DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di laurea in Ingegneria dell'Automazione

A NMPC based adversarial drivers for driving simulators

Laureando

Niccolò BRONCA

Relatore

Prof. Mattia BRUSCHETTA

Correlatore

Prof. Ruggero CARLI

Data di Laurea

July 18, 2022

ANNO ACCADEMICO 2021/2022

Contents

Sommario

L'obiettivo del progetto è realizzare un emulatore di guida avversario da inserire in un contesto di simulatori di guida, ovvero costruire uno strumento che basato su un Controllore Predittivo per Modelli Non lineari (NMPC) renda possibile l'interazione tra un veicolo avversario e un veicolo pilotato da un umano. A tal proposito inizialmente sarà considerato un singolo veicolo virtuale sul tracciato per verificare la miglior configurazione del controllore.
Successivamente due veicoli saranno pilotati dallo stesso controllore in un contesto simulato. In questo caso il modello interno del controllore comprenderà entrambi i veicoli, allo scopo di predire anche il comportamento del veicolo umano in caso di sorpasso. Infatti la gestione del sorpasso si rivelerà un fattore importante per descrivere la relazione tra i due veicoli.
In seguito un veicolo sarà controllato dal driver virtuale e l'altro avrà una traiettoria fissa predeterminata. Questa simulazione avrà come scopo quello di verificare la risposta del controllore al comportamento inatteso del secondo veicolo.
Per concludere il controllore che realizza questo obbiettivo dovrà tenere conto di una configurazione variabile. Infatti la forma del tracciato e le condizioni iniziali dei due veicoli risulteranno dei fattori determinanti per la loro interazione e la gestione del sorpasso su pista.

Abstract

The goal of the project is to realize an adversarial driving emulator in a driving simulator context. That is to build a tool that based on a Nonlinear Model Predictive Controller (NMPC) makes possible the interaction between an adversarial vehicle and a human-driven vehicle. For this purpose, initially a single virtual vehicle on the track will be considered to test the best controller configuration.

Subsequently, two vehicles will be driven by the same controller in a simulated context. In this case, the internal model of the controller includes both vehicles in order to also predict the behaviour of the human vehicle in case of overtaking. In fact, the overtaking management will prove to be an important factor in describing the relationship between the two vehicles.

Thereafter, one vehicle will be controlled by the virtual driver and the other will have a fixed predetermined trajectory. This simulation will aim to verify the controller's response to the unexpected behaviour of the second vehicle.

To conclude, the controller that realizes this goal will have to take into account a variable configuration. In fact, the shape of the track and the initial conditions for the two vehicles will turn out to be determining factors for their interaction and the overtaking management on the track.

Chapter 1

Introduction

In recent years, ADAS (Various Advanced Driver Assistance Systems) were introduced in commercial vehicles to manage simple driving tasks such as last-second crash prevention, semi-autonomous parking and autonomous cruise control. However, more challenging situations require more sophisticated control algorithms [21],[22]. In particular, for autonomous racing applications Model Predictive Control (MPC) has been shown to be a reliable and efficient control technique for self-driving ground vehicles. In fact, MPC is able to handle a multi-variable input-output systems (MIMO) subjected to physical and safety constraints and to states and inputs constraints. [18]. The key feature of MPC is to determine a control sequence at the initial time in order to optimize a cost function which represents the future behavior of the process to be controlled over a prediction horizon. Then, the entire optimization procedure is repeated for the next sampling istant [4].
Several implementations of the MPC and Nonlinear Model Predictive Controller (NMPC) techniques can be found in the literature for autonomous driving applications. For example, [21] developed a control system based on the NMPC formulation that aimed to achieve minimum-time driving for a race car respecting boundaries of the track.
Another application of NMPC is related to an autonomous vehicle system subjected to wind gusts and presented in [14]. In this case, a NMPC has been designed to stabilize the vehicle along a double lane change maneuver. In addiction, the controller was able to stabilize the vehicle up to wind gusts speed of $10m/s$.
Another NMPC implementation could be found in [3]. In this case, the controller was able to drive an autonomous ground vehicle, which was successful on tracking the lane centerline while avoiding obstacles collisions.
A NMPC based virtual driver has been proposed in [2], aiming at controlling a vehicle with high performance driving. In particular, the autonomous vehicle driven by NMPC proved to be effective for severe stability test at high velocity.
[17] describes a different autonomous driving scenario for race cars that is based on two different control formulations. The first controller was called HRHC (Hierarchical Receding Horizon Controller). It employed a two-level structure, consisting of a path planner and a nonlinear model predictive controller (NMPC) for tracking. The second controller was MPCC (Model

Predictive Contouring Controller), which combined both tasks in one nonlinear optimization problem (NLP) following the ideas of contouring control. Both the two strategies were adopted for the purpose of static obstacle avoidance, and the retrived results were compared. In particular, HRHC was not able to directly plan a path around all obstacles from the beginning. Thus, it avoided the first three cars, without predicting how to overtake the next two cars. On the other hand, MPCC planned the path around all obstacles before it even reached the first car due to its long prediction horizon.

The objective of this thesis concerns the development of a control system based on a Nonlinear Model Predictive Controller (NMPC) capable of behaving like an adversary in a simulated driving context. More specifically, the controller's task will be to manage a fully autonomous virtual vehicle interacting with a human-driven vehicle over a simulated race scenario.
This project differs from the last exposed applications since two virtual vehicles will compete each other in a race contest. Therefore, the obstacle is moving from the perspective of the faster vehicle that has to surpass. This last will be called *user vehicle*, the other one will be named as *adversarial vehicle*, which is the one that must deviate when overtaking occurs.
At the preliminary stage, it will be necessary that the controller is able to control a single vehicle in an autonomous racing scenario. Next, this control system will be configured to drive simultaneously a pair of vehicles competing on the track. The final step will be to test the behaviour of the NMPC system even in the case where *user vehicle* is not actually controlled by NMPC. In fact, it has completely independent dynamics of those predicted by the NMPC controller.
The last scenario will show that the controller is able to react differently through the fully autonomous vehicle (*adversarial vehicle*) depending on whether the vehicle with fixed dynamics is perceived by the system. In a future perspective, and by taking into account the limitations that will emerge, it will be possible to implement such a tool in a driving simulator in which the NMPC controller can behave as an adversary with respect to a vehicle driven by a real person.

The structure of this thesis is the following:

- **Chapter 2** introduces the model of the plant to be controlled. In other words, the vehicle dynamics are defined for a *bicycle* model.
- **Chapter 3** deals with the Nonlinear Model Predictive Controller (NMPC) adopted for this project and implemented through *MATMPC* toolbox. In particular, the model dynamics are taken into account in the Non Linear Programming Problem, which is solved by NMPC.
- **Chapter 4** defines the optimization problem arised from the previous Chapter. That is the *two vehicles problem*, which is formulated by considering the system dynamics, the cost function and particular inequality/equality constraints of the system.
- **Chapter 5** presents single vehicle simulations over the entire Paul Ricard circuit. The results will be evaluated according to three different reference strategies. Then, a benchmark

for the controller configuration will be retrived.

- **Chapter 6** deals with manegement of two vehicles in the same driving simulation. The problem will require to change the controller configuration because it depends on the shape of the track and the initial vehicle conditions. Consequently, the problem will be studied in different sections of the track, also altering the initial state of the system. In particular, *Paired departure* and *Delayed departure* tests will be carried out on single curves.

- **Chapter 7** illustrates two vehicles simulations under framework of *user vehicle* fixed trajectory. In other words, the fastest vehicle will have a fixed trajectory while the other one is still actually driven by the controller. In this perspective, the previous simulations will be repeated aiming to verify the controller's response to the unexpected behaviour of *user vehicle*.

- **Chapter 8** describes the final results retrived from the entire thesis and possible future developments.

Chapter 2

Bicycle model

The vehicle model is very important since it is the plant controlled by a Nonlinear Model Predictive Controller.
This Chapter discusses the reasons behind such model choice. Furthermore, the bicycle model dynamics will be defined after listing all the work assumptions. Then, a spatial reformulation of the state space model will be taken into account.
This model will be modified a bit in Chapter 4 since two vehicles will be considered.

2.1 Introduction and model choice

In literature many vehicle models can be found. There exists models that have 14 Degrees of freedom (Dof) [11], [24]. 6 Dof are related to lateral, longitudinal and vertical motion along x, y and z direction with roll, pitch and yaw angles. Then, 8 Dof are given by each wheel rotation and vertical movement. These models are very complex and they take into account several factors like load transfer, vertical dynamics, tire suspensions and each single wheel behaviour. Instead, a simpler model can be the *four wheels vehicles* [13], [10] which considers 3 Dof for CG (longitudinal, lateral and rotational movement around z axis) and 1 Dof for each wheel. In any case, the simplest one is the bicycle model [22].

In particular, for this project a 3 Dof *dynamic bicycle* model is considered, instead of a *kinematic bicycle* one. In fact, a *kinematic bicycle* model does not take into account the dynamics and actual lateral forces affecting the vehicle [15]. For this reason, the wheels velocities are directed exactly along their direction [1].
At high speeds, the assumption that the force vector of each wheel is in the direction of the wheel is no longer valid. As a consequence, a *dynamic bicycle* model results more suitable for the purpose of this thesis.

An important observation has to be clarified. In particular, it is considered a Nonlinear bicycle model referred to a car and not to a motorcycle. In fact, the two left and right front wheels of the car are represented by one single wheel at point A [19]. Similarly, the rear wheels

[1]This semplified model is suitable for urban driving applications like "stop-and-go" scenario [15]

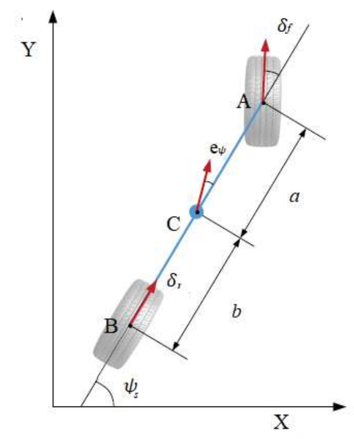


Figure 2.1: Bicycle model scheme

of the car are represented by one central rear wheel at point B (figure 2.1).
It is also possible to observe the steering angles for the front and rear wheels in the same image. They are represented by δ_f and δ_r. Moreover, it is assumed that only the front wheel can turn so that $\delta_r = 0°$.
Let the point C be the Center of Gravity (CG), then the distances from CG to the front and rear axles are called a and b.
The lateral distance of the car (defined as c) is not considered for the dynamics of the model even though this term will be important to define the lateral minimum distance between two vehicles in Chapter 4.
Addictional assumptions have to be exposed before introducing the model dynamics. In particular, the *cornering forces* generated by the tires are taken into account with aerodynamic effects. Instead, the load transfer, pitching and rolling effects and also the suspension system are not assumed.
To conclude, a trade-off between accuracy of the model and computational complexity has to be made [18]. In fact, on one hand assuming a more realistic model is good to get an accurate prediction trajectory, but on the other hand a sufficiently small computational burden is required to achieve a real-time feasibility.

2.2 Model dynamics

The vehicle has a planar motion so that three coordinates are necessary to describe the pose of the vehicle CG with respect to the Absolute reference frame: $[X\ Y\ \psi]^T$. $[X\ Y]^T$ describes the position of the car CG while the yaw angle ψ is related to its orientation.
$[x\ y]^T$ is assumed as the body frame coordinates attached to the bicycle CG, then it can be applied the *Newton's second law* for the motion along y axis:

$$m\, a_y = F_{y_f} + F_{y_r} \tag{2.1}$$

where a_y is the inertial acceleration along the y axis, and F_{y_f}, F_{y_r} are the lateral forces in the vehicle frame of the front and rear wheel. Addictionally, two acceleration terms contribute the overall acceleration a_y. Namely, the *centripetal* acceleration and the linear one along the y axis so that: $a_y = \ddot{y} + \dot{x}\,\dot{\psi}$. By substituting this relation in the previous equation, it can be obtained the following relation:

$$m\,\ddot{y} = -m\,\dot{x}\,\dot{\psi} + F_{y_f} + F_{y_r} \tag{2.2}$$

Similar considerations for the x axis can be made to get the acceleration expression. However, due to the fact that both a_x and δ_f will be the inputs of the system[2], then the resulting motion along the x axis is given by:

$$m\,\ddot{x} = m\,\dot{y}\,\dot{\psi} + F_{xf} + m\,a_x - F_x^d \tag{2.3}$$

In particular, F_{xr} is not considered since a_x is the control input of the longitudinal acceleration. Instead, the *centripetal* acceleration term has an opposite sign with respect to eq (2.2). Moreover from [12], the *drag force* depends on the frontal surface area of the car, the air density and the *drag coefficients* according to the following expression:

$$F_x^d = \frac{\rho}{2}\,C_{d_{long}}\,A_{d_{long}}\,\dot{x}^2 \tag{2.4}$$

Concerning to the angular acceleraton, it can be applied the *Principle of conservation of angular momentum*:

$$I_z\,\ddot{\psi} = a\,F_{y_f} - b\,F_{y_r} \tag{2.5}$$

By expressing all the previous equations with respect to the linear and angular accelerations, the continuous time bicycle model, which describes the planar motion, can be retrived:

$$\begin{aligned} \ddot{x} &= \dot{y}\,\dot{\psi} - \frac{F_x^d}{m} + \frac{1}{m}F_{xf} + a_x \\ \ddot{y} &= -\dot{x}\,\dot{\psi} + \frac{1}{m}(F_{y_f} + F_{y_r}) \\ \ddot{\psi} &= \frac{1}{I_z}(a\,F_{y_f} - b\,F_{y_r}) \end{aligned} \tag{2.6}$$

Taking into account figure 2.2, the forces of each wheel in vehicle frame are:

$$\begin{cases} F_{x_f} = -2\,F_{c_f} sin(\delta_f) \\ F_{y_f} = 2\,F_{c_f} cos(\delta_f) \\ F_{y_r} = 2\,F_{c_r} \end{cases} \tag{2.7}$$

From eq (2.7), the forces consider a factor of 2. This is the consequence given by the fact that the two frontal wheels of the car are merged in just one single frontal wheel, and the same happens for the rear wheel (Sec 2.1). Moreover, in eq (2.7), the second input of the system is

[2]In Chapter 4 the 1^{st} order derivatives of both a_x and δ_f will be considered as control inputs.

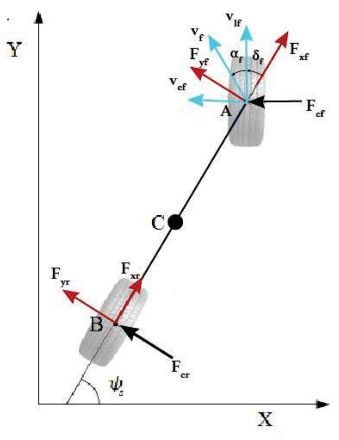


Figure 2.2: Longitudinal and lateral forces applied to the wheels

present. It is the steering angle δ_f of the frontal wheel (the only one that can turn).
Another important observation is that only the lateral forces of the tires F_{c_i} are considered. However, instead of computing them through the nonlinear Pacejka tire model (and a possible implementation is available in [16]), they are obtained through the following relation:

$$F_{c_i} = -C_{\alpha_i}\alpha_i \qquad \forall i = f, r \tag{2.8}$$

where C_{α_i} are the cornering stiffness coefficients which represent the slopes around the origins of the Pacejka functions. In fact, as reported in [20], they are evaluated as:

$$\begin{cases} C_{\alpha_i} = BCD \\ B = \frac{K_{y\alpha}}{CD+\xi} \end{cases} \tag{2.9}$$

In which it is possible to observe that B is the *stiffness factor*, C is the *shape factor* and D is the *peak factor*. In any case, the cornering stiffness coefficients can be approximated as: $C_{\alpha_i} \simeq K_{y_\alpha}$. Besides, the side slip angles of the wheels α_i are computed by the following equations:

$$\begin{aligned}
v_{x_f} &= \dot{x} \\
v_{y_f} &= \dot{y} + a\dot{\psi} \\
v_{x_r} &= \dot{x} \\
v_{y_r} &= \dot{y} - b\dot{\psi} \\
\alpha_f &= atan2\left(v_{y_f} cos(\delta_f) - v_{x_f} sin(\delta_f), v_{y_f} sin(\delta_f) + v_{x_f} cos(\delta_f)\right) \\
\alpha_r &= atan2\left(v_{y_r}, v_{x_r}\right)
\end{aligned} \tag{2.10}$$

To be precise, the *cornering forces* are proportional to the slip angles α_i if they are small. So that eq (2.8) is valid only in this case. In general, as said in [23], a *sigmoid function* can approximate the cornering forces computed by the *Pacejka tire* model. This is done because

Pacejka tire model considers several parameters. In particular, the lateral forces on the wheels are described by eq (2.11).

$$f_{y_i}^{sat} = A + \frac{K - A}{1 + exp(-BF_{ci})} \qquad \forall i = f, r \tag{2.11}$$

where A is the lower asyptote, K is the upper asyptote and B is the growth rate.
In this way, the logistic functions result very similar to those of Pacejka model (figure 2.3). In particular, there are three distinct phases: linear, transient and saturation. In the linear phase, the lateral friction forces are linearly dependent on the slip angles. This occurs for sideslip angle values less than 4°. In the next phase, the transient function is non linear and the forces reach their maximum values. The maximum lateral forces are between the static tire forces: $[-\mu\ F_z; \quad +\mu\ F_z]$. The last phase is a saturation phase, in which the cornering forces do not grow.

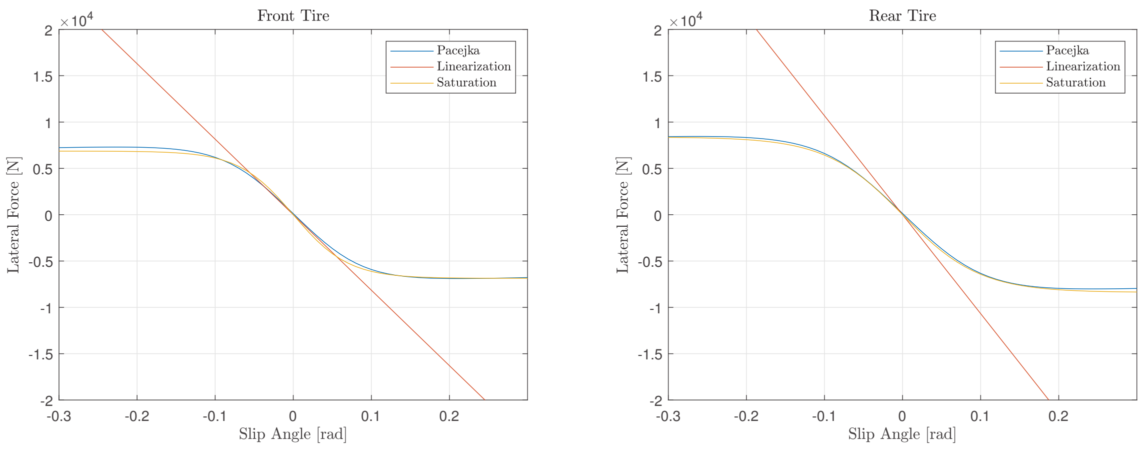


Figure 2.3: Cornering forces modeled by *logistic functions* w.r.t. Pacejka model ones

To conclude, the dynamics of the bicycle model are computed as:

$$\begin{aligned} \ddot{\psi} &= \frac{2}{I_z}(a\ F_{c_f}\ cos(\delta_f) - b\ F_{c_r}) \\ \ddot{y} &= -\dot{x}\ \dot{\psi} + \frac{2}{m}(F_{c_f}\ cos(\delta_f) + F_{c_r}) \\ \ddot{x} &= \dot{y}\ \dot{\psi} - \frac{F_d}{m} + \frac{2}{m}(-F_{c_f}\ sin(\delta_f)) + a_x \end{aligned} \tag{2.12}$$

Table 2.1 illustrates some parameters about the model.

Parameter	Value	
a	1.5064	$[m]$
b	1.2336	$[m]$
c	0.8422	$[m]$
m	$1.5543 \cdot 10^3$	$[kg]$
I_z	$2.1994 \cdot 10^3$	$[kg \cdot m^2]$

Table 2.1: Model parameters

2.3 Spatial reformulation of the system dynamics

As reported in [18], it is possible to allow the controller some freedom to choose the maximum speed. Therefore, a spatial reformulation is adopted for the system dynamics. In other words, eq (2.12), which describes the center of mass motion, will no longer depend on the time variable but by the arc lenght s. This quantity describes the relation between the actual pose of the vehicle CG: $[XY\psi]^T$ and its projection along the reference cuve σ: $[X_sY_s\psi_s]^T$. This concept is represented in figure 2.4.

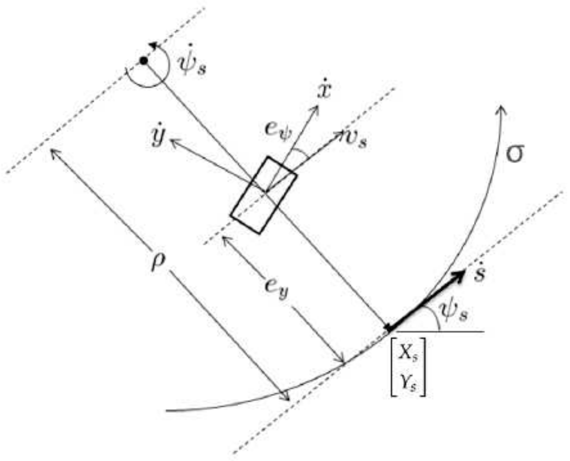


Figure 2.4: Spatial reformulation for bicycle model

In this way the tracking errors of the vehicle with respect to the reference are defined as:

$$\begin{cases} e_\psi = \psi - \psi_s \\ e_y = \pm \left\| \begin{bmatrix} X - X_s \\ Y - Y_s \end{bmatrix} \right\|_2 \end{cases} \tag{2.13}$$

These errors are referred to the vehicle direction and the lateral distance with respect to the road reference. In particular, if $e_y > 0$ then the car is on the left side otherwise it is on the right.

