (blue and yellow) and rear (red and purple) collision avoidance slack variables over time for the obstacles 1 and 2, respectively.

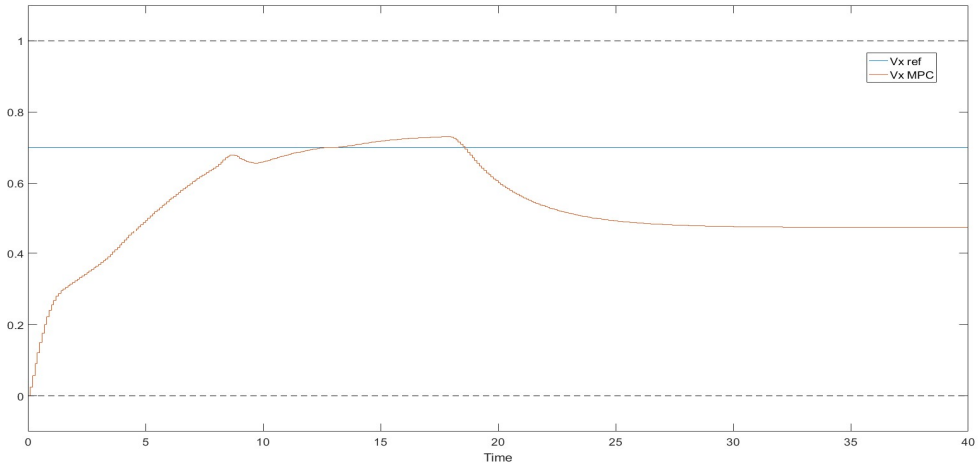


Figure 5.17: MPC output signal over time of the controlled vehicle longitudinal velocity (red) together with the reference (blue). All units are in SI.

In conclusion, it can be said that this approach is strongly tuning dependant, as for each scenario, the weights of the MPC cost function have to be tuned. Also, a small variation on the value of the weights or on the collision avoidance constraints parameters can lead to a completely different result (vehicle stopping and not overtaking, vehicle trying to drive in between the two obstacles, e.g.). In addition to this, the hidden relation between the vehicle relative speed with the obstacles and the collision avoidance slack variables weights, lead to undesirable behaviours that intensify the further the vehicle is from the obstacles and that cannot be solved using this approach.

5.5.2 Second approach

Because of the conclusions presented before, a new approach has been developed from the initial one. In this second approach, an activation and deactivation system has been implemented for each of the collision avoidance constraints. This is done through a binary parameter, represented with the symbol α_j, that only allows one collision avoidance constraint per obstacle j, to be active at the same time. With the addition of this parameter, the collision avoidance constraints from (5.26) to (5.29) are rewritten as the equations shown from (5.35) to (5.38) and constraint (5.34) to the one shown in equation (5.39). Then, the constraints from (5.30) to (5.33) are no longer needed, so they can be deleted from the model.

$$\alpha_1 \cdot \left(-\frac{x_1}{L_f} + \frac{x}{L_f} - \frac{y}{W} - \varepsilon_{f_1} + 1 + \frac{y_1}{W}\right) \leq 0 \tag{5.35}$$

$$(1-\alpha_1) \cdot \left(\frac{x_1}{L_r} - \frac{x}{L_r} - \frac{y}{W} - \varepsilon_{r_1} + 1 + \frac{y_1}{W}\right) \leq 0 \tag{5.36}$$

$$\alpha_2 \cdot \left(-\frac{x_2}{L_f} + \frac{x}{L_f} + \frac{y}{W} - \varepsilon_{f_2} + 1 - \frac{y_2}{W}\right) \leq 0 \tag{5.37}$$

$$(1-\alpha_2) \cdot \left(\frac{x_2}{L_r} - \frac{x}{L_r} + \frac{y}{W} - \varepsilon_{r_2} + 1 - \frac{y_2}{W}\right) \leq 0 \tag{5.38}$$

$$-\alpha_1 \cdot \varepsilon_{f_1}, (\alpha_1 - 1) \cdot \varepsilon_{r_1}, -\alpha_2 \cdot \varepsilon_{f_2}, (\alpha_2 - 1) \cdot \varepsilon_{r_2} \leq 0 \tag{5.39}$$

where all the parameters used, are the same described for the initial approach except the activation and deactivation parameter α_j.

The decision for the activation and deactivation of the FCC/RCC or, in other words, the way the value of the binary parameter α_j is set, is quite simple in this approach. Basically, $\alpha_j = 1$ when the obstacle j is ahead or in the same longitudinal position from the vehicle and $\alpha_j = 0$ otherwise.

Regarding the cost function, it remains unchanged respect the initial approach for ease reasons. Even if only one collision avoidance slack variable can be used per obstacle at same time we still have the fourth in the model, allowing to have different weights for each one in the cost function.

The aim of the modifications done in this approach, is mainly, to eliminate the undesired relation between the vehicle relative speed with the obstacles and the collision avoidance slack variables weights and also, to lower the parameter tuning dependence. Therefore, only one scenario is enough to check if these objectives are fulfilled.

The scenario shown for this approach, is very similar to the second one done for the initial approach and represented in figure 5.14. The vehicle has the same lane and speed references and the obstacles too. This time, the initial longitudinal coordinates for the vehicle and the two obstacles are $x_0 = -1, x_{1_0} = 2$ and $x_{2_0} = -4$, respectively.

After several simulations, it has been seen that having different weighting for each collision avoidance slack variable is not needed and, with this way, it can be assured less tuning parameter dependence in the problem. With that in mind, the MPC tuning parameters $Q_y = 1$, $Q_v = 40$, $R_x = R_y = 1$, $\Upsilon_1 = \Phi_1 = \Upsilon_2 = \Phi_2 = 5 \cdot 10^8$, $N_p = 30$ and $N_u = 12$ are considered the ones that fit the best. Regarding the collision avoidance constraints parameters, now, the longitudinal safety distance value is different for the FCC and RCC and they are set to $S_{df} = 1.5\ m$ and $S_{df} = 2\ m$. This allows a better control in the decision making of the vehicle but also, inserts certain tuning dependence in the model. The lateral safety distance remains unchanged, $W = 0.5\ m$.

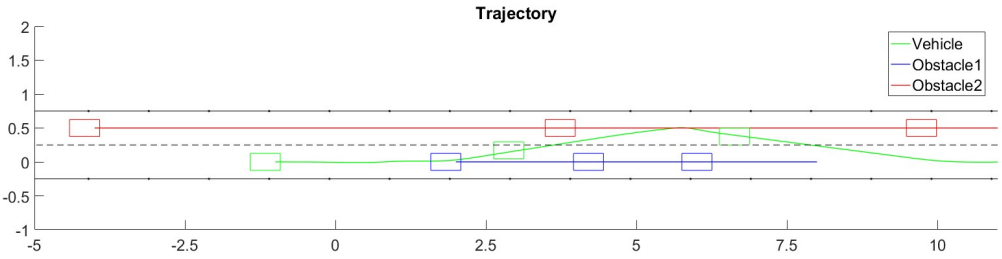


Figure 5.18: Trajectory (lateral vs longitudinal position) of the controlled vehicle (green), the obstacle 1 (blue) and the obstacle 2 (red) for the second approach simulation. Both axes expressed in metres. All units are in SI.

In the figure 5.18, it can be seen together the trajectories of the vehicle and the obstacles, as it has also shown in the second scenario of the initial approach. It has been represented with rectangles three different time instant positions again. Now, these time instants correspond to the simulation initial time, $t = 8$ s and $t = 14$ s.

As it has been seen in figure 5.18 and can also be seen in figures from 5.19 to 5.21, the collision avoidance manoeuvre has been executed successfully. This time, as it can be seen in the velocity plot shown in figure 5.21, the controlled vehicle slow down to let the obstacle 2 overtake the obstacle 1 without collisions and then, it executes its overtaking manoeuvre. In the lateral position plot shown in figure 5.19, it can be seen that now, the vehicle reaches the middle of the left lane, $y = 0.5\ m$, when overtaking.

When having a look at the collision avoidance slack variables in figure 5.20, it is easy to see that now, they only act on punctual step times, when there is no other option, as in this approach the collision avoidance constraints are "softer" than the rest. In this case they act when the obstacle 2, which is driving in the left lane, overtakes the vehicle. Anyway, as it can be seen in figure 5.18, no collision takes place in that exact instant or on any other.

Two last observations can be made when having a look at the velocity plot shown in figure 5.21. First, the output signal stabilizes in the desired velocity, which indicates that the hidden relations from the initial approach have been solved. Finally, it can be seen that an undesired deacceleration appears from $t = 10$ s to, $t = 13$ s, approximately. This event happens because the activation and deactivation of the collision avoidance constraints is decided considering only the actual state and not over the whole prediction horizon. This means that the control cannot predict when the activation and deactivation parameter will change. Hence, it slows down when approximating the intersection between the FCC and the upper road limit constraint and it speeds up in the instant when the FCC is deactivated and the RCC is activated.

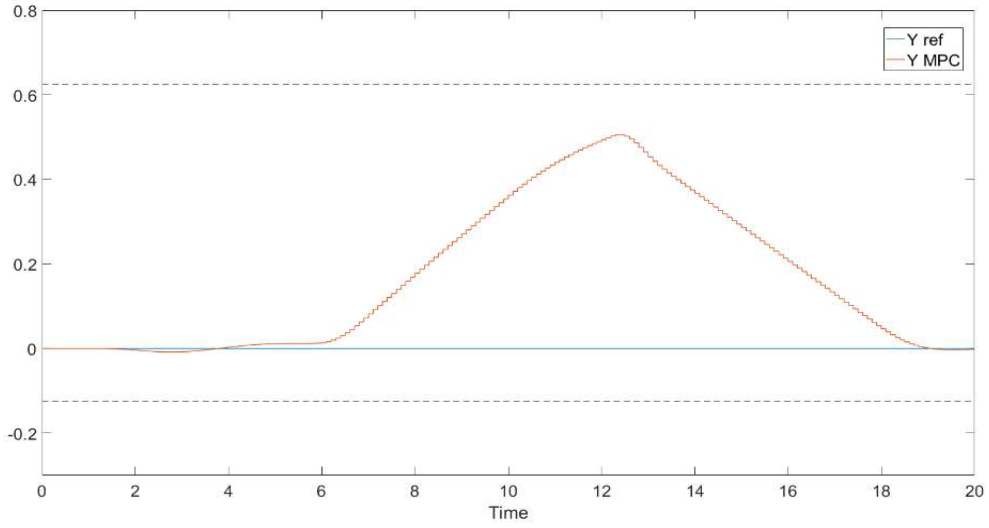


Figure 5.19: MPC output signal over time of the controlled vehicle lateral position (red) together with the reference (blue). All units are in SI.

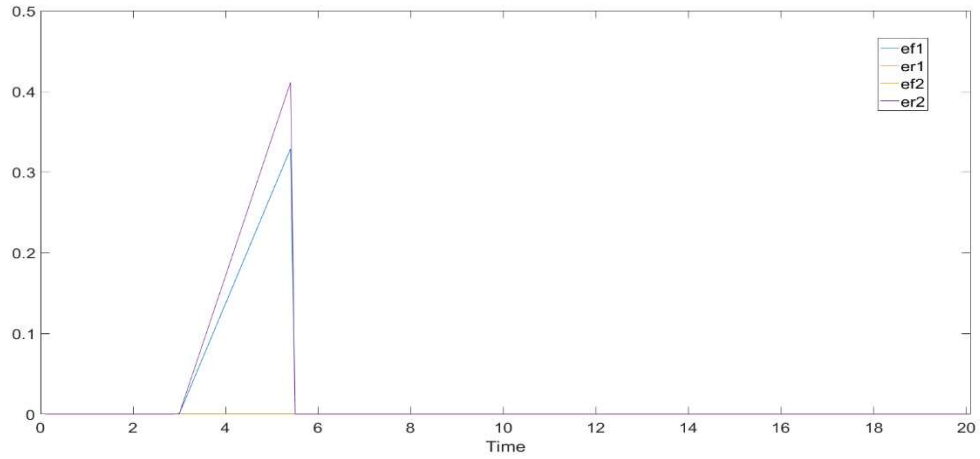


Figure 5.20: Front (blue and yellow) and rear (red and purple) collision avoidance slack variables for the obstacles 1 and 2, respectively.

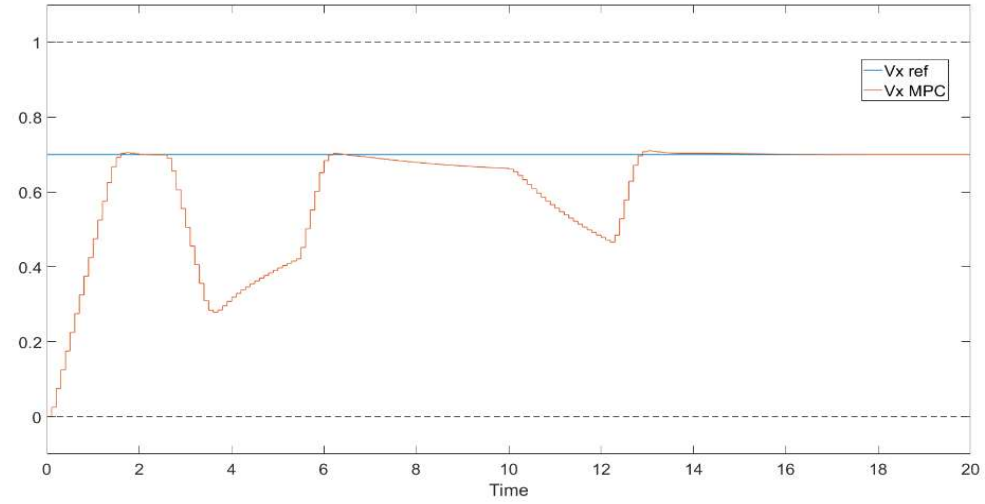


Figure 5.21: MPC output signal over time of the controlled vehicle longitudinal velocity (red) together with the reference (blue). All units are in SI.

In conclusion, it can be said that the issues that were aimed to be solved with this second approach have disappear, but new ones have come out and there is still room for improvement. The most critical issue to be solved is the undesirable deacceleration that appears just before the activation/deactivation of the collision avoidance constraints when overtaking, due to the reasons explained before. Another improvement that can be made, is regarding the overtaking manoeuvre trajectory. As it can be seen in figure 5.19, the vehicle stays in the left lane a brief period of time, just a couple of seconds, describing a triangular trajectory. This is not an overtaking trajectory characteristic of a human driving in a highway. It is thought that a trapezoidal trajectory would be more desirable for highway overtaking manoeuvres, therefore, this will also be aimed to achieve with the next approach.

5.5.3 Third approach

First, to achieve the desired trapezoidal trajectory when overtaking, a new collision avoidance constraint per obstacle has been added to the model. This constraint, which is named lateral collision avoidance constraint (LCC), represents the lateral boundary when the vehicle is in the opposite lane of the obstacle that is overtaking, as it is shown in the right picture of figure 5.22.

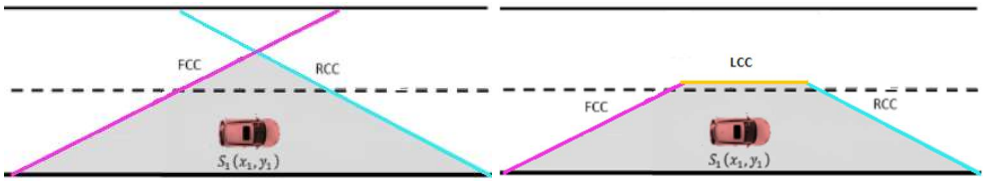


Figure 5.22: In the left, it can be seen the representation of the CC used in the first and second approach. In the right side, it is represented the new CC formulation, which adds the LCC shown in colour yellow. Modified from [11].

With the addition of the LCC to each collision avoidance obstacle formulation, the activation and deactivation parameter used in the previous approach α_j, is rewritten now as α_{ji}, where the subscript j still defines the obstacle and the subscript i, is used to identify the CC (1 for the FCC, 2 for the LCC and 3 for the RCC). This leads to the CC described in equations from (5.40) to (5.45)

$$\alpha_{11} \cdot \left(-\frac{x_1}{L_f} + \frac{x}{L_f} - \frac{y}{W_f} + 1 + \frac{y_1}{W_f} \right) \leq 0 \tag{5.40}$$

$$\alpha_{12} \cdot (-y + W + y_1) \leq 0 \tag{5.41}$$

$$\alpha_{13} \cdot \left(\frac{x_1}{L_r} - \frac{x}{L_r} - \frac{y}{W_r} + 1 + \frac{y_1}{W_r} \right) \leq 0 \tag{5.42}$$

$$\alpha_{21} \cdot \left(-\frac{x_2}{L_f} + \frac{x}{L_f} + \frac{y}{W_f} + 1 - \frac{y_2}{W_f} \right) \leq 0 \tag{5.43}$$

$$\alpha_{22} \cdot (y + W - y_2) \leq 0 \tag{5.44}$$

$$\alpha_{23} \cdot \left(\frac{x_2}{L_r} - \frac{x}{L_r} + \frac{y}{W_r} + 1 - \frac{y_2}{W_r} \right) \leq 0 \tag{5.45}$$

Notice that now, different lateral safety distances are introduced for the FCC, LCC and RCC. These new safety distance parameters are defined as follows:

$$W_f = L_f \cdot \frac{W}{(L_f - L)} \qquad W_r = L_r \cdot \frac{W}{(L_r - L)}$$

where W is now the lateral safety distance for the LCC, which basically indicates the lateral position where the horizontal boundary that defines the LCC is situated. Then, reminding that x_j is the longitudinal coordinate of the obstacle j, $(x_j - L)$ and $(x_j + L)$ are the longitudinal positions where the LCC intersects with the FCC and the RCC, respectively. This parameter L, is key for the decision making on the activation and deactivation of the CC, as it is explained below.

Now, with this new approach, there are six binary parameters used for the activation and deactivation of the CC, but only one per obstacle should have a non-zero value at the same time. The decision making in this approach is done as follows: $\alpha_{j2} = 1$ when the longitudinal distance between the obstacle j and the vehicle is smaller than L (are included in this condition the limit points $x = x_1 + L$ and $x = x_2 - L$ for $\alpha_{12} = 1$ and $\alpha_{22} = 1$, respectively). Then, if the vehicle is further than a distance L from the obstacle 1, $\alpha_{13} = 1$ and also, if the obstacle 2 is further than a distance L from the vehicle, $\alpha_{21} = 1$. Otherwise, $\alpha_{11} = 1$ and $\alpha_{23} = 1$.

Additionally, it is reminded that one of the objectives of this approach, is to make the deacceleration happening in the previous approach in the moment when the vehicle overtakes, disappear. This issue is aimed to be solved doing the activation and deactivation of the CC over the whole prediction horizon. This is done by first, calculating the positions $x(k)$, $x_1(k)$ and $x_2(k)$ for the horizon prediction ($k = 1, \dots, N_p$) through the plant model and then, taking into account the longitudinal positions obtained for each time step k of the prediction horizon, decide the value of the $\alpha_{ij}(k)$ binary variables, obtaining a 6 x N_p binary variables matrix.

