POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in
Ingegneria Meccanica

Model Predictive Control for an Autonomous Vehicle

Relatore: Prof. Francesco BRAGHIN

Tesi di Laurea di:

Nicola DE VAL	Matr. 779111
Andrea FUSO	Matr. 783291

Anno Accademico 2012 - 2013

Contents

Abstract

The purpose of this thesis is to design a Model Predictive Control based Motion Planner unit for an Autonomous Vehicle. The unit should provide the trajectories of the inputs of the vehicle such that certain references are followed and, at the same time, fixed and moving obstacles are avoided safely.

The first controller studied is an Active Front Steering system, where the input is only the steering angle. The controller is a linear model predictive control based on linearization of the nonlinear vehicle model and it was used to thoroughly investigate its limits, especially for what concerns the purpose of avoiding obstacles.

This linear controller is then enhanced including also the control of the brake and throttle pedals. This allows us to further delineate properties and defects of the linear controller.

Once the limits of the linear controller have been understood, the design of the nonlinear model predictive control, again dedicated to define the trajectories of the steering angle and of the brake and throttle pedal positions, has been carried out. We have then challenged it with fixed obstacles, with blind alleys and with moving obstacles. Also an extensive analysis on the controller sensitivity for what regards obstacle shapes, optimization algorithm and optimization initial conditions has been carried out in order to define the best setup.

Keywords

Model Predictive Control, Motion Planner, Obstacle avoidance systems, Autonomous vehicle.

Sommario

Lo scopo di questa tesi è quello di progettare un'unità Motion Planner basata sul Model Predictive Control da poter essere utilizzata su un veicolo autonomo. L'unità dovrà provvedere a fornire le traiettoria dei controlli del veicolo in modo tale da permettere di seguire determinati riferimenti e, allo stesso tempo, di evitare ostacoli sia fissi che in movimento.

Il primo controllore studiato è un sistema Active Front Steering, dove l'input del sistema è solamente l'angolo di sterzo. Il controllore è un Model Predictive Control lineare basato sulla linearizzazione del modello non lineare del veicolo, ed è stato usato per indagare a fondo i suoi limiti, specialmente per quanto riguarda la capacità di evitare ostacoli.

Questo controllore lineare è stato quindi sviluppato in modo tale da includere anche il controllo del pedale del freno e dell'acceleratore. Questo ha permesso di definire più approfonditamente i pro e i contro di questo controllore.

Infine si è progettato il controllore Model Predictive non lineare, volto a definire le traiettorie dell'angolo di sterzo e della posizione del pedale del freno e dell'acceleratore. Sono state quindi valutate le sue performance nei confronti di ostacoli fissi nello spazio, di vicoli ciechi e di ostacoli in movimento. Inoltre è stata condotta un'analisi dettagliata in funzione della forma dell'ostacolo, degli algoritmi di ottimizzazione e delle condizioni iniziali della ottimizzazione.

Parole chiave

Model Predictive Control, Motion Planner, Sistemi di aggiramento ostacoli, Veicolo autonomo.

List of Figures

Chapter 1

State of the art

In the last few years the world of mechatronics developed interest in the topic of fully autonomous vehicles. The attention in vehicles was then only based on driver assistance, while now it is more about creating completely independent cars.

1.1 Controlled vehicles overview

1.1.1 Semi autonomous systems

The term semi autonomous is used to indicate those systems that are capable of governing one or more particular actions, but need also the presence of an operator to cope with any other needs. This type of systems are simpler with respect to completely autonomous ones, and this is the reason why these systems were used in the past. The main difference is due to the measurement system. In fact semi autonomous systems manage only a small part of the whole system, and therefore need measurements and informations only about that particular part. For example, a cruise control needs only information about the speed of the vehicle, while a fully autonomous vehicle needs precise informations about the whole state of the vehicle and also information about the environment. Other examples of semi autonomous systems applied to vehicles are the well known Antilock Braking System (ABS) or the Electronic Stability Control (ESP) systems, but also more complicate Active Front Steering (AFS) systems were studied. Little by little every mechatronic system, although being semi autonomous, started to become more and more complicated and invasive. At the beginning of this year Toyota [11] announced a vehicle that is capable of autonomously avoid collisions. The system is still to be considered semi autonomous since the vehicle, in spite of being very sophisticated especially in the measurement system, is not able to reach destinations on its own. However the measurement system and the driving capabilities of the car over a short track are absolutely commensurate to the much more famous and fully autonomous Google car.

1.1.2 Autonomous systems

On the other hand fully autonomous system are, as already said, a lot more complicated. Another big difference between the two types, in addition to the complexity of measurement system that sometimes (as for the Toyota) might be comparable, is in the management of references or paths. Semi autonomous systems in fact operate the system only over a short track. A fully autonomous system instead must guide the vehicle to a known destination: it must manage stops, starts and, of course, the unexpected. Until a few years ago fully autonomous vehicle were only slow driving Automated Guided Vehicle (AGV). Only in the last decade the interest on medium-high speed applications started growing. In 2007 the Defense Advanced Research Projects Agency (DARPA) organized the DARPA Urban Challenge where fully autonomous vehicles were challenged in a course filled with typical urban obstacles. The winner of that race, the car named Boss, was driven by a system composed of three different control layers: the Mission layer creates the reference trajectory, or global path, depending on a known destination; the Behavioural layer is specialized in managing intersections and lane changes; finally the Motion Planning layer defines a local trajectory taking into account obstacles and informations from the upper layers. The Motion Planning is also the part of the control logic that actually drives the car, [1].

1.1.3 Model Predictive Control

The Model Predictive Control was developed to manage production plants of chemistry factories or refineries in the early 80s. Based on a dynamic model of the system, the controller makes a prediction of the outputs and therefore defines the values of the manipulated variables such that a certain cost function is minimized. Typically this functions is the error with respect to defined references. One of the major advantages of this type of control is, in addition to the capability of optimizing performances, the capability of managing constraints both in the manipulated variables and in the system outputs. An example could be the need to drive a valve that is only capable to open up to a certain value of flow (constraint on the manipulated variable), such that the system obtains a certain level of pressure but not over certain value (constraint on the system output).

A prediction horizon is required, which is the time interval that is considered for one optimization. On the top of that also a control horizon is set such that the control needs to calculate the control actions only for a shorter period of time, thus reducing the computational cost. Once the control actions for the future horizon have been defined, the control action is only applied for the first step. Then the optimization process is repeated. That is called receding strategy and allows to cope with modelling errors and unexpected changes in constraints.

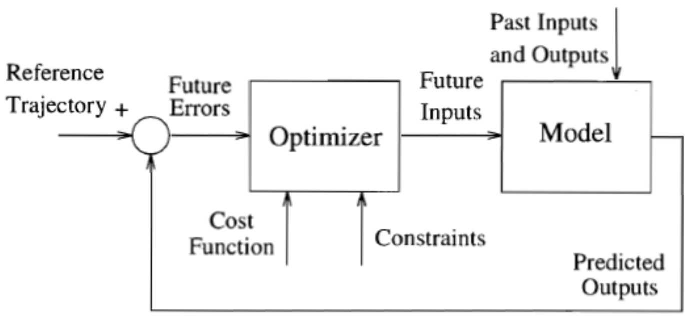

Figure 1.1: MPC basic structure

Model Predictive Control (MPC) for both linear and non-linear systems were studied, the second ones called Non linear Model Predictive Control (NMPC). We must say that the linear MPCs significantly reduce the computational time. In fact for a linear model it is very easy and fast to extrapolate a future prediction, even far, and, writing the optimization problem as a quadratic problem, it is very easy to find solvers (for example quadprog in Matlab) that are able of solving said problems in a very short time. NMPCs apply exactly the same concepts of linear MPCs, but both the prediction and the solver require higher computational costs to be solved by standard computers.

The control of industrial process is guaranteed by the very slow dynamics of such systems. Only recently, with the increase of the processing power NMPCs have applied to systems having faster dynamics.