Moreover, by taking into account the previous figure, the speed along the reference trajectory is computed as:

$$\dot{s} = \frac{ds}{dt} = \rho\,\dot{\psi}_s = \frac{\rho\, v_s}{\rho - e_y} \tag{2.14}$$

where $v_s = \dot{x}\, cos(e_\psi) - \dot{y}\, sin(e_\psi)$ and the curvature is defined as: $k = \rho^{-1}$. Thus, the dynamics of eq (2.13) are computed as:

$$\begin{cases} \dot{e}_\psi = \dot{\psi} - k\,\dot{s} \\ \dot{e}_y = \dot{x}\, sin(e_\psi) + \dot{y}\, cos(e_\psi) \end{cases} \tag{2.15}$$

Drawing inspiration from the procedure developed in [18], [6], [2], it is possible to express the vehicle dynamics with respect to a space formulation instead of a time one. Considering as state vector $\xi = [\dot{x}\,\dot{y}\,\dot{\psi}\,\dot{e}_\psi\,\dot{e}_y]^T$, then its derivative with respect to the arc length s is obtained by using the *chain rule*:

$$\xi' = \frac{d\xi}{ds} = \frac{d\xi}{dt}\frac{dt}{ds} = \frac{d\xi}{dt}\frac{1}{\dot{s}} = \frac{\dot{\xi}}{\dot{s}} \tag{2.16}$$

Chapter 3

NMPC implemented in MATMPC toolbox

In this Chapter the fast *Nonlinear Model Predictive Controller* (NMPC) adopted for this project will be described. It was developed and well presented in Yutao Chen Ph.D. thesis [4]. Addictionally NMPC was realized in *Matlab* language, resulting in *MATMPC* open source toolbox which is available at [5]. Moreover, it is designed to facilitate modelling controller design and simulation for a wide class of NMPC applications [6].
The following sections will deal with the Non Linear Optimization Problem (NLP) which is solved by this controller. Then, NLP will be formulated starting from a parametrization of the Optimal Control Problem (OCP) and a 1^{st} order necessary condition of optimality will be exposed.
To solve NLP, a *Sequential Quadratic Programming* (SQP) algorithm will be described. It consists to solve a series of *Quadratic Programming Problems* (QPs). In fact, QP will be obtained from a local quadratic approximation of NLP. Hence, QP can be solved iteratively (according to SQP algorithm) or through a *Real-Time iteration* scheme. Its solutions will be used to update those of NLP, according to a globalization strategy.

3.1 Nonlinear Model Predictive Control

A breaf introduction to *Model Predictive Control* (MPC) is presented in this section since a fast *Nonlinear Model Predictive Controller* (NMPC) is implemented in this project.
As cited in [4]: "MPC is an advanced computer control algorithm that exploits process models and constraints with the power of optimization. The key feature of MPC is to determine a control sequence at time t_0 in order to optimize a cost function representing the future behavior of the controlled process over a prediction horizon $[t_0, t_f]$. However, only the first control input is forwarded to the plant and the optimization procedure is repeated at the next sampling instant. As a result, MPC refers to a control strategy that solves a sequence of optimization problems on-online (when system is running), with respect to the latest state measurement of the plant".
Thus MPC, as reported in [1], also refered to as moving horizon control. In fact, the first part of the input signal is implemented untill new measurements become available. Based on the new information, the OCP is solved again and the whole procedure is repeated.

NMPC refers to MPC exploiting nonlinear plant models. In fact, by capturing the nonlinear behaviours of plants, NMPC can ensure better control performance. However, the main issue consists in solving the optimization problems within a real-time restriction. This is due to the fact that it is very hard to solve at each sampling istant, a *Nonlinear Optimal Control Problem* (OCP) respecting states, controls and safety constraints.
Figure 3.1 shows the building blocks of MPC.

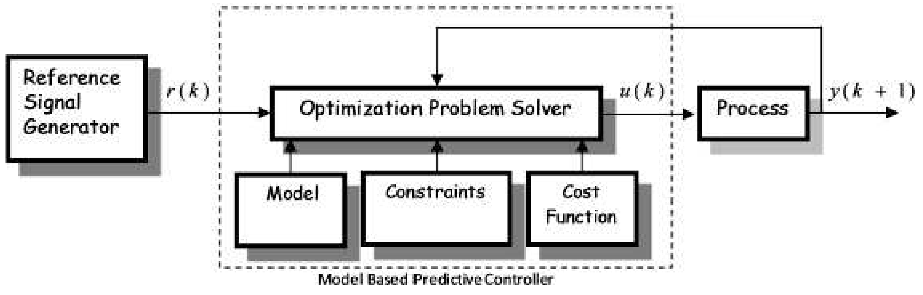

Reference Signal Generator
$r(k)$
Optimization Problem Solver
$u(k)$
Process
$y(k+1)$
Model
Constraints
Cost Function
Model Based Predictive Controller

Figure 3.1: A building block of MPC

Some key characteristics and properties of NMPC could be exposed:

- NMPC allows the direct use of nonlinear state space models for prediction.
- NMPC allows the explicit consideration of state and input constraints.
- In NMPC a specified performance criteria is minimized on-line.
- In NMPC the predicted behaviour is in general different from the closed loop one.
- For several NMPC applications, real-time feasibility is necessary.
- To perform the prediction, the system states must be measured or estimated.

3.2 From OCP to NLP

The *Nonlinear Programming Problem* (NLP), which is solved by NMPC, comes from a discretization of the *Optimal Control Problem* (OCP).
Considering a non linear plant referred to the 1^{st} order system of Ordinary Differential Equations (ODE):

$$\dot{x}(t) = f(t, x(t), u(t); p) \tag{3.1}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$ and $p \in \mathbb{R}^{n_p}$ are rispectively the plant states, controls and parameter variables. Then, starting with an initial condition $x(0) = \hat{x_0}$ and given a control trajectory $u(t)$, eq (3.1) is an Initial Value Problem (IVP). Moreover the existence of an unique solution $x(t)$

over the time interval $t \in [t_0, t_f]$ is guaranteed by the Picard's Existence theorem [1].

NMPC requires to solve an *Optimal Control Problem* (OCP) at each sampling istant, taking also into account the dynamics of the nonlinear plant (3.1). Therefore lets define the OCP problem as:

$$\begin{aligned} \min_{x(t),u(t)} & \int_{t_0}^{t_f} \phi(t, x(t), u(t); p)\, dt + \Phi(x(t_f)) \\ s.t.\, & x(t_0) = \hat{x}_0, \\ & \dot{x}(t) = f(t, x(t), u(t); p),\ \forall t \in [t_0, t_f], \\ & r(x(t), u(t); p) \leq 0,\ \forall t \in [t_0, t_f], \\ & l(x(t_f)) \leq 0 \end{aligned} \tag{3.2}$$

where ϕ and Φ are the optimization objectives. r is the path constraint which includes also states and inputs constraints while l refers to the boundary conditions.
To summarize, given an initial state measurement $\hat{x}_0$, then an optimal solution trajectory $(x_{opt}(t), u_{opt}(t))$ (defined in the prediction horizon $[t_0, t_f]$) could be computed by solving OCP (3.2).

In order to solve OCP (3.2), many kind of methods can be applied. Essentially there exist three bigger families: *Dynamic Programming*, *Indirect methods* and *Direct methods*. The last ones are widely used in *Non Linear Model Predictive Control* (NMPC) applications even though they require to satisfy the 1^{st} order necessary condition of optimality for OCP. This condition will be exposed in Sec 3.3.
MATMPC toolbox adopts a *Direct method* which consists to perform a finite dimensional parametrization of the continuous time OCP (3.2). To achieve this feature, *Multiple shooting* technique is implented in order to obtain a discretization version of the OCP, which is called *Non Linear Programming Problem* (NLP). So that the NLP can be solved by state of art numeric solvers.

The *Multiple shooting* approach consists to divide the time interval $[t_0, t_f]$ into N intervals $[t_k, t_{k+1}]$ $\forall k = 0, 1..., N-1$ in such a way the control trajectory is parametrized through a piece wise costant representation:

$$u(t) = u_k \quad \forall t \in [t_k, t_{k+1}] \tag{3.3}$$

Instead the states are evaluated on the time grid points t_k as:

$$x_k(t_k) = x_k \quad \forall k = 0, 1..., N-1 \tag{3.4}$$

Moreover x_k is called shooting point and it is the initial condition for the next Initial Value Problem (IVP) of type:

[1]if $f(t, x(t), u(t))$ is lipschitz continuous in $x(t)$, $u(t)$ and continuous in t then $\exists$ an unic solution of the IVP

$$\dot{x}_k(t) = f(t, x_k(t), u_k(t); p), \quad x_k(t_k) = x_k \quad \forall t \in [t_k, t_{k+1}] \tag{3.5}$$

So that the dynamic constraint of the OCP becomes:

$$x_{k+1} = \Xi_k(t_k, x_k, u_k; p) \quad \forall k = 0, 1..., N-1 \tag{3.6}$$

Where Ξ_k is the integration operator that solves the IVP (3.5) and returns the solution at the next grid point t_{k+1}.
Also the path constraint is parametrized on the discrete time points as:

$$r(x_k, u_k; p) \leq 0 \quad \forall k = 0, 1..., N-1 \tag{3.7}$$

While the optimization objective of the OCP could be approximated using a discrete sum:

$$\sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} \phi(t_k, x_k, u_k; p)\, dt + \Phi(x_N) \approx \sum_{k=0}^{N-1} \phi(t_k, x_k, u_k; p) + \Phi(x_N) \tag{3.8}$$

In particular, *MATMPC* considers a cost function which is defined by a linear sum of quadratic terms:

$$\frac{1}{2}\sum_{k=0}^{N-1} ||h(x_k, u_k) - h_{ref}^k||_{W_k}^2 + \frac{1}{2}||h_N(x_N) - h_{ref}^N||_{W_N}^2 \tag{3.9}$$

Finally, given the previous parametrizations for states, controls, equality and inequality constraints, then the *Non Linear Programming Problem* (NLP) is formulated:

$$\begin{aligned} \min_{x_k, u_k} & \frac{1}{2}\sum_{k=0}^{N-1} ||h(x_k, u_k) - h_{ref}^k||_{W_k}^2 + \frac{1}{2}||h_N(x_N) - h_{ref}^N||_{W_N}^2 \\ s.t.\, & x_0 = \hat{x}_0, \\ & x_{k+1} = \Xi_k(t_k, x_k, u_k; p), \quad \forall k = 0, 1, \ldots, N-1, \\ & r(x_k, u_k; p) \leq 0 \quad \forall k = 0, 1, \ldots, N-1, \\ & l(x_N) \leq 0 \end{aligned} \tag{3.10}$$

where

$$\begin{aligned} \mathbf{x} &= \left[x_0^\top, x_1^\top, \ldots, x_N^\top\right]^\top, \\ \mathbf{u} &= \left[u_0^\top, u_1^\top, \ldots, u_{N-1}^\top\right]^\top \end{aligned} \tag{3.11}$$

are the discrete states and controls variables. While h and h_N are the objective functions whose definition will be discussed in Chapter 4.

3.3 Optimality Condition and NLP features

For completeness, the Karush-Kuhn-Tucker (KKT) 1^{st} order necessary condition for optimality of NLP (3.10) have to be expressed. This is quite important since an optimality check of the retrived solution is required for *Sequential Quadratic Programming* (SQP) algorithm.
As said by *Yutao Chen* Ph.D. thesis (see [4], *pp. 18-19*), it is considered the following compact formulation of the NLP (3.10):

$$\begin{aligned} &\min_z a(z) \\ &s.t.\, b(z) = 0 \\ &\quad c(z) \leq 0 \end{aligned} \tag{3.12}$$

where $z = \left[z_0^\top, z_1^\top, \dots, z_{N-1}^\top, x_N^\top\right]^\top \in \mathbb{R}^{n_z}$ with $z_k = \left[x_k^\top, u_k^\top\right]^\top \in \mathbb{R}^{n_x+n_u} \quad \forall k = 0, 1..., N-1$. Instead the equality and the inequality constraints take the form:

$$b(z) = \begin{bmatrix} \hat{x}_0 - x_0 \\ \Xi_0(t_0, x_0, u_0; p) - x_1 \\ \dots \\ \Xi_{N-1}(t_{N-1}, x_{N-1}, u_{N-1}; p) - x_N \end{bmatrix} \in \mathbb{R}^{n_b} \qquad c(z) = \begin{bmatrix} r(x_0, u_0; p) \\ \dots \\ r(x_{N-1}, u_{N-1}; p) \\ l(x_N) \end{bmatrix} \in \mathbb{R}^{n_c} \tag{3.13}$$

Assuming $z^* \in \mathbb{R}^{n_z}$ be a local minimizer of (3.12), there exist KKT multipliers $\lambda^* \in \mathbb{R}^{n_b}$, $\mu^* \in \mathbb{R}^{n_c}$ corresponding to the constraints $b(z) = 0$ and $c(z) \leq 0$ respectively, such that the following conditions are satisfied:

1. Stationarity condition: $\nabla_z \mathcal{L}(z^*, \lambda^*, \mu^*) = \nabla_z a(z^*) + \nabla_z b(z^*)^\top \lambda^* + \nabla_z c(z^*)^\top \mu^* = 0$

2. Primal feasibility: $b(z^*) = 0$ and $c(z^*) \leq 0$

3. Dual feasibility: $\mu^* \geq 0$

4. Complementary slackness: $\mu_k^* c_k(z^*) = 0 \quad \forall k = 0, 1, \dots, n_c$

The optimal primal and dual solution is the KKT point: $y^* = \left[z^{*\top}, \lambda^{*\top}, \mu^{*\top}\right]^\top$.

In *MATMPC*, it is exploited the boundaries function l and the path constraint r, in such away it is more clear the boundary conditions both for states and controls. Meaning that NLP (3.10) is rewritten as:

$$
\begin{aligned}
\min_{x_k,u_k} \ & \frac{1}{2}\sum_{k=0}^{N-1} d_k(h(x_k,u_k) - h^k_{ref})_{W_k} + \frac{1}{2} d_N(h_N(x_N) - h^N_{ref})_{W_N} \\
s.t.\ & 0 = x_0 - \hat{x}_0, \\
& 0 = x_{k+1} - \Xi_k(x_k,u_k),\ k = 0,1,\dots,N-1, \\
& \underline{x}_k \leq x_k \leq \overline{x}_k,\ k = 0,1,\dots,N, \\
& \underline{u}_k \leq u_k \leq \overline{u}_k,\ k = 0,1,\dots,N-1, \\
& \underline{r}_k \leq r_k(x_k,u_k) \leq \overline{r}_k,\ k = 0,1,\dots,N-1, \\
& \underline{r}_N \leq r_N(x_N) \leq \overline{r}_N,
\end{aligned}
\tag{3.14}
$$

where $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \longrightarrow \mathbb{R}^{n_y}, h_N : \mathbb{R}^{n_x} \longrightarrow \mathbb{R}^{n_{y_N}}$ are vector functions of state and control (x,u), with corresponding references h^k_{ref} and h^N_{ref}. Note that h, h_N can be nonlinear and nonconvex. The outer objective functions $d : \mathbb{R}^{n_y} \longrightarrow \mathbb{R}, d_N : \mathbb{R}^{n_{y_N}} \longrightarrow \mathbb{R}$ are assumed convex, e.g. linear sum or quadratic. W_k, W_N are weights for each term in d for stage k. $\hat{x}_0$ is the measurement of the current state. The function $r(x_k,u_k) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \longrightarrow \mathbb{R}^{n_c}$ and $r_N(x_N) : \mathbb{R}^{n_x} \longrightarrow \mathbb{R}^{n_{c_N}}$ can be linear or nonlinear, with lower and upper bound $\underline{r}_k, \overline{r}_k$. $\Xi_k(x_k,u_k)$, as said before, solves IVP (3.5) and returns the next grid point.

The formulation of NLP (3.14) presents the following features:

- An initial guess is required to solve NLP (3.10) over the entire prediction horizon $[t_0, t_f]$.
- The resulting NLP is numerically stable even if the model dynamics (3.1) is unstable as proved in [4], *pp.62-65*.
- There exist many types of software which solves both the IVP (3.6) and the subproblems araised from NLP (3.10) (in fact, NLP is usually reformulated as a *Quadratic Problem*). In particular, for this project, a numeric integrator called Runge Kutta of 4^{th} order is adopted (whose implementation is discussed in [4], *p.103*) to solve (3.6). Concerning to (3.10), *MATMPC* toolbox considers two possible approaches. The first one is the SQP algorithm while the second one consits in RTI scheme. Both them will be presented in Sec 3.4.
- Since *MATMPC* realizes this NMPC controller, there is no need to redesign it. Moreover it can be used for controlling different kind of plants depending on the application.
  For this project, the automotive application is related to develop a NMPC based adversarial driver for driving simulators. Additional details for this problem formulation will be taken into account in Chapter 4.

3.4 Sequential Quadratic Programming algorithm

A *Sequential Quadratic Programming* (SQP) algorithm solves NLP (3.14) iteratively by using a local quadratic approximation of the objective and linearized constraints at each iteration.

However, for this project, it will be considered a *Real Time Iteration scheme* as in [6]. The reasonings are concerned to real time feasibility. Therefore, a suboptimal solution will be computed by using a fast NMPC algorithm. In fact, at the i^{th} sampling istant, RTI performs only one SQP iteration to solve NLP.
At iteration i, given the initial trajectory $(\mathbf{x}^i, \mathbf{u}^i)$, NLP (3.14) is linearized and resulted in the following *Quadratic programming* (QP) problem:

$$\begin{aligned}
\min_{\Delta\mathbf{x},\Delta\mathbf{u}} \quad & \sum_{k=0}^{N-1}\left(\frac{1}{2}\begin{bmatrix}\Delta x_k\\ \Delta u_k\end{bmatrix}^\top \begin{bmatrix}Q_k^i & S_k^i\\ S_k^{i\top} & R_k^i\end{bmatrix}\begin{bmatrix}\Delta x_k\\ \Delta u_k\end{bmatrix} + \begin{bmatrix}g_{x_k}^i\\ g_{u_k}^i\end{bmatrix}^\top \begin{bmatrix}\Delta x_k\\ \Delta u_k\end{bmatrix}\right) + \frac{1}{2}\Delta x_N^\top Q_N^i \Delta x_N + g_{x_N}^{i\top}\Delta x_N \\
s.t. \quad & \Delta x_0 = \hat{x}_0 - x_0, \\
& \Delta x_{k+1} = A_k^i \Delta x_k + B_k^i \Delta u_k + a_k^i, \; k = 0, \ldots, N-1 \\
& \underline{x}_k - x_k^i \leq \Delta x_k \leq \overline{x}_k - x_k^i, \; k = 1, \ldots, N \\
& \underline{u}_k - u_k^i \leq \Delta u_k \leq \overline{u}_k - u_k^i, \; k = 0, \ldots, N-1 \\
& \underline{c}_k^i \leq C_k^i \Delta x_k + D_k^i \Delta u_k \leq \overline{c}_k^i, \; k = 0, \ldots, N-1 \\
& \underline{c}_n - c_N^i \leq C_N^i \Delta x_N \leq \overline{c}_n - c_N^i,
\end{aligned} \tag{3.15}$$

where

$$\begin{aligned}
\Delta\mathbf{x} &= \mathbf{x} - \mathbf{x}^i, \\
\Delta\mathbf{u} &= \mathbf{u} - \mathbf{u}^i
\end{aligned} \tag{3.16}$$

and

$$\begin{aligned}
& Q_k^i = \frac{\partial (d^i)^2}{\partial^2 x_k}, \quad S_k^i = \frac{\partial (d^i)^2}{\partial x_k \partial u_k}, \quad R_k^i = \frac{\partial (d^i)^2}{\partial^2 u_k}, \quad Q_N^i = \frac{\partial (d_N^i)^2}{\partial^2 x_N} \\
& g_{x_k}^i = \frac{\partial d^i}{\partial x_k}, \quad g_{u_k}^i = \frac{\partial d^i}{\partial u_k}, \quad g_{x_N}^i = \frac{\partial d_N^i}{\partial x_N}, \\
& A_k^i = \frac{\partial \Xi_k}{\partial x_k}, \quad B_k^i = \frac{\partial \Xi_k}{\partial u_k}, \quad a_k^i = \Xi(x_k^i, u_k^i) - x_{k+1}^i, \\
& C_k^i = \frac{\partial r_k}{\partial x_k}, \quad D_k^i = \frac{\partial r_k}{\partial u_k}, \quad C_N^i = \frac{\partial r_N}{\partial x_N}, \\
& \overline{c}_k^i = \overline{r}_k - r_k(x_k^i, u_k^i), \quad \underline{c}_k^i = \underline{r}_k - r_k(x_k^i, u_k^i), \\
& \overline{c}_N^i = \overline{r}_N - r_N(x_N^i), \quad \underline{c}_N^i = \underline{r}_N - r_N(x_N^i)
\end{aligned} \tag{3.17}$$

The Hessian matrices Q_k, S_k, R_k can be approximated by *Gauss-Newton* (GN) method when NLP (3.14) has a least square cost function. To be more precise, only the 1st order derivatives for the objective function are considered:

$$H_k^i = \begin{bmatrix}Q_k^i & S_k^i\\ S_k^{i\top} & R_k^i\end{bmatrix} = \frac{\partial d^i}{\partial (x_k, u_k)}^\top \frac{\partial d^i}{\partial (x_k, u_k)} \tag{3.18}$$

Then, a state of art solver *High Performance Interior Point Method* (HPIPM) is adopted to get the solution of QP (3.15). This HPIPM state-of-art solver was developed by *Gianluca Frison* and well presented in [9] while the open-source toolbox is available in [8].

Once that the primal solutions are computed: $(\Delta\mathbf{x}^{i^*}, \Delta\mathbf{u}^{i^*})$, then they can be used to update the solution of NLP (3.14) through the following relations:

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \alpha^i \Delta\mathbf{x}^{i^*},\ \mathbf{u}^{i+1} = \mathbf{u}^i + \alpha^i \Delta\mathbf{u}^{i^*}, \tag{3.19}$$

Adictionally, *Real-Time Iteration* scheme considers a full Newton step about $\alpha^i = 1$.

3.5 MATMPC

The following figure, better summarizes what exposed till now. In particular how NMPC is implemented in *Matlab* code, resulting in *MATMPC* open-source package.