Another interesting addition implemented in this approach is the MPC reference change. Thanks to including the LCC to the model together with its activation/deactivation parameter α_{12}, the MPC reference can be computed depending on this parameter. This allows to change the lateral position reference to the left lane or increase the vehicle speed when the LCC_1 is active (only when overtaking obstacle 1 through the left lane). In the end, only the lateral position reference change is used in the simulations and tests of this approach due to the low-level velocity control not being accurate enough to make the small velocity changes transcendent.

Regarding the cost function, as in the previous approach it was seen that there is no need to have different weightings for each slack variable of the CC, they have been eliminated from the model and the global slack variable ϵ, is used in their place. Now, the CC constraints will also use the weighting vector τ, to indicate their level of hardness compared with the other constraints. This vector is defined as follows: 0.5 for the acceleration constraints (u), 0 for the output constraints (hard constraints),

1 for the variation of the acceleration constraints (Δu), 0.25 for the state constraints (specifically constraint defined in equation (5.22)) and 0.001 for the CC. All this can be written as shown in equation (5.46).

$$\Omega \cdot \Delta\widehat{U}(k) \leq \omega + \tau\epsilon \tag{5.46}$$

with

$$\Omega = \begin{bmatrix} \mathcal{F} \\ G \cdot \Theta \\ E \\ J \cdot A_{ba} \end{bmatrix}, \qquad \omega = \begin{bmatrix} -\mathcal{F}_1 \cdot u_{k-1} - f \\ -G \cdot [\Psi \cdot X_k + \Upsilon \cdot u_{k-1}] - g \\ e \\ -J \cdot [A_a \cdot X_k + A_b \cdot u_{k-1}] - j \end{bmatrix}$$

where all the parameters have already been defined in previously.

As this approach was showing satisfactory results, it has proceeded to do the simulations with the overall control. When the collision avoidance trajectory generation problem was being developed and proved in a simulation environment, the lane change approach was already finished. That means that the testing for the lane change, explained in a later chapter, specifically chapter 6.3, was already carried out successfully. With that in mind, the low-level control parameters used for this approach, are the ones obtained in the definitive testing tuning of the lane change approach. The reason behind that, is that the hardware and environment conditions are the same, hence, the definitive tuning parameters for this approach should not differ too much from the lane change one and no excessive time wants to be spend tuning the low-level control parameters in simulation, as they will have to be tuned again in the testing phase. Therefore, the low-level control variables, which have been described previously in chapter 4.3, take now the following values: K_p = 1.2, Kstan = 0.5 and Ktheta = 0.

For this approach, multiple scenarios have been posed, so they can be tested later in the truck lab in case the simulations are carried out successfully. A total of five scenarios have been prepared.

In the first two scenarios, both obstacles remain static, $v_j = 0$. In the first one, the obstacle 2 is situated before the obstacle 1 and in the second scenario, the obstacle 2 is situated ahead of the obstacle 1.

The third scenario, consists on only one obstacle, driving in the right lane with a speed of $0.3\ m/s$.

In the fourth and fifth scenarios, there are again both obstacles active but now driving with a certain speed. The speeds of the obstacles will be changed so that in the fourth scenario, the vehicle slows down to let the obstacle 2 overtake the obstacle

1 first and in the fifth one, the vehicle speeds up to overtake the obstacle 1 before the obstacle 2.

Regarding the MPC tuning parameters, it has been tried to find a global set of values that lead to the best results for this approach in all the posed scenarios. That means that the values of the tuning parameters will not be changed in each scenario if it is not strictly needed, as this is thought to add more autonomy to the approach.

After several simulations, the MPC tuning parameters $Q_y = 1,\ Q_v = 40,\ R_x = R_y = 1,\ \rho = 5 \cdot 10^8,\ N_p = 30,\ N_u = 12$ are considered the ones that fit the best for most of the scenarios. Regarding the collision avoidance constraints parameters, they are set to $S_{df} = 1.5\ m$, $S_{dr} = 1\ m$, $W = 0.4\ m$ and $L = 0.7\ m$.

Now that everything is described, let us move to the first simulation scenario. The positions and safety distances have been adapted to fit in the truck lab dimensions. The exact initial longitudinal coordinates for the vehicle and the obstacles are $x_0 = -4,\ x_{1_0} = 1$ and $x_{2_0} = -2$, and the desired longitudinal velocity for the controlled vehicle is set to $v_x = 0.8\ m/s$.

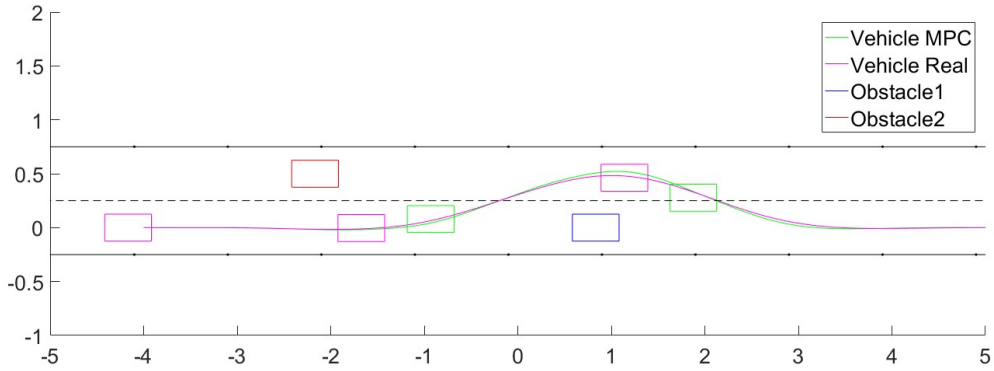


Figure 5.23: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and by the kinematic model (magenta), together with the obstacles 1 (blue) and 2 (red). All units are in SI.

In the figure 5.23, it is shown the trajectory of the vehicle and the static obstacles position. For this approach, as we are using the overall control, it can also be plotted the vehicle trajectory generated by the kinematic model, which represents the trajectory that the vehicle would truly take. It has also been represented with rectangles three different time instant position. This allows to see that the trajectory generated by the MPC is always ahead of the true trajectory and consequently, it can be followed successfully. These time instants correspond to the simulation initial time, $t = 5$ s and $t = 8.5$ s.

As it has been seen in figure 5.23 and can also be seen in figures from 5.24 to 5.29, the collision avoidance manoeuvre has been executed successfully. This time, as it can be seen in the velocity plot shown in figure 5.24, there are three longitudinal velocity signals represented. The new signal, represented in colour yellow, corresponds to the one generated by the kinematic model. The interesting thing to notice for the longitudinal velocity plot, is that the deacceleration happening when overtaking has been eliminated, giving way to a constant velocity signal. The slower acceleration in the true signal (yellow) compared with the MPC one, is due to the reaction time of the vehicle, and it helps to keep the MPC trajectory, which is used as reference for the low-level control, ahead.

In the lateral position plot, shown in figure 5.25, it has also been added the signal generated by the kinematic model. There, it can also be noticed the lateral position reference change, happening when the LCC is activated. The last interesting thing to comment about the lateral position plot, is that in this scenario the trajectory obtained is smother than the triangular one from the other approach, but it is not a fully trapezoidal trajectory neither. This is due to the limited size that the truck lab has for straight line testing and the intentionality of applying it in the simulation scenarios, for the reasons explained before in this chapter. If the simulation was carried in a longer road, it would be enough to increase the CC parameter *L*, to make the vehicle stay longer in the left lane and obtaining this way, a more trapezoidal trajectory (provided that the safety distance with the obstacles allows it).

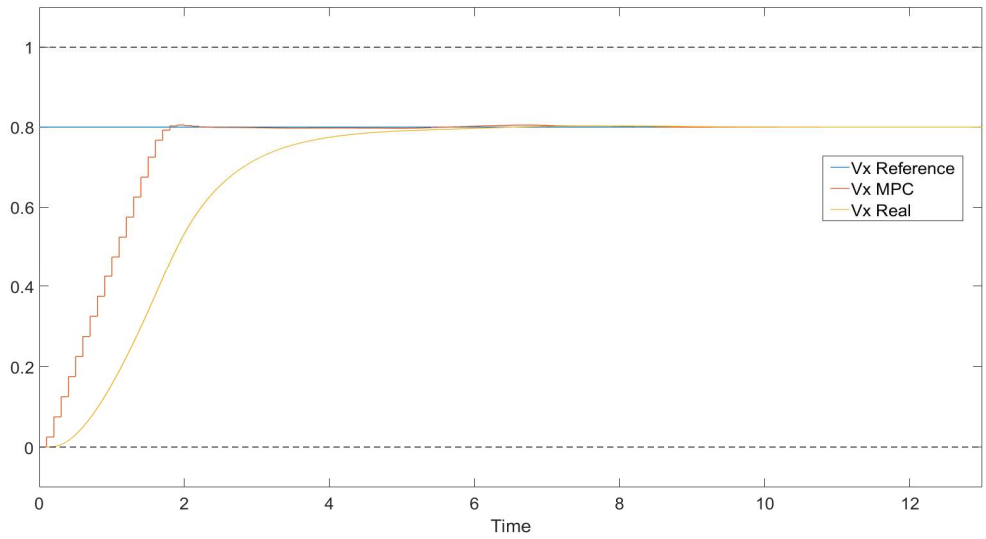


Figure 5.24: Plot over time of the vehicle longitudinal velocity signals. Desired velocity in blue, generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

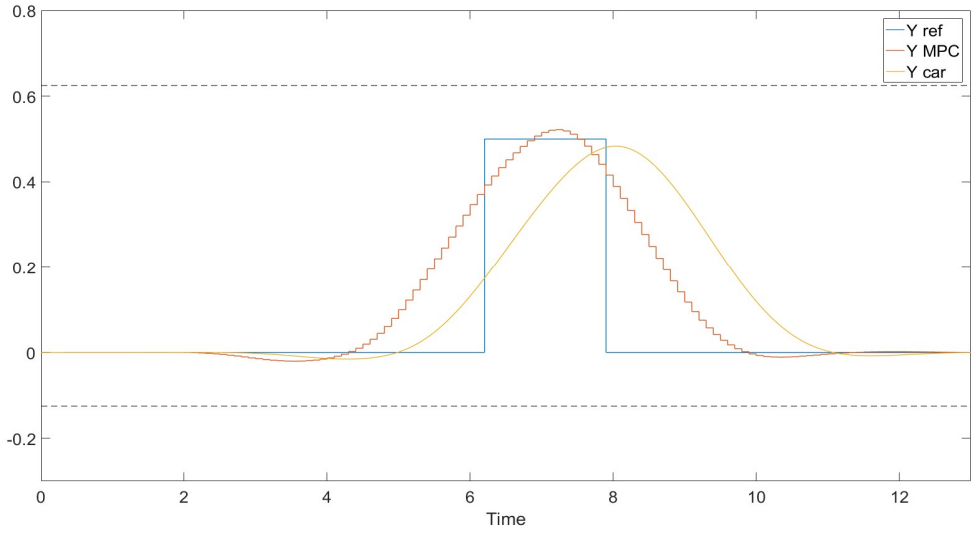


Figure 5.25: Plot over time of the vehicle lateral position signals. The desired velocity in blue, the generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

Figure 5.26 shows the vehicle steering angle signal generated by the low-level control. It can be seen there, that the signal is far from the vehicle capabilities of ± 0.3 radians. This happens manly due to the restriction added for the collision avoidance trajectory generation approach, defined in equation (5.22), which does not allow orientation angles outside the ± 0.3 radians limits for the vehicle. The compliance of this constraint, can be seen in figure 5.27. Figures 5.28 and 5.29 also show constraints compliance, this time for the accelerations and its variation, respectively.

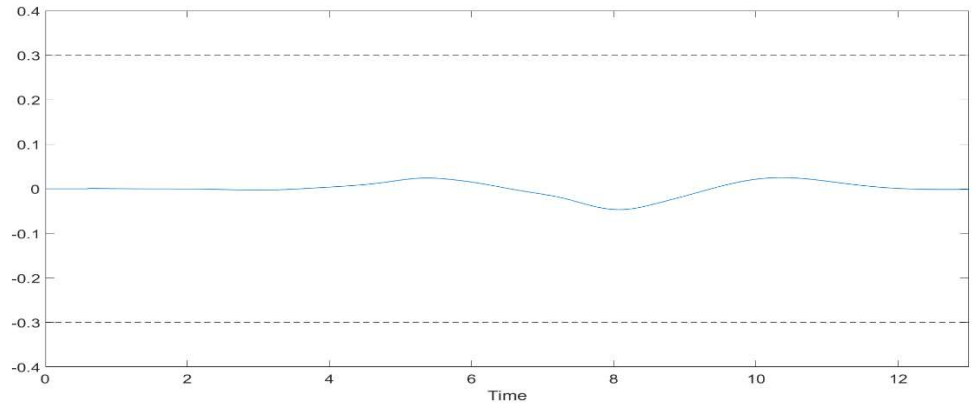


Figure 5.26: Steering angle over time of the vehicle calculated in the low-level control in radians.

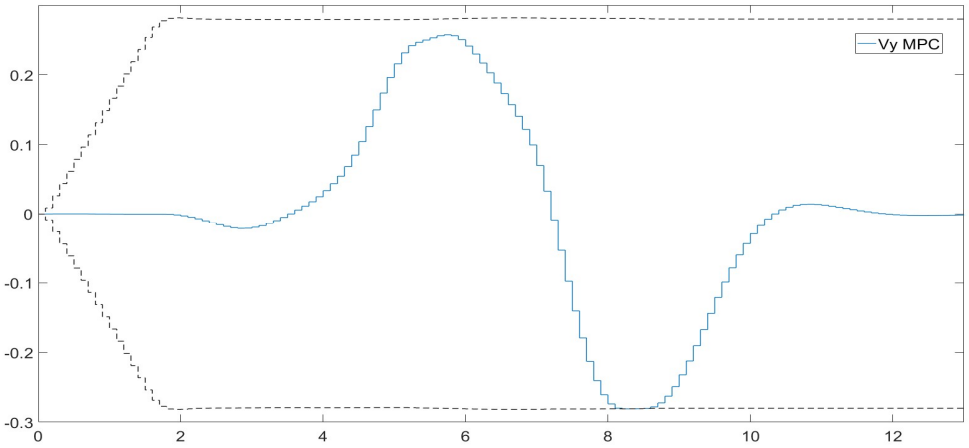


Figure 5.27: Vehicle lateral velocity signal over time generated by the MPC. All units are in SI.

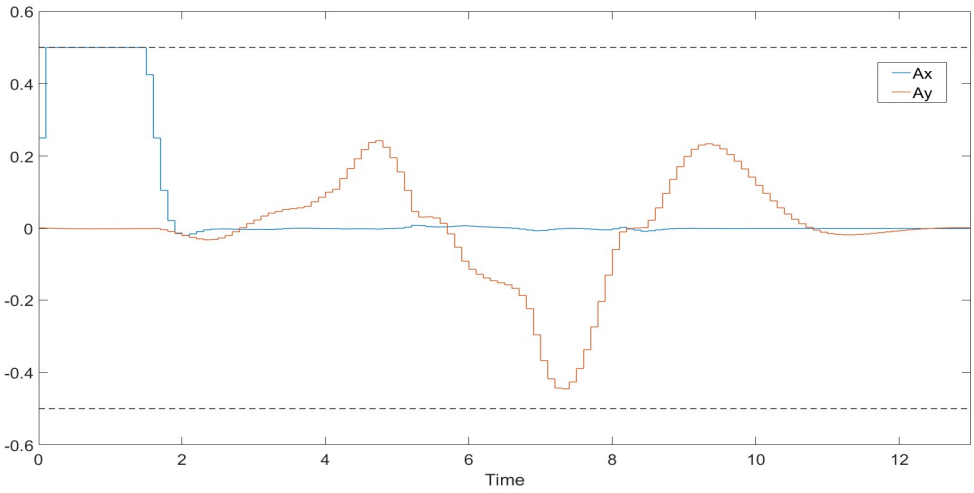


Figure 5.28: Vehicle acceleration signals generated by the MPC. All units are in SI.

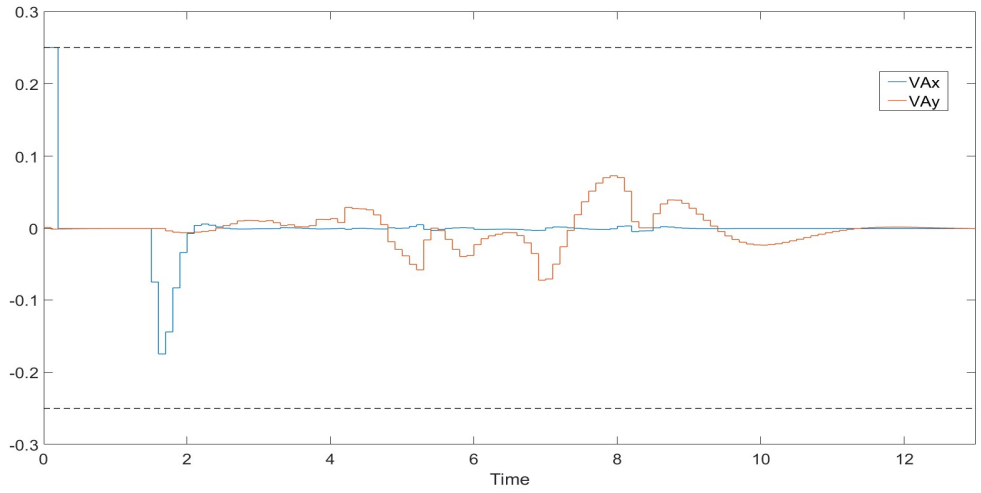


Figure 5.29: Vehicle acceleration variation signals generated by the MPC. All units are in SI.

Let us move now to the second scenario, which is quite similar to the first one, just been analysed. The only parameters that change from the previous scenario are the obstacles longitudinal positions, which now are set to $x_{1_0} = -1$ and $x_{2_0} = 2$.

As it can be seen in figure 5.30, now the vehicle overtakes first the static obstacle situated in the right lane and then moves back to the right lane before colliding with the obstacle situated in the left lane. As said in the previous approach, the trajectory plot, can also show the true trajectory generated by the kinematic model. This will also happen for the rest of scenarios presented in this approach. The three rectangle sets representing the trucks correspond to the same time instants used for the first scenario, which are the simulation initial time, $t = 5$ s and $t = 8.5$ s.

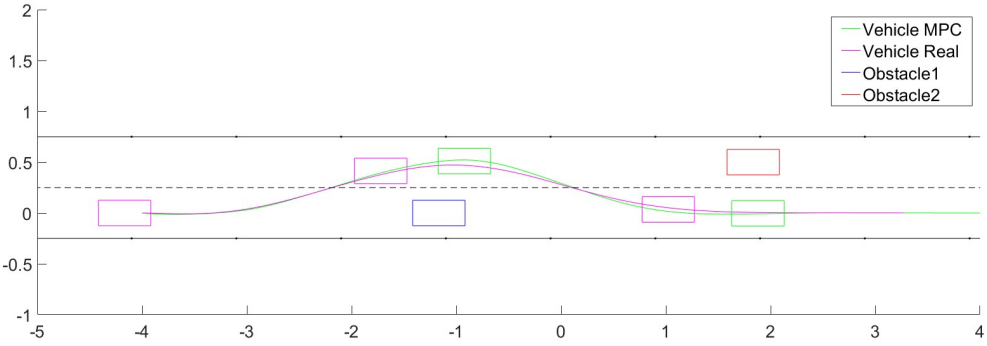


Figure 5.30: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and by the kinematic model (magenta), together with the obstacles 1 (blue) and 2 (red). All units are in SI.