1.1.4 MPC guided vehicles

MPC is capable of managing only aspects very close to the real driving, i.e. at the Motion Planning layer. In some papers MPC is applied to only few aspects of the driving; i.e. only the steering is taken into account. In [2] and

[3] we find the first interesting applications of AFS obtained with a MPC. In these examples a known trajectory is provided to the controller (thus the control is not capable of taking into account obstacles) with the goal of verifying the stability of the vehicle in particular conditions. In [2] the attention is focused on the comparison between NMPCs and Linear Time Variant (LTV)-MPCs while managing icy roads. In [3] the research is focused on stability with the vehicle forced by lateral wind with NMPC only. We can find evolutions of these controls in [4] where two different levels of MPC controllers were used. The first one with the task of creating a reference trajectory, using a simplified mass-point vehicle with a NMPC, while the second one with the task of driving the non-linear vehicle model with a LTV-MPC. Although this tasks division allows a faster controller and thus a faster real-time implementation, obstacles were not taken into account and therefore it is not known if the first level, using a simplified dynamic system, is capable of creating feasible trajectories for the second level to follow. Moreover with these simplification we are losing the concept of optimization, in fact the trajectory is not generated for the reason being the optimal one for the vehicle, but from a simplified model that is very far from the real one.

It is clear that those systems are not useful enough to create a fully autonomous vehicle. In fact they are missing the control for the longitudinal dynamics of the vehicle, i.e. accelerator and brake pedal, thus missing to take into account the joint lateral and longitudinal dynamics. The only papers that consider at least the braking capabilities of the vehicle are [5], [6] and [7] where the slip ratio notion is introduced. In [5] there is the hypothesis that a brake system capable of creating the four required slip ratios exists. The MPC is thus used to decide the four slip ratios and the steering to follow a known trajectory. We find an evolution of this system in [6] where also a model of the braking system is introduced. In another example, [7] uses steering and blocking differential to control the yaw of the vehicle. Although exercises for the longitudinal dynamic control exist, as in [9], nobody has never considered MPC capable of driving a vehicle using common inputs like steering, accelerator and brake pedals while avoiding obstacles at medium-high speeds.

We appreciated from the beginning the choice of the MPC. Thanks to the prediction based on the system dynamic model and thanks to the receding strategy, its behaviour is as close to a racing driver as possible: it knows the vehicle so well that it is able to predict the future state of the vehicle, but it is also capable of managing unforeseen events. A common PID controller is not capable of complying all these things and for these reasons we decided to use the MPC as driver for our vehicle.

1.2 Linear Discrete MPC theory

As we have already explained, the general design objective of model predictive control is to compute a trajectory of a future manipulated variable u in order to optimize the future behaviour of the plant output y. The optimization is performed within a limited time window by giving system information at the start of the time window. Being the optimization based on the model of the system, it is obvious that an accurate system model is required. Typically this is a very big problem with chemical plants, but in our case the vehicle model is accurate. There are different approaches to predictive control design. Each approach uses a different model structure. Among many we find step response, transfer functions and state space models. The more used and more adapt to cope with mechanical systems is the state space model.

Then there is the need to discretize the model of the system. The control is in fact created only to cope with steps. This is because the control has to obtain, through the optimization process, the actual values of the manipulated variables over the whole time window. If the control is discrete, it means that the optimization process has to provide, step by step, the values of the manipulated variables. So, for example, a $2s$ time window with $50ms$ steps results in 40 different manipulated variable values that the control has to determine. Now if the control is not discrete, it means that the optimization process should give us a continuous signal for the manipulated variables and this, even though it can be made using primitive functions, is not typically applied because the optimization process, already very tricky, becomes more more complicated.

At this point we have explained the basic idea behind the control and we have introduced the essential parameters for his functioning. To clarify a little better, these are the fundamental components of our control:

- Discretization step ts;
- Prediction horizon n_y;
- Control horizon n_c;

During a single optimization the discretization step represents how often there will be a change in the control forces. For this reason the controller can only act every ts seconds, for example every $50ms$. Therefore the optimization result will consist in the values of the manipulated variables every ts seconds. As soon as the optimization is completed, the controller applies only the first value of the manipulated variables. The system will then move for an interval equal to the discretization step and then the whole optimization process will be repeated.

The value of the prediction horizon defines how long will the time window be. Typically n_y is expressed as a number of steps. With this convention the following will be true: $W_l = n_y \cdot ts$, where W_l is the time window length in seconds.

For what regards the control horizon, it is defined as the number of steps of the manipulated variables that are considered in the time window. In fact we may want to consider only the actions of the manipulated variables in the first part of the window to increase the speed of the optimization. In this way the optimizator will have only to obtain the first n_c values of the manipulated variables and then, for the rest of the window, it will consider them to remain constant.

The choice of these three parameters is of fundamental importance and strictly depends on the system under control. As suggested in [12], the length of the time window must be as long as the slowest dynamics of the system. In this way the control is able to consider and optimize control actions that allow the system to conclude the transient motion. Of course the longer the better, but that will effect the speed of the optimization process and it might be so slow that a real-time implementation will not be possible.

The value of the control horizon is, in the same way, to be kept as long as possible, but typically it is taken to be equal to half of n_y to allow faster optimizations. Bigger values of n_c means increasing the number of variables that the optimizator must handle.

While the time window length must be kept as long as possible to consider slow dynamics, the value of ts must be kept as short as possible to consider fast dynamics. It is easy to understand that, if the system has got very fast dynamics, a rough discretization will filter out part of his behaviour thus producing control aliasing and this will greatly affect the stability of the controlled system.

1.2.1 Augmented model

Now it is required to use the discretized mechanical system. From the continuous vehicle model we obtain the discretized model thanks to *c2d* matlab function that uses the Zero-order hold method [16]. If we write it in the state-space form for a Multi Input Multi Output (MIMO) linear system:

$$\begin{aligned} \vec{x}_m(k+1) &= A_m\vec{x}_m(k) + B_m\vec{u}(k) \\ \vec{y}(k) &= C_m\vec{x}_m(k) \end{aligned} \tag{1.1}$$

where $\vec{x_m}$ is the state vector, $\vec{y}$ is the vector of outputs of the system and $\vec{u}$ is the vector of the manipulated variables. It is important to point out that an MPC controller is capable of managing nonsquare systems (i.e. systems with different number of outputs different from the number of manipulated variables), but best results are obtained with square systems. In fact if we have more outputs than manipulated variables we must accept offsets in some of the outputs. On the other hand, if we have more manipulated variables than outputs, we will have more conditions that give us the same results and some set-up of the optimizator will be required in order to obtain reasonable results.

To guarantee the perfect achievement of the references the state-space model is modified so that an embedded integrator is included [13].

Subtracting two subsequent time instants, from (1.1), we obtain:

$$\vec{x}_m(k+1) - \vec{x}_m(k) = A_m(\vec{x}_m(k) - \vec{x}_m(k-1)) + B_m(\vec{u}(k) - \vec{u}(k-1))$$

that can be easily rewritten as:

$$\vec{\Delta x}_m(k+1) = A_m \vec{\Delta x}_m(k) + B_m \vec{\Delta u}(k) \tag{1.2}$$

It is important to notice that the inputs of this systems are $\vec{\Delta u}(k)$ and not $\vec{u}$. In the same way, for the outputs we have:

$$\begin{aligned} \vec{y}(k+1) - \vec{y}(k) &= C_m(\vec{x}_m(k+1) - \vec{x}_m(k)) = C_m \vec{\Delta x}_m(k+1) \\ &= C_m A_m \vec{\Delta x}_m(k) + C_m B_m \vec{\Delta u}(k) \end{aligned} \tag{1.3}$$

Combining (1.2) and (1.3) we obtain another state-space model.

$$\begin{aligned} \overbrace{\begin{bmatrix} \vec{\Delta x}_m(k+1) \\ \vec{y}(k+1) \end{bmatrix}}^{\vec{x}(k+1)} &= \overbrace{\begin{bmatrix} A_m & 0_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \vec{\Delta x}_m(k) \\ \vec{y}(k) \end{bmatrix}}^{\vec{x}(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \vec{\Delta u}(k) \\ \vec{y}(k) &= \overbrace{\begin{bmatrix} 0_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} \vec{\Delta x}_m(k) \\ \vec{y}(k) \end{bmatrix} \end{aligned} \tag{1.4}$$

where $0_m = \overbrace{\begin{bmatrix} 00\dots0 \end{bmatrix}}^{n_1}$ and n_1 is the length of the state vector. The triplet (A, B, C) is called the augmented model, that will be used in the design of the predictive control.