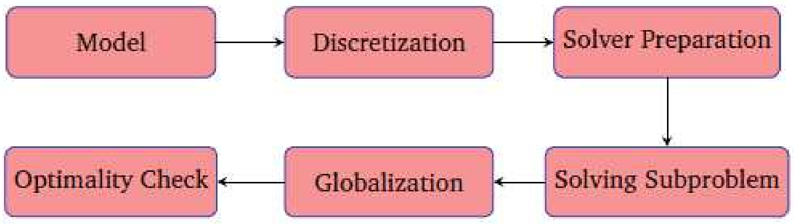


Figure 3.2: Structure of *MATMPC*

First of all the model is defined in a continuous time fashion through *CasAdi* language. Then, a spatial reformulation for the system dynamics is done.
The discretization of OCP is performed via *Multiple Shooting* technique so that the NLP (3.14) is obtained. During this parametrization, an Explicit Runge Kutta numeric integrator of the 4^{th} order is employed.
Furthermore, a solver preparation phase is implemented in *MATMPC*. Indeed, once that all ingredients for QP problem are ready (after performing a local quadratic approximation of the NLP), options are required for the specific solver. For this project, these settings are referred to a no condensed QP problem.
Regarding to get the solution of the subproblem, an *High Performance Interior Point Method* (*HPIPM*) solver is used.
Globalization strategy are adopted in order to update the solution of NLP (3.14) via eq (3.19). In particular, the step lenght α_i could be either computed by implementing a *line search* algorithm or by considering a *Real Time Iteration* (RTI) scheme. Since a real time restriction is important for this project, a suboptimal solution could be computed by terminating the SQP iteration early before convergence is achieved. For RTI approach, it is sufficient to use only one iteration with a full Newton step of $\alpha = 1$.
The optimality check is not taken into account for RTI approach since the solutions are suboptimal. On contrary, for SQP algorithm, the optimality check is required. This means that the KKT point must satisfy the 1^{st} order necessary condition of optimality, well exposed in Sec 3.3.

Chapter 4

Two vehicles problem

The purpose of this Chapter is to define all the quantities of NLP which refers to both *two vehicles problem* (Sec 4.1) and *single vehicle problem* (Sec 4.3). In fact, all the terms that define the optimization problem are quite important for the next Chapaters. In particular, the *single vehicle problem* will be dealt in Chapter 5. Then, the competition between two vehicles will be addressed and the overtaking manegement will be performed in Chapters 6 and 7.

4.1 NLP for two vehicles

From Chapter 3, the following NLP is considered:

$$\begin{aligned}
\min_{x_k,u_k} \ & \frac{1}{2}\sum_{k=0}^{N-1} d_k(h(x_k,u_k) - h^k_{ref})_{W_k} + \frac{1}{2} d_N(h_N(x_N) - h^N_{ref})_{W_N} \\
s.t.\ & 0 = x_0 - \hat{x}_0, \\
& 0 = x_{k+1} - \Xi_k(x_k,u_k),\ k = 0,1,\dots,N-1, \\
& \underline{x}_k \le x_k \le \overline{x}_k,\ k = 0,1,\dots,N, \\
& \underline{u}_k \le u_k \le \overline{u}_k,\ k = 0,1,\dots,N-1, \\
& \underline{r}_k \le r_k(x_k,u_k) \le \overline{r}_k,\ k = 0,1,\dots,N-1, \\
& \underline{r}_N \le r_N(x_N) \le \overline{r}_N,
\end{aligned} \tag{4.1}$$

where each term was explained in Sec 3.3.
Recall that the decision variables can be written in the compact form as:

$$\begin{aligned}
\mathbf{x} &= \left[x_0^\top, x_1^\top, \dots, x_N^\top\right]^\top, \\
\mathbf{u} &= \left[u_0^\top, u_1^\top, \dots, u_{N-1}^\top\right]^\top
\end{aligned} \tag{4.2}$$

The states and controls are defined by evaluating a single stage k:

$$x_k = \left[\dot{x}_1, \dot{y}_1, \dot{\psi}_1\, e_{\psi_1}, e_{y_1}, \delta_{f_1}, a_{x1}, t_1, \dot{x}_2, \dot{y}_2, \dot{\psi}_2\, e_{\psi_2}, e_{y_2}, \delta_{f_2}, a_{x2}, t_2\right]^\top, \tag{4.3}$$

$$u_k = \left[\dot{\delta}_{f_1}, \dot{a}_{x1}, \epsilon_{y_1}, \epsilon_{g_1} \dot{\delta}_{f_2}, \dot{a}_{x2}, \epsilon_{y_2}, \epsilon_{g_2}\right]^\top \tag{4.4}$$

where the quantities $*_1$ and $*_2$ are referred to the first and to the second vehicle respectively. In particular, the vehicles velocities, the tracking errors, the steering angles of the frontal wheels, the longitudinal accelerations and the time variables are considered as state variables.
Notably, eq (4.4) presents $\dot{\delta_{f_i}}$ and $\dot{a_{xi}}$ $\forall i = 1, 2$ in order to modify the smoothness of inputs at will. In other words, the derivative of the steering angle of the front wheels is needed to reduce the effects of the vehicle sway. The derivative of longitudinal acceleration improves the regularity of acceleration/braking.
Some slack variables are considered as controls and they impact on the general constraint definition (eq (4.13)).

Regarding to the system dynamics, it will be formulated with respect to the independent variable s (space traveled along the curve) instead of the time one:

$$x'(s) = \frac{\dot{x}(t)}{\dot{s}}$$

Then, this relation is referred to the equality constraint:

$$x_{k+1} = \Xi_k(x_k, u_k), \ k = 0, 1, \ldots, N-1$$

since the entire OCP problem was discretized (through *Multiple shooting* technique) obtaining NLP (4.1). Moreover, the integration done with respect to the space allows to define the time within the system. Therefore, the initial time variable (def 4.3) can be used to make vehicles pass at the same point on the track at different time instants.

The cost function defines the values that the solver will attempt to minimize by solving the optimization problem. Ideally, the objective function can be defined only for the lap time. Unfortunately, this is not possible because there are many other factors that are representative for the actual behaviour of the vehicles. The inclusion of these elements, allows the solver to avoid unrobust and particularly vibrant behaviours from a numerical point of view. In other words, the regularization of the vehicle behavior is a very important factor that has to be taken into account.
The cost function is defined from the difference between the predicted values and the relative references. In particular, it is defined for each shooting point:

$$d_k(x_k, u_k) = \frac{1}{2}(h(x_k, u_k) - h^k_{ref})^T \cdot W_k \cdot (h(x_k, u_k) - h^k_{ref}) \tag{4.5}$$

and for the final state:

$$d_k(x_N, u_N) = \frac{1}{2}(h_N(x_N) - h^N_{ref})^T \cdot W_N \cdot (h_N(x_N) - h^N_{ref}) \tag{4.6}$$

It is important to say that the cost function depends on the specific type of reference. In particular, three different strategies are proposed to describe the reference errors (lateral error distance and direction error of the vehicles with respect to a reference trajectory) which have a significant

impact on performance. In fact, it will be seen that the inclusion of the tracking errors in the cost functions will prove to be a decisive factor in evaluating numerical robustness.
The first strategy is to consider no tracking errors reference. Indeed, the vehicle can be forced to stay within the road limits (eq (4.17)) so that the controller minimizes the lap time (also the other quantities) without taking into account the reference errors. Therefore, for the *No tracking errors reference* strategy, the vectors $h(x_k, u_k)$ and $h_N(x_N)$ are defined as:

$$h(x_k, u_k) = \left[t_1, \dot{\delta}_{f1}, \dot{a}_{x1}, \epsilon_{y_1}, \epsilon_{g_1}, t_2, \dot{\delta}_{f2}, \dot{a}_{x2}, \epsilon_{y_2}, \epsilon_{g_2}\right]^\top \tag{4.7}$$

$$h_N(x_N) = [t_{1N}, t_{2N}]^\top \tag{4.8}$$

while the reference vectors are $h^k_{ref} = \mathbf{0}$ and $h^N_{ref} = \mathbf{0}$ according to the proper dimension.
Observing that in eq (4.7) to minimize: $\dot{\delta}_{f_i}$, $\dot{a}_{xi}$, ϵ_{y_i}, ϵ_{g_i} allows the solver to improve the regularity of the solution. For example, it is important to minimize $\dot{\delta}_{f_i}$ of the frontal wheels to avoid harsh steering. Another important element is $\dot{a}_{xi}$. This term should be minimized to prevent the vehicle from suddenly accelerating or decelerating.

Nevertheless, to define a reference trajectory also for the tracking errors can facilitate the solver to find the optimal solution. In the sense that if the solver will try to minimize also the tracking errors related to a reference trajectory, then this action can improve the regularity of the vehicle behaviour. For this reason, two reference trajectories are considered. Namely, a *Centerline* reference and a *Curve Cutting* reference.
Regarding to the strategy that considers a *Centerline* trajectory, the cost function predictive vectors are:

$$h(x_k, u_k) = \left[e_{y_1}, e_{\psi_1} + \alpha_1, t_1, \dot{\delta}_{f1}, \dot{a}_{x1}, \epsilon_{y_1}, \epsilon_{g_1}, e_{y_2}, e_{\psi_2} + \alpha_1, t_2, \dot{\delta}_{f2}, \dot{a}_{x2}, \epsilon_{y_2}, \epsilon_{g_2}\right]^\top \tag{4.9}$$

$$h_N(x_N) = \left[e_{y_{1N}}, e_{\psi_{1N}} + \alpha_1, t_{1N}, e_{y_{2N}}, e_{\psi_{2N}} + \alpha_2, t_{2N},\right]^\top \tag{4.10}$$

where α_1 and α_2 are the vehicles slips[1]. They are taken into account with e_{ψ_i} in the cost functions in order to improve the regularity of the vehicle directions.
Instead, the reference vectors are $h^k_{ref} = \mathbf{0}$ and $h^N_{ref} = \mathbf{0}$ (with same dimension of the previous $h(x_k, u_k)$ and $h_N(x_N)$).
The last strategy is referred to consider a *Curve cutting* trajectory so that the inner objective vectors are equal to the previous ones while the reference vectors are defined as:

$$h^k_{ref} = \left[e_{y_{ref}}, e_{\psi_{ref}}, 0, 0, 0, 0, 0, e_{y_{ref}}, e_{\psi_{ref}}, 0, 0, 0, 0, 0\right]^\top \tag{4.11}$$

$$h^N_{ref} = \left[e_{y_{ref_N}}, e_{\psi_{ref_N}}, 0, e_{y_{ref_N}}, e_{\psi_{ref_N}}, 0\right]^\top \tag{4.12}$$

The quantities $e_{y_{ref}}$, $e_{y_{ref_N}}$, $e_{\psi_{ref_N}}$ and $e_{\psi_{ref_N}}$ are referred to the *Curve cutting* trajectory,

[1]$\alpha_j = atan2(\dot{y}_j, \dot{x}_j) \qquad \forall j = 1, 2$

which is based on the curvature computation.

The general constraint function is defined as:

$$r_k(x_k, u_k) = \begin{bmatrix} e_{y1} + \epsilon_{y1} \\ (\frac{\ddot{x}_{ext,1}}{g\,\mu_x})^2 + (\frac{\ddot{y}_{ext,1}}{g\,\mu_y})^2 + \epsilon_{g1} \\ e_{y2} + \epsilon_{y2} \\ (\frac{\ddot{x}_{ext,2}}{g\,\mu_x})^2 + (\frac{\ddot{y}_{ext,2}}{g\,\mu_y})^2 + \epsilon_{g2} \\ \sqrt{(d_{long})^2 + (d_{lat})^2} \\ e_{\psi 1} + \alpha_1 \\ e_{\psi 2} + \alpha_2 \end{bmatrix} \tag{4.13}$$

while $r_N(x_N)$ is not defined.
The general constraint takes into account the path constraints related to the vehicle distances from references and the vehicle direction errors (with vehicle slips). The elements $e_{y_i} + \epsilon_{y_i}$ define the limits of the track in which vehicles are supposed to run. Due to the presence of the slack variables ϵ_{y_i}, these constraints can be violated (in fact, they are called *soft constraints*).
The second and the fourth constraint of $r_k(x_k, u_k)$ are acceleration constraints inspired by the typical grip ellipsoid induced by the tire dynamics.
Finally, the constraint $\sqrt{(d_{long})^2 + (d_{lat})^2}$ is referred to the relative distance between vehicles.
In particular, the longitudinal and lateral distances are calculated as follows:

$$\begin{aligned} d_{long} &= |t_1 - t_2|\,\frac{\dot{x}_1 + \dot{x}_2}{2}, \\ d_{lat} &= |e_{y1} - e_{y2}| \end{aligned} \tag{4.14}$$

4.2 Controller setup

The controller setup for this automotive application is described in this section. In particular, *MATMPC* options are defined through Table 4.1.
A sparse QP solver is used because one rule-of-thumb is that the sparse formulation is faster for large values of N (number of shooting points) while the dense one is suitable for problems with small values of N [7]. In fact, a long prediction horizon $H = N\,Ts = 400\,m$ is taken into account because both vehicles reach very high velocities.
To conclude, a nonuniform grid is adopted for *Multiple Shooting* technique in order to reduce the number of computational points i.e. $r = 140$.

	Selected module
Integrator	Explicit Runge Kutta 4
Condensing	Non
QP Solver	HPIPM
Globalization	Real-Time Iteration
Sampling space	$T_s = 1\,[m]$
Shooting interval	$T_s t = T_s$
No. of shooting points	$N = 400$
No. of input move blocks	$r = 140$

Table 4.1: *MATMPC* options

Regarding to the initial conditions, an initial longitudinal velocity of $\dot{x}_1 = \dot{x}_2 = 50\,m/s^2$ and a position with respect to the *Centerline* reference of $e_{y_1} = 2\,m$, $e_{y_2} = -2\,m$ are considered. Thus, the first vehicle (named as *user vehicle*) starts on the left side from the *Centerline* reference. Instead, the second vehicle (called *adversarial vehicle*) is positioned on the right side.
Instead, if the *Curve Cutting* reference is taken into account, then the initial values of tracking errors will be different for each test.
In general, the other quantities for both x_0 and u_0 are set to zero. Sometimes also the vehicle initial time istants will be set differently (this will turn out for *Delayed departure* tests in Chapters 6 and 7)
In NLP (4.1), the constraints on states are intrinsic to the problem. They allow to determine when the problem is infeasible. In particular, the angles of the front tires cannot be greater than 30° while the longitudinal acceleration cannot exceed 5 g.

$$\underline{x}_k = \begin{bmatrix} -\pi/6 \\ -5 \\ -\pi/6 \\ -5 \end{bmatrix} \leq \begin{bmatrix} \delta_{f1} \\ a_{x1} \\ \delta_{f2} \\ a_{x2} \end{bmatrix} \leq \overline{x}_k = \begin{bmatrix} \pi/6 \\ 5 \\ \pi/6 \\ 5 \end{bmatrix} \tag{4.15}$$

Constraints on inputs have the dual function of helping the solver by narrowing the set of possible solutions and preventing drastic changes in the inputs. These constraints are not physiological but if left too wide, they can cause instability in the solution [2].

$$\underline{u}_k = \begin{bmatrix} -2.3 \\ -40 \\ -20 \\ -20 \\ -2.3 \\ -40 \\ -20 \\ -20 \end{bmatrix} \leq u_k = \begin{bmatrix} \dot{\delta}_{f1} \\ \dot{a}_{x1} \\ \epsilon_{y1} \\ \epsilon_{g1} \\ \dot{\delta}_{f2} \\ \dot{a}_{x2} \\ \epsilon_{y2} \\ \epsilon_{g2} \end{bmatrix} \leq \overline{u}_k = \begin{bmatrix} 2.3 \\ 40 \\ 20 \\ 20 \\ 2.3 \\ 0 \\ 20 \\ 20 \end{bmatrix} \tag{4.16}$$

Finally the general constraint bounds are defined as:

$$\underline{r}_k = \begin{bmatrix} -3.5 \\ 0 \\ -3.5 \\ 0 \\ d_{min} \\ -\pi/2 \\ -\pi/2 \end{bmatrix} \leq r_k(x_k, u_k) = \begin{bmatrix} e_{y1} + \epsilon_{y1} \\ (\frac{\ddot{x}_{ext,1}}{g\,\mu_x})^2 + (\frac{\ddot{y}_{ext,1}}{g\,\mu_y})^2 + \epsilon_{g1} \\ e_{y2} + \epsilon_{y2} \\ (\frac{\ddot{x}_{ext,2}}{g\,\mu_x})^2 + (\frac{\ddot{y}_{ext,2}}{g\,\mu_y})^2 + \epsilon_{g2} \\ \sqrt{(d_{long})^2 + (d_{lat})^2} \\ e_{\psi 1} + \alpha_1 \\ e_{\psi 2} + \alpha_2 \end{bmatrix} \leq \overline{r}_k = \begin{bmatrix} 3.5 \\ \mu^2 \\ 3.5 \\ \mu^2 \\ 1000 \\ \pi/2 \\ \pi/2 \end{bmatrix} \tag{4.17}$$

In particular, the constraints related to the errors are selected arbitrary while the others are intrinsic of the problem [3].
Observe in eq (4.17) that the vehicles crash distance constraint cannot be violeted and the

[2]The limits for $\dot{\delta}_{f_i}$ are evaluated in $[rad/s]$ and the bounds of $\dot{a}_{xi}$ are expressed in 9.81 $[m/s^3]$.

[3]$e_{y_i} + \epsilon_{y_i}$ is evaluated in $[m]$ while $e_{\psi_i} + \alpha_i$ are expressed in $[rad]$. The relative distance constraint is measured in $[m]$ while the others are adimensional.

minimum value is computed as:

$$d_{min} = \sqrt{(a+b)^2 + (2c)^2} + d_{add} \tag{4.18}$$

where the parameters a, b, c are those of the vehicle dimensions[4] while the term d_{add} plays the role of an additional distance constant to ensure collision avoidance.

Taking into account NLP (4.1), the weigths matrices of the objective function are defined as follows [5]:

$$W_k = diag\,(q_{e_{y_1}}, q_{e_{\psi_1}}, q_{t_1}, q_{\dot{\delta_{f_1}}}, q_{\dot{a_{x1}}}, q_{\epsilon_{y_1}}, q_{\epsilon_{g_1}}, q_{e_{y_2}}, q_{e_{\psi_2}}, q_{t_2}, q_{\dot{\delta_{f_2}}}, q_{\dot{a_{x2}}}, q_{\epsilon_{y_2}}, q_{\epsilon_{g_2}}) \tag{4.19}$$

$$W_N = diag\,(q_{e_{y_1 N}}, q_{e_{\psi_1 N}}, q_{t_1 N}, q_{e_{y_2 N}}, q_{e_{\psi_2 N}}, q_{t_2 N}) \tag{4.20}$$

W_k and W_N are important since they allow to control properly the two vehicles simultaneously. Thus, if one weight takes on a very high value, then the related variable will be minimized even more.
An example of calibration is provided by Table 6.2.
This concludes the controller setup for two vehicles in order to solve the Nonlinear Programming Problem introduced in Sec 4.1. All the tests related to the race between these vehicles will be evaluated in Chapter 6 and 7.

4.3 NLP for single vehicle

If a single vehicle behaviour is taken into account, then the formulation of NLP (4.1) should be reformulated with respect to different kind of states, controls, cost function and general constraint.

In particular, the states become:

$$x_k = \left[\dot{x}, \dot{y}, \dot{\psi}\ e_\psi, e_y, \delta_f, a_x, t\right]^\top \tag{4.21}$$

while the controls are:

$$u_k = \left[\dot{\delta_f}, \dot{a_x}, \epsilon_y, \epsilon_g\right]^\top . \tag{4.22}$$

As a consequence it follows that:

$$h(x_k, u_k) = \left[e_y, e_\psi, t, \dot{\delta_f}, \dot{a_x}, \epsilon_y, \epsilon_g\right]^\top \tag{4.23}$$

[4]In the bicycle model the lateral distance between the wheel and the center of mass is $c = 0$. So in d_{min} definition, the quantity about $2\,c$ plays the role of minimum lateral distance between the bicycles CGs.

[5]In case of *No tracking errors weigths* reference strategy, both W_k and W_N do not consider the weights of tracking errors.

$$h_N(x_N) = \left[e_{y_N}, e_{\psi_N}, t_N\right]^\top \tag{4.24}$$

$$r_k(x_k, u_k) = \begin{bmatrix} e_y + \epsilon_y \\ (\frac{\ddot{x}_{ext}}{g\,\mu_x})^2 + (\frac{\ddot{y}_{ext}}{g\,\mu_y})^2 + \epsilon_g \\ e_\psi + \alpha \end{bmatrix} \tag{4.25}$$

with the proper states, controls and general constraint bounds (equal to the previous ones but referred to a single vehicle).
Notably, eq (4.25) doesn't take into account the relative distance between vehicles since there is only one car to control.

Concerning to the controller setup, Table 4.1 can be applied again with weights matrices defined as:

$$W_k = diag\ (q_{e_y}, q_{e_\psi}, q_t, q_{\dot{\delta_f}}, q_{\dot{a_x}}, q_{\epsilon_y}, q_{\epsilon_g}) \tag{4.26}$$

$$W_N = diag\ (q_{e_{y_N}}, q_{e_{\psi_N}}, q_{t_N}) \tag{4.27}$$

All the single vehicle simulations will be presented in Chapter 5.

Chapter 5

Simulations with a single vehicle

The objective of this thesis is to provide an NMPC-based tool to emulate a controller that interacts with human.
This Chapter aims to define a benchmark for the NMPC controller configuration. In particular, to identify a starting calibration for a single vehicle which can be adopted also for the two vehicles scenario.
The inclusion of e_y, e_ψ, e_{y_N} and e_{ψ_N} in the cost functions is an important factor that is carefully considered within this thesis project. For such purpose, in the previous Chapter three different strategies are described in details for the reference errors that are named as: *No tracking errors reference*, *Centerline* reference and *Curve cutting* reference.
To determine which strategy will be better than the others, two important characteristics will be evaluated. Namely, the minimization of the lap time and the numerical robustness of the system. Consequently, these three scenarios will be tested for a single vehicle on the Paul Ricard circuit (figure 5.1).

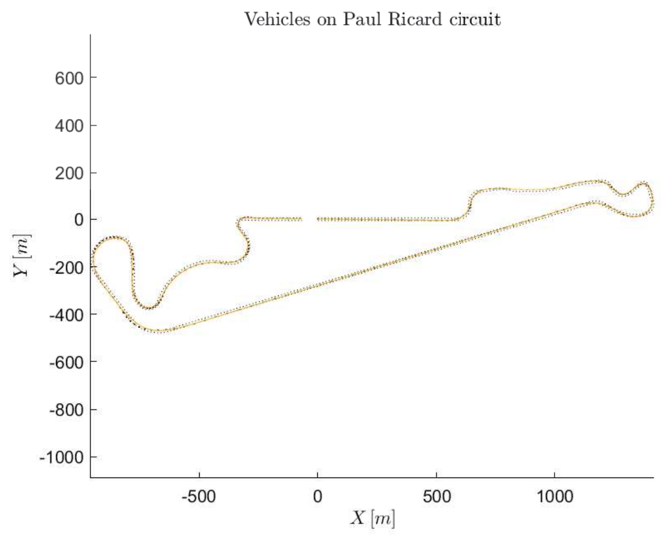


Figure 5.1: Paul Ricard circuit

Table 5.1 describes the specific calibration of the weights for the single vehicle with respect to the three reference strategies. Instead, Table 5.2 presents some additional parameters and results obtained from each simulation.