Figures 5.31 and 5.32, which show the plots of the vehicle longitudinal velocity signals and the lateral position signals, respectively, reaffirm the successful resolve

of the second scenario posed for this approach. This time, all the plots showing the compliance of the constraints are not presented, although, all of them have been fulfilled. This is done this way, aiming to make the chapter more pleasant for the reader.

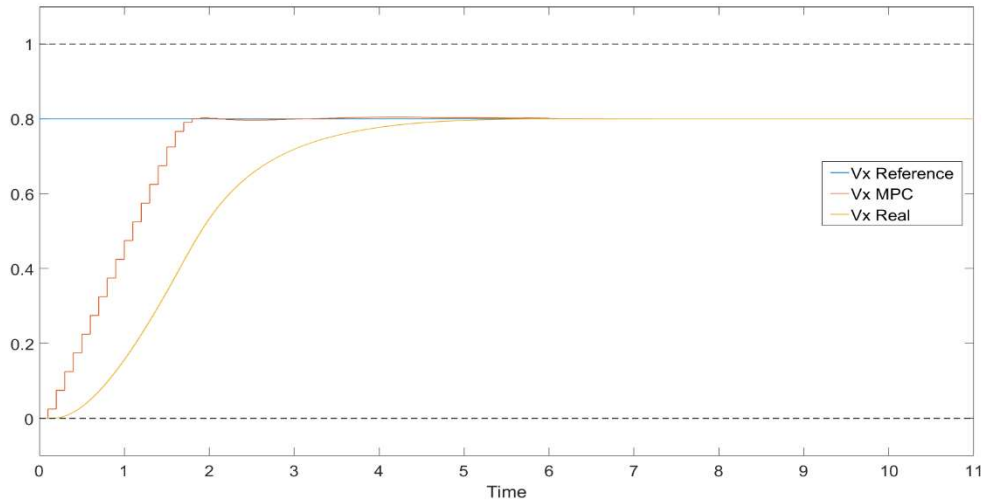


Figure 5.31: Plot over time of the vehicle longitudinal velocity signals. Desired velocity in blue, generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

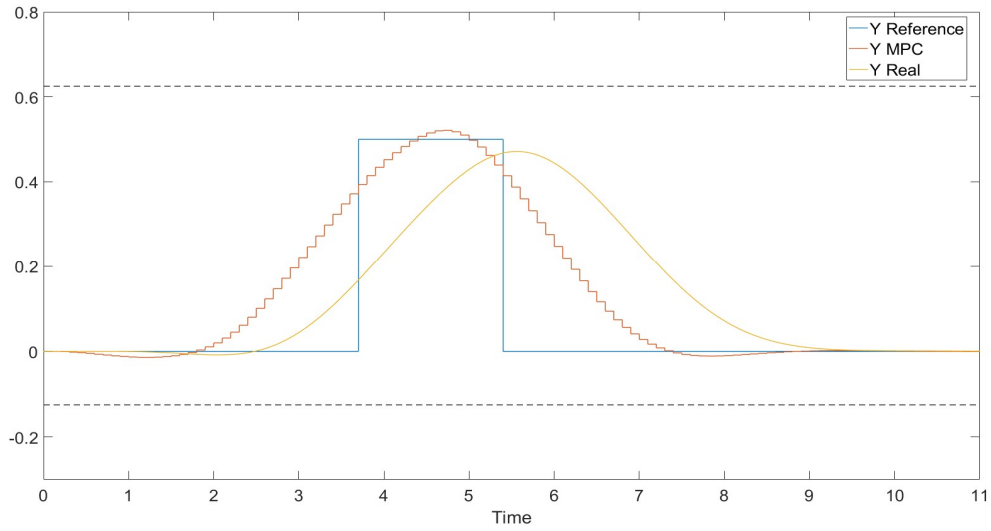


Figure 5.32: Plot over time of the vehicle lateral position signals. The desired velocity in blue, the generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

The third scenario to analyse, consists of only one obstacle driving in the right lane with a speed of $v_1 = 0.3\ m/s$ and initial position of $x_{1_0} = -1.5\ m$. The controlled vehicle parameters remain unchanged.

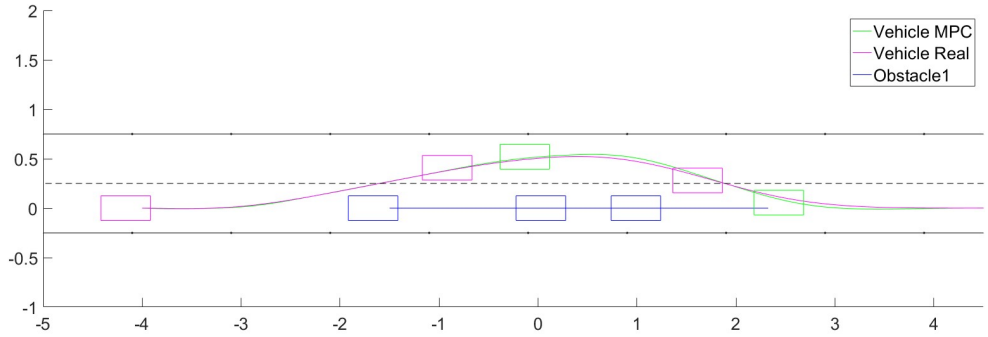


Figure 5.33: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and by the kinematic model (magenta), together with the obstacle 1 (blue). All units are in SI.

As it can be seen in figure 5.33, now the vehicle is forced to do longer overtaking manoeuvre, as the obstacle situated in the right lane is also moving. As in each scenario, it has been represented with rectangles three different time instant position. These time instants correspond to the simulation initial time, $t = 6$ s and $t = 9.2$ s. The instant time $t = 9.2$ s is chosen to show that if the time gap between the trajectory generated in the MPC and the true trajectory of the vehicle is too high it could generate a collision without violating any constraint. To avoid that, the low-level control parameters are used. In the tests, the boot acceleration of the obstacle is quite lower than the used in the simulation. For these reason, the event just described is not really worrying and the low-level control tuning will be studied in the testing phase.

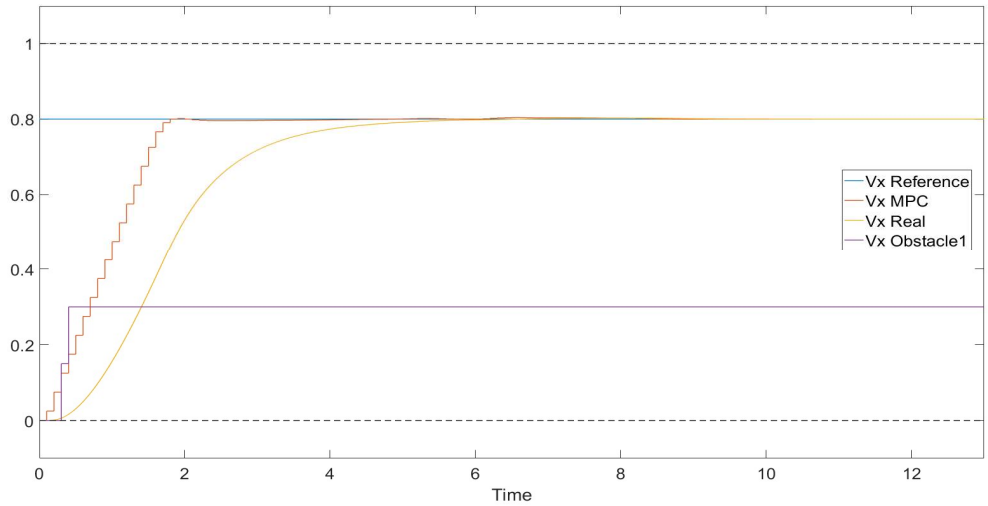


Figure 5.34: Plot over time of the longitudinal velocity signals for the vehicle and obstacle 1 (purple). Vehicle desired velocity in blue, generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

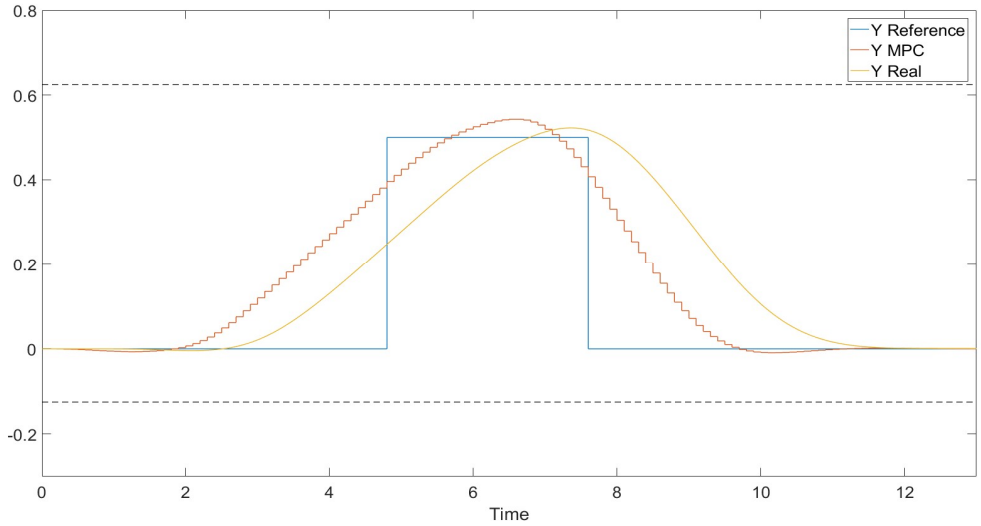


Figure 5.35: Plot over time of the vehicle lateral position signals. The desired velocity in blue, the generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

Figures 5.34 and 5.35, which show the plots of the vehicle longitudinal velocity signals and the lateral position signals, respectively, reaffirm the successful resolve of the second scenario posed for this approach. This time, in the longitudinal velocity plot it can also be seen the speed of obstacle 1.

As already said for the previous scenario, all the plots showing the compliance of the constraints are not presented this time neither, although, all of them have been fulfilled. This will be done this way for the rest of the simulation scenarios described in this approach if the constraints are not violated.

The fourth scenario to analyse, consists in one obstacle driving in the left lane with a speed of $v_2 = 0.7\ m/s$, while another obstacle stands still in the right lane. The initial positions of the vehicle and the obstacles are set to $x_0 = -2$, $x_{1_0} = 1$ and $x_{2_0} = -4.5$. The controlled vehicle longitudinal speed is set to $v_x = 0.6\ m/s$.

As it can be seen in figure 5.36, the controlled vehicle overtakes the obstacle 1 situated in the right lane before the obstacle 2. As in each scenario, it has been represented with rectangles three different time instant position. These time instants correspond to the simulation initial time, $t = 5$ s and $t = 9$ s.

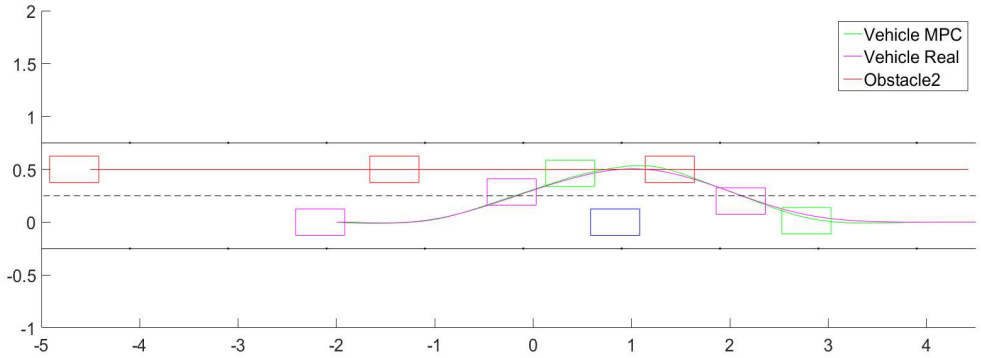


Figure 5.36: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and by the kinematic model (magenta), together with the obstacle 2 (red). All units are in SI.

Figures 5.37 and 5.38, which show the plots of the vehicle longitudinal velocity signals and the lateral position signals, respectively, reaffirm the successful resolve of the second scenario posed for this approach. This time, in the longitudinal velocity plot it can also be seen the speed of obstacle 2.

Last thing to mention, is that all the constraints have been respected in this scenario.

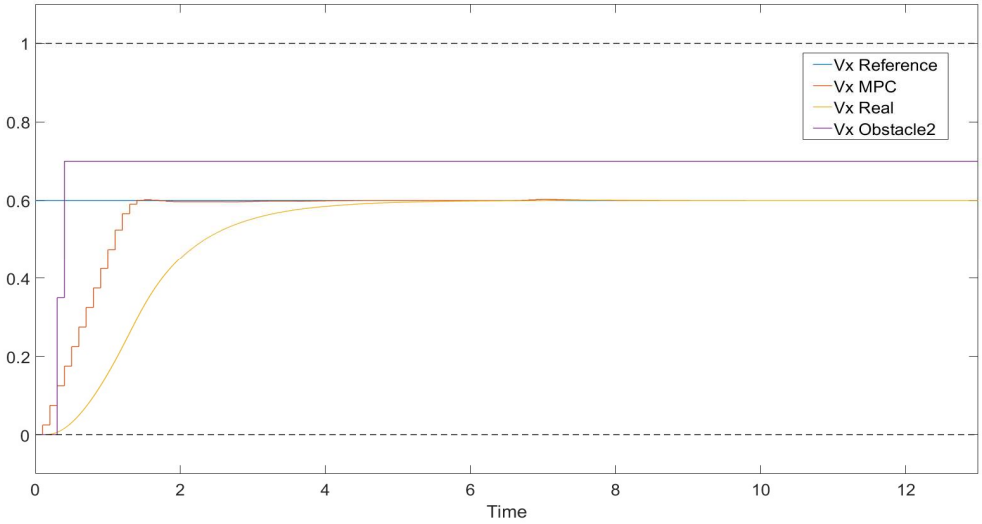


Figure 5.37: Plot over time of the longitudinal velocity signals for the vehicle and obstacle 2 (purple). Vehicle desired velocity in blue, generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

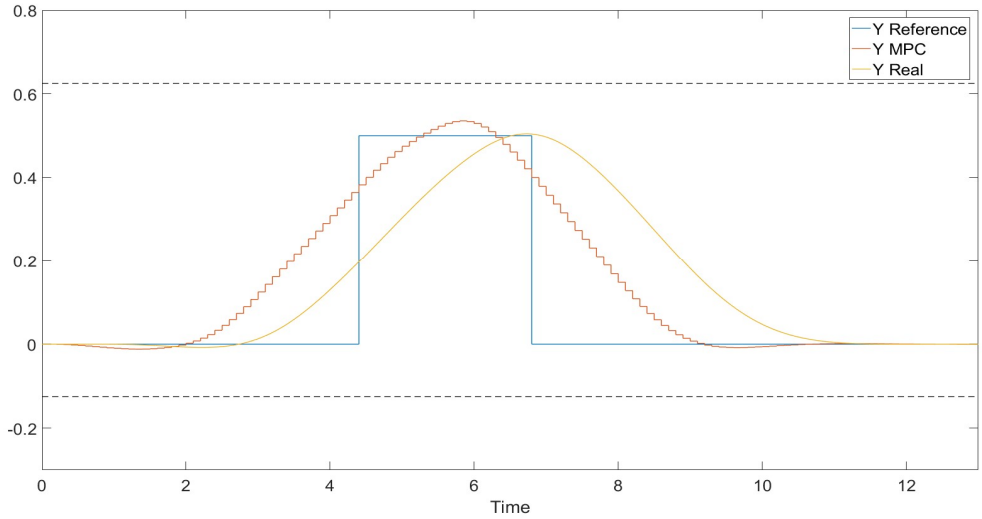
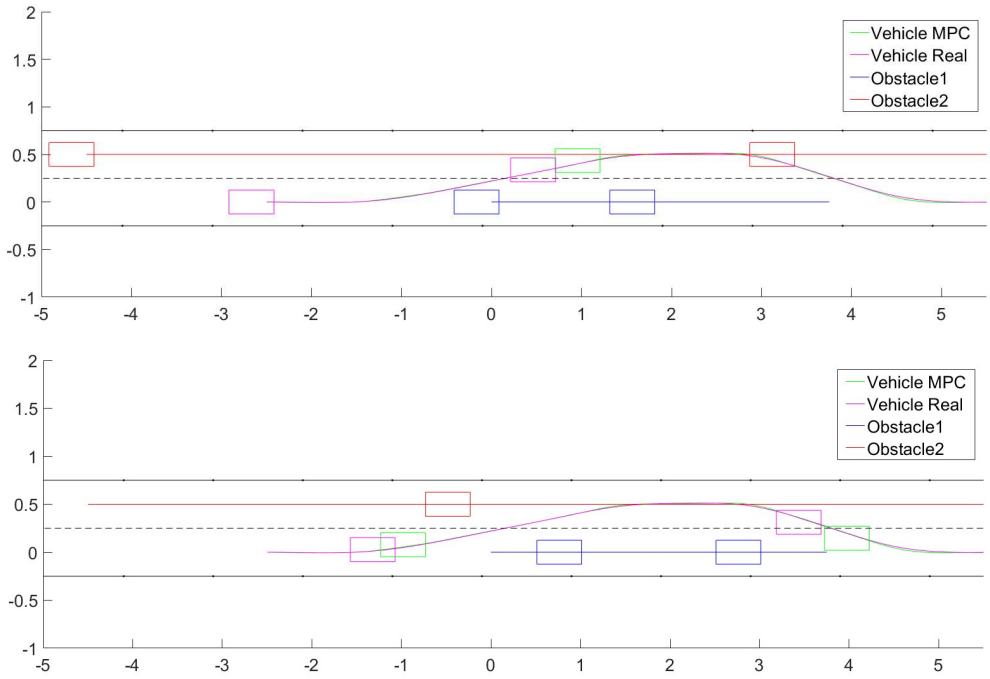


Figure 5.38: Plot over time of the vehicle lateral position signals. The desired velocity in blue, the generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

In the fifth, and last, scenario to analyse, both obstacles are driving in their lanes with a respective speed of $v_1 = 0.2\ m/s$ and $v_2 = 0.9\ m/s$. The initial positions of the vehicle and the obstacles are set to $x_0 = -2.5$, $x_{1_0} = 0$ and $x_{2_0} = -4.5$, and the controlled vehicle longitudinal speed is set to $v_x = 0.5\ m/s$.

For this scenario, the trajectory plots are divided in two figures, 5.39 and 5.40, for an easier understanding, as four different time instant positions for the vehicle and obstacles are represented. Both figures show how the controlled vehicle slows down to let the obstacle 2 overtake obstacle 1, before executing the manoeuvre itself. In figure 5.39, the truck positions corresponding to the simulation initial time and $t = 9$ s are shown, while in figure 5.40, the truck positions corresponding to $t = 5$ s and $t = 15$ s are represented. For $t = 15$ s, obstacle 2 is outside the plot limits as it is not transcendent anymore.