1.2.2 Linear prediction

Since we are now dealing with linear systems, the prediction can be analytically written.We assume that the whole state vector for each prediction is known. Thanks to this hypothesis the state vector at the time k is known. The future control trajectory is denoted by:

$$\vec{\Delta u}(k),\ \vec{\Delta u}(k+1),\ \dots,\ \vec{\Delta u}(k+n_c-1)$$

where n_c is the already introduced control horizon. The prediction of the state of our system starting from k, using the notion of [13], will be written as:

$$\vec{x}(k+1|k),\ \vec{x}(k+2|k),\ \dots,\ \vec{x}(k+n_y|k)$$

where n_y is the already introduced prediction horizon. Being the system a MIMO system, every $\vec{x}$ and $\vec{\Delta u}$ is a vector. Now considering the triplet (A, B, C) we can easily write the future prediction of the state vector of our linear system[1].

$$\begin{aligned}
\vec{x}(k+1|k) &= A\vec{x}(k) + B\vec{\Delta u}(k) \\
\vec{x}(k+2|k) &= A\vec{x}(k+1) + B\vec{\Delta u}(k+1) \\
&= A^2\vec{x}(k) + AB\vec{\Delta u}(k) + B\vec{\Delta u}(k+1) \\
\vec{x}(k+n_y|k) &= A^{n_y}\vec{x}(k) + A^{n_y-1}B\vec{\Delta u}(k) + \cdots + A^{n_y-n_c}B\vec{\Delta u}(k+n_c-1)
\end{aligned}$$

In the same way we can proceed with the outputs.

$$\begin{aligned}
\vec{y}(k+1|k) &= CA\vec{x}(k) + CB\vec{\Delta u}(k) \\
\vec{y}(k+2|k) &= CA\vec{x}(k+1) + CB\vec{\Delta u}(k+1) \\
&= CA^2\vec{x}(k) + CAB\vec{\Delta u}(k) + CB\vec{\Delta u}(k+1) \\
\vec{y}(k+n_y|k) &= CA^{n_y}\vec{x}(k) + CA^{n_y-1}B\vec{\Delta u}(k) + \cdots + CA^{n_y-n_c}B\vec{\Delta u}(k+n_c-1)
\end{aligned}$$

We can now define the matrices P_{xx}, H_x, F and Φ:

[1]It is important to remember that the triplet is not the discretized model, but the augmented version of it.

$$\underbrace{\begin{bmatrix} \vec{x}(k+1|k) \\ \vec{x}(k+2|k) \\ \vec{x}(k+3|k) \\ \vdots \\ \vec{x}(k+n_y|k) \end{bmatrix}}_{\underline{\vec{x}}} = \underbrace{\begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^{n_y} \end{bmatrix}}_{P_{xx}} \vec{x}(k) + \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ A^2B & AB & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ A^{n_y-1}B & A^{n_y-2}B & \dots & A^{n_y-n_c}B \end{bmatrix}}_{H_x} \underbrace{\begin{bmatrix} \vec{\Delta u}(k) \\ \vec{\Delta u}(k+1) \\ \vec{\Delta u}(k+2) \\ \vdots \\ \vec{\Delta u}(k+n_y-1) \end{bmatrix}}_{\underline{\vec{\Delta u}}}$$

and

$$\underbrace{\begin{bmatrix} \vec{y}(k+1|k) \\ \vec{y}(k+2|k) \\ \vec{y}(k+3|k) \\ \vdots \\ \vec{y}(k+n_y|k) \end{bmatrix}}_{\underline{\vec{y}}} = \underbrace{\begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{n_y} \end{bmatrix}}_{F} \vec{x}(k) + \underbrace{\begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ CA^2B & CAB & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ CA^{n_y-1}B & CA^{n_y-2}B & \dots & CA^{n_y-n_c}B \end{bmatrix}}_{\Phi} \underline{\vec{\Delta u}}$$

At this point we have completely defined the future prediction of our system starting from the state k. In particular we have managed to express the prediction using a combination of only the triplet (A, B, C) multiplied by the known initial state vector $\vec{x}(k)$ and by the unknown control trajectory $\underline{\vec{\Delta u}}$ that the optimizator must define.

1.2.3 Optimization

We have now arrived at the main component of our control: the optimizator. We will now need to write down in mathematical terms what we would like the system to do, and this results in writing down a cost function for our optimizator. However, before defining the cost function, we have to introduce a reference vector. The control must in fact obtain a control trajectory feasible with respect to the imposed constraints such that a certain reference path is followed. The reference vector is introduced as such:

$$\vec{R}_s^T = \left[\vec{y}_{ref}(k)^T, \quad \vec{y}_{ref}(k+1)^T, \quad \dots, \quad \vec{y}_{ref}(k+n_y)^T\right] \tag{1.5}$$

where $\vec{y}_{ref}(k_i)^T$ is a vector since we are dealing with a MIMO system. Each of these vectors contain the set points that all the outputs have to obtain at a certain time k_i.

We can now introduce the cost function:

$$J = ||\tilde{Q}(\vec{R}_s - \underline{\vec{y}})||_2^2 + ||\tilde{R}\ \underline{\vec{\Delta u}}||_2^2 \tag{1.6}$$

where $\tilde{Q}$ and $\tilde{R}$ are diagonal and positive definite matrices of weights. The first matrix is used to weight differently the outputs, while the second one is used to differently weight the control forces. As a common advice, the set up of the control should start with normalized weights. As we can see, this cost function grows when there is error between the outputs and their references, and when control forces are different from zero.

The minimization problem is written as follows:

$$\begin{aligned} \min_{\vec{\Delta u}} \qquad & J = ||\tilde{Q}(\vec{R}_s - \underline{\vec{y}})||_2^2 + ||\tilde{R}\ \underline{\vec{\Delta u}}||_2^2 \\ \text{subj. to} \qquad & A_{ineq}\ \underline{\vec{\Delta u}} \leq B_{ineq} \end{aligned} \tag{1.7}$$

A detailed discussion about the type of constraints, that explains the formulation of the A_{ineq} and B_{ineq} matrices, will be dealt with in the next paragraph.

By simply substituting the prediction as obtained in the previous paragraph into (1.6), knowing that $\underline{\vec{y}} = F\vec{x}(k) + \Phi\underline{\vec{\Delta u}}$, we obtain:

$$J = (\vec{R}_s - F\vec{x}(k))^T\tilde{Q}(\vec{R}_s - F\vec{x}(k)) - 2\underline{\vec{\Delta u}}^T\Phi\tilde{Q}(\vec{R}_s - F\vec{x}(k)) + \underline{\vec{\Delta u}}^T(\Phi^T\Phi + \tilde{R})\underline{\vec{\Delta u}} \tag{1.8}$$

where we neglect the term $(\vec{R}_s - F\vec{x}(k))^T\tilde{Q}(\vec{R}_s - F\vec{x}(k))$ since it is a constant term that is not depending on $\underline{\vec{\Delta u}}$ and therefore it cannot be minimized. If we divide by 2 we obtain:

$$J = \frac{1}{2}\underline{\vec{\Delta u}}^T H_{qp}\underline{\vec{\Delta u}} + F_{qp}^T\underline{\vec{\Delta u}}$$

where

$$\begin{aligned} H_{qp} &= (\Phi^T\Phi + \tilde{R}) \\ F_{qp} &= \Phi^T\tilde{Q}(-\vec{R}_s + F^T\vec{x}(k)) \end{aligned}$$

i.e we can use quadratic programming to obtain the optimized control trajectory $\underline{\vec{\Delta u}}$.

1.2.4 Linear constraints

The A_{ineq} and B_{ineq} matrices can be used to impose constraints to the optimizator, i.e. we can introduce upper and lower limits in the manipulated variables so that the control understands that it cannot use a certain value of control force:

$$\vec{u}^{min}(k_i) \leq \vec{u}(k_i) \leq \vec{u}^{max}(k_i)$$

We can also introduce limits in the variation of the manipulated variable. In this way the controller understands that it cannot move the manipulated variable too fast:

$$\vec{\Delta u}^{min}(k_i) \leq \vec{\Delta u}(k_i) \leq \vec{\Delta u}^{max}(k_i)$$

Finally we can introduce constraints in the outputs of the system:

$$\vec{y}^{min}(k_i) \leq \vec{y}(k_i) \leq \vec{y}^{max}(k_i)$$

Since the constraints in the quadratic program must be written in a specific manner, i.e. $A_{ineq}\ \underline{\vec{\Delta u}} \leq B_{ineq}$, we have to rewrite them.