Reference strategy	*Centerline*	*Curve Cutting*	*No tracking errors ref*
Weight	Numeric values		
q_{e_y}	$4 \cdot 10^{-2}$	10^{-2}	–
q_{e_ψ}	$4 \cdot 10^{-2}$	10^{-2}	–
q_t	$5 \cdot 10^{-1}$	$5 \cdot 10^{-1}$	$1 \cdot 10^{-1}$
$q_{\dot{\delta_f}}$	10^2	10^2	$1 \cdot 10^2$
$q_{\dot{a_x}}$	$5 \cdot 10^{-2}$	$5 \cdot 10^{-2}$	$1 \cdot 10^{-1}$
q_{ϵ_y}	$2 \cdot 10^1$	$2 \cdot 10^1$	$3 \cdot 10^3$
q_{ϵ_g}	$1 \cdot 10^2$	$1 \cdot 10^2$	$2 \cdot 10^1$
$q_{e_y N}$	$3 \cdot 10^2$	$5 \cdot 10^1$	–
$q_{e_\psi N}$	10^2	$5 \cdot 10^1$	–
q_{tN}	10^2	10^1	$1 \cdot 10^{-1}$

Table 5.1: Controller calibration for a Single vehicle

Reference strategy	*Centerline*	*Curve Cutting*	*No tracking errors ref*
Parameters	Numeric values		
Average CPT $[ms]$:	10.3947	9.8167	17.81
Maximum CPT $[ms]$	43.7391	45.741	46.7307
Maneuver Time $[s]$	119.6643	118.6931	120.4514
Max speed $[m/s]$	104.8493	105.4278	103.3584

Table 5.2: Results for each reference strategy

The *No tracking errors reference* strategy is very sensitive to the variation on weights in the sense that a small variation of the weights leads to fail the simulation.
Conversely, the algorithm results more robust when using references (*Curve Cutting* and *Centerline*). This is proved by the fact that if q_t,$q_{\dot{\delta_f}}$, $q_{\dot{a_x}}$, q_{ϵ_y}, q_{ϵ_g} and q_{tN} are changed, then simulation of a single vehicle correctly works. Instead, if a small perturbation is performed for q_{e_y}, q_{e_ψ}, $q_{e_y N}$ and $q_{e_\psi N}$, then the controller provides a trajectory whose maneuvering time turns out to be similar to the previous one.

For example, considering *Curve Cutting* reference, if the following weigths are imposed:

$$q_{e_y} = 10^{-1}, \qquad q_{e_\psi} = 10^{-1}, \qquad q_{e_{yN}} = 10^{2}, \qquad q_{e_{\psi N}} = 10^{2},$$

then the lap time is 118.7302 s.

From Table 5.2, the smallest lap time is related to the *Curve Cutting* reference so that this strategy proved to be the best one. For this reason, the benchmark for the NMPC controller is defined by adopting the *Curve Cutting* strategy weigths. In addiction, the vehicle simulation path under *Curve Cutting* is presented in the following section. Instead, the other two cases are reported in the Appendix A.1.

5.1 Single vehicle test with Curve cutting strategy

Figures 5.2 and 5.3 show the vehicle trajectory on Paul Ricard circuit.
Given that an high value for the time weigth q_t and a small value for the longitudinal acceleration weigth $q_{\dot{a}_x}$ are imposed, then the vehicle goes slightly off the track during some curves. In particular, during the fast curve (called *Courbe de Signes* curve and located at 3700 m), the errors take the worst values due to high speed (figure 5.3).
In these figures the car is plotted at each 0.5 s and it is colored differently for each time istant.

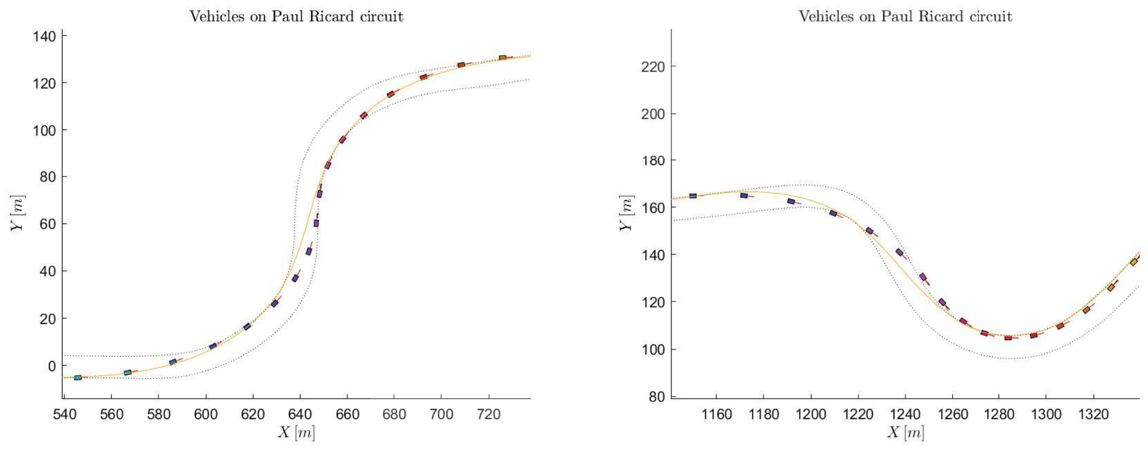
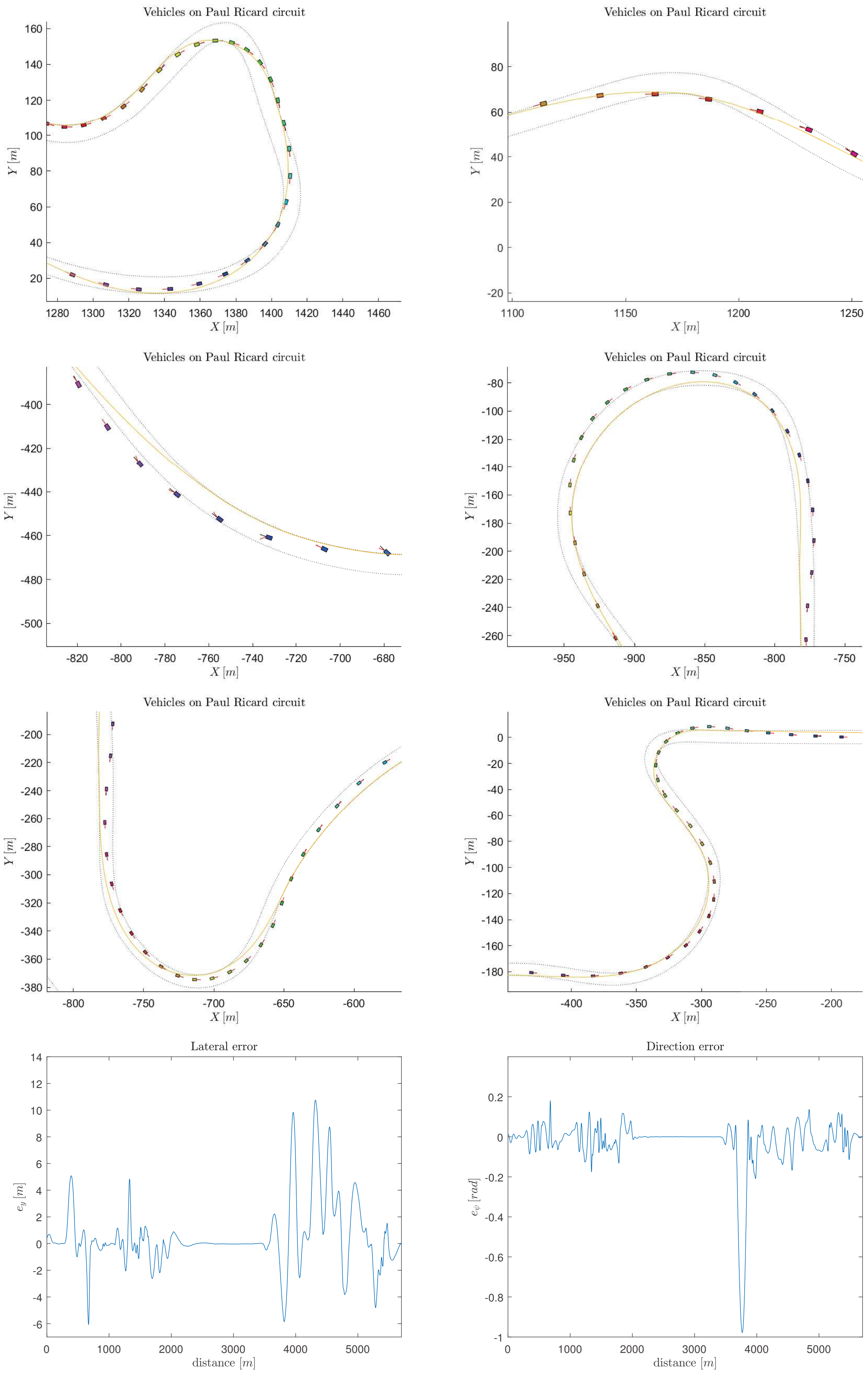


Figure 5.2: Simulation Path under *Curve cutting* reference



(a) Lateral error

(b) Direction error

Figure 5.3: Simulation Path and tracking errors under *Curve cutting* reference

Chapter 6

Simulations with two vehicles

The purpose of the project is to build a NMPC system that allows to drive a vehicle that behaves like an opponent in a driving simulation context. To achieve this goal, it is necessary to realize a controller that allows the management of two vehicles within the same control system. This fact will be addressed and described in this Chapter.

For sake of clarity I recall that the first vehicle is named *user vehicle* because it will represent the vehicle that will have to be driven by the human. The second one is called *adversarial vehicle* since it will be the one actually driven by the controller.
Before dealing with the main part of the Chapter, a remark has to be made: the NLP (4.1) is solved by adopting the controller setup imposed in Sec 4.2.

At the preliminary stage, the controller was analyzed with a single vehicle to establish a benchmark for the controller calibration (Chapter 5). The second phase must involve structuring the controller for two vehicles, which can be realized from that benchmark configuration. Therefore, this Chapter will deal with the competition between two virtual vehicles (driven by the same NMPC controller) on the Paul Ricard circuit.
The overtaking management will become a fundamental element in describing the interaction between these vehicles. It will be found that overtaking depends on many factors including: the shape of the road, the initial position of the vehicles relative to the curve and the initial state of the model. For these reasons, to use a single calibration will not be possible to ensure overtaking in any situation. Therefore, the benchmark calibration will work only for the *Paired departure* test developed on the entire circuit. In this particular situation *user vehicle* will depart at the same time istant of the *adversarial vehicle* one. Then, *user vehicle* will surpass *adversarial vehicle* during the first straight section. After that, both vehicles will complete the track. This simulation will be described in Sec 6.1.
Sec 6.2 will focus on single curves tests where the calibration of Table 5.1 will not work anymore. Moreover, both *Paired departure* tests and *Delayed departure* tests will be taken into account for each curve and in general, they will require different calibrations. To be more specific, the *Delayed departure* tests are referred to the cases where *user vehicle* starts later than *adversarial*

vehicle. In this latter circumstances the race will be more challenging since it is desired that *user vehicle* overtakes *adversarial vehicle* within the end of the curve.
In conclusion, all these tests will be performed to prove that the development of the desired tool is very difficult. In fact, the overtaking manegement will have to be handled appropriately. This is because, as already argued, the competition between two virtual vehicles will have to take into account a variable calibration for the controller depending on the specific case. Some overtaking considerations will be presented in Sec 6.3.

6.1 Paired departure test on Paul Ricard circuit

In this section the *Paired departure* test is considered. For this scenario, both the vehicles start their maneuver at the same initial time istant.
To accomplish this simulation, the benchmark defined in the previous Chapter can be adopted. Thus, the same calibration of the single vehicle (the one related to *Curve Cutting* reference) can be assigned to both *user vehicle* and *adversarial vehicle*. To be more specific, only the time weigths are different so that the time weight of the second vehicle is reduced: $q_{t_1} = 5{\cdot}10^{-1} > q_{t_2} = 5{\cdot}10^{-2}$. This relation allows the cost function to minimize the lap time of the first vehicle more than the second one. This is reflected in Table 6.1 and *user vehicle* is able to surpass *adversarial vehicle* within the first 80 m (figure 6.1a).
The choice of small tracking errors weigths allows both vehicles to move away from the reference to accomplish overtaking otherwise they would have been forced to follow the reference.

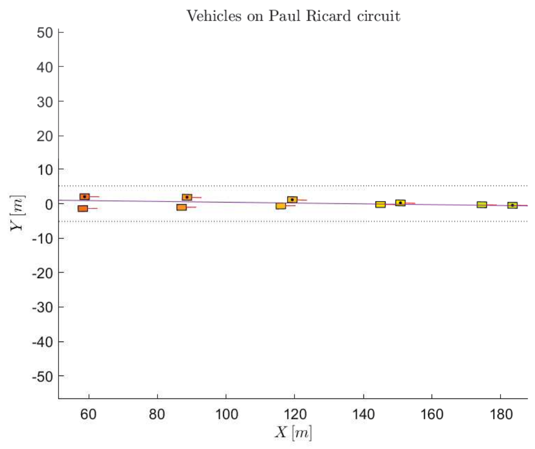
It is also necessary to add that *user vehicle* will be highlighted with a "black dot" applyed to its Center of Gravity (for each image related to the simulation path). This is done in order to discriminate its path from the *adversarial vehicle* one.

Test type	*Paired departure* test
Reference strategy	*Curve Cutting*
Parameters	Numeric values
Average CPT [ms]:	31.9414
Maximum CPT [ms]	168.6339
user vehicle maneuver time [s]	119.4682
adversarial vehicle maneuver time [s]	124.1292
user vehicle max speed [m/s]	105.1201
adversarial vehicle max speed [m/s]	104.3369

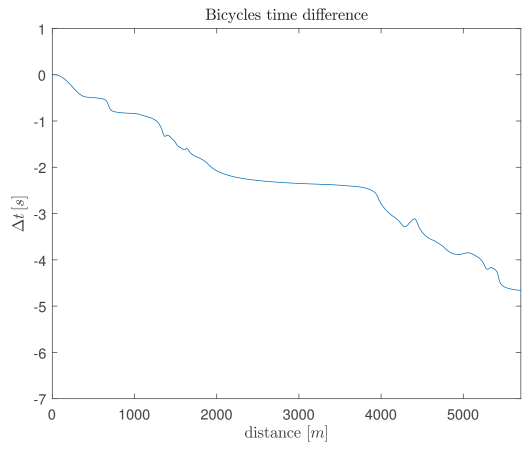
Table 6.1: Performances about *Paired departure* test

Figures 6.1 and 6.2 show the simulation results. In particular, the overtaking (figure 6.1a),

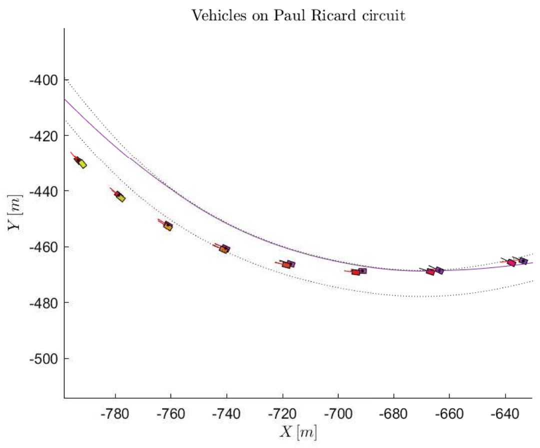
the negative gap time Δ_t which shows a time advantage for *user vehicle* (figure 6.1b), two critical curves (figures 6.1c and 6.1d), the reference errors (figures 6.1e, 6.1f) and the side slip angles of the wheels (figures 6.2a and 6.2b).
As for single vehicle test, also in this case the vehicle paths are plotted at each 0.5 s and they are colored differently for each time istant.



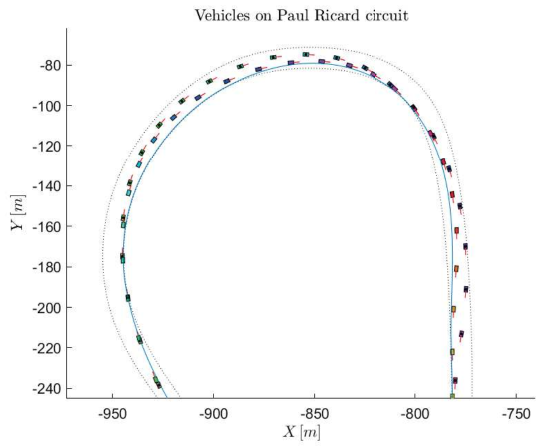
(a) Overtaking phase



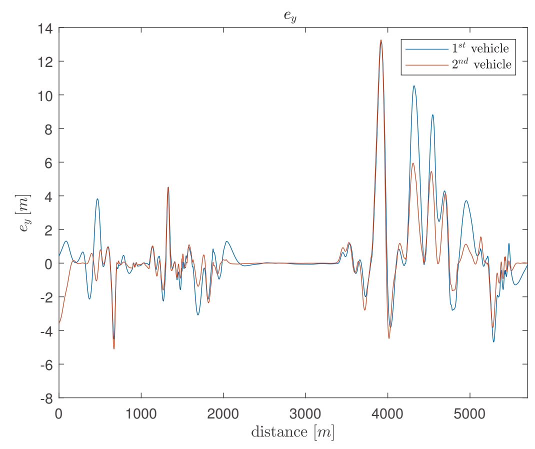
(b) Delta time between vehicles



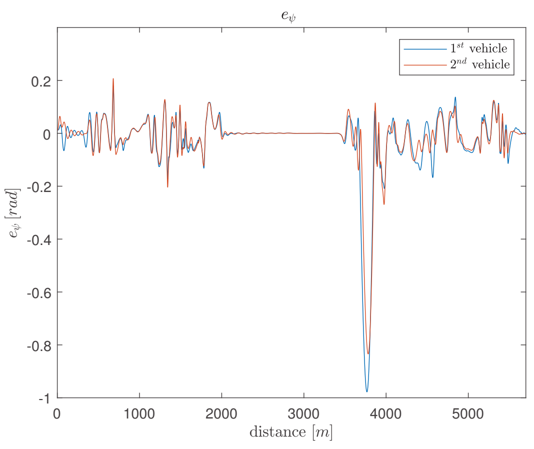
(c) Courbe de Signes curve



(d) Beausset curve

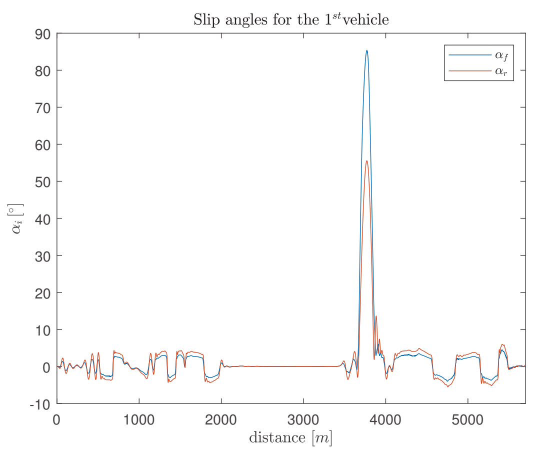


(e) Lateral error

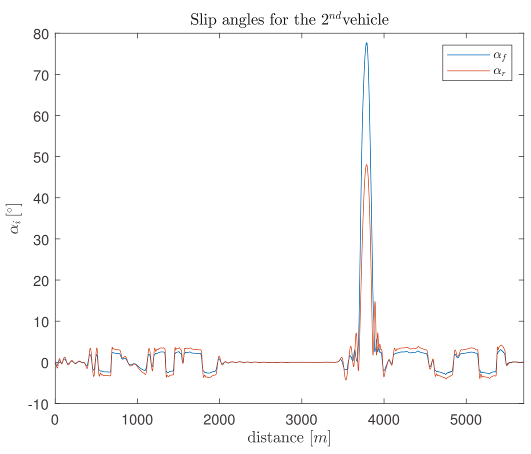


(f) Direction error

Figure 6.1: Results of *Paired departure* test with *Curve Cutting* strategy



(a) Slips of the 1^{st} bicycle



(b) Slips of the 2^{nd} bicycle

Figure 6.2: Results of *Paired departure* test with *Curve Cutting* strategy

The reason why both vehicles go out from *Courbe de Signes* curve (figure 6.1c) is related to the high values of the slip angles. In fact, the vehicles slip too much due to high speeds like in the single vehicle simulation. This happens because the *user vehicle* trajectory was originally optimized (in Chapter 5) in order to minimize the maneauver time. This causes vehicle velocities bigger than 100 m/s (Table 6.1) in the proximity of this curve.
It is desired that the vehicles have similar trajectories outside the overtaking zone. The tracking errors values have to be considered to study this phenomenon. Indeed, taking into account e_{y_i} and e_{ψ_i} $\forall i = 1, 2$ (figures 6.1e and 6.1f) there are only two curves which are crossed wrongly. The first one is *Courbe de Signes* curve (the 8^{th} one) for the reasons described above. Instead, *user vehicle* crosses *Beausset* curve (the 9^{th} one reported in figure 6.1d) externally with respect to *adversarial vehicle* since it goes faster. This last situation proves that if one vehicle is much faster than the other, it will surely have a very different path.
The entire simulation path is reported in the Appendix A.2 for compliteness and in general, the two vehicles have a very similar path (except for the two situations described earlier).