One interesting thing to be noticed in these figures, is that we can detect that the controlled vehicle has slowed down in $t = 5$ s, because the longitudinal distance between the trajectory generated by the MPC and the true trajectory is lower than in the following times.



Figures 5.39 and 5.40: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and by the kinematic model (magenta), together with the obstacles 1 (blue) and 2 (red). All units are in SI.

Figures 5.41 and 5.42, which show the plots of the vehicle longitudinal velocity signals and the lateral position signals, respectively, reaffirm the successful resolve of the second scenario posed for this approach. This time, in the longitudinal velocity plot it can be seen the speed of the obstacles 1 and 2 in colour purple and green, respectively. In this figure, it can be easily seen the deacceleration of the controlled vehicle to let obstacle 2 come first. It has to be said that the boot acceleration of the obstacles in simulation is higher than in reality, especially for obstacle 2 when reaching the high speed $v_2 = 0.9\ m/s$.

This time, in the plot of the vehicle lateral position signals, it can be seen a trapezoidal trajectory, because the vehicle stays longer with the LCC active, due to the scenario parameters.

Last thing to mention, is that all the constraints have been respected in this scenario.

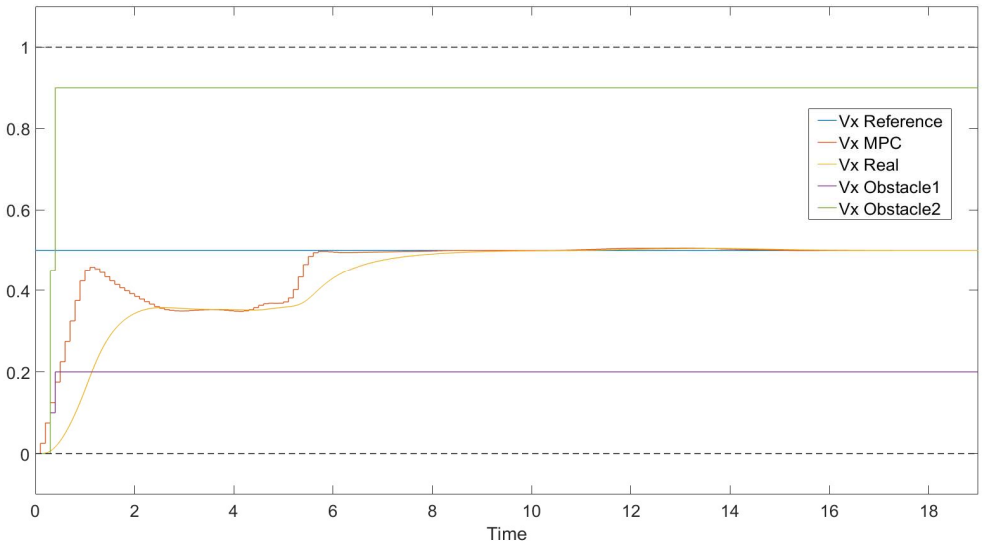


Figure 5.41: Plot over time of the longitudinal velocity signals for the vehicle and obstacles 1 (purple) and 2 (green). Vehicle desired velocity in blue, generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

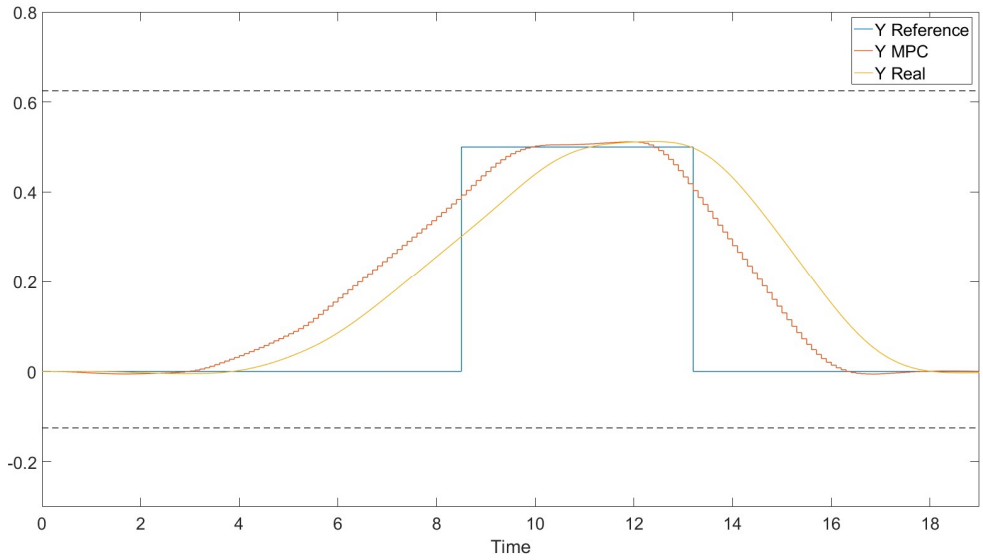


Figure 5.42: Plot over time of the vehicle lateral position signals. The desired velocity in blue, the generated by the MPC in red and the generated by the kinematic model in yellow. All units are in SI.

6 Testing

6.1 Hardware and software

In this chapter, the software and hardware used in the testbed, also named in this thesis as truck lab, will be described. To make it easier for the reader to comprehend how the testing procedure works, it will be explained in a structured way, which follows the data flow: starting from the MATLAB Simulink model used in the simulations and ending with the online feedback system. The testbed system is shown schematized in figure 6.1.

The first thing to do after the simulations of the model have been carried out successfully, is to adapt the Simulink model for the small-scale trucks to be able to run it. This is done through a Simulink library that has been created by the Insitute of Automation and Control. It includes the RT-MaG Toolbox[1], which is used to generate real-time capable programs, UDP inputs, to receive the feedback data, and the BeagleBone Support package, to generate the outputs and logfiles. Once the model is adapted with the Simulink testing library blocks needed, it is converted to C code through the RT-MaG Toolbox, and then sent to an FTP-Server where the truck also has access to.

The small-scale (1:14) trucks are equipped with a BeagleBone Black Board[2] that is an online single-board computer, in charge of compiling and running the C code of the model. Linux Debian[3] is used as operating system (OS) on the board. The communications to send and receive data are done through Wi-Fi protocol, using a WLAN module equipped in the trucks. For the position tracking, each truck is

[1]RT-MaG: http://www.gipsa-lab.fr/projet/RT-MaG/
[2] BeagleBone Black Board: https://beagleboard.org/black
[3] Linux Debian: http://elinux.org/BeagleBoardDebian

equipped with a different AprilTag[4], which is a visual marker designed for robust detection by cameras. Finally, the trucks are actuated by a motor and servo motors for gear changing and steering. However, gear changing will not take place in the tests, as it is not needed.

As said before, the truck tracking is done through AprilTags. These Tags are detected by four WebCams installed on the lab ceiling. The images captured by the cameras are processed on a dedicated Linux workstation using OpenCV library. Then, the trucks positions are sent via Wi-Fi as an UDP-Packet, which contains the ID of the detected truck, its x and y coordinates and its orientation angle. The detection is done with a rate of 10Hz and an accuracy of $0.03\ m$.

Finally, the data logged in the BeagleBone Black Board of the trucks can be accessed through a software program called WinSCP.

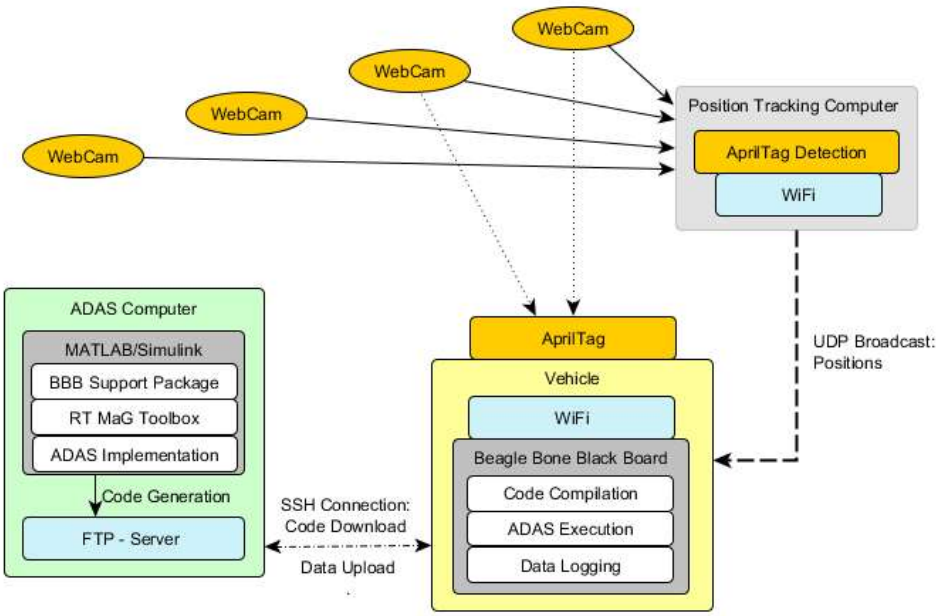


Figure 6.1: Scheme of the Testbed setup.

[4] AprilTag: https://april.eecs.umich.edu/wiki/AprilTags

6.2 Code generation changes

Once the MATLAB Simulink model that is tested has passed the simulation phase successfully, it is converted to C code so it can be sent to and run by the BeagleBone Black Board installed on the testing trucks. In this section, the changes that were required in order to successfully generate the code are presented. The addition of the required Simulink testing library blocks, such as input and output conditioning blocks, are not contemplated in this chapter.

First of all, Interpreted MATLAB Function blocks are not supported for code generation. This means that we have to use another kind of block for the optimizer computation. In this case, we have decided to use a regular MATLAB Function block, specifying in the Model Explorer which function inputs are real inputs and which are just parameters.

Second, the optimization formulation *Yalmip* and its corresponding functions used in the simulation phase are not allowed for code generation. Later on, it will also be seen that the *quadprog* function is not supported neither. To solve this, the *mpcqpsolver*[5] function introduced in the MATLAB version R2015b will be employed.

This optimization function solves the QP problem using an active-set method, the KWIK algorithm, described in Schmid [14], which requires the Hessian matrix to be positive definite. This algorithm uses the inverse of lower-triangular Cholesky decomposition of the Hessian matrix, instead of directly using the Hessian matrix, as in other optimizers. This additional computation of the Hessian needed for this optimizer, can produce small errors that lead to non-symmetry or a non-definite positive matrix. To prevent these errors some adjustments of the Hessian matrix are implemented in the MATLAB code before the optimizer call.

As explained before, the code generation is done using the RT-MaG Toolbox. For an unknown reason, even if the program sends a successful code generation feedback message, when the BeagleBone tries to compile the code, it sends an error message due to invalid generated code. After an exhaustive investigation, a solution was found. This solution consists on deleting manually, from the C code generated script *main.c*, the numbers of the parameters *step0* and *step1,* leaving just the name

[5] Mpcqpsolver: https://www.mathworks.com/help/mpc/ref/mpcqpsolver.html

step. It is known that this is not the best solution, as it does not solve the cause of the problem and it requires to manually modify the C code generated script each time that a new model is needed to be compiled, but it is the only one found.

6.3 Lane change test results

In this chapter, the testing procedure and the results obtained for the lane change approach, will be explained.

Once the lane change model is fully adapted, so it can be compiled and run by the BeagleBone Black Board, the test is ready to be executed. Before executing each test, the small-scale truck is manually positioned to the initial coordinates $x_0 = -3$, $y_0 = 0$ and $\theta_0 = 0$. As the positioning is done manually, small initial deviations are likely to happen.

The overall control parameters used, consisting in MPC and low-level control, are the ones obtained during the simulation tuning, described in chapter 4.3. The MPC parameters are expected to remain the same during the whole testing phase, if the tuning was done correctly, but the low-level control ones are most likely to change, as the kinematic model used in the simulations can differ from the real dynamics of the small-scale trucks. Hence, the initial testing parameters are: $K_p = 5,\ K_{stan} = 4,\ K_{theta} = 1,\ Q_v = 15,\ Q_y = 60,\ R_x = 1,\ R_y = 60,\ N_p = 30,\ N_u = 12.$

The first simulation results are shown from figure 6.1 to 6.5.

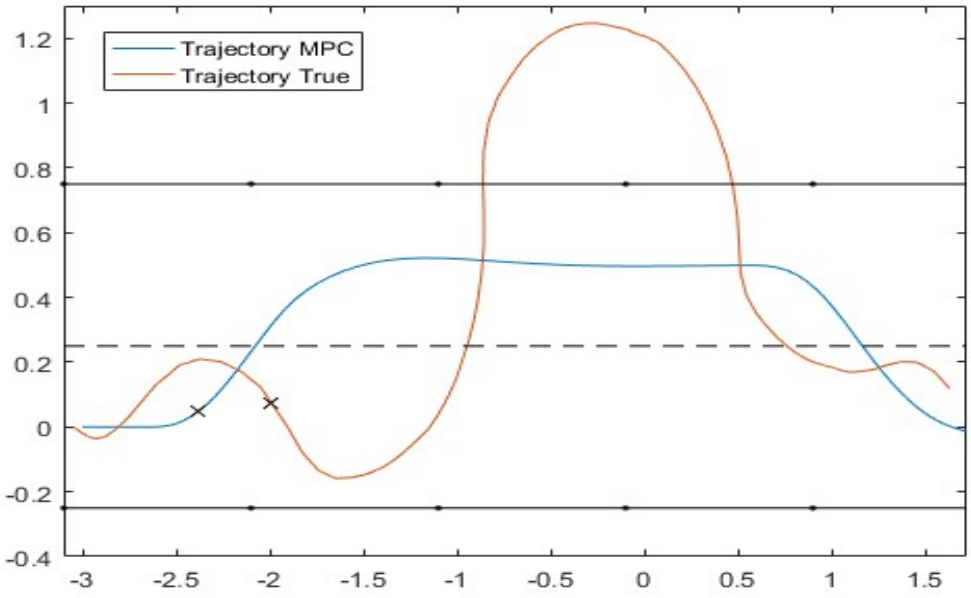


Figure 6.2: Trajectory (lateral vs longitudinal position) of the truck generated by the MPC (blue) and the actual one (red). The cross marks belong to the trajectory positions at time instant $t = 2s$. All units are in SI.

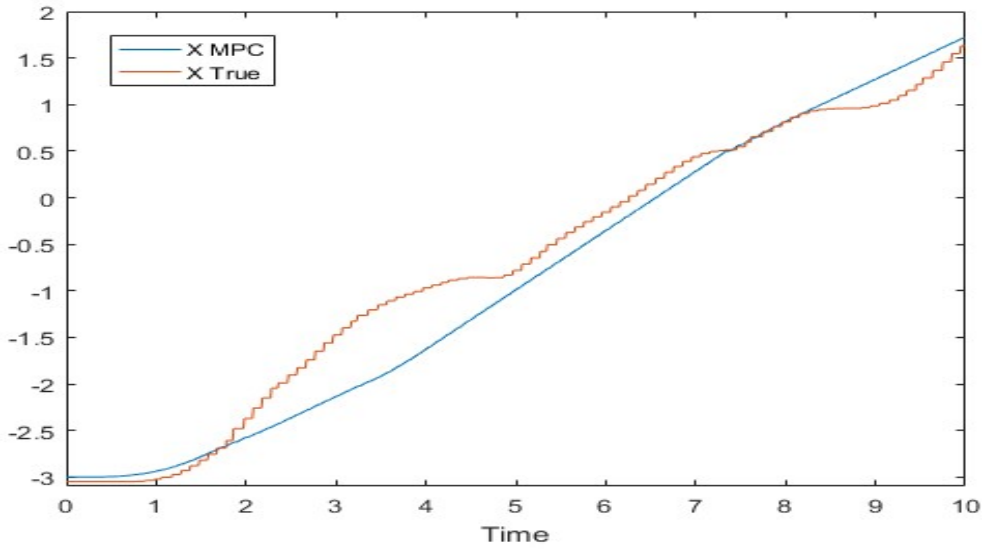


Figure 6.3: Longitudinal position over time of the truck generated by MPC (blue) and the actual one (red). All units are in SI.

As it can be seen in the trajectory plot shown in figure 6.2, The true trajectory of the truck differs a lot from the one given by the MPC. Figure 6.3 corresponds to the longitudinal position plot and it shows that the actual position of the truck is higher than the MPC signal, used as reference for the low-level controller. If we also plot in figure 6.2 a cross mark in the position corresponding to the time instant e.g. $t = 2s$, it can be confirmed that the vehicle is driving ahead of its trajectory reference. This explains the oscillating trajectory generated by the truck.

Having a look at the velocity in figure 6.4, it can be seen that the actual truck speed (red) has a higher value than the one designated by the MPC (blue) mostly all the time. This is the reason why the truck drives ahead of the MPC trajectory. The yellow signal, corresponding to the low-level control speed output which is sent to the truck, indicates that the issue is in the low-level velocity control, as the actual signal is just following the one given as reference.

Another cause for the actual trajectory to be ahead of its reference could be the sample time being lower than the time required by the Beagle Bone to execute the MPC tasks. To check this, execution times are plotted in figure 6.5, where it can be seen that the MPC execution time mean is around $0.03\ s$, while its sample time is set to $0.1\ s$. The low-level controller execution time mean is also lower than its sample time, which corresponds to $0.01\ s$.

Finally, figure 6.6 shows the CPU load in percentage and its mean, which is situated around $22.5\ \%$.

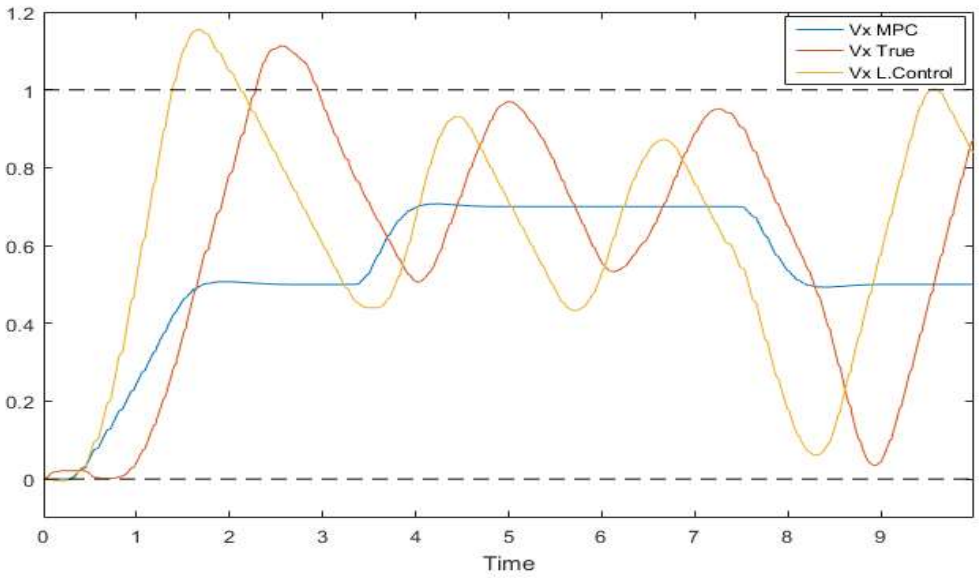


Figure 6.4: Longitudinal velocity of the truck generated by MPC (blue), generated by the low-level controller (yellow) and the actual one (red). All units are in SI.