For what regards the constraints on the variation of the control forces we can simply express them using two inequalities:

$$\begin{aligned} -\underline{\vec{\Delta u}} &\leq -\underline{\vec{\Delta u}}^{min} \\ \underline{\vec{\Delta u}} &\leq \underline{\vec{\Delta u}}^{max} \end{aligned}$$

The notion $\underline{\vec{\Delta u}}$ is used, since now we are talking about constraints for any sample times and not only for the instantk_i. This, in matrix form, becomes:

$$\begin{bmatrix} -I \\ I \end{bmatrix} \underline{\vec{\Delta u}} \leq \begin{bmatrix} -\underline{\vec{\Delta u}}^{min} \\ \underline{\vec{\Delta u}}^{max} \end{bmatrix} \tag{1.9}$$

where $\underline{\vec{\Delta u}}^{max}$ and $\underline{\vec{\Delta u}}^{min}$ are column vectors with $n_c \cdot N$ elements of $\vec{\Delta u}^{max}$ and $\vec{\Delta u}^{min}$ respectively, N being the number of inputs.

In the case of a manipulated variable constraint, we write:

$$\underbrace{\begin{bmatrix} \vec{u}(k) \\ \vec{u}(k+1) \\ \vdots \\ \vec{u}(k+n_c-1)) \end{bmatrix}}_{\underline{\vec{u}}} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \vec{u}(k-1) + \begin{bmatrix} I & 0 & \dots & 0 \\ I & I & \dots & 0 \\ \vdots & & & \\ I & I & \dots & I \end{bmatrix} \underbrace{\begin{bmatrix} \vec{\Delta u}(k) \\ \vec{\Delta u}(k+1) \\ \vdots \\ \vec{\Delta u}(k+n_c-1) \end{bmatrix}}_{\underline{\vec{\Delta u}}} \tag{1.10}$$

We can re-write (1.10) using C_1 and C_2.

$$\begin{aligned} -(C_1\vec{u}(k-1) + C_2\underline{\vec{\Delta u}}) &\leq -\underline{\vec{u}}^{min} \\ (C_1\vec{u}(k-1) + C_2\underline{\vec{\Delta u}}) &\leq \underline{\vec{u}}^{max} \end{aligned} \tag{1.11}$$

where $\underline{\vec{u}}^{min}$ and $\underline{\vec{u}}^{max}$ are column vectors with $n_c \cdot N$ elements of $\vec{u}^{min}$ and $\vec{u}^{max}$ respectively.

The output constraints are expressed in terms of $\underline{\vec{\Delta u}}$:

$$\underline{\vec{y}}^{min} \leq F\vec{x}(k) + \Phi\underline{\vec{\Delta u}} \leq \underline{\vec{y}}^{max} \tag{1.12}$$

We can finally say that the constraints matrices are:

$$A_{ineq} = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}; \qquad B_{ineq} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} \tag{1.13}$$

where

$$\begin{gathered} M_1 = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix}; \ N_1 = \begin{bmatrix} -\underline{\vec{u}}^{min} + C_1\vec{u}(k-1) \\ \underline{\vec{u}}^{max} - C_1\vec{u}(k-1) \end{bmatrix}; \\ M_2 = \begin{bmatrix} -I \\ I \end{bmatrix}; N_2 = \begin{bmatrix} -\underline{\vec{\Delta u}}^{min} \\ \underline{\vec{\Delta u}}^{max} \end{bmatrix}; \\ M_3 = \begin{bmatrix} -\Phi \\ \Phi \end{bmatrix}; N_3 = \begin{bmatrix} -\underline{\vec{y}}^{min} + F\vec{x}(k) \\ \underline{\vec{y}}^{max} - F\vec{x}(k) \end{bmatrix}; \end{gathered} \tag{1.14}$$

1.3 Non linear model predictive control theory

The NMPC has got, of course, the same characteristics that we have already discussed about in the linear control section. The control is still a feedback control based on an optimization method that requires a prediction of the state vector based on a detailed model of the system. As before the prediction is compared with the references and, through nonlinear optimizator, we aim at obtaining the optimal future trajectories of the manipulated variables that allow to follow in the best possible way these references while fulfilling the constraints. Also in this case the receding strategy is applied: only the first step of the manipulated variables is applied and then the whole optimization process is repeated. As for the linear control, the constraints can be introduced in the control variables, in their rate of change or in the outputs. However, for the case being, constraints are both linear and nonlinear, i.e. constraints functions of the manipulated variables. This is one of the biggest advantage of using a non linear model predictive control.

The other great advantage is the possibility of using a more realistic model of the system. Even hybrid models are allowed, i.e. models that exhibit both continuous and discrete dynamic behaviour. In wider terms we will have, for a multi input multi output system:

$$\dot{\vec{x}} = g(\vec{x}(t), \vec{u}(t)) \tag{1.15}$$

where g is a general non linear function. The model still needs to be discretized and therefore, knowing the discretization step ts, we will need to define a discrete function f such that:

$$\vec{x}(k+1) = f(\vec{x}(k), \vec{u}(k)) \tag{1.16}$$

All parameters of this type of control have absolutely the same meaning as explained in the linear section. Both n_y and n_c are present here and the choice of these parameters follows the same guidelines as the linear ones.

As we can see, now that we have introduced the non linear model, everything is pretty much identical to the linear model. Though we have to keep in mind that the way of actually implementing the control is completely different than for the linear case.

1.3.1 Non linear programming

As for the linear control, the optimization problem can be written as:

$$\begin{aligned} \min_{\underline{\vec{u}}} \qquad & J = ||\tilde{Q}(\vec{R}_s - \underline{\vec{y}})||_2^2 + ||\tilde{R}\ \underline{\vec{u}}||_2^2 \\ \text{subj. to} \qquad & C_{ineq} \ \leq 0 \end{aligned} \tag{1.17}$$

where $\vec{R}_s$ is defined in 1.5, $\underline{\vec{y}}$ are the future trajectories of the outputs of the system that can be a component of the state vector but also a general non linear function of the state vector, and the matrices $\tilde{R}$ and $\tilde{Q}$ have exactly the same meaning of the linear case. The difference between the linear and non linear problem is that now we cannot simplify the above equation in a way that can be used in quadratic programming.

Thus, we can decide to use full discretization or recursive discretization. The full discretization approach expects that the optimizator obtains, both the trajectory of the future manipulated variables, and the trajectory of every single component of the state vector. Thus the number of variables that need to be optimized is very big. We have also to introduce the system dynamics in the constraints matrix. Being the constraint matrix simply a column vector containing a list of generic non linear equations, we will need to add the following equation for each sample time k:

$$\vec{x}(k+1) - f(\vec{x}(k), \vec{u}(k)) = 0 \tag{1.18}$$

The advantage in using this approach is the ease of implementation, at the cost of a very high number of variables thus resulting in high computational time and difficulties in solution.

The other available method is the recursive discretization. In this approach the number of variables to be optimized is much smaller compared to the full discretization because only the control variables are considered. On the other hand there is the need to introduce the evaluation of the system dynamics, which is the trajectory of the system, function of the manipulated variables determined by the optimizator. In this sense the verification of the dynamics is carried out outside the optimizator and the fact of not considering it within the optimizator algorithm can cause numerical difficulties since we are neglecting some information of the system inside the optimization. Anyhow this leads to a remarkable reduction in computational time and, for this reason, is preferred and was used in our work.

1.3.2 Non linear prediction and constraints

For a non linear system it is not possible to analytically write the expression of the prediction as we did for the linear case. The prediction needs to be simulated for each guess. The optimizator, starting from the initial guess, will make many attempts to combine various trajectories of the manipulated variables, and every single attempts requires to use a non linear solver. The solver provides the outputs that are used at the end of the process to calculate the cost function. Then, depending on the optimizator algorithm, the variables under optimization will be changed, a new attempt will be made, until an optimal solution is found.

Thus, for the programmer point of view, the most complicated control to implement is the linear one, since it requires the writing of many matrices of generous size. In the non linear control instead the complication is all left to the optimizator, in our case the *fmincon* function of Matlab. This brings many problems for what regards the control setup. Weights and parameters became fundamental both for the control stability and the for good results, particularly if the constraints are pressing.

Also the constraints, compared to the linear control, are much more easy to write. We can include non linear constraints and also constraints dependant on boolean functions. Of course we can include also constraints on the control variables, but this time these constraints can be more sophisticated. For example we might want the throttle pedal to be used at most at 80% in first gear, while leaving the freedom of using full throttle in any other gear.

Note that it cannot be taken for granted that the optimizator is capable of dealing with all these constraints. In fact it might happen that the optimizator gives back a solution that fullfills only few constraints.