6.2 Single curve tests

The result described in the previous section is related to a simple case of interest. In particular, the overtaking took place at the beginning of the competition along the first straight section of the road. In general, the previous calibration is not suitable to accomplish an overtaking, especially when vehicles are crossing a curve. The motivations have already been described at the beginning of the Chapter. In particular, the shape of the track affects the problem of managing the two vehicles. Therefore, different calibrations have to be selected for different sections of the circuit to solve the problem. In general, a good calibration must have a low tracking error weigths. This implies that vehicles can move away from the reference trajectory to accomplish the overtaking. But on the other hands these weights should not be too small to avoid off-road scenarios [1]. Regarding to the time weigths and the acceleration ones, they should be adjusted based on making a vehicle faster or slower. In general, it is preferable to keep an equal acceleration profile and change only the time weigths. In fact, from the perspective of a realistic scenario, the two vehicles should accelerate/brake smoothly and similarly.

This section deals with single curves tests where both vehicles can compete in a very particular situation. The aim is to accomplish the overtaking maneuver along three different curves. They are named *first Chicane* (1^{st} and 2^{nd} curve), *L'Ecole* (7^{th} curve) and *Beausset* (9^{th} curve). For each one of them, both *Paired departure* and *Delayed departure* tests will be carried out, respectively in Sec 6.2.1, Sec 6.2.2, Sec 6.2.3.
Table 6.2 reports a possible calibration which considers small values for $q_{e_{y_i}}$, $q_{e_{\psi_i}}$, high values for $q_{e_{y_i N}}$, $q_{e_{\psi_i N}}$ and $q_{t1} > q_{t2}$. This leads to have an overtaking in most cases. To be more specific, this calibration allows the controller to accomplish the overtaking for all the *Paired departure* tests. In *delay 0.5 s* test, *user vehicle* is still able to surpass *adversarial vehicle* in all the curves, except when crossing *L'Ecole* curve. This happens due to the shape of this specific curve and its length. In fact, even if q_{t1} is increased and q_{t2} is reduced, the controller is not able to drive *user vehicle* to surpass *adversarial vehicle* neither in *delay 0.5 s* test, nor in *delay 1s* test.
Instead, this calibration has to be properly modified for the *first Chicane* and *Beausset* curves to achieve the overtaking in *delay 1s* test. This proves again that the shape of these curves impacts on vehicles performances.

[1]Recall that the road limits are not hard constraints but soft constraints as described by the general constraint definition (4.13)

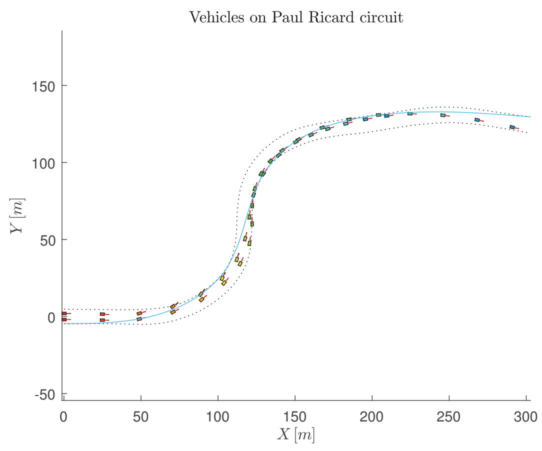
Curve type	*first Chicane*	*L'Ecole*	*Beausset*
Weight	Numeric values		
$q_{e_{y1}}$	$5 \cdot 10^{-2}$		
$q_{e_{\psi 1}}$	$5 \cdot 10^{-2}$		
q_{t_1}	$5 \cdot 10^{-1}$		
$q_{\dot{\delta}_{f1}}$	10^{2}		
$q_{\dot{a}_{x1}}$	$5 \cdot 10^{-1}$		
$q_{\epsilon_{y1}}$	10^{2}		
$q_{\epsilon_{g1}}$	10^{1}		
$q_{e_{y1N}}$	$5 \cdot 10^{2}$		
$q_{e_{\psi 1N}}$	$5 \cdot 10^{2}$		
$q_{t_1 N}$	$5 \cdot 10^{1}$		
$q_{e_{y2}}$	$5 \cdot 10^{-2}$		
$q_{e_{\psi 2}}$	$5 \cdot 10^{-2}$		
q_{t_2}	$5 \cdot 10^{-3}$		
$q_{\dot{\delta}_{f2}}$	10^{2}		
$q_{\dot{a}_{x2}}$	$5 \cdot 10^{-1}$		
$q_{\epsilon_{y2}}$	10^{2}		
$q_{\epsilon_{g2}}$	10		
$q_{e_{y2N}}$	$5 \cdot 10^{2}$		
$q_{e_{\psi 2N}}$	$5 \cdot 10^{2}$		
$q_{t_2 N}$	5		

Table 6.2: Controller calibration for *Paired departure* tests on single curves

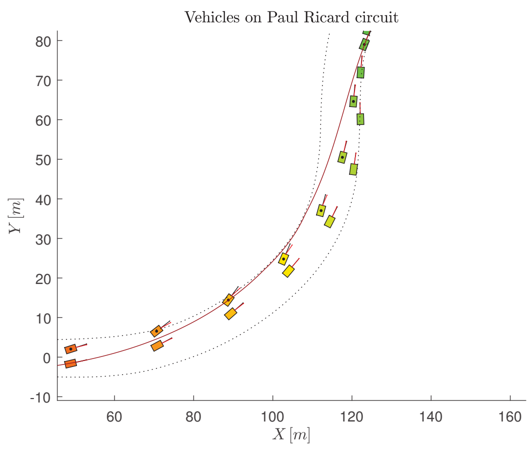
6.2.1 First Chicane curve

Starting from the *Paired departure* test on the *first Chicane* curve, the *user vehicle* can overtake the *adversarial* one by cutting it off internally during the first curve (figure 6.3b). This happens since the first vehicle has a bigger q_{t_1} weight and it is initially located internally with respect to the curve.

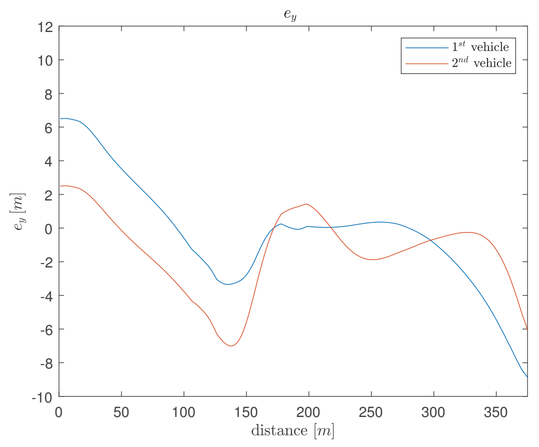
An important phenomenon comes out during this test. In fact, the slower vehicle knows that *user vehicle* is going faster and has to win. So it moves externally during the first curve in order to satisfy the constraint about the vehicles relative distance. After that, *adversarial vehicle* follows *user vehicle* according to the tracking errors reported in figures 6.3c and 6.3d.



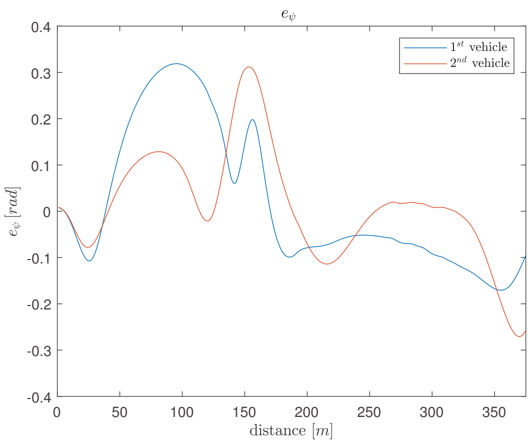
(a) 1^{st} and 2^{nd} curve



(b) Overtaking



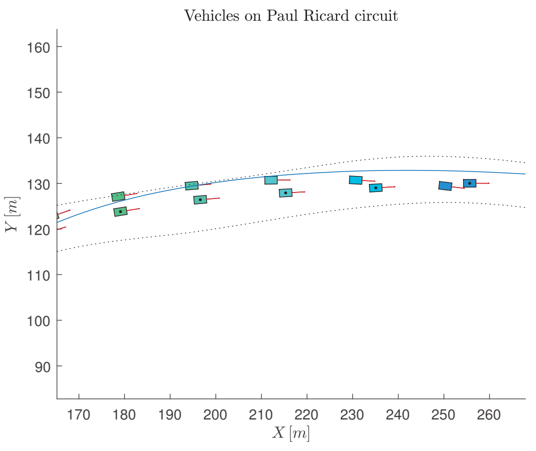
(c) $e_{y_i} \quad \forall i = 1, 2$



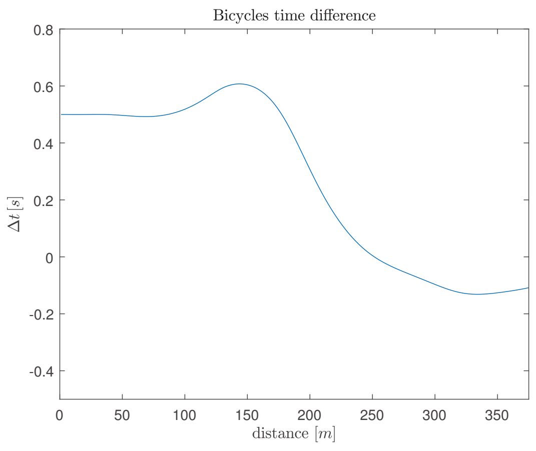
(d) $e_{\psi_i} \quad \forall i = 1, 2$

Figure 6.3: Results of *Paired departure* test on the *first Chicane* curve

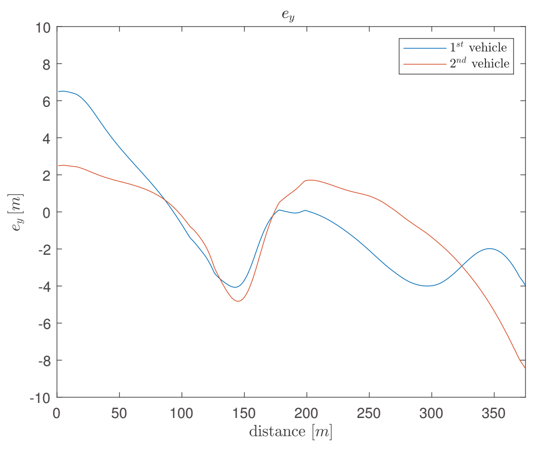
Then *Delay 0.5 s* test is performed on the same curve. As previously said, *user vehicle* is supposed to pass away *adversarial vehicle* even with a disadvantage about the initial time istant. The previous W_k and W_N matrices allow the controller to achieve the goal. In fact, the gap between q_{t_1} and q_{t_2} is sufficiently big. As a consequence, the overtaking takes place during the end of the second curve. In particular, the first vehicle cuts internally, while the second one moves externally in order to let *user vehicle* win (figures 6.4a and 6.4b). Figures 6.4c and 6.4d present high tracking errors since a low tracking errors weigths are adopted.



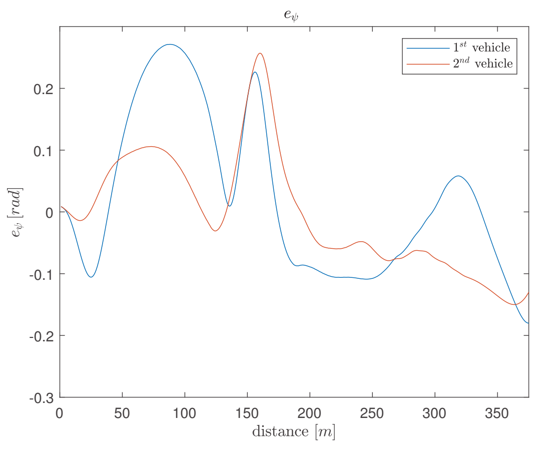
(a) Overtaking



(b) Delta time between vehicles



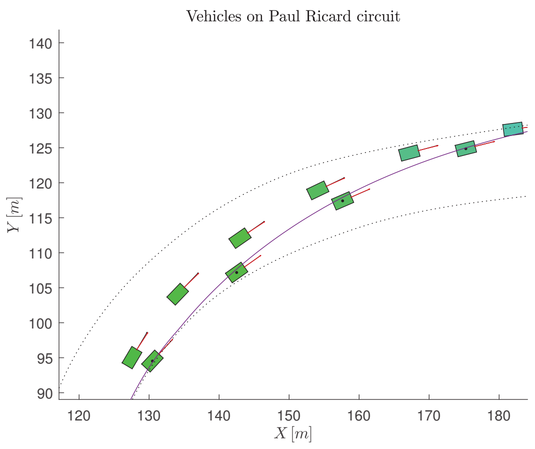
(c) e_{y_i} $\forall i = 1, 2$



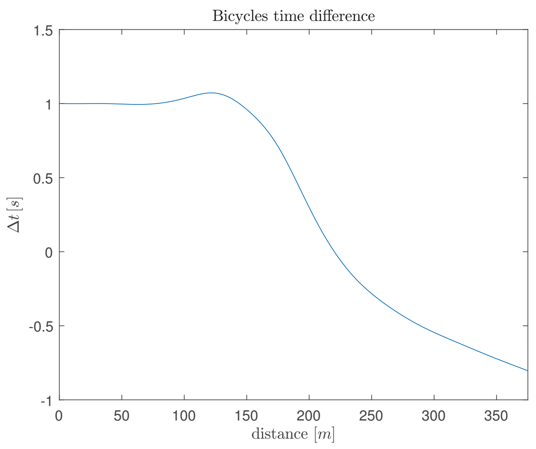
(d) e_{ψ_i} $\forall i = 1, 2$

Figure 6.4: Results of *Delay 0.5 s* test on the *first Chicane* curve

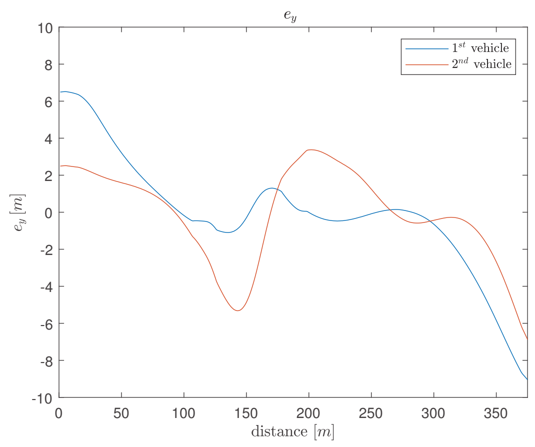
Since in the previous scenario the overtaking took place at the end of the curve, for *Delay 1 s* test is necessary to anticipate it. Therefore, a different calibration is considered. Specifically, the control weigths ϵ_{y_1} and ϵ_{y_2} are increased and also the time weigths are modified. The first ones impact on general constraint lateral errors performances. The second ones allow to change the gap time between vehicles by increasing the speed for *user vehicle* and by reducing the velocity for *adversarial vehicle*. Therefore, the overtaking is anticipated at 216 m (picture 6.5a). *User vehicle* cuts again internally the second curve while *adversarial* one turns externally.



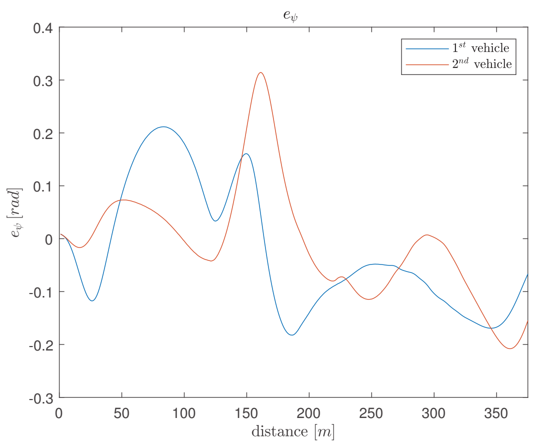
(a) Overtaking



(b) Δ_t between vehicles



(c) $e_{y_i} \quad \forall i = 1, 2$



(d) $e_{\psi_i} \quad \forall i = 1, 2$

Figure 6.5: Results of *Delay 1 s* test on the *first Chicane* curve

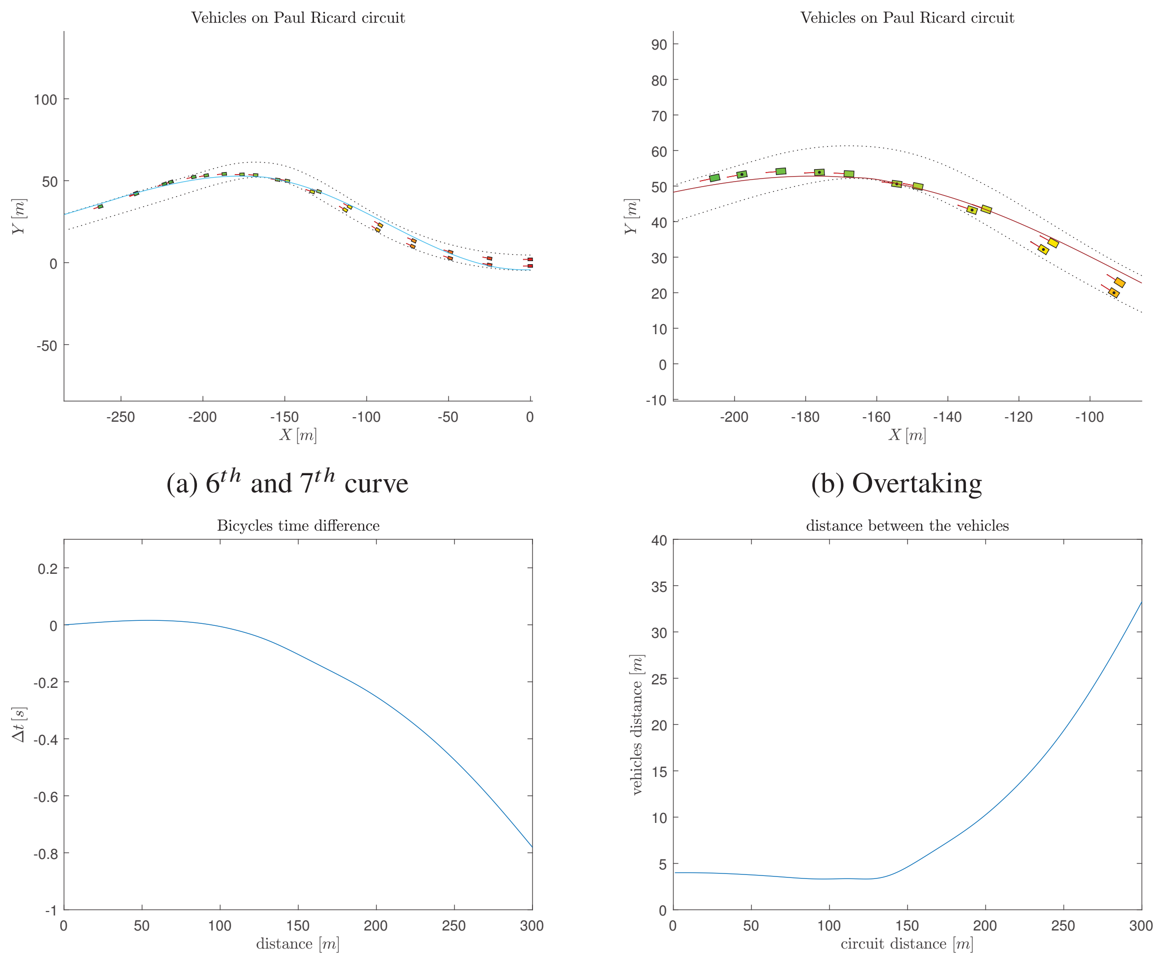
To conclude, Table 6.3 reports the *first Chicane* curve results of these three tests. Observe that the vehicles maneuver times of *Delayed departure* tests take into account the *user vehicle* starting time of $t_0 = 0.5$ s and $t_0 = 1$ s.

Test type	*Paired departure*	*Delay 0.5 s*	*Delay 1 s*
Reference strategy	*Curve cutting*		
Parameters	Numeric values		
Average CPT $[ms]$:	32.6556	31.1221	32.6643
Maximum CPT $[ms]$	52.2583	48.6123	58.704
user vehicle maneuver time $[s]$	9.7357	10.6074	10.3649
adv. vehicle maneuver time $[s]$	12.3277	10.6799	11.1716
user vehicle max speed $[m/s]$	50	50	50
adv. vehicle max speed $[m/s]$	50	50	50

Table 6.3: Performances about the *first Chicane* curve tests

6.2.2 L'Ecole curve

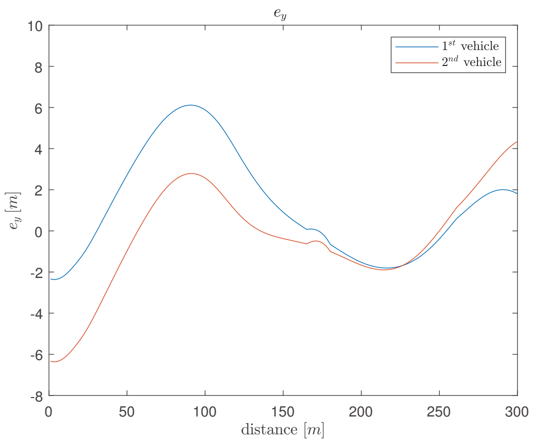
L'Ecole curve refers to the end of the 6^{th} and the 7^{th} curves of the Paul Ricard circuit.
The same previous simulations with two vehicles are repeated. In particular, the *user vehicle* surpasses *adversarial vehicle* by cutting it off inside the 7^{th} curve during the *Paired departure* test (figure 6.6b). This result is obtained by using Table 6.2 weights.
Figure 6.6a shows that *adversarial vehicle* has an advantage during the departure phase. In fact, it is initially positioned internally with respect to the 6^{th} curve. Moreover $\Delta_t > 0$ till the first 100 m (figure 6.6c). Then *adversarial vehicle* decelerates in the proximity of the 7^{th} curve to leave sufficient space maneuver for the overtaking.
Additional information about the relative distance between vehicles (that must be always bigger than d_{min}) and the tracking errors are shown in figures 6.6d, 6.7a and 6.7b.