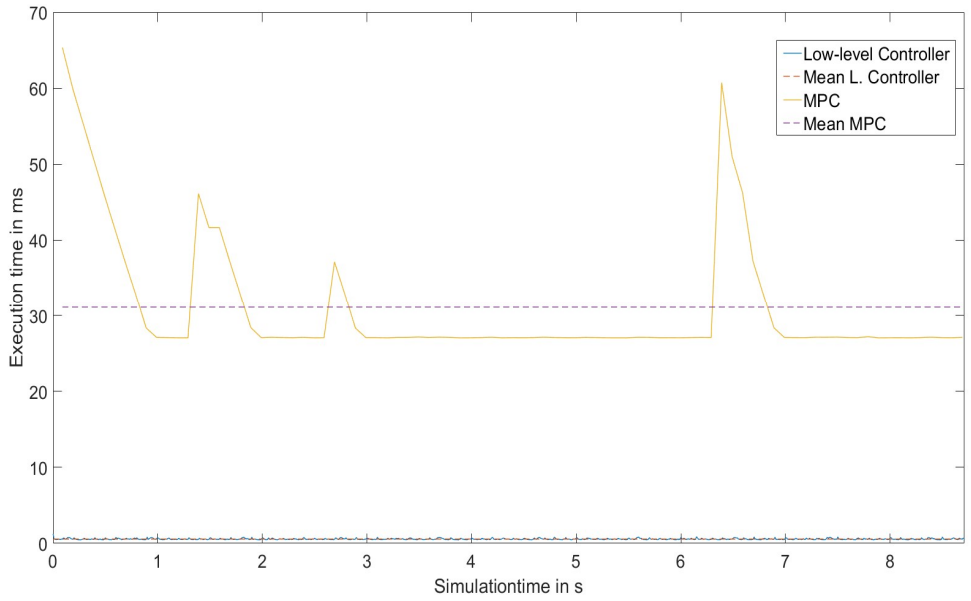


Figure 6.5: Execution times of the Low-level controller tasks (blue) and the MPC tasks (yellow), together with its means (red and purple, respectively).

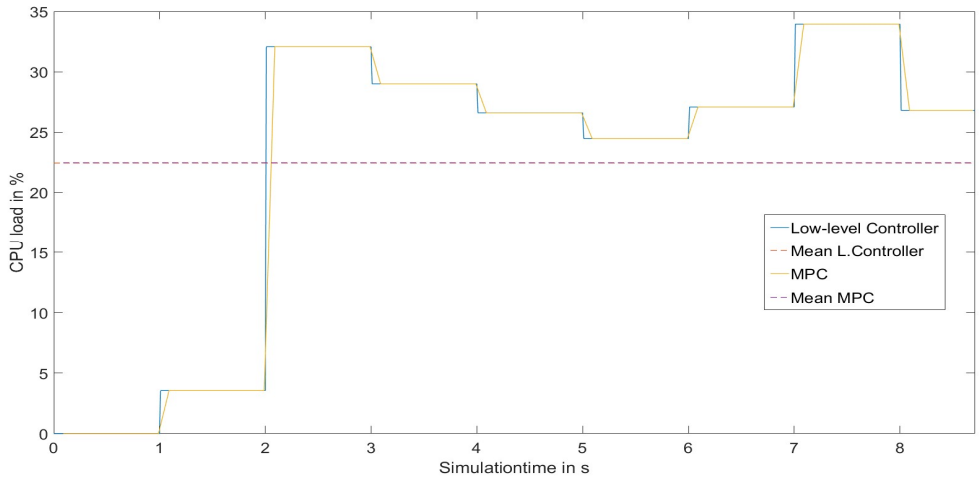


Figure 6.6: CPU load of the Low-level controller tasks (blue) and the MPC tasks (yellow), together with its means (red and purple, respectively).

The conclusions made from the results obtained in the first test have led to a tuning phase for the low-level control parameters. A series of tests have been carried out, resulting in the parameters $K_p = 0.9$, $K_{stan} = 0.5$ and $K_{theta} = 0$ as the ones fitting best for this approach. As described in chapter 4.3, K_{thet} corresponds to the gain of the vehicle orientation angle error signal for the low-level steering control. Zero for this parameter means that the signal is not used. The reason behind this decision, is that a truck trajectory with the same shape as the one given by the MPC is sought, even with a delay.

The test results obtained with the low-level control parameters tuned are shown in figures 6.7 to 6.10.

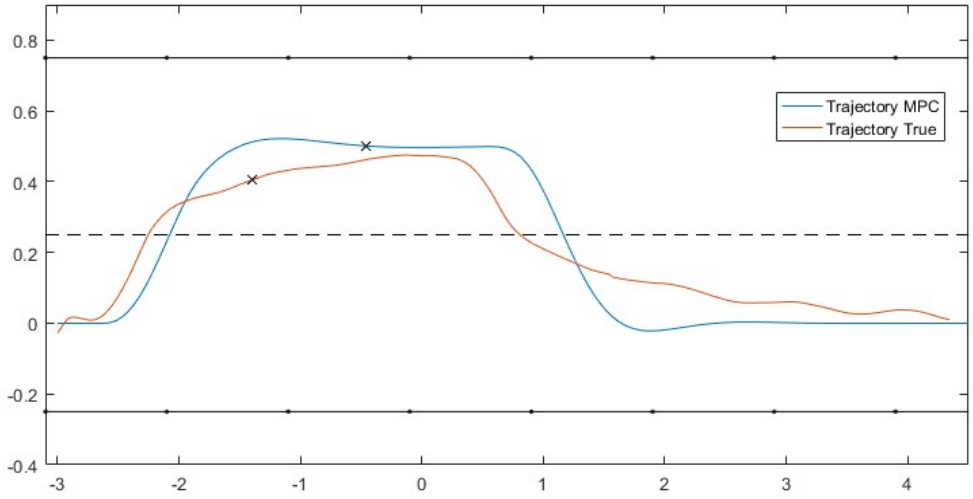


Figure 6.7: Trajectory (lateral vs longitudinal position) of the truck generated by the MPC (blue) and the actual one (red). The "x" belong to the trajectory positions at time instant $t = 4.5\ s$. All units are in SI.

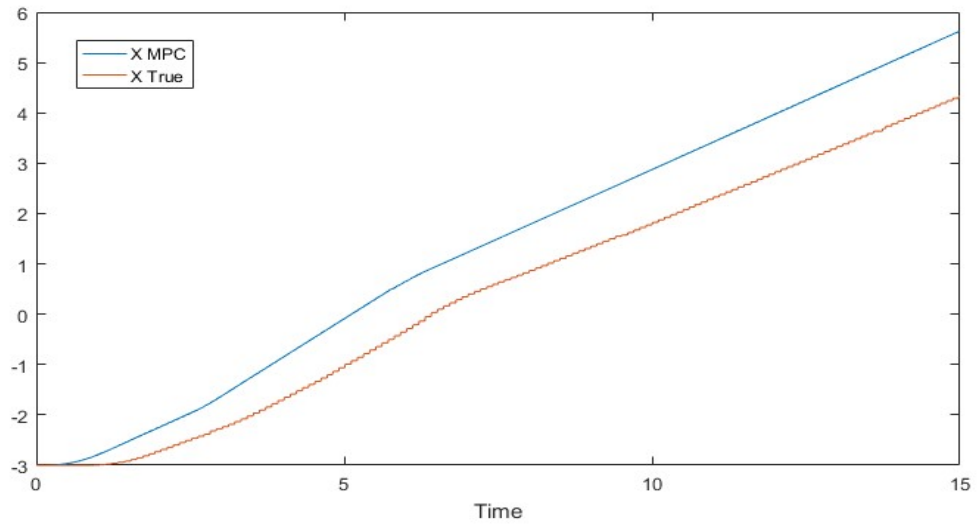


Figure 6.8: Longitudinal position over time of the truck generated by MPC (blue) and the actual one (red). All units are in SI.

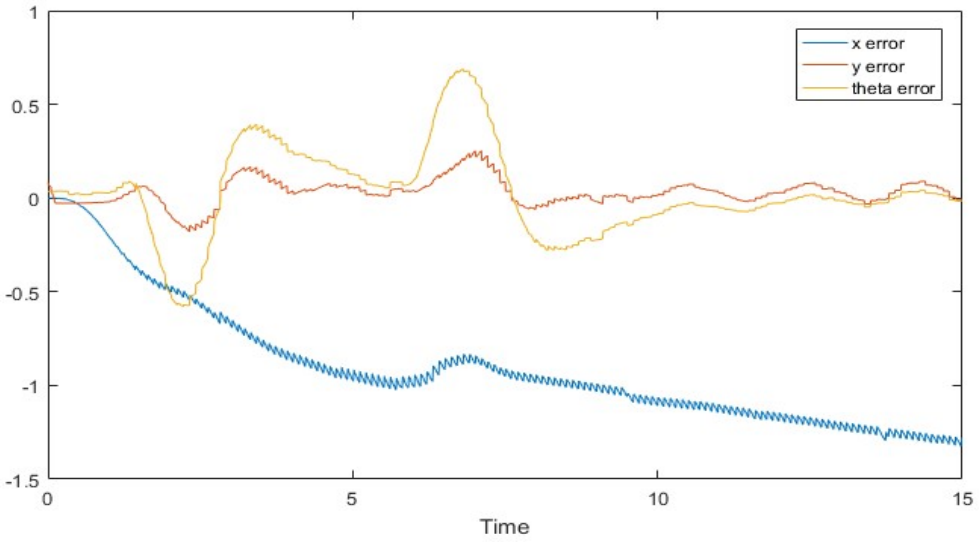


Figure 6.9: Longitudinal and lateral position errors (blue and red, respectively) and of the orientation angle error (yellow). All units are in SI.

Figure 6.7 shows that the truck follows the MPC trajectory better than with the previous parameters. Therein, the cross mark corresponding to the time instant $t = 4.5\ s$. Together with the longitudinal position plot shown in figure 6.8, it can be concluded that the truck no longer drives ahead of the reference at any time.

In figure 6.9, the errors computed from the difference between the true vehicle signals and the MPC outputs for the position coordinates and the orientation angle are plotted. As it can be seen in this figure, and in figure 6.8, the issue now is that the longitudinal position error keeps incrementing overtime. Hence, a mechanism

controlling this parameter is needed in order to not allow the vehicle to go ahead of the reference at the start, but at the same time, prevent high differences in the long run.

The velocity signals are shown in figure 6.10, where it can be seen that the loss of speed with respect to the MPC generated signal takes place in the initial phase of the test, when the vehicle accelerates from zero velocity.

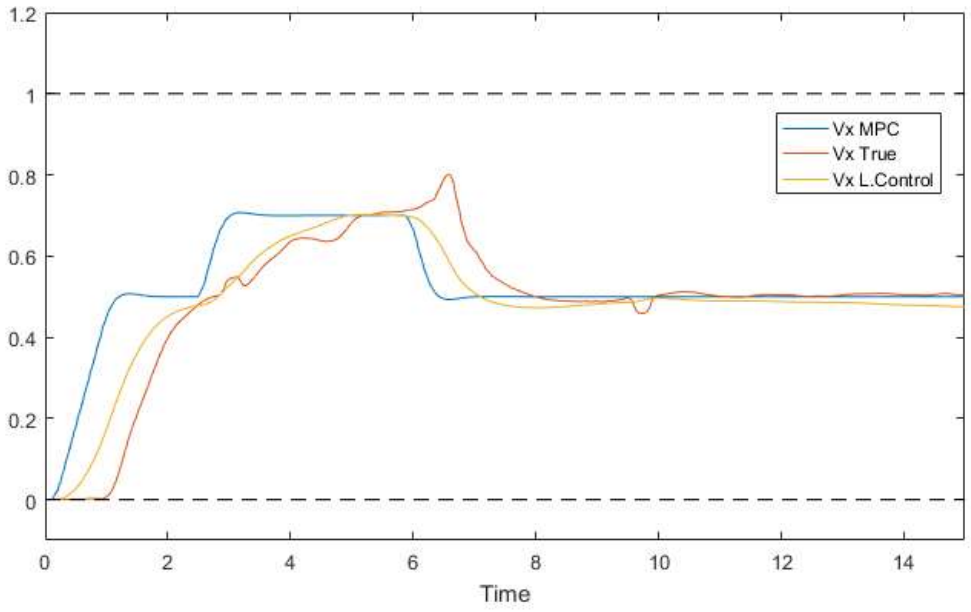


Figure 6.10: Truck longitudinal velocity generated by MPC (blue), generated by the low-level controller (yellow) and the actual one (red). All units are in SI.

As mentioned before, the distance between the true truck trajectory and the one generated by the MPC needs to be controlled. For this reason, the longitudinal position error signal is added to the low-level speed control, resulting in a PI control. The addition of the signal is done through a gain, defined as K_I, and a tolerance, as shown in equation (6.1). This tolerance is used to allow a certain error and thus prevent the truck to be ahead of its reference at any time.

$$v_i = K_p \cdot e_v + K_I \cdot (-e_x - tolerance) \tag{6.1}$$

where e_v and e_x represent the velocity and the longitudinal position errors, respectively.

Obviously, due to the low-level speed control modification, its parameters must be tuned again. After several tests, the following speed control parameters fit best for this approach: $K_p = 1.25$, $K_I = 0.5$ and tolerance $= 0.25$. The rest of the tuning variables remain unchanged from the last tests.

The results obtained with these new parameters are plotted in figures 6.11 to 6.16.

As it can be seen in figure 6.11, the trajectory generated by the MPC can be followed by the small-scale truck, especially in the first half of the test. The cross mark in the positions corresponds to the time $t = 9\ s$. Around this time happens the biggest lateral error between the actual trajectory and the one generated by the MPC and the longitudinal distance seems appropriate for the controller to be able arrange it faster. Nonetheless, a higher value on the lateral error gain, K_{stan}, leads to worse results.

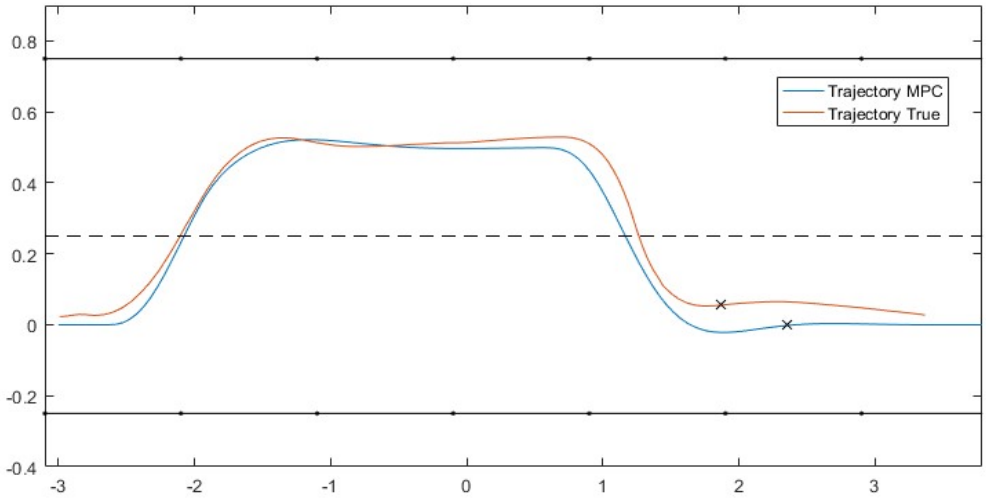


Figure 6.11: Trajectory (lateral vs longitudinal position) of the truck generated by the MPC (blue) and the actual one (red). The "x" belong to the trajectory positions at time instant $t = 9\ s$. All units are in SI.

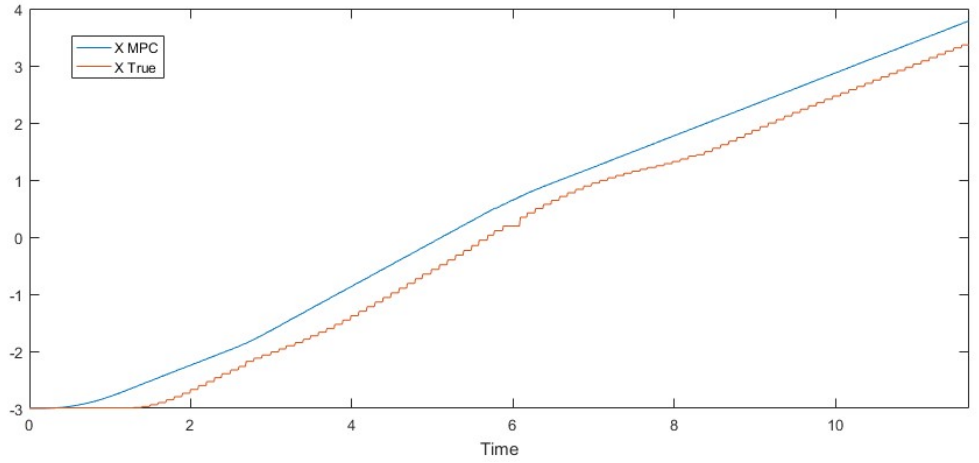


Figure 6.12: Longitudinal position over time of the truck generated by MPC (blue) and the actual one (red). All units are in SI.

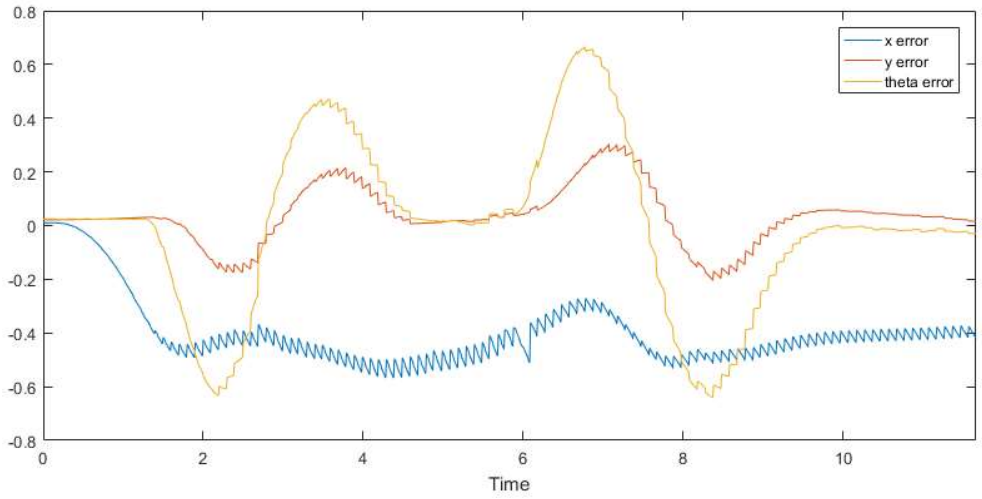


Figure 6.13: Longitudinal and lateral position errors (blue and red, respectively) and the orientation angle error (yellow). All units are in SI.

The longitudinal position and errors plots, shown in figures 6.12 and 6.13, let us conclude that the issue with the longitudinal position error increasing over time has been solved successfully, thanks to the new speed control implementation. On the other hand, this implementation leads to an actual speed signal that differs from the one generated by the MPC, as it can be seen in figure 6.14. Anyway, following perfectly the MPC velocity output is not considered important and the results are accepted if the speed boundaries are respected, so the integrity of the truck engine is not in danger. The small peaks that differ from the low-level control speed signal (yellow) are due to detection disturbances of the cameras.