Chapter 2

LTV MPC Active front steering

This chapter will focus on the steering angle control done using a LTV-MPC controller. We started with a simple controller on purpose in order to examine thoroughly the limits of this type of control. At a later stage the control will be refined adding the management of the throttle and brake pedal in order to get closer to a realistic Motion Planner inside a fully autonomous vehicle. Note that until now nobody has ever taken into consideration this problem.

We must say that we have not considered the requirements necessary to achieve a real-time control system. In fact, sample time by sample time, the simulation time is stopped, the calculation of the future control variables is made, and only then the simulation is started back again. This is due to the fact that Matlab, and not a compiled code, is used. In any case considerations about the time required by the control will be made: there will be a dedicated section to show the control calculations speed and few simplifications on the control model will be made to accelerate the optimization process.

2.1 Vehicle models and linearization

Since we are programming a control that acts only on the steering angle, we must necessarily neglect the longitudinal dynamic of the vehicle. Thus from the vehicle model explained in the appendix we simplify the longitudinal forces acting on the tyres, as well as the aerodynamic forces. The vehicle needs an initial speed to be set. We are well aware of the important assumptions that we are doing but, as we have already said, the importance of the AFS is dedicated to studying and understanding the linear MPC.

Furthermore, in the vehicle model that the controller is using, we neglect both the load transfer and the relaxation lengths, while the simulated model, albeit with no longitudinal forces, remains as sophisticated as possible. It is obvious but necessary to remember that a vehicle model, as simplified, is extremely nonlinear both for the presence of strong non linearity due to the tyres behaviour and for the geometric non linearities that we find already in the balance of forces. Its implementation in a linear controller may be felt as inconsistent and needs further explanations. What happens is that we are using a standard linear controller, following slavishly the theory which concerns it, but instead of using a normal linear model, we are using a local linearization of a nonlinear model.

The nonlinear 2 contact single track vehicle model that the controller uses:

$$\begin{cases} \dot{V_x} = (m\dot{\sigma}V_y - F_{tf}\sin(\delta))/m \\ \dot{V_y} = (-m\dot{\sigma}V_x + F_{tf}\cos(\delta) + F_{tr})/m \\ \ddot{\sigma} = (-F_{tr}b + F_{tf}a\cos(\delta))/J_z \\ \dot{\sigma} = \dot{\sigma} \\ \dot{x_g} = V_x\cos(\sigma) - V_y\sin(\sigma) \\ \dot{y_g} = V_x\sin(\sigma) + V_y\cos(\sigma) \end{cases} \tag{2.1}$$

where the expressions of the two tyre transversal forces, functions of the slip angles, have been omitted for clarity, and where a slightly different expression for $\dot{x_g}$ and $\dot{y_g}$, that we remember being the speed with respect to the absolute reference system, has been used. At each k we will have a defined condition and therefore a defined state vector, around which we are going to linearize the system calculating:

$$A_m = \begin{bmatrix} \frac{\vartheta\dot{V_x}}{\vartheta V_x} & \frac{\vartheta\dot{V_x}}{\vartheta V_y} & \frac{\vartheta\dot{V_x}}{\vartheta\dot{\sigma}} & 0 & 0 & 0 \\ \frac{\vartheta\dot{V_y}}{\vartheta V_x} & \frac{\vartheta\dot{V_y}}{\vartheta V_y} & \frac{\vartheta\dot{V_y}}{\vartheta\dot{\sigma}} & 0 & 0 & 0 \\ \frac{\vartheta\ddot{\sigma}}{\vartheta V_x} & \frac{\vartheta\ddot{\sigma}}{\vartheta V_y} & \frac{\vartheta\ddot{\sigma}}{\vartheta\dot{\sigma}} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{\vartheta\dot{x_g}}{\vartheta V_x} & \frac{\vartheta\dot{x_g}}{\vartheta V_y} & 0 & \frac{\vartheta\dot{x_g}}{\vartheta\sigma} & 0 & 0 \\ \frac{\vartheta\dot{y_g}}{\vartheta V_x} & \frac{\vartheta\dot{y_g}}{\vartheta V_y} & 0 & \frac{\vartheta\dot{y_g}}{\vartheta\sigma} & 0 & 0 \end{bmatrix}; \qquad B_m = \begin{bmatrix} \frac{\vartheta\dot{V_x}}{\vartheta\delta} \\ \frac{\vartheta\dot{V_y}}{\vartheta\delta} \\ \frac{\vartheta\ddot{\sigma}}{\vartheta\delta} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.2}$$

and those matrices A_m and B_m need to be, as explained, augmented to obtain the triplet (A, B, C) that is then used inside the controller to make the linear prediction and therefore to obtain the future trajectories of the manipulated variables. It is fundamental to understand how both the prediction and thus the whole optimization is made with the strong hypothesis of keeping constant the matrices of the system. In this way we are introducing important prediction errors due to the strong non linearities of the system, but those will become acceptable within certain limits thanks to the receding strategy.

At the next step the linearization will be redone, new (A, B, C) matrices will be obtained and the whole optimization process will be made again.

As we know it is possible to introduce constraints, provided that they are linear or rather not function of the inputs $\vec{\Delta u}$. They can also vary with k, that is what we will do to introduce the obstacles in our work. For now we limit ourselves to introducing constraints on the manipulated variable and his rate of change. We can think about it directly as the steering angle imposed to the wheel, and this hypothesis is not strong because the angle at the wheel is the angle at the steering wheel multiplied by a constant. With this consideration and the considerations made by [2], a maximum and minimum value of $\pm\frac{\pi}{8}rad$ is set for the steering angle, and of $\pm 0.2rad/s$ for what regards its rate of change. These were introduced because a normal vehicle has a maximum physical value of steering that we can impose that was not introduced in the model itself. For what regards the rate of change we have set it like so because the control, not having the relaxation lengths in its model, is not capable of understanding that the transversal forces are not algebraic. Of course this could be limiting in some occasions. The constraints on the rate of change could be used also to consider the limits in variations that an actuator, that we have considered ideal, could possibly have.

2.2 Path follower

In this section we want to show the control for what it was developed in its early stage. An MPC controller was in fact born with the purpose to follow a reference in the best way possible, respecting some constraints that are typically imposed on the manipulated variables. We will impose a reference on the lateral position of the vehicle, therefore an output of the system, and we will ask for the trajectory of the steering that allows to drive the vehicle as close as possible to the reference. Nevertheless the controller has to consider the constraints on the maximum steering angle and on its maximum rate of change. It is important to notice that, at least for the linear controller, the constraints are fulfilled only during the prediction, and so the controller might not follow the limits in the rate of change for what regards the first value he needs to choose for each prediction. Anyhow he will never void the constraints during a prediction, but only when considering the first value. This is a clarification that we needed to say, but it has never been a problem for us. In fact, even though the linearization is a strong hypothesis, the differences between two optimizations are very little.

The reference must be expressed as a function of time and is developed, in this example, as a vector in which each value corresponds to the absolute

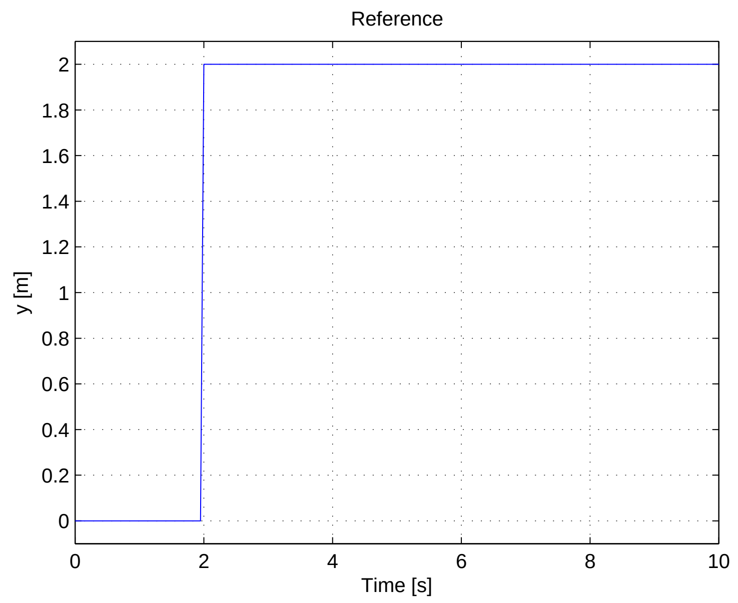

Figure 2.1: Reference signal

lateral position in meters that the vehicle has to obtain at each sample time k, as we can see in figure 2.1. Actually we can impose a reference in whatever output of the system, as long as it is a linear combination of the state vector since we are using a linear controller. Also we still need to consider many references as control variables to maintain a squared control.