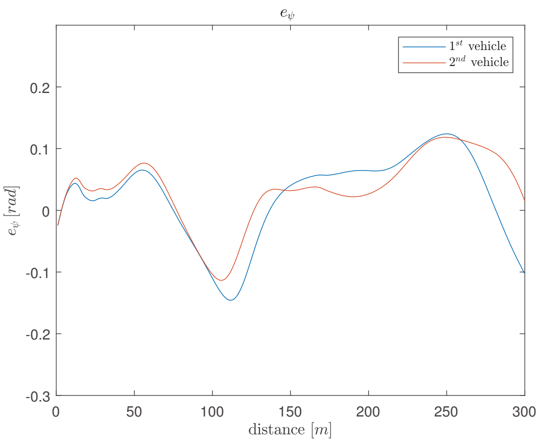
(a) 6^{th} and 7^{th} curve (b) Overtaking

(c) Delta time between vehicles (d) Relative distance

Figure 6.6: Results of *Paired departure* test on *L'Ecole* curve



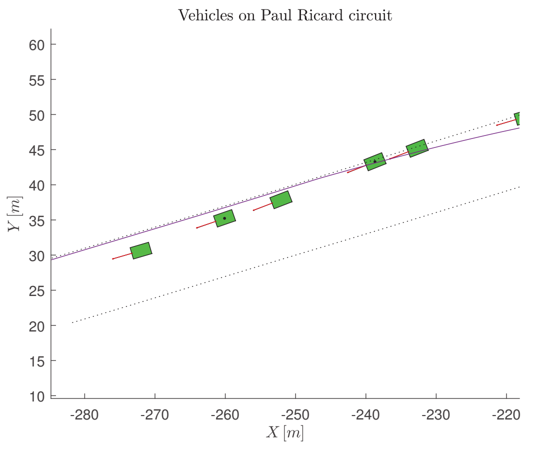
(a) e_{y_i} $\forall i = 1, 2$



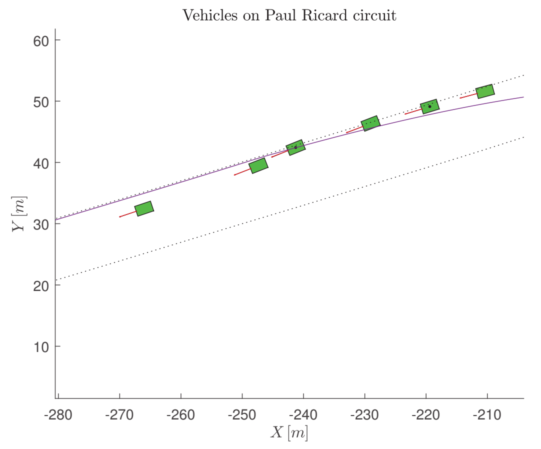
(b) e_{ψ_i} $\forall i = 1, 2$

Figure 6.7: Results of *Paired departure* test on *L'Ecole* curve

For both *delay 0.5 s* and *delay 1 s* tests, *user vehicle* is not able to win the competition (figure 6.8). The reasons are related to the path length (about just 300 m) and to the curve structure. In fact, as previously mentioned, *adversarial vehicle* has already an advantage about the departure since it is located internally with respect to the 6^{th} curve. Consequently, the shape of the curve, the length of simulation path and the initial positions of vehicles have a significant impact for overtaking manegement. In fact, even if q_{t1} is increased and q_{t2} is decreased, then the overtaking cannot be accomplished.



(a) Overtaking attempt for *delay 0.5 s* test



(b) Overtaking attempt for *delay 1 s* test

Figure 6.8: Results of *Delayed departure* tests on *L'Ecole* curve

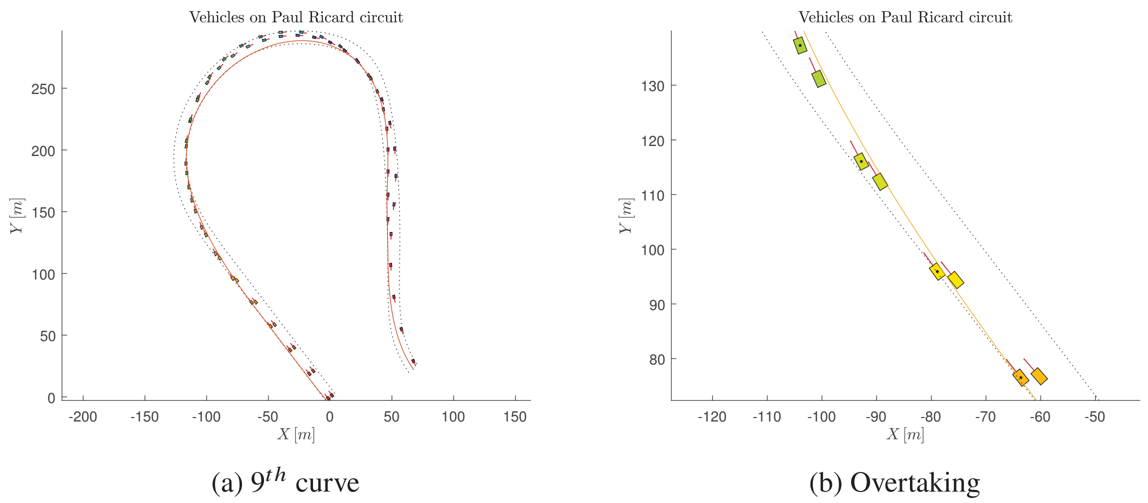
Table 6.4 reports more informations about *L'Ecole* curve tests. As usual, the first vehicle maneuver time considers also the initial time disadvantage for *Delayed departure* tests.

Test type	*Paired departure*	*Delay 0.5 s*	*Delay 1 s*
Reference strategy	*Curve cutting*		
Parameters	Numeric values		
Average CPT $[ms]$:	30.485	32.9637	29.3896
Maximum CPT $[ms]$	72.0689	89.5459	66.016
user vehicle maneuver time $[s]$	6.477	7.0415	7.4564
adv. vehicle maneuver time $[s]$	7.2646	6.7946	6.9664
user vehicle max speed $[m/s]$	50	50	50
adv. vehicle max speed $[m/s]$	50	50	50

Table 6.4: Performances about *L'Ecole* curve tests

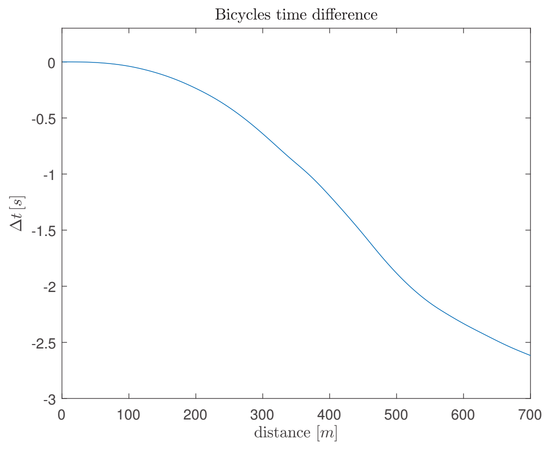
6.2.3 Beausset curve

Table 6.2 can also be applied to *Beausset* curve (the 9th one of the circuit).
For the *Paired departure* test, this approach guarantees the *user vehicle* victory since it surpasses the *adversarial* one during the first straight stretch of road (figure 6.9b). But since it reaches an higher speed before the curve, then its e_{y_1} value becomes bigger than e_{y_2}. Meaning that the controller almost lost its capability to keep the first car within the road limits (figure 6.9a). Instead the second vehicle enters slowly, thus it crosses the curve more softly.
Figures 6.10a, 6.10b, 6.10c and 6.10d show respectively the gap time Δ_t, the relative distance and the tracking errors.

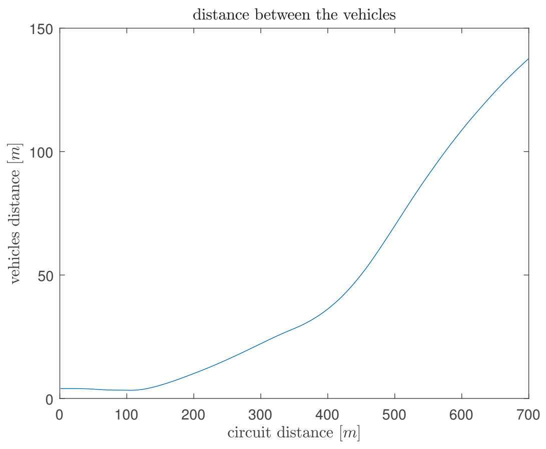


(a) 9th curve (b) Overtaking

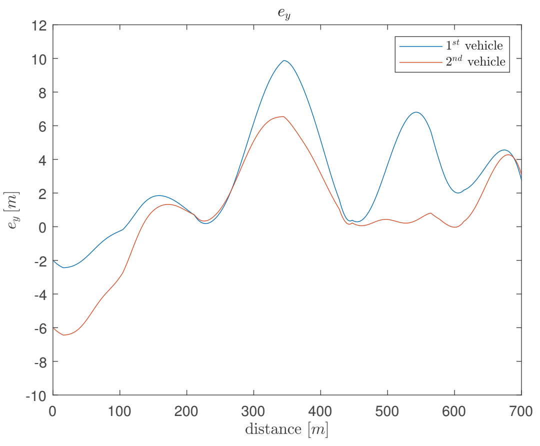
Figure 6.9: Results of *Paired departure* test on *Beausset* curve



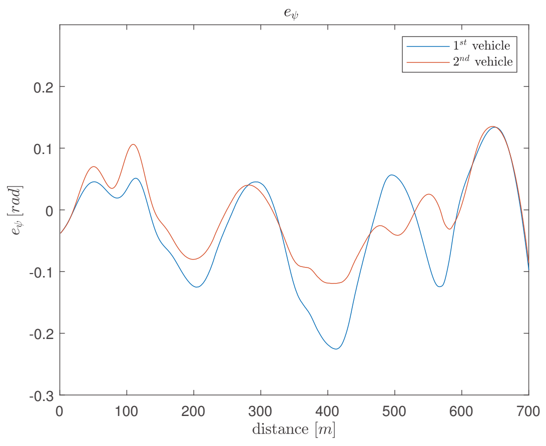
(a) Delta time between vehicles



(b) Relative distance



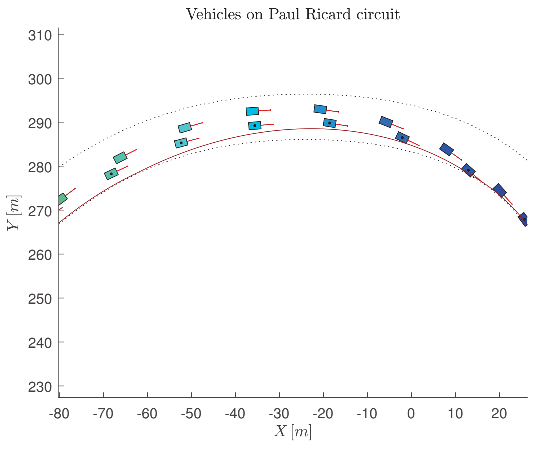
(c) $e_{y_i} \qquad \forall i = 1, 2$



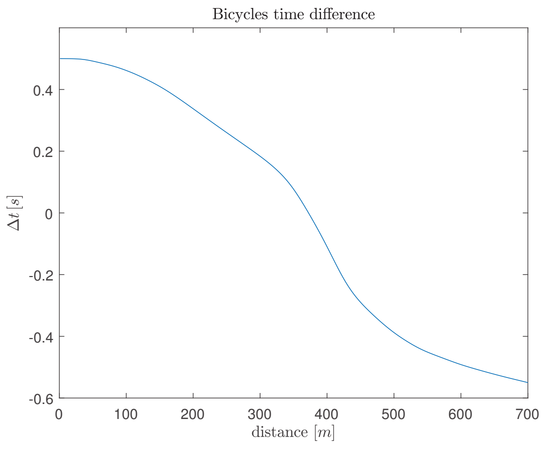
(d) $e_{\psi_i} \qquad \forall i = 1, 2$

Figure 6.10: Results of *Paired departure* test on *Beausset* curve

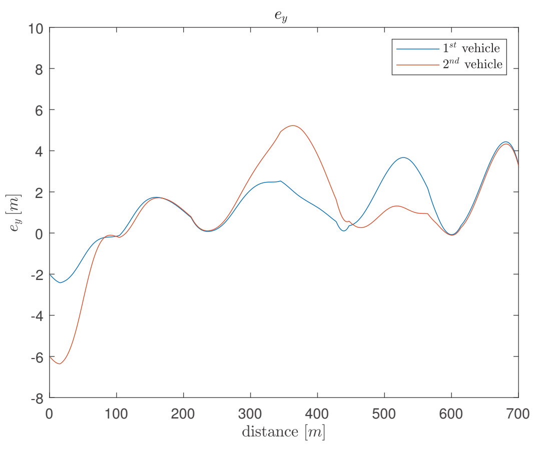
Concerning to *Delayed departure* tests, the previous calibration has to be changed. In fact, to avoid outroad scenarios, two possibilities are available. The first one is to modify the tracking error weights so that both vehicles trust on the reference a bit more than before. The second possibility consists to modify the time weigths. The first choice is adopted to achieve the desired result for *Delay 0.5 s* test. This strategy causes a slight increment of the first vehicle maneuver time (Table 6.5) and a reduction of the reference errors (figures 6.11c and 6.11d). As a consequence, the overtaking is accomplished at 350 m (figures 6.11a and 6.11b) where *user vehicle* surpasses internally *adversarial vehicle*.



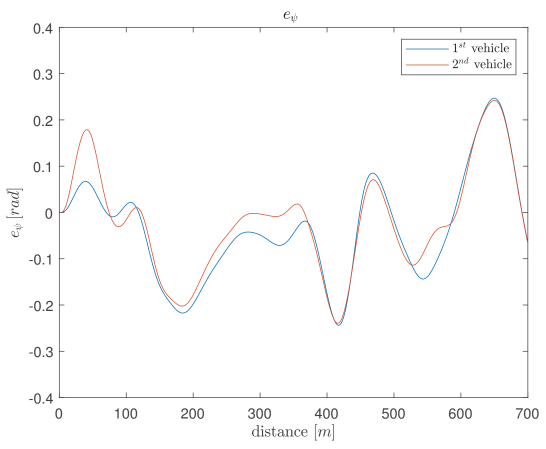
(a) Overtaking for *delay 0.5 s* test



(b) Δt for *delay 0.5 s* test



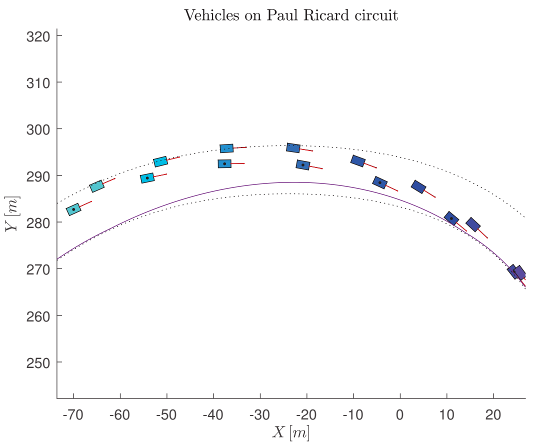
(c) e_{y_i} $\forall i = 1, 2$ for *delay 0.5 s* test



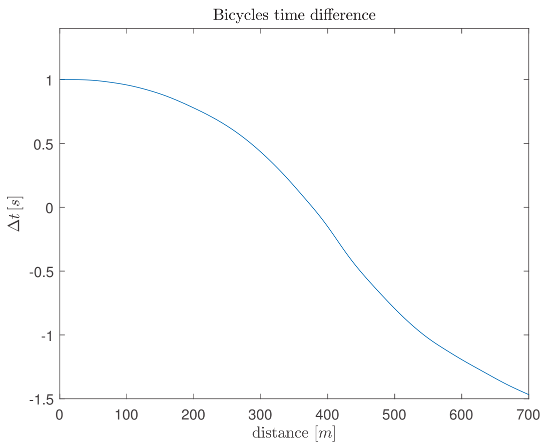
(d) e_{ψ_i} $\forall i = 1, 2$ for *delay 0.5 s* test

Figure 6.11: Results of *Delay 0.5 s* test on *Beausset* curve

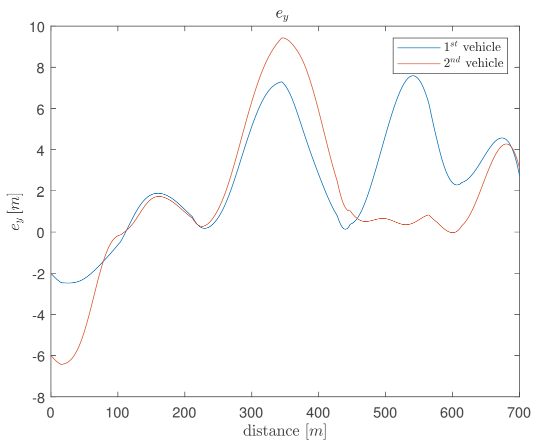
Delay 1 s test is performed by increasing the gap between the time weights to make the overtaking manegement possible. In particular, the first vehicle surpasses the second one at 370 *m* (figures 6.12a 6.12b). In this latter case, the vehicles have similar lateral errors to the *Paired departure* test (figures 6.12c and 6.12d).



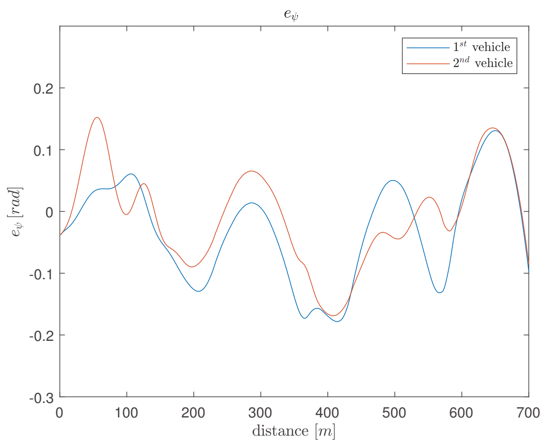
(a) Overtaking for *delay 1 s* test



(b) Δt for *delay 1 s* test



(c) $e_{y_i}\ \forall i = 1,2$ for *delay 1 s* test



(d) $e_{\psi_i}\ \forall i = 1,2$ for *delay 1 s* test

Figure 6.12: Results of *Delay 1 s* tests on *Beausset* curve

Test type	*Paired departure*	*Delay 0.5 s*	*Delay 1 s*
Reference strategy	*Curve cutting*		
Parameters	Numeric values		
Average CPT $[ms]$:	28.9143	29.2132	29.0886
Maximum CPT $[ms]$	54.7951	53.6551	55.1383
user vehicle maneuver time $[s]$	16.1139	16.6396	17.0544
adv. vehicle maneuver time $[s]$	18.7336	17.1901	18.5243
user vehicle max speed $[m/s]$	56.372	55.4222	56.0827
adv. vehicle max speed $[m/s]$	50	53.7742	50

Table 6.5: Performances about *Beausset* curve tests

6.3 Overtaking considerations

The analysis of overtaking deserves a brief description. In fact, the overtaking manegement rappresents the interaction between *user vehicle* and *adversarial vehicle*. This phenomenon depends on several elements. One determining factor is the shape of the road. In fact, it is much more simple to achieve overtaking on a straight stretch than on a curve.
For example, the *Paired departure* test developed on the entire circuit (Sec 6.1) is very different from the *first Chicane* curve one (Sec 6.2.1). In fact, in the former case the vehicles start with a specific initial speeds and positions (see x_0 definition in Sec 4.2). Then, both vehicles increase their speeds in the straight section where *user vehicle* immediately performs the overtaking. Succesively, they enter on the first curve according to well defined trajectories (computed by the NMPC controller). In particular, at that location, they are one behind the other and the state value is very different from x_0 of the *Paired departure* test on the *first Chicane* curve. In fact, for this test the dynamics of the system is computed differently by the controller. This is because they start paired and they have different tracking errors from those calculated by the controller in the previous case. Also the velocities that they take on that curve is different and this fact impacts on performances.
The vehicle initial conditions are another important element. In fact, if they are changed, then the controller calibration has to be modify. In particular, the calibration reported in Table 6.2 does not achieve the vehicles manegement in all the cases, especially for *Delayed departure* tests (which consider different x_0 value than *Paired departure* ones). Those tests are harder to solve than *Paired departure* ones. The reason is that in the second case, if the vehicles initial relative distance is bigger than d_{min}, and one vehicle is faster than the other, then the overtaking starts immediately (even with higher errors weights). Instead, the first case is much more challenging because the high-performing vehicle (*user vehicle*) is positioned behind the other one and must achieve it. Sometimes it can be useful to make *user vehicle* even faster and *adversarial vehicle* even slower.
Finally, also the vehicles initial positions impact on the overtaking manegement. For example, on *L'Ecole* curve *user vehicle* is initially positioned externally with respect to the curve. This proved to be a major disadvantage for *Delayed departure* tests, since the faster vehicle was not able to achieve the goal. Perhaps, if the simulation path had been longer, then it would have been successful. But in any case, it would not have been able to surpass *adversarial vehicle* by the end of the curve.

Chapter 7

Fixed trajectory simulations

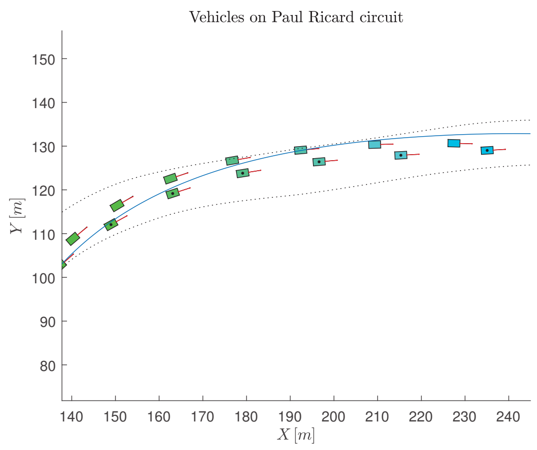
In the previous Chapter, both vehicles are driven by the Nonlinear Model Predictive Controller (NMPC) and in general, a different calibration is required to manage the vehicle behaviours.
The purpose of this Chapter is related to test the developed NMPC controller, as it acts as an adversary to another vehicle that it cannot control (because it simulates an human driven vehicle). In other words, during the previous simulations, the vehicles predictive trajectories were provided by the same controller online. Now, the competition can be implemented by considering a predetermined fixed trajectory for the first vehicle. Namely, all the data related to the dynamics of the *user vehicle* are loaded in a new simulation scenario. Then, if the calibration of *user vehicle* is manually changed, then the NMPC controller could see a different evolution of the system with respect to what it had computed.
For example, if q_{t1}, q_{ey_1} and $q_{e\psi_1}$ are increased [1], then the controller tries to reduce the maneuver time and the tracking errors, but this ineffective since it can not control the first vehicle. In fact, during the simulation the controller has to deal with an autonomous *user vehicle* (whose trajectory is fixed) which is slower and less precise with respect to what the controller expected. In this case, *MATMPC* could properly control only the second vehicle, i.e. to avoid collisions and to possibly allow the overtaking. On contrary, if q_{t1}, q_{ey_1} and $q_{e\psi_1}$ are reduced, then the controller will see that the first vehicle will be faster and will trust on the reference more than what the controller predicted. As a consequence, the overtaking could be either postponed or anticipated along the simulation path. Moreover, the overtaking can be performed in a different way depending on the *adversarial vehicle* trajectory.
Taking into account the results about single curves tests (described in Sec 6.2), those simulations will be repeated under framework of fixed trajectories applied to *user vehicle*.
In the following sections, only those simulations that show a very different overtaking maneuver along the curve will be described. For example, *l'Ecole* curve tests will not be considered because the results are very similar to those reported in Sec 6.2.2. This happens since the curve shape, the curve length and the initial locations of vehicles have significantly impacted on overtaking manegement. As a consequence, even forcing the first vehicle to be faster, it could not overcome *adversarial vehicle* under *Delayed departure* tests.