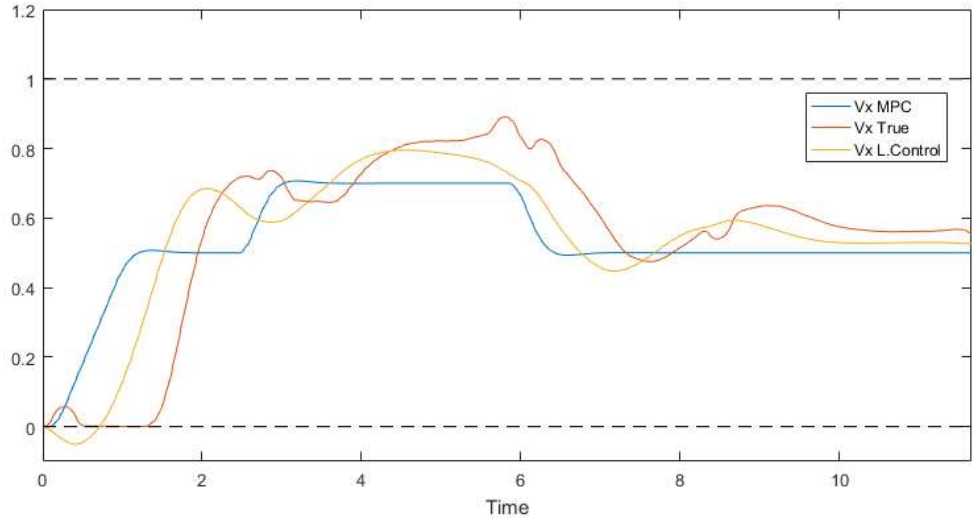


Figure 6.14: Truck longitudinal velocity generated by MPC (blue), generated by the low-level controller (yellow) and the actual one (red). All units are in SI.

After the first changes in the model during the testing phase, the execution time logs have not been taken correctly, mixing low-level controller tasks with the MPC ones. This causes the appearance of multiple periodical peaks over time. The period of this peaks is related with the sample times of the MPC and the low-level control. The cause of this issue could not be detected. Hence, it has been decided to plot the execution times together using the same colour, as it still shows an overall idea of it. This can be seen in figure 6.15.

Finally, several changes were made for the last tests, in order to reduce the computation requirements of the model without affecting the other results. This can be seen in figure 6.16, where the CPU load in percentage is shown with a mean of 6% approximately, a lot lower than the 22.5% obtained in the first test shown in figure 6.6.

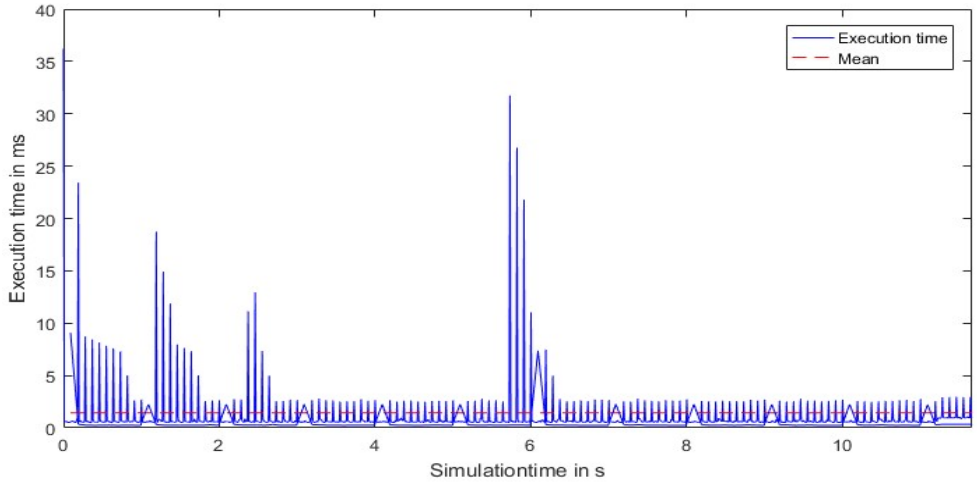


Figure 6.15: Overall execution times (blue) together with its mean (red).

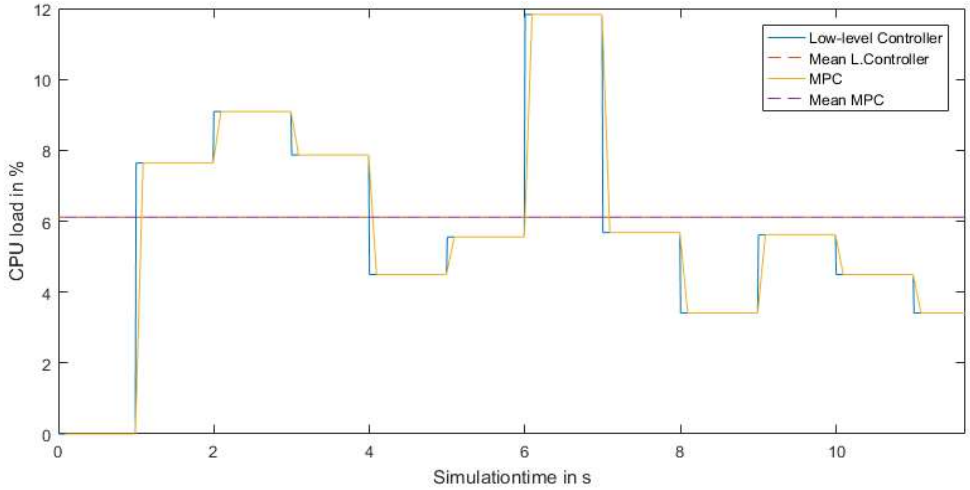


Figure 6.16: CPU load of the Low-level controller tasks (blue) and the MPC tasks (yellow), together with its means (red and purple, respectively).

6.4 Collision avoidance test results

In this chapter, the collision avoidance trajectory generation approach is tested and the final test results are presented and discussed.

The low-level control parameters obtained in the lane change testing phase and presented in the previous chapter are used for the collision avoidance approach. The truck used on these tests is the same, as well as the environment where it drives through. Hence, the low-level control parameters should not change a lot. These parameters are: $K_p = 1.25,\ K_I = 0.5$, tolerance $= 0.25$, $K_{stan} = 0.5$ and $K_{theta} = 0$.

Regarding the MPC parameters, including the collision avoidance parameters, the ones that fit the best for the simulations of the last collision avoidance trajectory generation approach are used. These parameters, already described in chapter 5.5, are: $Q_y = 1,\ Q_v = 40,\ R_x = R_y = 1,\ \rho = 5 \cdot 10^8,\ N_p = 30,\ N_u = 12,\ S_{df} = 1.5\ m$, $S_{dr} = 1\ m$, $W = 0.4\ m$ and $L = 0.7\ m$. The MPC parameters are expected to remain unchanged during the testing phase.

When executing the first test, it is rapidly seen that something is wrong, as the truck describes an oscillating trajectory shown in figure 6.17. Having in mind the issues experienced in the lane change testing, the first guess is that the truck might be moving ahead of its reference. To check that, the plot showing the actual longitudinal position of the truck and the one generated by the MPC is used (Figure 6.18). There, it can be seen that the suspicions were true, i.e., the truck is ahead of its reference.

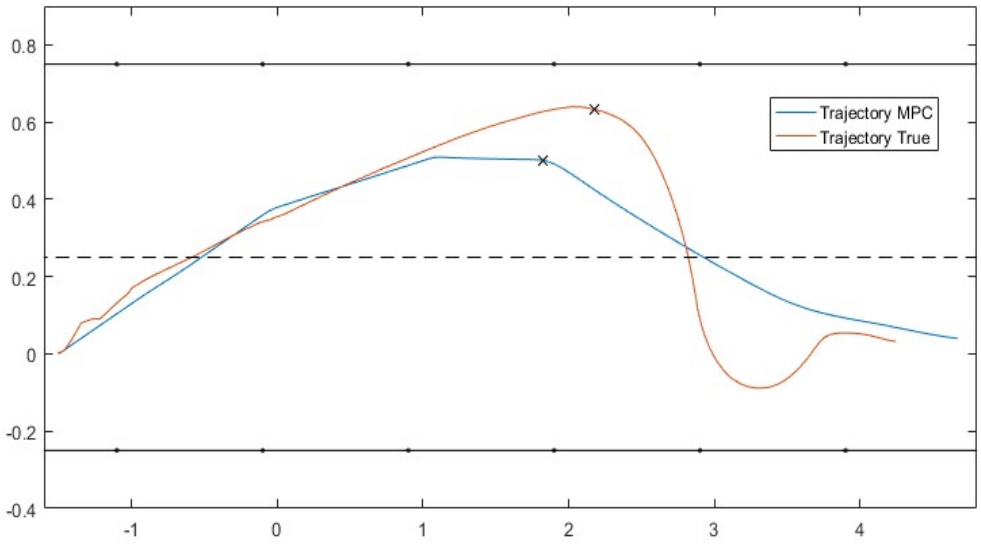


Figure 6.17: Trajectory (lateral vs longitudinal position) of the truck generated by the MPC (blue) and the actual one (red). The "x" belong to the trajectory positions at time instant $t = 6.5\ s$. All units are in SI.

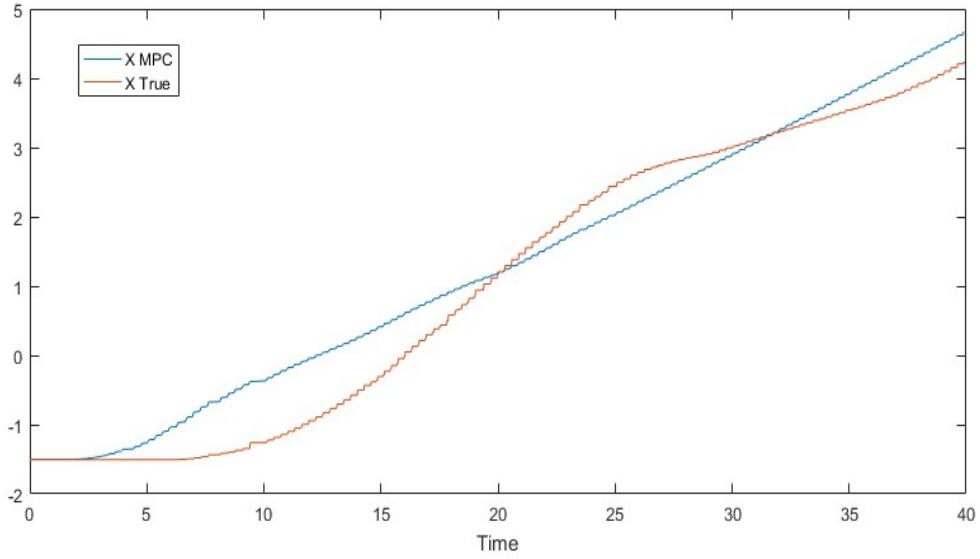


Figure 6.18: Longitudinal position over time of the truck generated by MPC (blue) and the actual one (red). All units are in SI.

As explained in the previous chapter, there are two potential causes for this to happen: the low-level velocity control parameters let the truck move faster than its reference, as it initially happened for the lane change approach, or that the computation time required for the beagle bone is higher than the sample time used and consequently, the MPC output trajectory is not sent in time. The first cause is quite unlikely to be happening, because the low-level control parameters used should not differ that much from the tuned ones, for the reasons explained in the previous paragraph. Anyway, to check if this issue is happening, the velocity plot of the truck and its reference is used (Figure 6.19). There, it is seen that overall, the truck is not moving in a faster speed than its reference. Hence, the first cause is discarded and to check if the second one is happening, the execution task times are analysed in figure 6.20. There, it is finally seen that a sample time of $0.1\ s$ for the MPC is too low, as there are too much spikes above $100\ ms$. Consequently, a new sample time of $0.2\ s$ is proposed for the MPC and, as shown in figure 6.22, it is found to be high enough.

In the CPU load plots, it can also be noticed that with a sample time of $0.1\ s$ the Beagle Bone is almost at its limits (Figure 6.21), while after increasing the sample time to $0.2\ s$ it is lowered until a mean of 9 %, approximately (Figure 6.23).

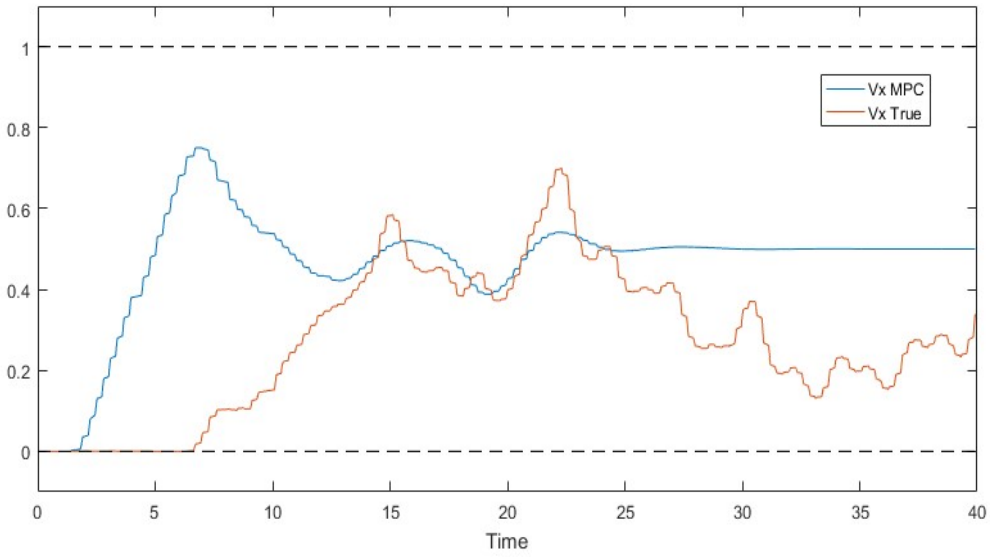


Figure 6.19: Truck longitudinal velocity over time generated by MPC (blue) and the actual one (red). All units are in SI.

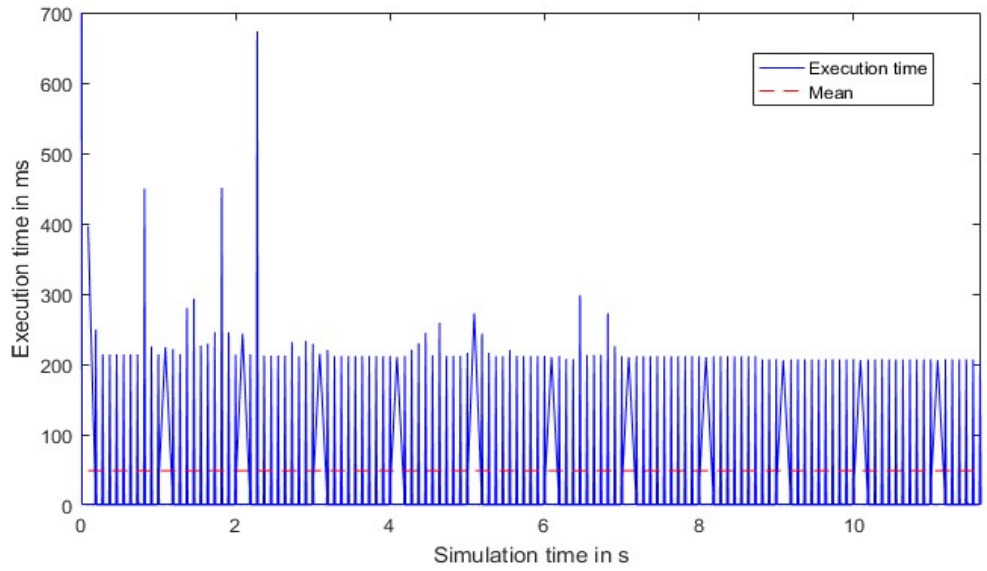


Figure 6.20: Overall execution times (blue) together with its mean (red) at $T_s = 0.1$.

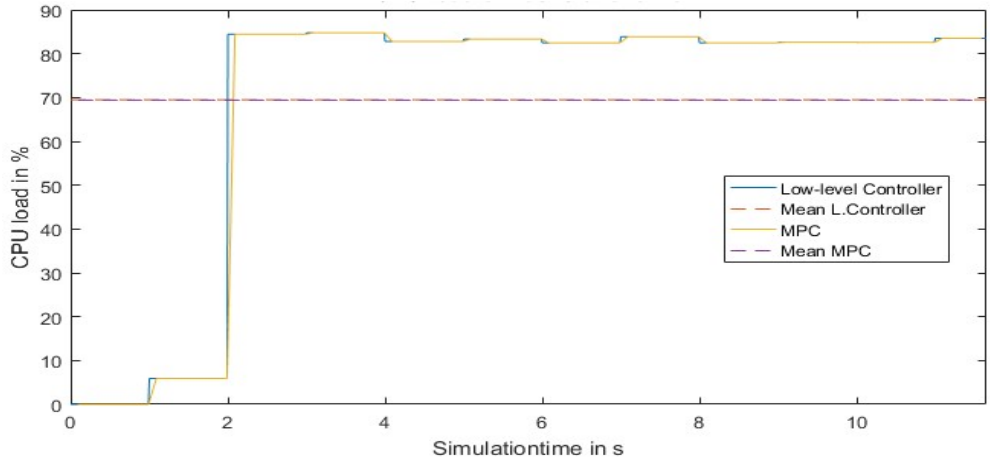


Figure 6.21: CPU load of the Low-level controller tasks (blue) and the MPC tasks (yellow), together with its means (red and purple, respectively) at $T_s = 0.1$.

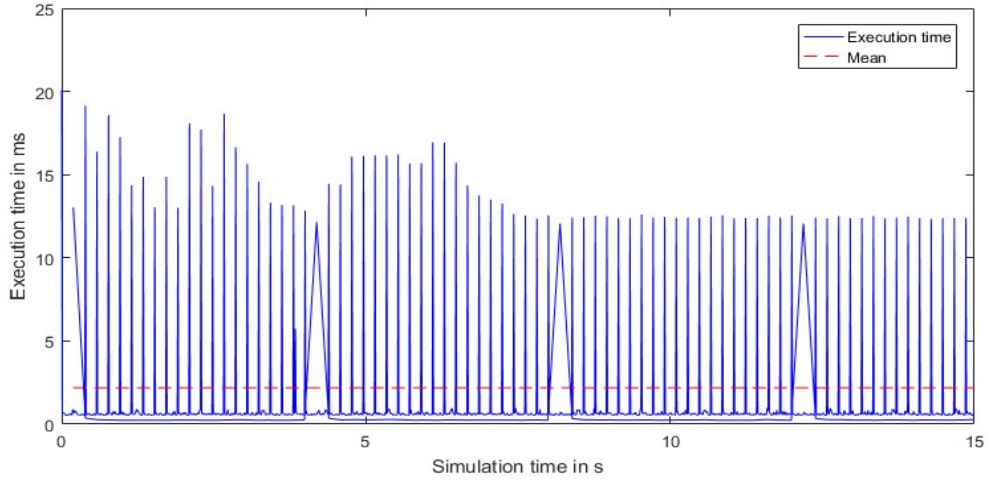


Figure 6.22: Overall execution times (blue) together with its mean (red) $T_s = 0.2$.