As we can see from figure 2.2, the controller is capable of following the reference without any problems and is, unlike the classic PID controller, very good at anticipating the step so that the error is minimized. In figure 2.3 we can see the behaviour of the control variable, that is the steering. We can see that it is shown with stems to underline the fact that the control is discrete. The vehicle is in fact driven every ts seconds, in this example every $50ms$, and for each of this steps we can see the corresponding value of the steering. From now on, to clarify, the steering signal will be shown as a continuous line, as in figure 2.4, but we must never forget that a discrete controller is always used. The trial is done at an initial speed of $50km/h$ with parameters and weights that we found to be optimal. We want to remember that $\tilde{Q}$ is the matrix of outputs weights and $\tilde{R}$ is the matrix of control forces weights. Both are a single value in this case. In table 2.1 we find the list of all the values used.

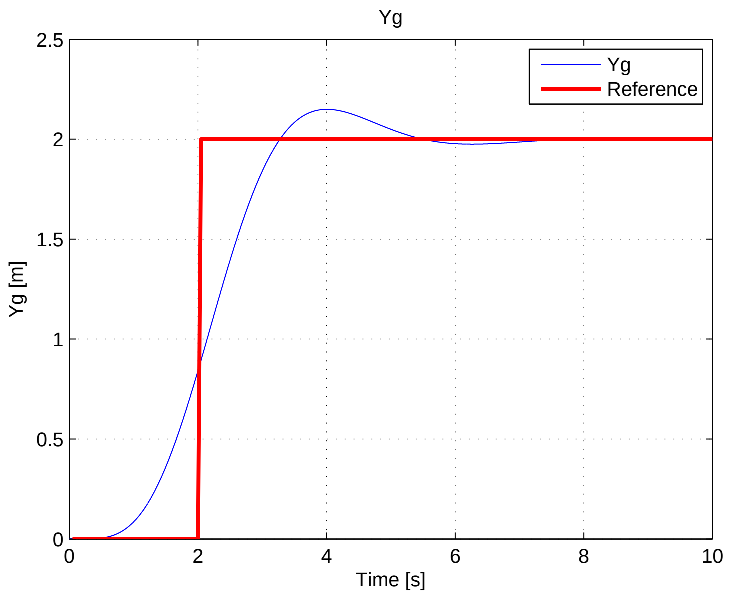

Figure 2.2: Path follower - Lateral displacement

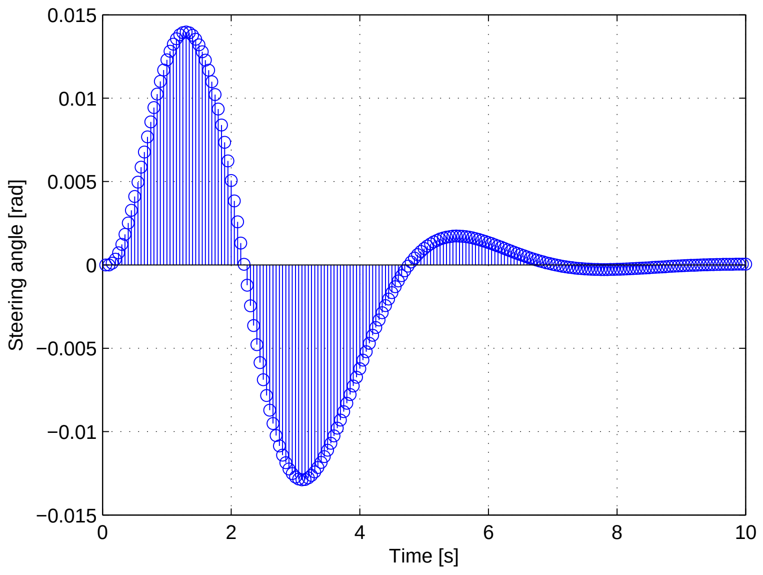

Figure 2.3: Path follower - Steering stems

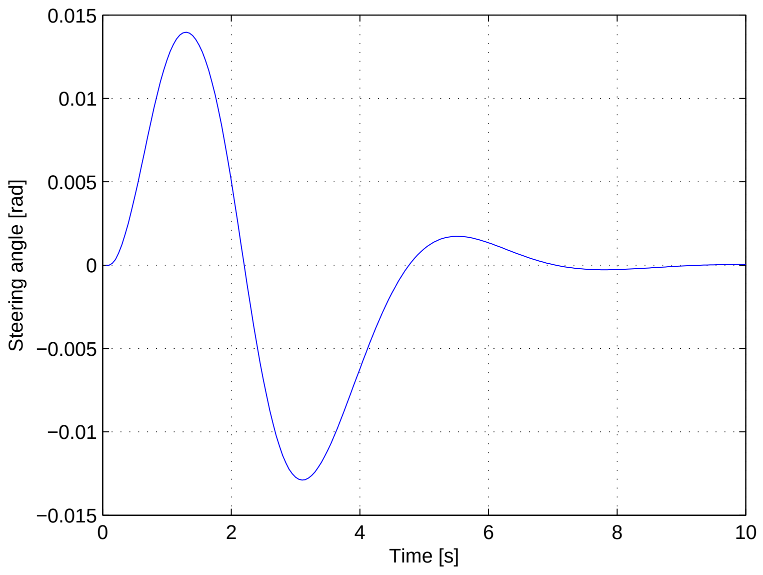

Figure 2.4: Path follower - Steering

n_y	n_c	$ts\ [s]$	$\tilde{Q}$	$\tilde{R}$
50	10	0.05	1	50000

Table 2.1: Standard parameters

2.3 Obstacle

Finally we have arrived at the point where we can challenge our controller with an obstacle. The proposed obstacle is a lorry that is represented by a constraint $15m$ long and $2m$ wide. The width, that might look small, is half the width of the lorry plus half the width of the car. Even though this could look like a strong hypothesis, in any case we are just interested to show how the control is capable of considering encumbrances and to act in order to change his path. Also the fact of knowing the length of the obstacle could be felt as not credible, but in reality from previous experience we know the length of obstacles that are commonly found in roads. Everybody expects a normal vehicle to be of the length that he typically is and this statement is true also for lorries, people or animals. It is feasible to give to the controller a length of the obstacle and nevertheless the information can be corrected as soon as it is updated from the measurement system. We have made the hypothesis that a measurement system is capable of detecting obstacles well before our controller, while our MPC will be able to understand the presence of an obstacle only if it appears in his prediction window. The n_y parameter becomes fundamental.

Even though we talked about meters, the obstacle must be defined in time. For now, for the AFS systems that cannot cause big modifications in the longitudinal velocity of the vehicle, the obstacle is imposed considering constant speed. So, thanks to the measurement system we will know that we will get to the obstacle in, for example, $25m$, and those became, if travelling at $50km/h$, $1.8s$. In this way we will say that until the next $1.8s$ the road will be clear, while from $1.8s$ to $2.9s$, if the obstacle is the lorry, the road will not be practicable below $2m$. Obviously this brings errors because the speed is not constant over the manoeuvre, but these errors are small enough not to cause any problems. Apart from being written in the time domain, the obstacle, as well as the reference and the whole control, is discretized every ts. It might happen that, if the speed is too high, a very short obstacle could not be seen by the control. In fact if the obstacle is shorter than ts, not even a single point will exist to define it. An obstacle that is $1m$ long will be, at $150km/h$, as long as $0.024s$ and this is shorter than our discretization step. Of course this is not a big issue, since to solve it we can simply impose obstacles at least $50ms$ long, but it is important to specify.