[1]Sec 4.2 describes some possible values of weigths bounds

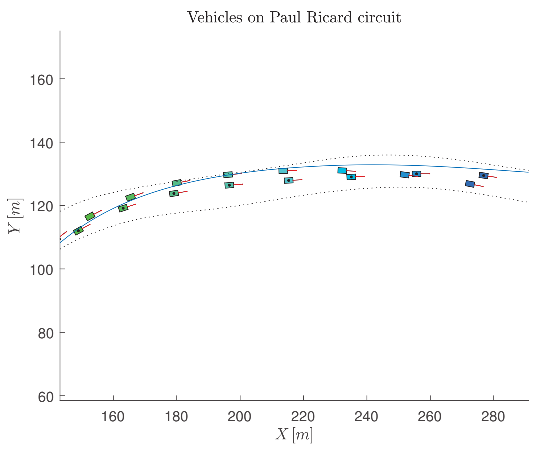
Concerning to *Paired departure* tests, they are not even reported because the overtaking takes place in the same previous locations [2]. This happens because *user vehicle* has already a time advantage when the competition starts. Conversely, *Delayed departure* tests show a very different evolution of the system for *adversarial vehicle* (the one that is actually controlled by *MATMPC*). Moreover, with respect to the original tests, the anticipation and postponement of the overtaking is much more visible.

7.1 First Chicane curve

Starting from the *first Chicane* curve, *Delay 0.5 s* test shows that if q_{t1} is reduced with respect to the original case (no matter if it is bigger or smaller than q_{t2}), then the overcoming happens at 238 m instead of 250 m (figures 7.1a and 6.4a). In particular, the controller detects the presence of *user vehicle*, which is going faster than what it predicted. So *adversarial vehicle* moves to the left earlier. It remains externally to let *user vehicle* to complete the overtaking by cutting the curve internally. Instead, if q_{t1} is increased then the overtaking happens later than before, at 270 m (figure 7.1b).
Delay 0.5 s test is repeated according to a similar procedure. This time, $q_{e_{y}1}$ and $q_{e_{\psi}1}$ weigths are reduced so that another different competition is carried out (figure 7.1b). In fact, the overtaking is anticipated at 213 m along the second curve, where the controller drives *adversarial vehicle* so that it turns externally. In this way, it allows *user vehicle* to complete the desired maneuver. In this situation, the controller detects that the first vehicle follows the *Curve Cutting* reference more than what it expected, and this result confirms that the overtaking is not ensured by having only a bigger speed. In fact, also the manner in which the curve is crossed impacts on performances. If the procedure is repeated with bigger $q_{e_{y}1}$ and $q_{e_{\psi}1}$ weigths, then the overtaking is a bit postponed (figure 7.2b).



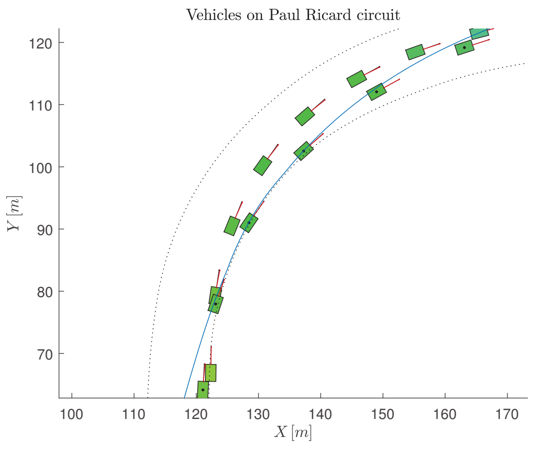
(a) Overtaking with smaller q_{t1}



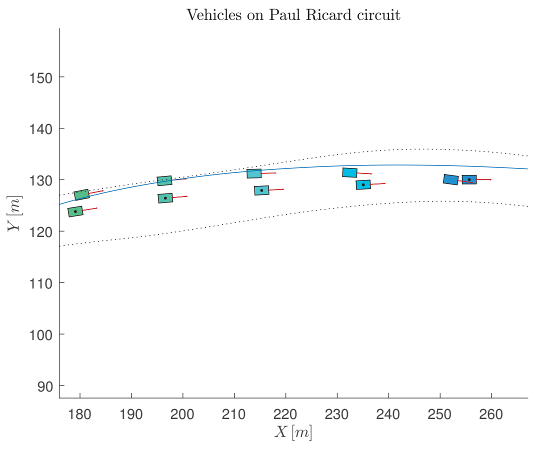
(b) Overtaking with bigger q_{t1}

Figure 7.1: Results of *Delay 0.5 s* test on *Chicane curve*

[2]Since they are *Paired departure* tests the overtaking cannot be anticipated since *user vehicle* immediately starts to surpass. Regarding to postponed overtakings, they happened a few meters later than the original ones.



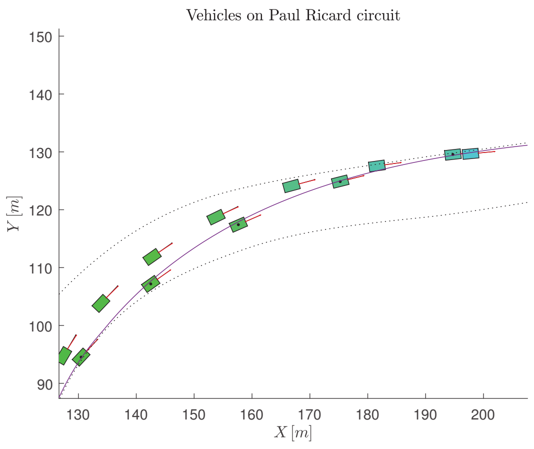
(a) Overtaking with smaller $q_{e_{y1}}, q_{e_{\psi 1}}$



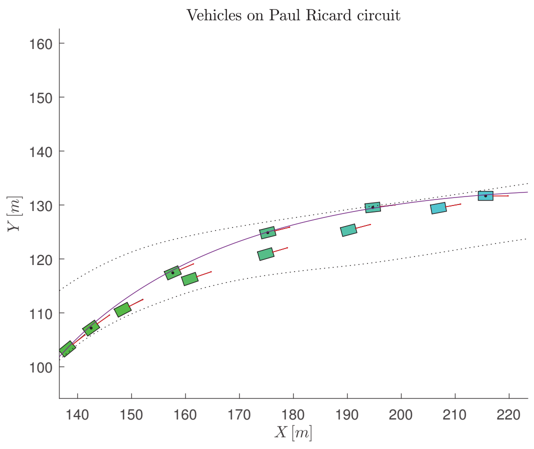
(b) Overtaking with bigger $q_{e_{y1}}, q_{e_{\psi 1}}$

Figure 7.2: Results of *Delay 0.5 s* test on *Chicane curve*

Recall that the original calibration was changed to ensure overtaking in *Delay 1 s* test on the same curve (see Sec 6.2.1). So that *user vehicle* surpassed *adversarial vehicle* at 216 m.
As for *Delay 0.5 s* test, also *Delay 1 s* test is carried out with a fixed trajectory for *user vehicle*. Again, different calibrations of the tracking errors and the lap time lead to have a different *adversarial vehicle* paths. Each result is reported in figures 7.3a, 7.3b, 7.4a and 7.4b. Rispectively, the overtaking is anticipated for the first case, postponed for the second and third cases. For the last case, it is carried out similarly as in the original one. Notably figures 7.3b and 7.4a present a different *adversarial vehicle* maneuver where it moves in the opposite direction with respect to the original case.

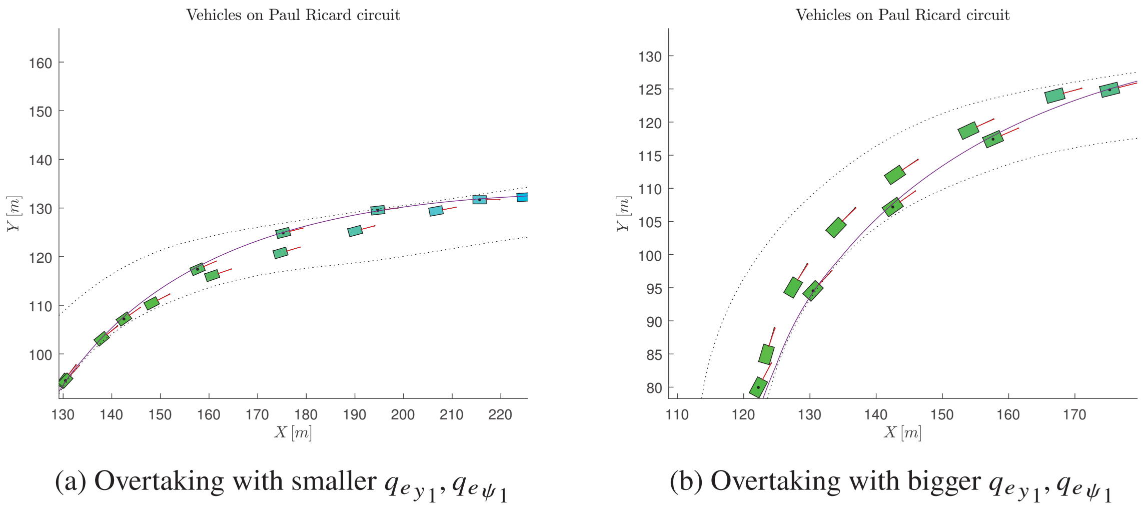


(a) Overtaking with smaller q_{t1}



(b) Overtaking with bigger q_{t1}

Figure 7.3: Results of *Delay 1 s* test on *Chicane curve*



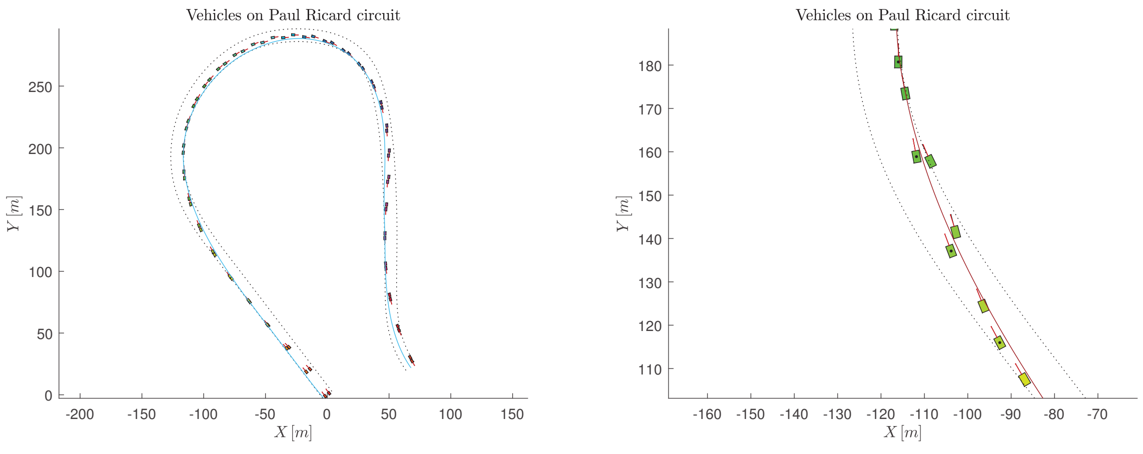
(a) Overtaking with smaller $q_{e_y 1}, q_{e_\psi 1}$ (b) Overtaking with bigger $q_{e_y 1}, q_{e_\psi 1}$

Figure 7.4: Results of *Delay 1 s* test on *Chicane curve*

7.2 Beausset curve

Reffering to original *Delayed departure* tests on *Beausset* curve (Sec 6.2.3), the entire previous procedure is repeated.

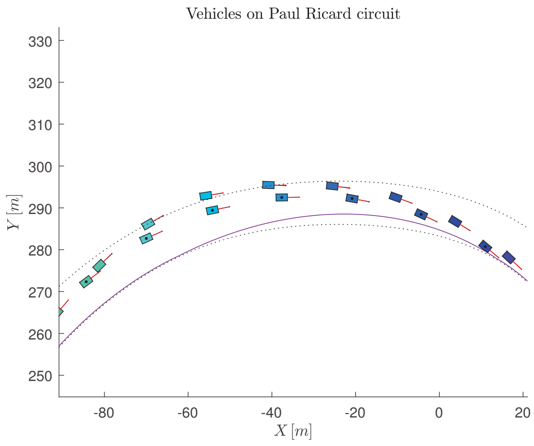
Considering *delay 0.5 s* test and modifying the weigths, two cases turn out to be very different from the original ones. Figure 7.5a presents no overtaking when q_{t1} is increased and tracking errors weigths are reduced. In particular, the controller forces an higher *adversarial vehicle* speed because it believes that *user vehicle* is too slow. So that the second vehicle finishes its maneuver with 16.5775 s versus the first vehicle one of 16.6396 s.
The second case takes into account the opposite calibration (figure 7.5a). As a consequence, *user vehicle* is faster with respect to what the controller planned and the overtaking is anticipated at 184 m (figure 7.5b). To accomplish such maneuver, the second vehicle changes its direction to leave sufficient space for *user vehicle*.



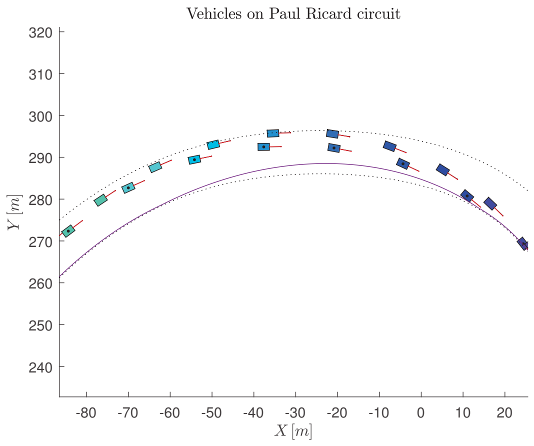
(a) No overtaking with smaller $q_{e_y 1}, q_{e_\psi 1}$ and bigger q_{t1} (b) Overtaking with bigger $q_{e_y 1}, q_{e_\psi 1}$ and smaller q_{t1}

Figure 7.5: Results of *Delay 0.5 s* test on *Beausset curve*

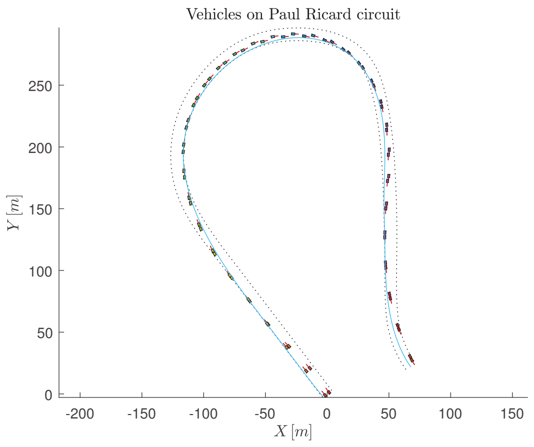
Regarding to *Delay 1 s* test, the most different scenarios with respect to the original ones are reported in figures 7.6a, 7.6b, 7.6c and 7.6d. In particular if q_{t1} is reduced then the overtaking is anticipated at 336 m. In the opposite scenario it is postponed a bit (at 380 m). Also the tracking errors weights are important. In fact, if they are increased then *user vehicle* may not be able to overcome *adversarial vehicle*. In particular, if it is too slow from the controller point of view (q_{t1} bigger), it fails to overtake. Otherwise, the overtaking is possible, but it takes place much later 477 m.



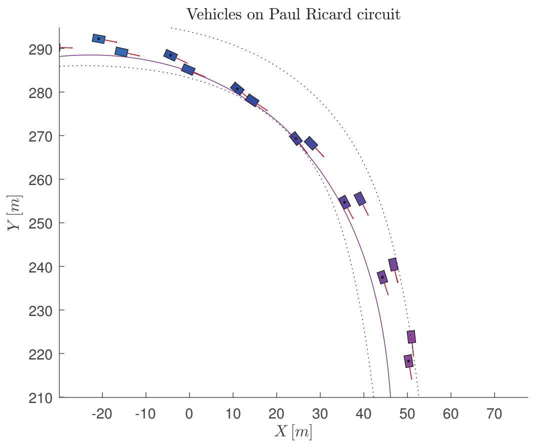
(a) Overtaking with smaller q_{t1}



(b) Overtaking with bigger q_{t1}



(c) No overtaking with smaller $q_{e_{y1}}, q_{e_{\psi 1}}$ and bigger q_{t1}



(d) Overtaking with smaller $q_{e_{y1}}, q_{e_{\psi 1}}$ and smaller q_{t1}

Figure 7.6: Results of *Delay 1 s* test on *Beausset curve*

Chapter 8

Conclusion

The purpose of the project is to realize a tool based on a NMPC controller that behaves like an adversary to *user vehicle*. To achieve this goal many steps were performed, which will be briefly described below.
First of all the dynamics of the vehicle were modeled as those of the bicycle model [15], [19]. This model choice was made by taking into account both accuracy and the computational complexity required to solve the problem. In other words, the simplest vehicle model was adopted because during some tests the controller had to handle two vehicles at the same time.
The second step concerned the NMPC controller setup. NMPC aims to solve a well defined optimization problem that takes into account several factors. In particular, they are the model dynamics, the system constraints, the references and the cost function, which represents the future behaviour of the entire system that the solver attempts to minimize.
Given the complexity of the problem, in the first stage the analysis was performed for a single vehicle to test the best controller configuration. A benchmark configuration was found for the controller, which was able to properly control a single vehicle, i.e. to minimize the lap time and to ensure numerical robustness.
Then, two virtual vehicles were driven simultaneously by the same controller. At this point, the system proved to be a very complex problem to solve. In fact, the previous calibration was able to control the vehicles only under a *Paired departure* test conditions. As a consequence, the problem of making a controller that behaves like an opponent in the driving simulator cannot be solved by using a single calibration. It is precisely for this reason that the management of two vehicles was analyzed in very specific cases. In particular, by taking into account different curves and by considering different kind of tests depending on the initial conditions of the vehicles, i.e. intial time istants and initial positions with respect to the curve.
Finally, the previous tests were repeated with a fixed trajectories that aimed to verify the controller's response to the unexpected behaviour of *user vehicle*. In particular, to simulate a real behaviour of a human-driven vehicle, *user vehicle* had a fixed predetermined trajectory. As a consequence, the controller could only act on *adversarial vehicle*. In this way, NMPC behaved like an opponent in a driving simulation. Performances changed significantly if the trajectory calculated by the controller was very different from that of the *user vehicle*. In particular, the

overtaking resulted anticipated, postponed and sometimes dealt with a different way with respect to the original simulations. For example, if q_{t1} is reduced, then the controller detected a vehicle which was faster than what it planned. So the controller was forced to move *adversarial vehicle* earlier to avoid collision. Even the modification of tracking error weights gave different results from the original simulation tests. Thus, not only the speed but even the way a vehicle entered on a curve could greatly influence the *adversarial vehicle* maneuver (which was actually driven by the controller). As a result, the overtaking was handled differently from the original case.

In conclusion, the realization of an adversarial driver is very complex. The vehicles manegement has to take into account a variable configuration of the controller for any different section of the circuit. In fact, a single calibration cannot achieve the goal for any possible scenario. However, a good calibration strategy can be described as a starting point for addressing the problem in any situation. In particular, the tracking errors weigths have to be set sufficiently small. In this way both vehicles are able to move away from the reference in order to perform the overtaking. Moreover, also the time and the acceleration weigths have to be selected carefully. In fact, they allow to command a vehicle to be even faster or even slower.

In a future perspective, a possible solution to the problem is to analyze the behavior of vehicles in each individual sector of the track to ensure an overtaking.
Another future development could be referred to a different model plant. In particular, a four wheels vehicle with 7 dof can be taken into account [24],[18], with a *Pacejka tire model* [16] before analyzing more complex models. Moreover, this model could represent the behaviour of an autonomous driving simulator better than bicycle model. On the other hand, the four wheels vehicle model is not too complex as the 14 dof models. In particular, the possibility to solve the Nonlinear Programming Problem (NLP) with this model type could still satisfy a real time feasibility. In fact, even with two bicycle models, the computational burden is not too high (about 30 ms on average).
An additional development could be the cost function modification that represents the future behaviour of the process to be controlled. In the first instance, an accelerating/breaking coefficient β could be used as input [2] instead of a longitudinal acceleration term. Also the lateral error distance of each steering wheel from the reference could be taken into account. Consequently, the general constraint should consider the bounds of these errors.
Another possibility to solve the problem is to focus on *SQP* algorithm to solve the NLP problem without RTI scheme. This will lead to neglect a real time property, but it could be useful to approach an optimal solution instead of a suboptimal one.

Appendix A

Additional Tests

A.1 Additional single vehicle tests

In this section only the tests related to the single vehicle maneuver are carried out after setting NMPC controller properly (see Sec 4.3). In particular, the simulations of *Centerline* and *No tracking errors reference* strategies will be reported.

A.1.1 Centerline reference strategy

Figures A.1 and A.2 show the simulation path of the single vehicle *Centerline* strategy. The maneuver results similar to the *Curve Cutting* one. In fact, the same calibration is adopted with small tracking errors weights and by selecting a very high value for the time weight. However, the resulting maneuver time is bigger than the *Curve Cutting* one.