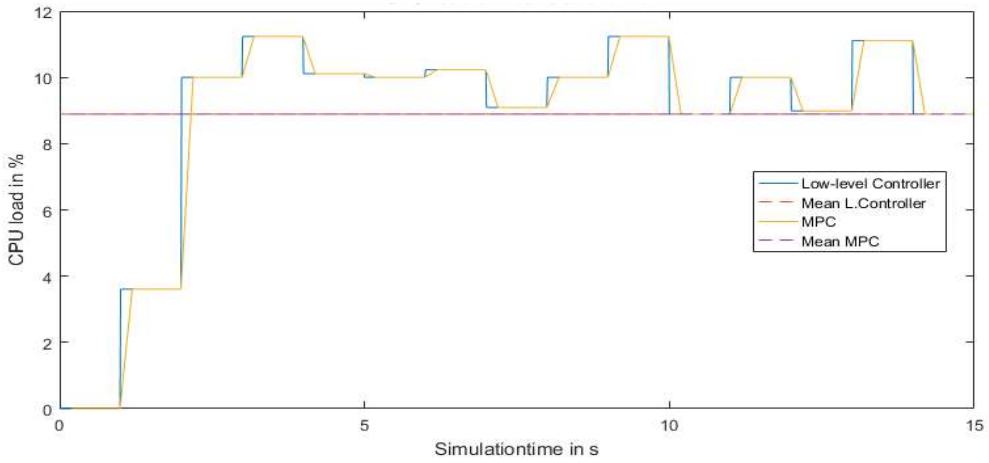


Figure 6.23: CPU load of the Low-level controller tasks (blue) and the MPC tasks (yellow), together with its means (red and purple, respectively) at $T_s = 0.2$.

With the sample time change, it is expected that the prediction and control horizons might change as well, as these parameters are related with the sample time. A series of simulations have been carried out and $N_p = 12,\ N_u = 6$ are considered the ones that fit the best. The rest of parameters remain unchanged.

Once the sample time issue has been solved, the actual tests can be started. The scenarios that have been tested in the small-scale trucks, are the ones that were successfully resolved in the simulation environment for the last approach, described in chapter 5.5.

In the first two scenarios, both obstacles remain static, $v_j = 0$. In the first scenario, the obstacle 2 is placed in front of the obstacle 1 and in the second one, the obstacle 2 is situated ahead of the obstacle 1.

The third scenario, consists of only one obstacle, driving in the right lane with a speed of $0.2\ m/s$.

In the fourth and fifth scenarios, there are again two obstacles, now driving with a certain speed. The speeds of the obstacles will change from one scenario to another, so that in the fourth scenario, the vehicle slows down to let the obstacle 2 overtake the obstacle 1 first and in the fifth one, the ego vehicle speeds up to overtake the obstacle 1 before the obstacle 2.

After several tests have been carried out, the following low-level control parameters fit best for the majority of the scenarios with this approach: $K_I = 0.2$, tolerance $= 0.25$, $K_{stan} = 0.55$ and $K_{theta} = 0$. The parameter K_p of the velocity controller has been changed depending on the desired vehicle speed. In the first scenario where the vehicle desired speed is $v_x = 0.8\ m/s$, $K_p = 0.6$ and for the rest of scenarios, where $v_x = 0.6\ m/s$, $K_p = 0.9$.

With all this in mind, the definitive test results for the first scenario are shown from figure 6.24 to 6.26.

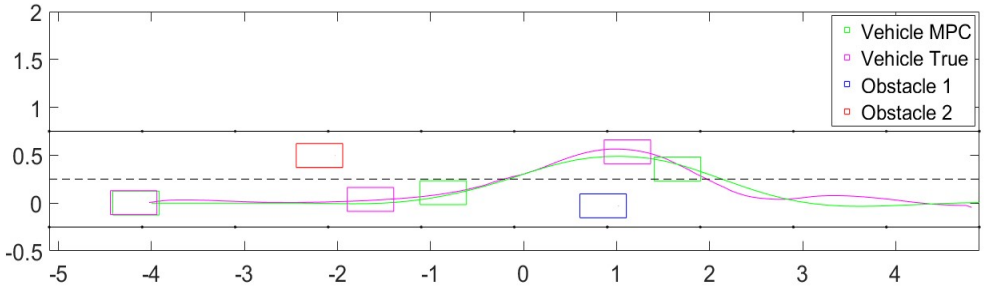


Figure 6.24: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and the actual one (magenta), together with the obstacles 1 (blue) and 2 (red). All units are in SI.

In figure 6.24, it can be seen that the overtaking manoeuvre has been realized successfully. Three different positions are also plotted with rectangles at time instants: $t = 0$ s, $t = 5$ s and $t = 8$ s.

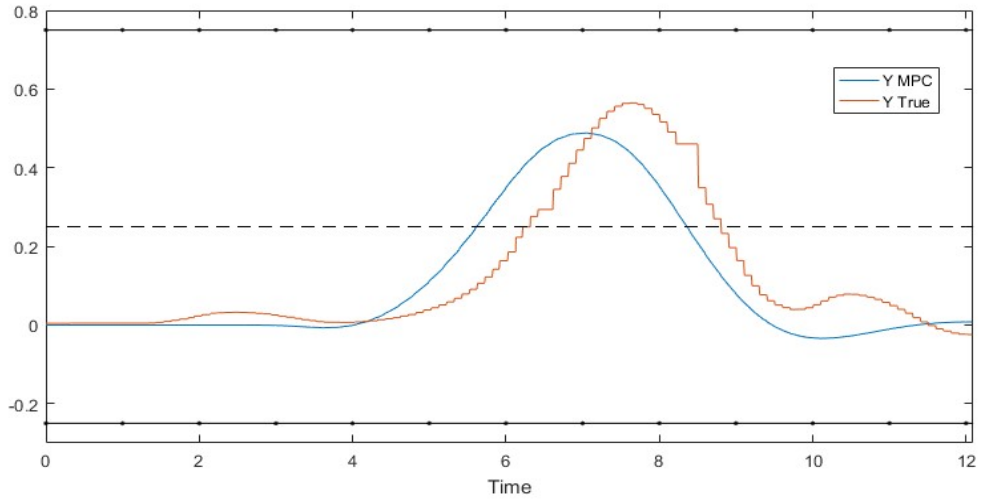


Figure 6.25: Truck lateral position over time generated by MPC (blue) and the actual one (red). All units are in SI.

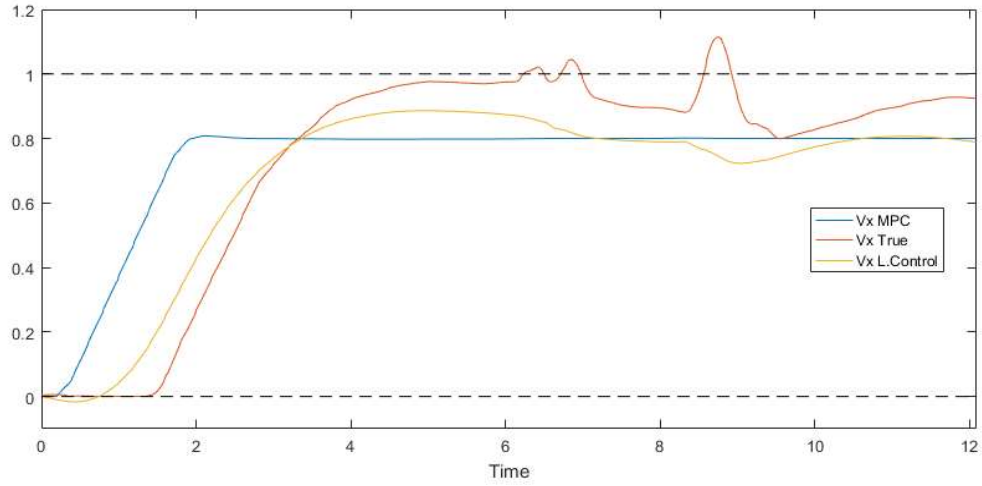


Figure 6.26: Truck longitudinal velocity generated by MPC (blue), generated by the low-level control (yellow) and the actual one (red). All units are in SI.

In figure 6.25, the lateral position of the controlled truck is plotted. There, it can be noticed two disturbances, one small around time 6.5 s and another around 8 s. These disturbances appear due to the cameras losing the truck for a small period of time. These same disturbances can also be noticed in the velocity signal plot, shown in figure 6.26. There, the low-level control output speed signal (yellow) is also plotted to show clearly that the peaks in the true speed signal are due to the camera disturbances just presented. This means that the truck has not actually crossed the

$1\ m/s$ boundary. Note that this boundary is not desired to be crossed but the real capability of the truck is at $1.5\ m/s$.

Let us move to the second scenario, where the obstacle longitudinal positions have been changed. In the previous scenario presented, the true vehicle velocity was close to its superior boundary, due to having to catch up with the MPC trajectory. For this reason, the controlled vehicle desired speed has been lowered from $0.8\ m/s$ to $0.6\ m/s$.

Figures from 6.27 to 6. 29, show the results obtained in the experiments of the second scenario.

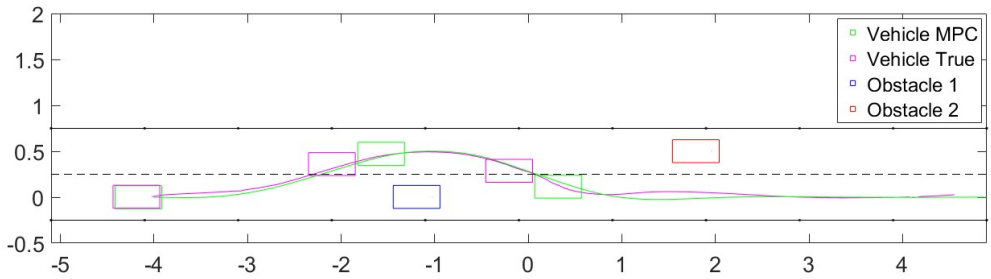


Figure 6.27: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and the actual one (magenta), together with the obstacles 1 (blue) and 2 (red). All units are in SI.

The vehicle trajectory generated by the MPC and the actual one are plotted in figure 6.27. As it can be seen, the overtaking manoeuvre has also been realized successfully in this scenario. Again, three different positions are plotted with rectangles at the same time instants as before ($t = 0$ s, $t = 5$ s and $t = 8$ s).

In figure 6.25, the lateral position of the controlled truck is plotted. This time, no huge disturbances appear. Only small peaks can be noticed in the velocity plot, when comparing the actual signal (red) with the low-level control output one (yellow), shown in figure 6.19.

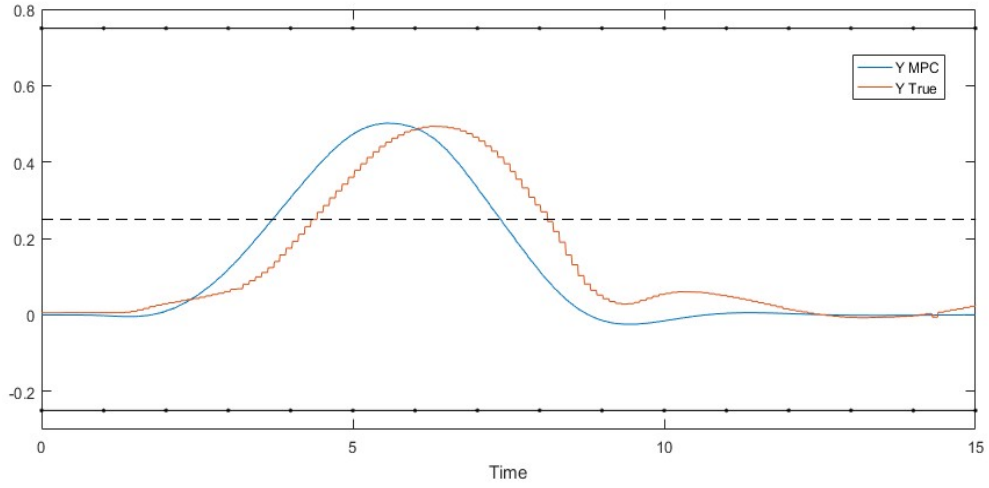


Figure 6.28: Truck lateral position over timegenerated by MPC (blue) and the actual one (red). All units are in SI.

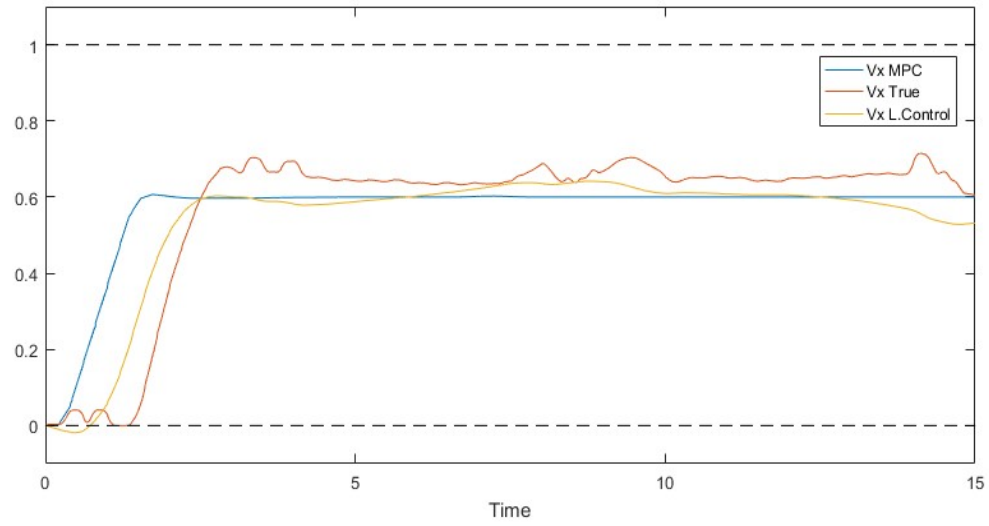


Figure 6.29: Truck longitudinal velocity generated by MPC (blue), generated by the low-level control (yellow) and the actual one (red). All units are in SI.

The third scenario results are shown in figures from 6.30 to 6.32. In the trajectories plot in figure 6.30, also the trajectory from the Obstacle 1 is shown. As it can be noticed, the obstacle drives slightly to the left, too close to the centre of the road. Nevertheless, the controlled truck can react to this kind of variations if necessary. Again, three different positions are plotted with rectangles at time instants: $t = 0$ s, $t = 6$ s and $t = 9$ s, for this scenario.

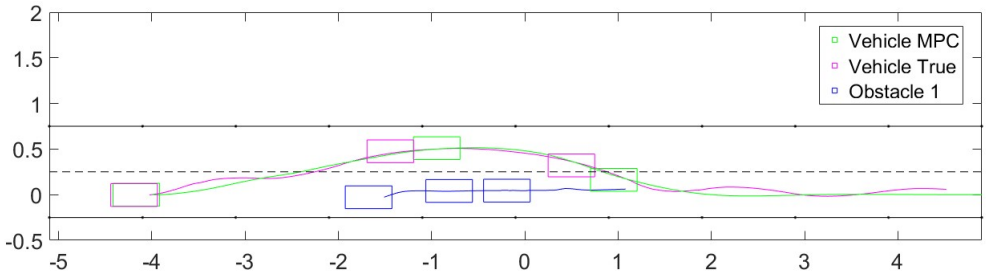


Figure 6.30: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and the actual one (magenta), together with the obstacle 1 (blue). All units are in SI.

In figure 6.31, the lateral position of the controlled vehicle is plotted. A deviation can be noticed happening around the $3\ s$ point and then a small oscillation when coming back to the right lane. This oscillation is most likely to stabilize if testing road was longer.

Finally, figure 6.32 shows the longitudinal velocity for the vehicle and also for the Obstacle 1 (yellow). There, a peak disturbance appears in the Obstacle 1 signal around the $11\ s$ mark, due to the camera losing the truck for a small period of time.

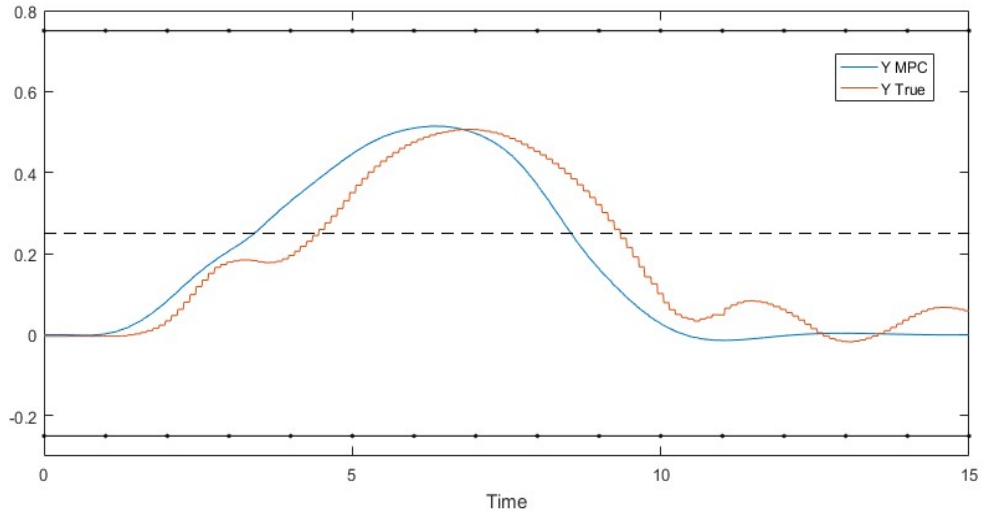


Figure 6.31: Truck lateral position over time generated by MPC (blue) and the actual one (red). All units are in SI.

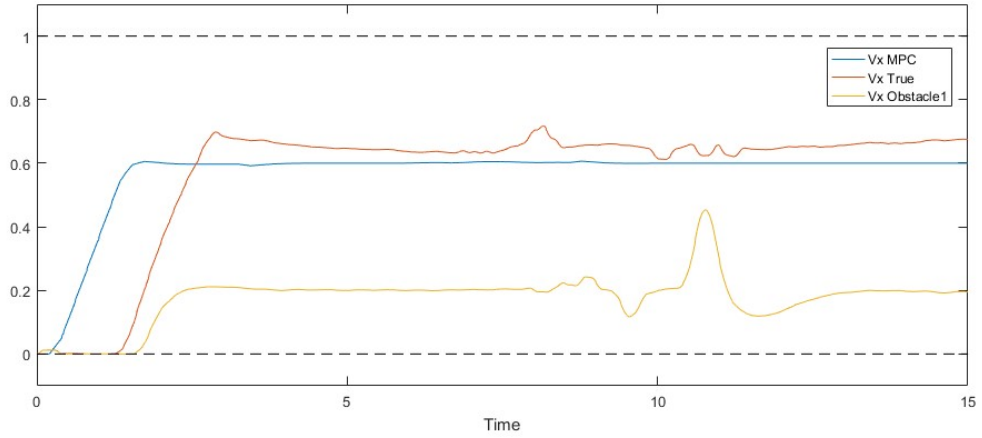


Figure 6.32: Longitudinal velocity over time of the controlled vehicle generated by MPC (blue) and the actual one (red) together with the longitudinal velocity of obstacle 1 (yellow). All units are in SI.