As we can see from figure 2.5 the discretization brought us to find another problem. The blue lines are predictions at various sample time k. We have highlighted a prediction that, even though it crosses over the obstacle, is considered valid. In fact the optimizator checks the error with respect to the reference and the validation of the constraints every ts, as we can see thanks

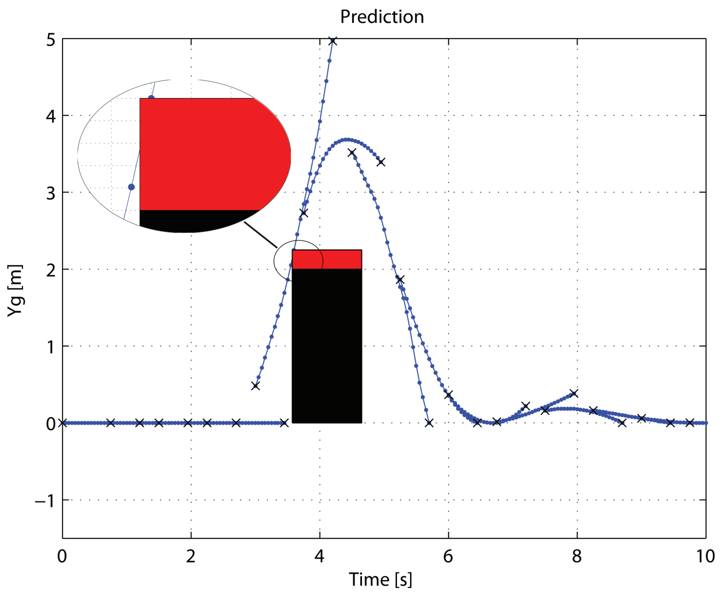

Figure 2.5: Obstacle - Discretization problem

to the small blue dots. In that trial randomly it happened that the step before the obstacle and the one right after are both out of the constrained zone. Unfortunately the trajectory will have to go across the obstacle, but the controller cannot understand that. A way for solving this is to reduce the discretization step, but this will cause huge slowdowns in the optimization process. What we decided to do was to slightly increase the obstacle width such that, though crossing the constrained zone, it will avoid the real obstacle. From now on we will see the real obstacle represented in black, while the obstacle that is given to the controller includes also the red zone.

Thanks to figure 2.6 we can understand that a reference must still be present. The controller must know that he has to follow the road, that in this case means staying around the value zero because we are simulating a straight line. It is important to notice that the controller is forced to increase the value of the cost function in order to avoid the constraint. The priorities of the controller are obvious. Anyhow the controller is very good at avoiding the obstacle, and in this case is also capable of avoiding completely also the red zone, as we can see from figure 2.7 and 2.8. The control parameters are the one of the standard trial shown in table 2.1. Those figures show the advantage of the MPC with respect to the classic PID controller. A PID controller would have required the definition of a trajectory in order to complete this manoeuvre, and that trajectory would have been defined by a programmer

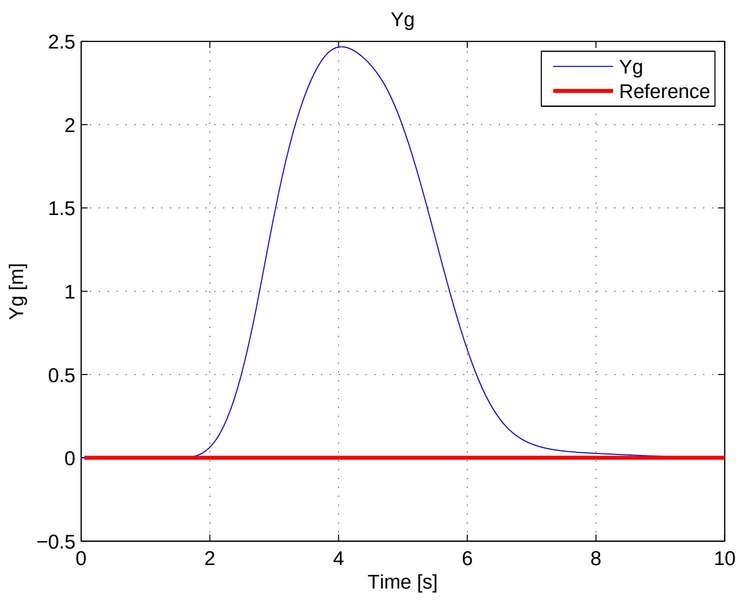

Figure 2.6: Obstacle - Reference

based on criteria that typically are not good enough to consider the various problems that can be found. The MPC on the other hand is capable of doing this all on his own. Every manoeuvre is engaged knowing the vehicle behaviour and every event is dealt with so that an optimal result is obtained.

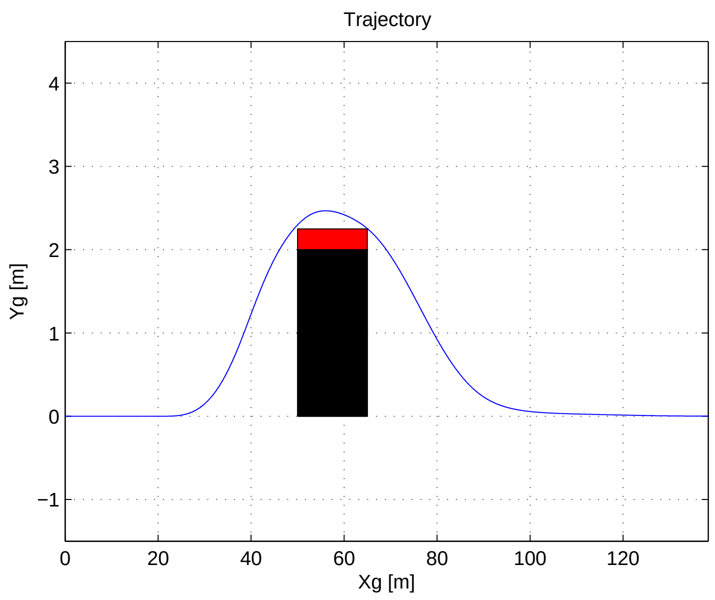

Figure 2.7: Obstacle - Trajectory

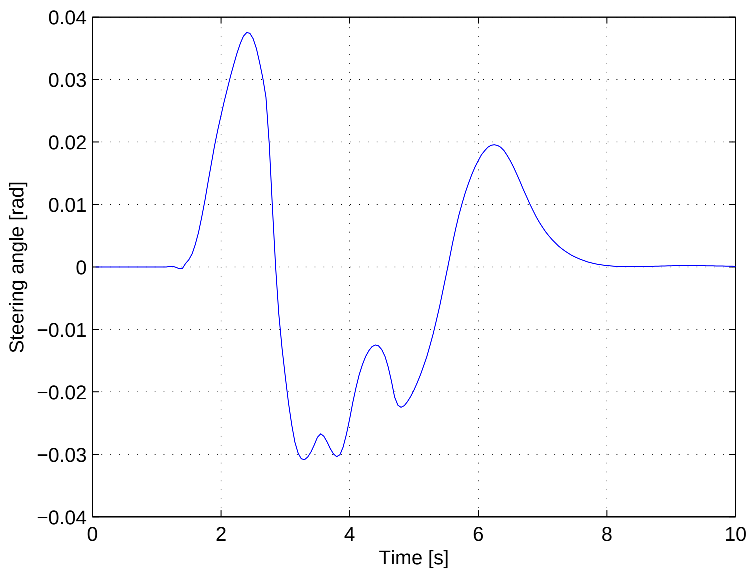

Figure 2.8: Obstacle - Steering

n_y	n_c	ts $[s]$	$\tilde{Q}$	$\tilde{R}$
25	10	0.05	1	50000

Table 2.2: $n_y = 25$ - Parameters

2.4 Parameters' choice

In this section we would like to show the reasons that guided us to choose the parameters that then we have used to produce this paper. As we have anticipated in the state of the art the choice of fundamental parameters, that are n_y, n_c and ts, is guided by the system characteristics. Of course with our nonlinear system used with a linear control is less easy to foresee the optimal parameters, and therefore many trials in different conditions were required to understand what was good and what was not. Here we just want to explain what a certain parameter is important for. n_y is the length of the time window in steps, and generally it must be kept as big as feasible with the time available for the optimization process and at least bigger than the slowest dynamics of the system. In this way the control is capable of understanding the future behaviour of the system until it gets to his steady state conditions. Unfortunately, in our situation, two facts became fundamental. The first one is the fact that the bigger n_y is, the sooner the control gets aware that a certain obstacle is present. So, even though smaller value of n_y could have been enough to follow a certain reference, it could not be enough to cope with a very high obstacle. As we can see from figure 2.9, the control with $n_y = 25$ is stable and that means that in that condition 25 steps are enough to drive a vehicle, but are too few to manage correctly the obstacle. In fact compared with the trials made with n_y equal to 50, there is a much bigger over elongation. This trial is made with parameters as shown in table 2.2.

On the other hand also the situation where we have used an higher n_y turned out to be less performing than usual. We can see that we have changed only that particular parameter in table 2.3. The trajectory in figure 2.9 has an odd shape but, more importantly, we find that the steering trajectory, again in figure 2.10, is a lot less accurate than before. This happens because since we are asking for a longer prediction, we are introducing very important errors due to the fact that the control forces a linearization of the system. Those error can be easily seen comparing figures 2.11 and 2.12. In the first one, even though prevision errors are present also due to the fact that we are using a value of n_c smaller than n_y, we can see that the previsions are coherent with the actual dynamic of the system, while in the last one much bigger variance is present.