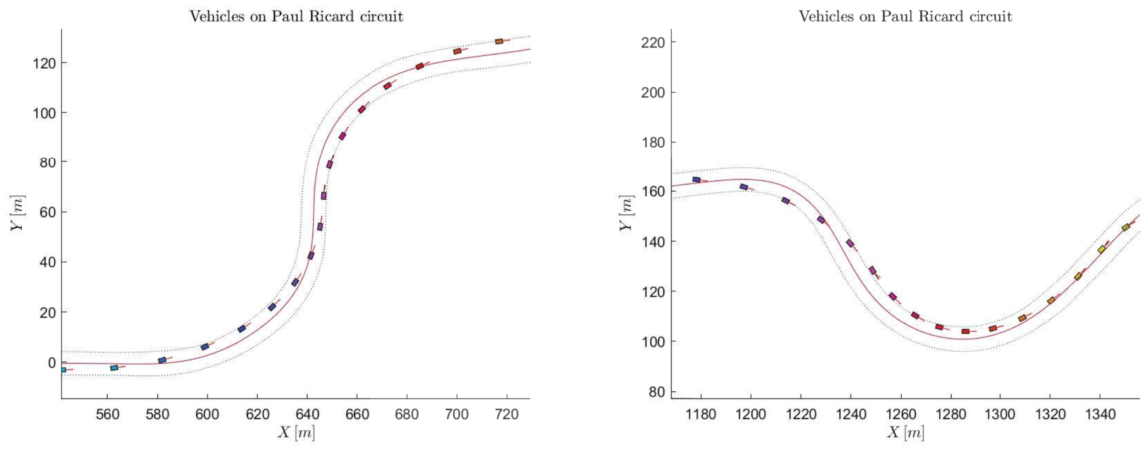


Figure A.1: Simulation of a single vehicle with *Centerline* strategy

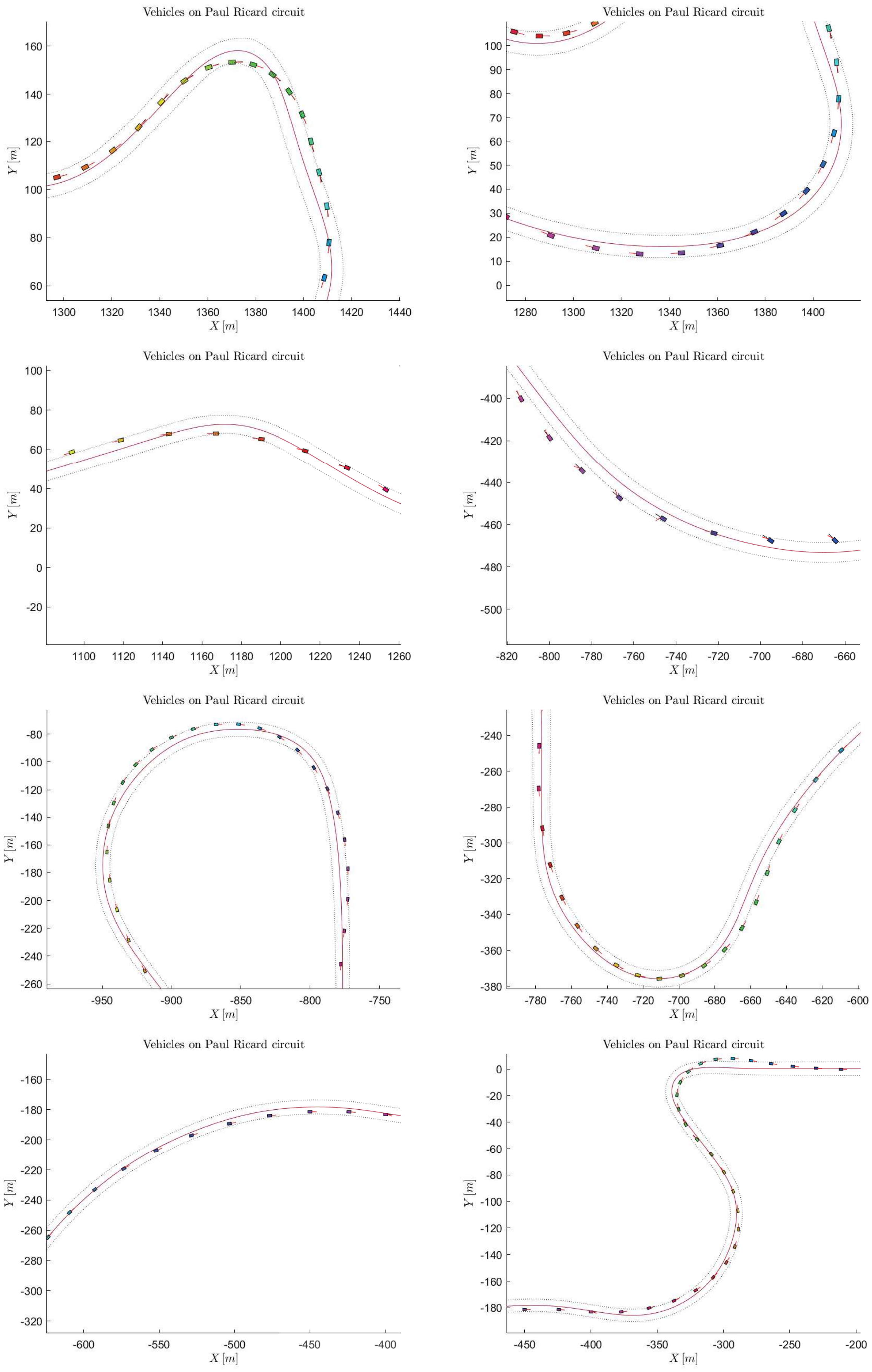


Figure A.2: Simulation of a single vehicle with *Centerline* strategy

A.1.2 No tracking errors reference strategy

As mentioned in Chapter 5, due to the lack of numerical robustness, the same *Curve Cutting* weigths: q_t, $q_{\dot{\delta_f}}$, $q_{\dot{a_x}}$, q_{ϵ_y} q_{ϵ_g} and q_{tN} cannot be adopted also for this strategy. Thus, the following path is 2 s longer than the *Curve Cutting* one. This result is reported in figures A.3 and A.4.

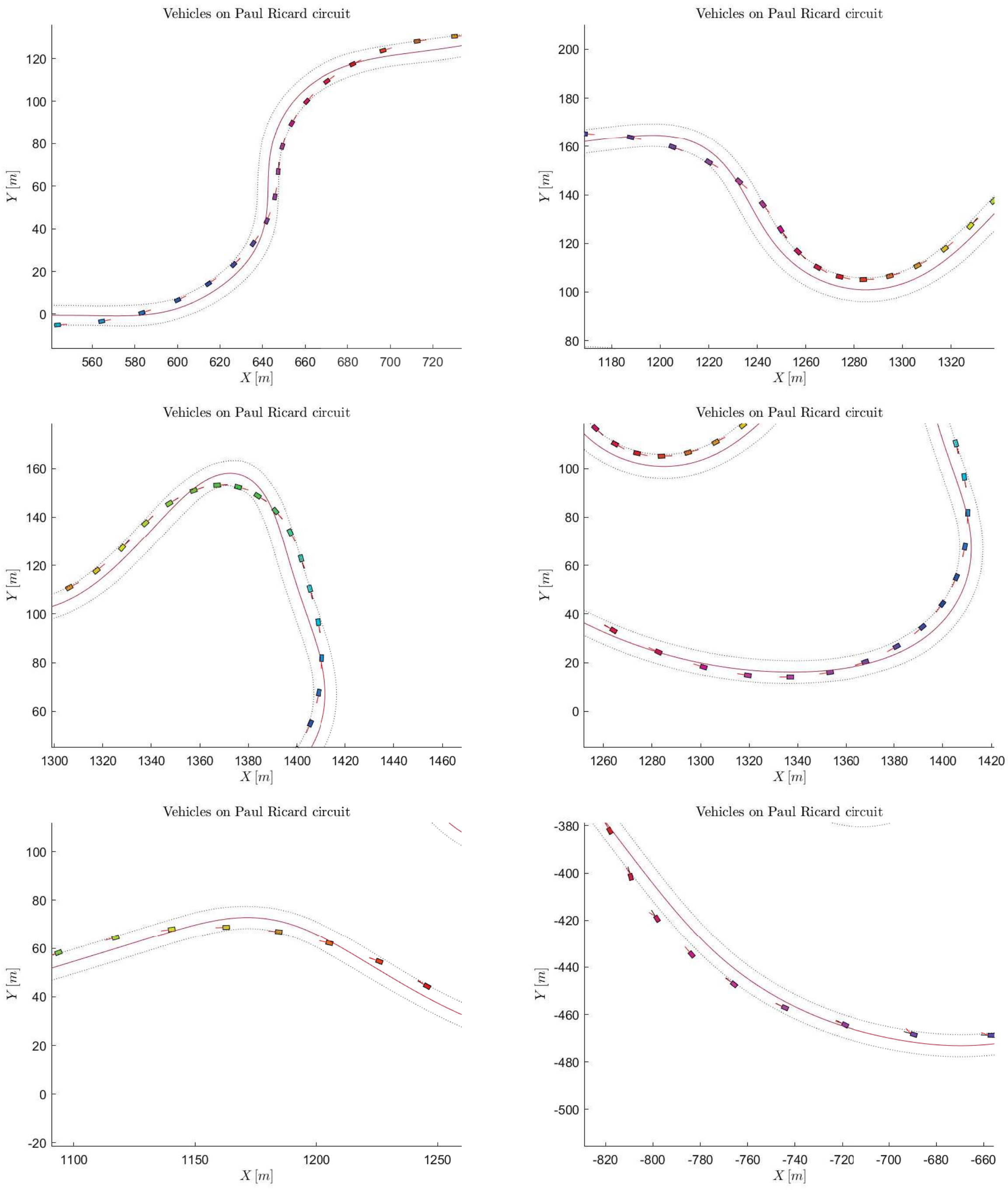


Figure A.3: Simulation of a single vehicle with *No tracking errors reference* strategy

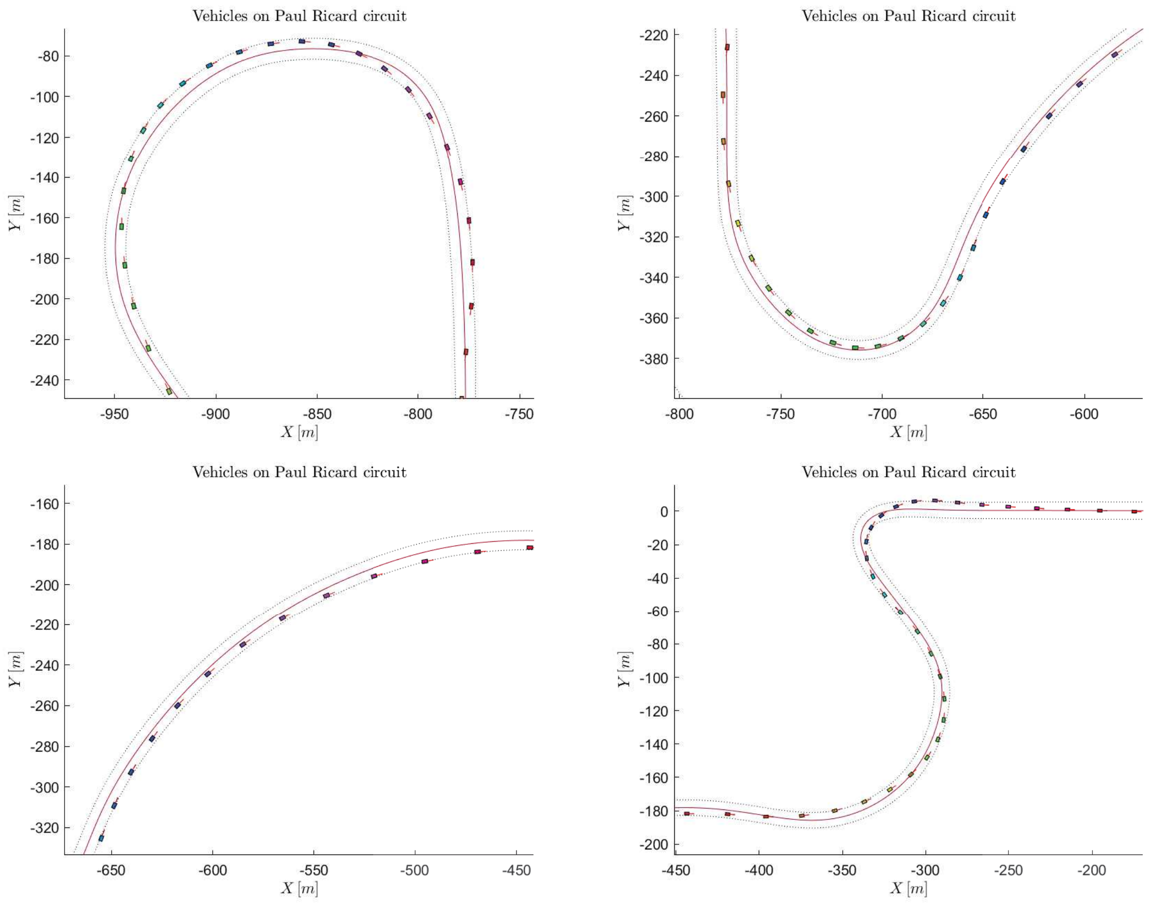


Figure A.4: Simulation of a single vehicle with *No tracking errors reference* strategy

A.2 Paired departure for two vehicles

Referring to Sec 6.1, the simulation paths on the entire circuit of two vehicles for *Paired departure* test are exposed in this section.

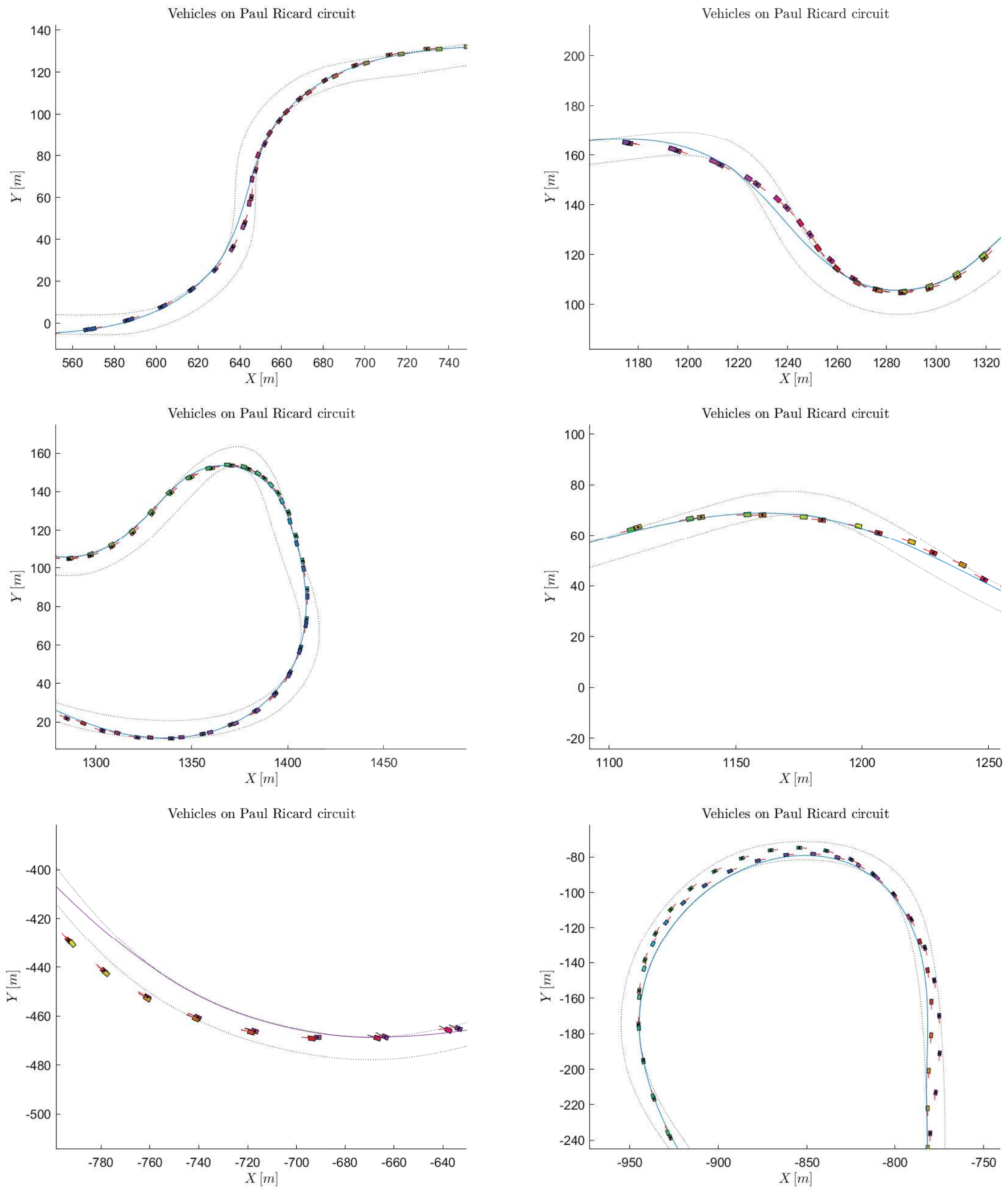
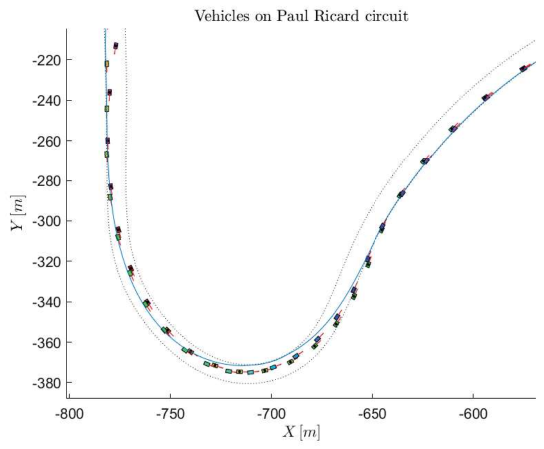


Figure A.5: Simulation of two vehicles with *Curve cutting* strategy



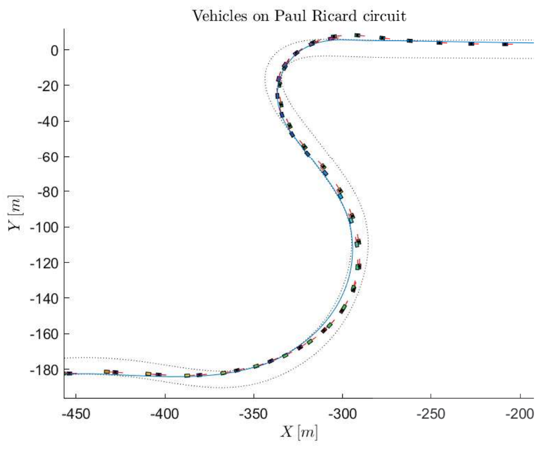


Figure A.6: Simulation of two vehicles with *Curve cutting* strategy

Bibliography

[1] F. Allgöwer, F. Rolf, and C. Ebenbauer. Nonlinear model predictive control. In *IEE Proceedings. Control Theory & Applications*, volume 152, pages 257–258. IEE, 2005.

[2] M. Bruschetta, E. Picotti, E. Mion, Y. Chen, A. Beghi, and D. Minen. A nonlinear model predictive control based virtual driver for high performance driving. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*, pages 9–14, 2019.

[3] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli. Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2335–2340, 2013.

[4] Y. Chen. *Algorithms and Applications for Nonlinear Model Predictive Control with Long Prediction Horizon*. PhD thesis, University of Padua, 2018.

[5] Y. Chen. Matmpc. https://github.com/chenyutao36/MATMPC, 2018.

[6] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi. Matmpc - a matlab based toolbox for real-time nonlinear model predictive control. In *18th European Control Conference (ECC)*, pages 3365–3370, 2019.

[7] A. Daniel. Controlling the level of sparsity in mpc. In *Systems and Control Letters*, volume 76, pages 1–7, 2015.

[8] G. Frison. Hpipm. https://github.com/giaf/hpipm, 2017.

[9] G. Frison and M. Diehl. Hpipm: a high-performance quadratic programming framework for model predictive control. In *Elsevier*, volume 53, pages 6563–6569, 2020.

[10] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. 2010.

[11] C. Ghike and T. Shim. 14 degree-of-freedom vehicle model for roll dynamics study. 04 2006.

[12] T. Gillespie and I. SAE. *Fundamentals of vehicle dynamics*, pages 97–98. Society of Automotive Engineers, 1992.

[13] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli. Predictive control for agile semi-autonomous ground vehicles using motion primitives. In *2012 American Control Conference (ACC)*, pages 4239–4244. IEEE, 2012.

[14] T. Keviczky, P. Falcone, F. Borrelli, J. Asgari, and D. Hrovat. Predictive control approach to autonomous vehicle steering. In *2006 American Control Conference*, volume 15, pages 6 pp.–, 2006.

[15] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.

[16] E. Kuiper and J. Van Oosten. The pac2002 advanced handling tire model. In *Vehicle System Dynamics*, volume 45, pages 153–167, 2007.

[17] A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36:628–647, 09 2015.

[18] E. Picotti. Controllo predittivo non lineare di un veicolo virtuale. Master's thesis, University of Padua, 2018.

[19] R. Rajamani. *Vehicle dynamics and control*, pages 20–31. Springer Science & Business Media, 2011.

[20] ScienceDirect. Magic formula. https://www.sciencedirect.com/topics/engineering/magic-formula, 2015.

[21] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl. Towards time-optimal race car driving using nonlinear mpc in real-time. In *53rd IEEE Conference on Decision and Control*, pages 2505–2510, 2014.

[22] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl. Time-optimal race car driving using an online exact hessian based nonlinear mpc algorithm. In *2016 European Control Conference (ECC)*, number 141-147, pages 141–147, 2016.

[23] P. Wojciech and B. Tomasz. Dynamic model of a logistic train with different steering systems and tire models. In *SciELO Brazil*, volume 18, pages 1–27, 2021.

[24] I. Youn, E. Youn, M. A. Khan, L. Wu, and M. Tomizuka. Combined effect of electronic stability control and active tilting control based on full-car nonlinear model. In *The Dynamics of Vehicles on Roads and Tracks: Proceedings of the 24th Symposium of the International Association for Vehicle System Dynamics (IAVSD 2015), Graz, Austria, 17-21 August 2015*, pages 345–347. CRC Press, 2016.