In the fourth scenario, two obstacles are moving with the left one moving at a speed of $0.7\ m/s$.

Figures from 6.33 to 6. 35, show the results obtained in the test for this scenario.

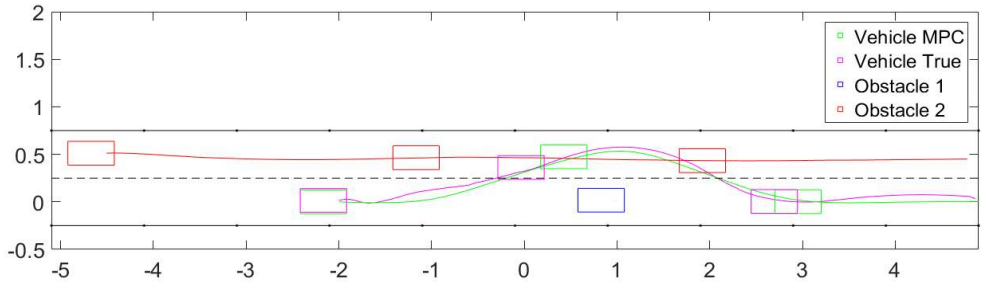


Figure 6.33: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and the actual one (magenta), together with the obstacles 1 (blue) and 2 (red). All units are in SI.

In figure 6.33, the trajectories from the controlled vehicle and from the obstacle 2 (red) are shown. As it can be seen, thanks to the three positions plotted with rectangles at time instants: $t = 0$ s, $t = 5$ s and $t = 9$ s, the controlled vehicle overtakes Obstacle 1, situated in the right lane, before Obstacle 2.

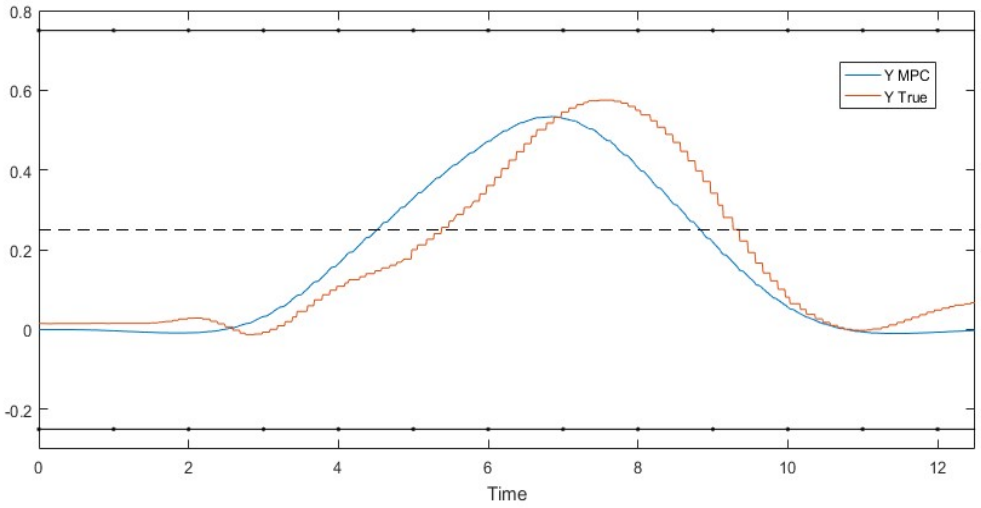


Figure 6.34: Truck lateral position over time generated by MPC (blue) and the actual one (red). All units are in SI.

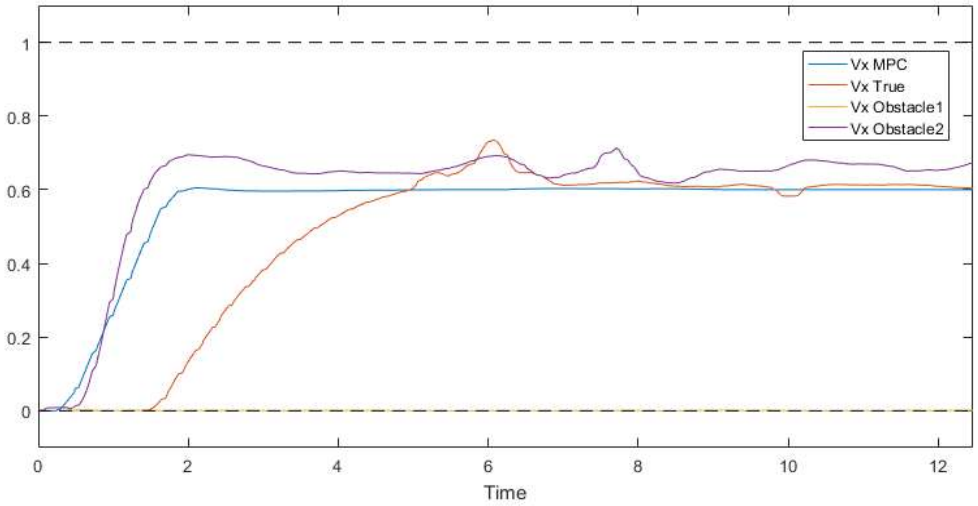


Figure 6.35: Longitudinal velocity over time of the controlled vehicle generated by MPC (blue) and the actual one (red) together with the longitudinal velocity of the obstacles 1 (yellow) and 2 (purple). All units are in SI.

The lateral position of the controlled vehicle presented in figure 6.34, does not show any huge disturbance. On the other hand, in the vehicle velocity plot shown in figure 6.35, a small disturbance can be noticed around $t = 6\,s$. In this figure, the speed signals of Obstacle 1 (yellow) and Obstacle 2 (purple) are also plotted, with the first one being zero for the whole scenario.

Finally, in the fifth scenario, both obstacles move at a longitudinal speed of $v_1 = 0.2\,m/s$ and $v_2 = 0.9\,m/s$, respectively. The desired speed of controlled truck is $v_x = 0.5\,m/s$.

As it can be seen in the trajectory of controlled vehicle shown in figure 6.36, the test was stopped before the vehicle had finished the overtaking manoeuvre. Obstacle 2 (red) reached the end of the road and hence, the camera vision limit. When this happens, the camera sends a [0,0] position for the obstacle, leading to undesirable results. However, the first half of the manoeuvre trajectory can be studied. In this figure, it can also be seen that Obstacle 2 tends to move towards the centre of the road, due to its control not being accurate enough. Again, three different positions are plotted with rectangles at time instants: $t = 0$ s, $t = 3.5$ s and $t = 9.18$ s (final instant), for this scenario.

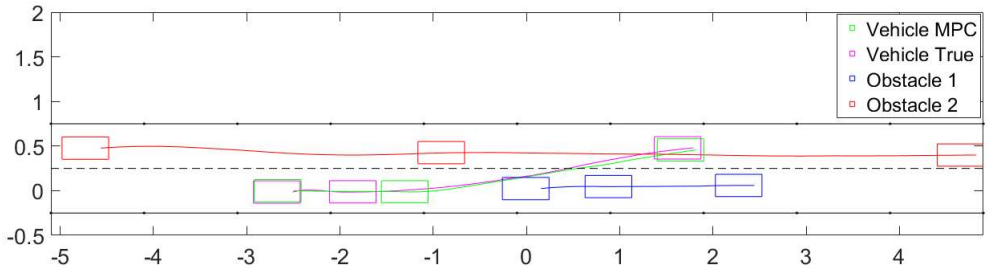


Figure 6.36: Trajectories (lateral vs longitudinal position) of the vehicle generated by the MPC (green) and the actual one (magenta), together with the obstacles 1 (blue) and 2 (red). All units are in SI.

Figure 6.37 shows a really good following of the lateral position trajectory by the controlled vehicle, with no disturbances.

In figure 6.38, the longitudinal velocity of the controlled truck together with the obstacles 1 (yellow) and 2 (purple) speed are plotted. The vehicle velocity has a slow acceleration this time, as it lets Obstacle 2 pass first. It can also be noticed that Obstacle 2 has a quite inaccurate speed control, as it oscillates around the desired value of 0.9 m/s, the peaks around $t = 5$ s and $t = 8$ s, could be due to disturbances from the cameras.

It can be concluded that the fifth scenario cannot be tested efficiently due to the limited size of the straight road in the truck lab. However, there are no signs in the first half of the trajectory that suggest an unsuccessful course of the scenario, as Obstacle 2 is far ahead with a higher speed and Obstacle 1 has been overtaken.

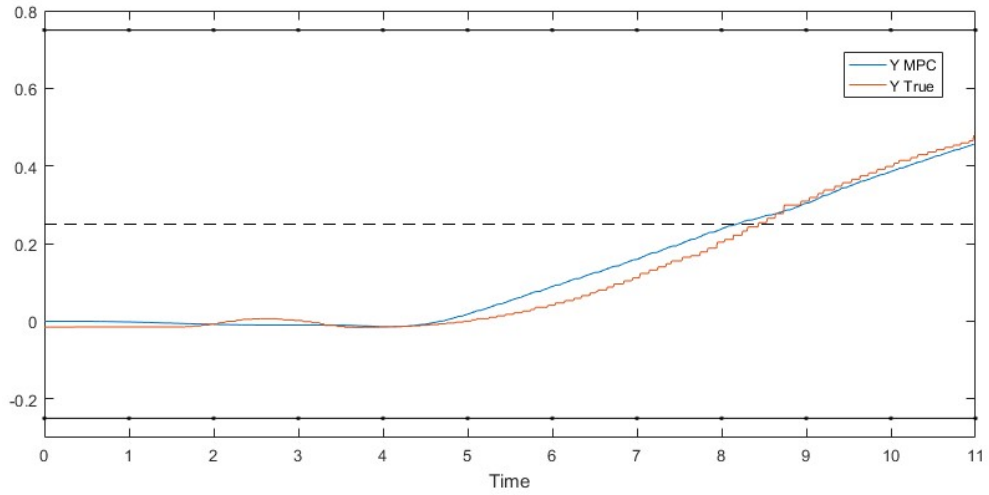


Figure 6.37: Truck lateral position over time generated by MPC (blue) and the actual one (red). All units are in SI.

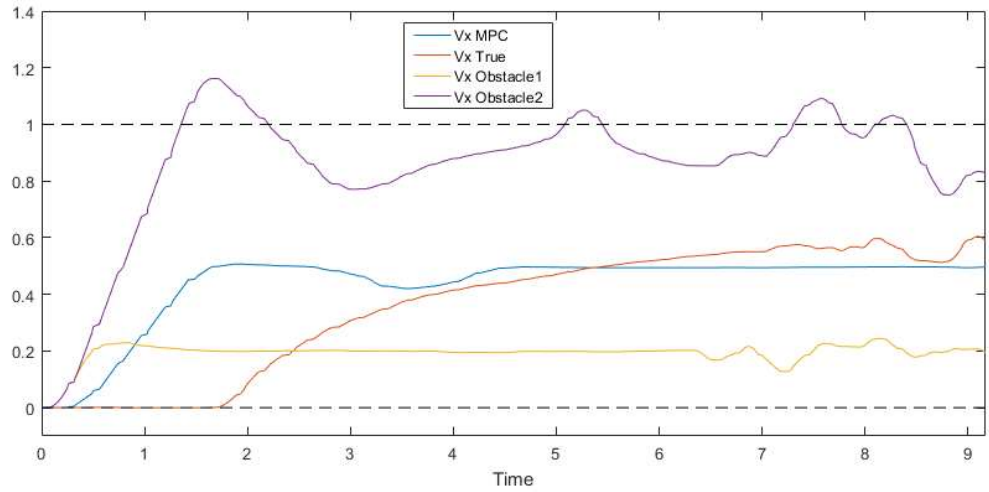


Figure 6.38: Longitudinal velocity over time of the controlled vehicle generated by MPC (blue) and the actual one (red) together with the longitudinal velocity of the obstacles 1 (yellow) and 2 (purple). All units are in SI.

7 Conclusions and Future work

7.1 Conclusions

The work presented in this thesis provides the development process, analysis and testing of a collision avoidance trajectory generator algorithm for highway environment using Model Predictive Control formulation.

To achieve this, first, the theory behind MPC formulation and its implementation methods have been studied, mainly through the book [9]. For the building of the MPC formulation, several plant models have been analysed. The final choice of using the point-mass kinematic vehicle model, due to its simplicity and the usage of a low-level controller, has been proved successful for both algorithms, the lane changing and the collision avoidance trajectory generator (this second with the corresponding extension taking into account the obstacles).

In order to make the path towards the main goal of this thesis less overwhelming, an intermediate MPC planner algorithm was developed. This algorithm, consisting of a lane change in a highway environment, was studied deeply through tuning and simulations. Then, the planner was linked with the low-level control, where signal conditioning was needed to implement because of having different sample times in each of the systems. After solving several code generating issues faced, the overall system was put to test in the Autonomous Driving Lab. Once tuned, the results obtained through the tests were satisfactory and they will be used in the development and tuning processes of the collision avoidance trajectory generator algorithm.

The first thing done during the development process of the collision avoidance algorithm was to choose an obstacle representation method. Again, several approaches where analysed and the decision of implementing the approach described in [11] as a starting point was considered the best.

After carrying out a tuning phase and several simulations, it was seen that the approach used needed some modifications to improve the algorithm response when facing obstacles. For this reason, several approaches where developed in a staged way, carrying out a tuning phase and several simulations after each huge change to prove the improvements made. Once the algorithm was showing satisfactory results

in the simulations, several scenarios were posed to test them experimentally in the lab using the small-scale trucks. All the scenarios were solved with the same MPC parameter configuration successfully. This gives a certain reliability to the approach developed as it proves low tuning dependency.

In the first tests, it was seen that the sample time of $T_s = 0.1\ s$ used for the planner in the lane changing algorithm was not enough for the collision avoidance trajectory generator algorithm and it was needed to increase to $T_s = 0.2\ s$. After tuning again the MPC parameters due to the sample time change, the scenarios posed were tested. All of them showed satisfactory results as the controlled vehicle avoided collision successfully with the obstacles. In the last scenario tested, the controlled vehicle could not complete the overtaking manoeuvre because the testbed showed not to have enough length of the straight road segment. Nonetheless, the test results indicated that if the testbed was larger the manoeuvre could had been finish successfully.

With the results obtained in this thesis, it can be concluded that it is possible to implement MPC based trajectory planers in a BeagleBone Black Board using an adequate sample time. Also, as it was expected, it could be proved that the MPC based lane changing algorithm developed has a higher computational cost than the polynomial based algorithm.

7.2 Future work

In this section, the next steps towards the improvement in performance and applicability range of the work presented in this thesis, are posed.

As it has been seen in the last test results presented in this thesis (chapter 6.4), the length of the straight road segment of the testbed was not enough to test one of the scenarios. Consequently, a major improvement that could be done is to extend the work of this thesis with an implementation for curved roads. This, would not only allow new scenarios to be tested in the Autonomous Driving Lab but also would increase the applicability range of the algorithm.

Another aspect that could be work on is the improvement of the MPC optimization iterative cycle in order to reduce the computation time required to execute this task and consequently, increase the overall performance of the algorithm.

Other obstacle representation strategies can be explored in order to develop new implementations. Then, benchmarking the resulting algorithms with the work presented in this thesis is an effective way toward the improvement in this field.

Finally, there are some issues faced during the development of this work that could not be solved in a definitive way, but only provisional. The cause of these issues is still unknown. Hence, an investigation could be done regarding the code generation, which was invalid to be used by the compiler straight away and manual modifications in the code were required previously, as it is explained in chapter 6.2. Also, as explained in chapter 6.3, the execution time logs are mixed between the tasks with different sample time, showing a spiking behaviour when plotted. Hence, an analysis of the execution time logs and their connexion with the different task sample times could be carried out, in order to obtain more reliable results.

8 References

[1] Carvalho, A., Gao, Y., Gray, A., Tseng, H. and Borrelli, F. (2013). Predictive Control of an Autonomous Ground Vehicle using an Iterative Linearization Approach. *IEEE Conference on Intelligent Transportation Systems*, pp.2335 - 2340.

[2] Chakravarthy, A. and Ghose, D. (1998). Obstacle avoidance in a dynamic environment: a collision cone approach. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28(5), pp.562-574.

[3] Eilbrecht, J. and Stursberg, O. (2017). Auction-based Cooperation of Autonomous Vehicle using Mixed-Integer Planning. *Proc. AAET Symposium*, pp.267-287.

[4] Emre Sancar, F., Fidan, B., Huissoon, J. and Waslander, S. (2014). MPC based collaborative adaptive cruise control with rear end collision avoidance. *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*.

[5] Eskandarian, A. (2012). *Handbook of intelligent vehicles*. London: Springer, pp.1271-1310.

[6] Falcone, P., Borrelli, F., Asgari, J., Tseng, H. and Hrovat, D. (2007). Predictive Active Steering Control for Autonomous Vehicle Systems. IEEE Transactions on Control Systems Technology, 15(3), pp.566-580.

[7] Ferrara, A. and Vecchio, C. (2006). Collision avoidance strategies and coordinated control of passenger vehicles. *Nonlinear Dynamics*, 49(4), pp.475-492.

[8] Katrakazas, C., Quddus, M., Chen, W. and Deka, L. (2015). Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60, pp.416-442.

[9] Maciejowski, J. (2002). *Predictive control*. Harlow, England: Prentice Hall.

[10] Mashadi, B., Majidi, M. (2014). Global optimal path planning of an autonomous vehicle for overtaking a moving obstacle. *Latin American Journal of Solids and Structures*, *11*(14), 2555-2572.

[11] Nilsson, J., Falcone, P., Ali, M. and Sjöberg, J. (2015). Receding horizon manoeuvre generation for automated highway driving. *Control Engineering Practice*, 41, pp.124-133.

[12] Nilsson, J., Falcone, P., Ali, M. and Sjöberg, J. (2013). Predictive manoeuvre generation for automated driving. *IEEE conference on intelligent transportation systems*, pp.418–423.

[13] Nilsson, J., Gao, Y., Carvalho, A. and Borrelli, F. (2014). Manoeuvre generation and control for automated highway driving. *IFAC Proceedings Volumes*, 47(3), pp.6301-6306.

[14] Schmid, C., and Biegler, L. T. (1994). Quadratic programming methods for reduced Hessian SQP. *Computers & Chemical Engineering*. Vol. 18, No. 9, pp. 817–832.

[15] Schouwenaars, T., DeMoor, B., Feron, E. and How, J. (2001). Mixed Integer Programming for Multi-Vehicle Path Planning. *European Control Conference,* pp. 2603-2608.

[16] Solis, B., Szymanki, J. and Littleton, A. (2017). The Race to 2021: The State of Autonomous Vehicles and a "Who's Who" of Industry Drivers. [online] Es.slideshare.net. Available at: https://es.slideshare.net/Altimeter/the-race-to-2021-the-state-of-autonomous-vehicles-and-a-whos-who-of-industry-drivers [Last Access 29 Sep. 2017].

[17] Varaiya, P. (1993). Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(2), pp.195-207.

[18] Williams, H. (2013). *Model building in mathematical programming*. Chichester, West Sussex: Wiley.

[19] Yoon, Y., Shin, J., Kim, H., Park, Y. and Sastry, S. (2009). Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Engineering Practice*, 17(7), pp.741-750.