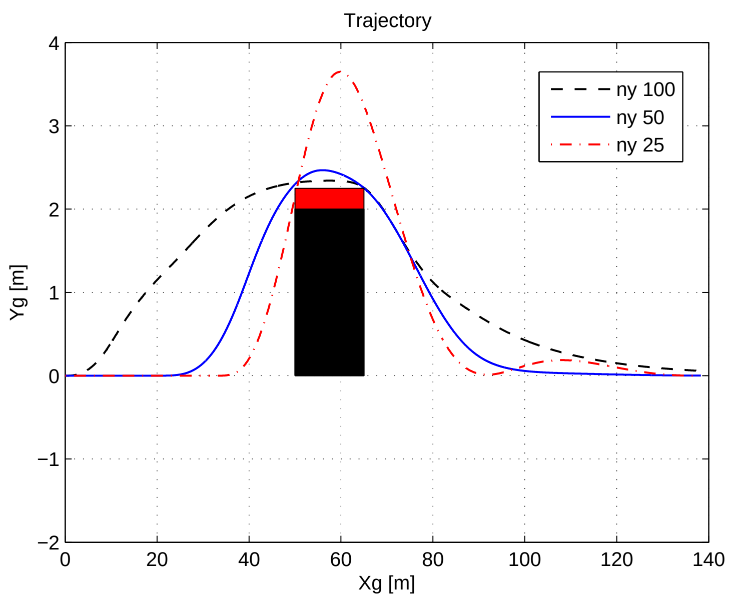

Figure 2.9: n_y comparison - Trajectory

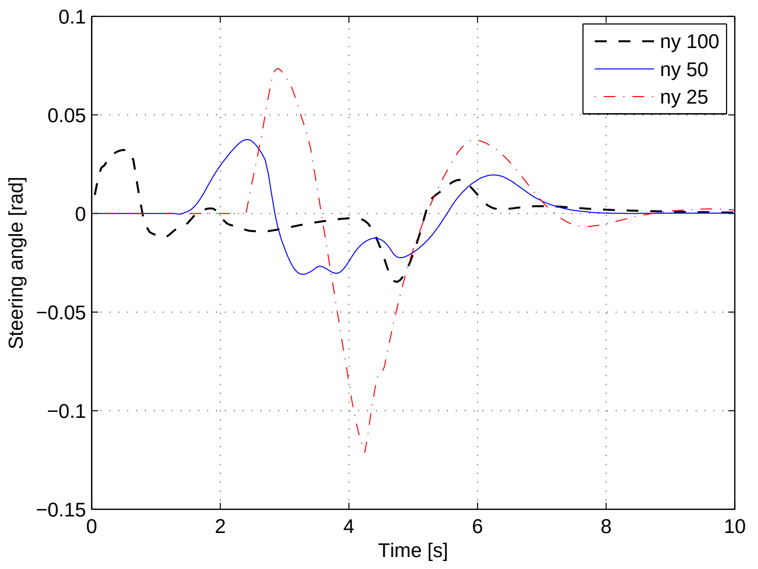

Figure 2.10: n_y comparison - Steering

n_y	n_c	$ts\ [s]$	$\tilde{Q}$	$\tilde{R}$
100	10	0.05	1	50000

Table 2.3: $n_y = 100$ - Parameters

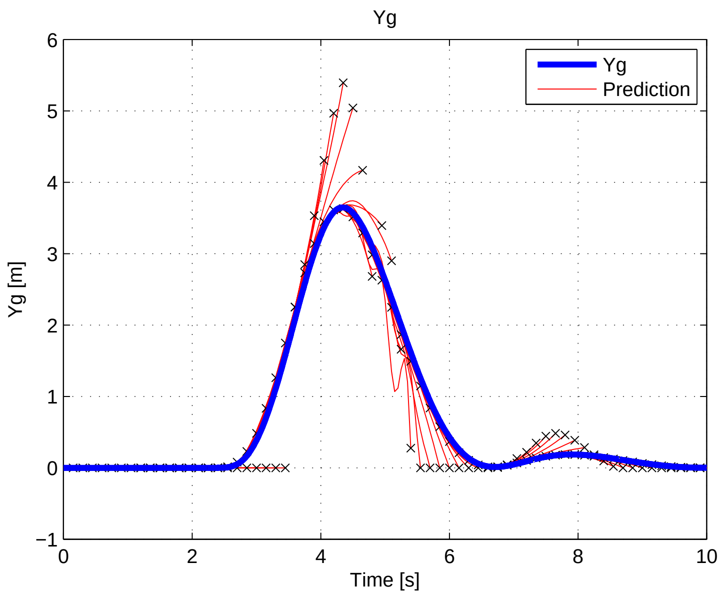

Figure 2.11: $n_y = 25$ - Predictions

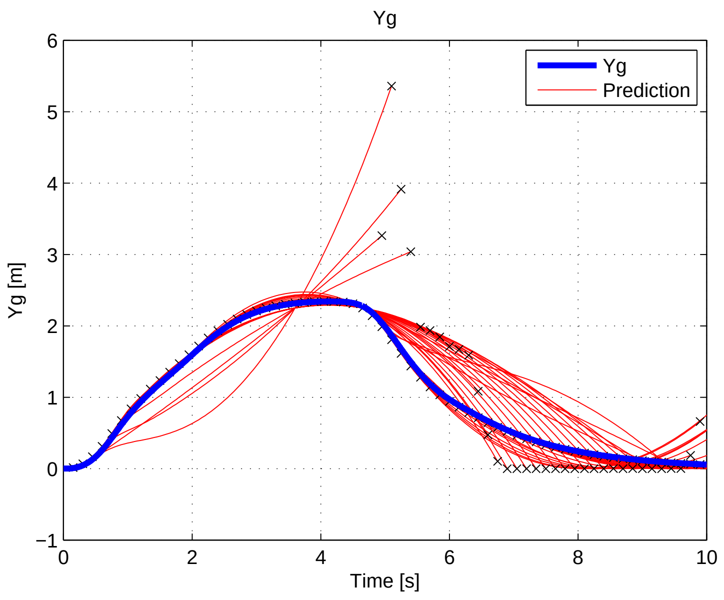

Figure 2.12: $n_y = 100$ - Predictions

n_y	n_c	$ts\ [s]$	$\tilde{Q}$	$\tilde{R}$
100	20	0.025	1	50000

Table 2.4: $ts = 0.025$ - Parameters

n_y	n_c	$ts\ [s]$	$\tilde{Q}$	$\tilde{R}$
5	1	0.5	1	50000

Table 2.5: $ts = 0.5$ - Parameters

We are for these reasons obliged to keep a value of the window length long enough to keep into consideration the presence of obstacles, but also short enough to avoid the prediction errors to became too important. All of these trials were made at the same $50km/h$, but it is important to notice that, being a very nonlinear system, results vary also considering various initials speed and obstacle shapes. Considering this we have found $n_y = 50$ to be the ideal value for us. Another important notice is that with a fixed value of n_y we have a different visual lengths available. For example 50 steps at $50km/h$ are almost $35m$, while at $150km/h$ they are almost $105m$. This effects, that is simply caused by the fact that we have to impose the obstacle in the time domain, even if it has a fixed length in meters, helps us dealing with obstacles at various speeds.

For what regards ts we have made few trials to show that, coherently with the values we have found in [2], [3] and [4], our discretization step is good enough. This value is to be kept as low as possible. Lower values will cause the optimization to be too long, while bigger value cannot guarantee a stable control. In the first trial we used values from table 2.4. You can see that we needed to adjust also the prediction and control horizons to allow the control to have the same window lenght to the previous trials. As we can see from figure 2.13, the trajectory is almost identical to the standard trial, but it has required a lot more to calculate since we have required a much thinner discretization as we can see from the number of stems in figure 2.14.

The second trial, as shown in table 2.5, was made with a much coarser discretization step. This brought us a much faster control, as we can see from the number of stems in figure 2.15, but with them is not possible to deal with the obstacle.

We have decided to avoid showing trials also for what regards n_c, $\tilde{Q}$ or $\tilde{R}$, but we have done many trials in different conditions (different obstacles and different speeds) to define the most stable and performing control we could.

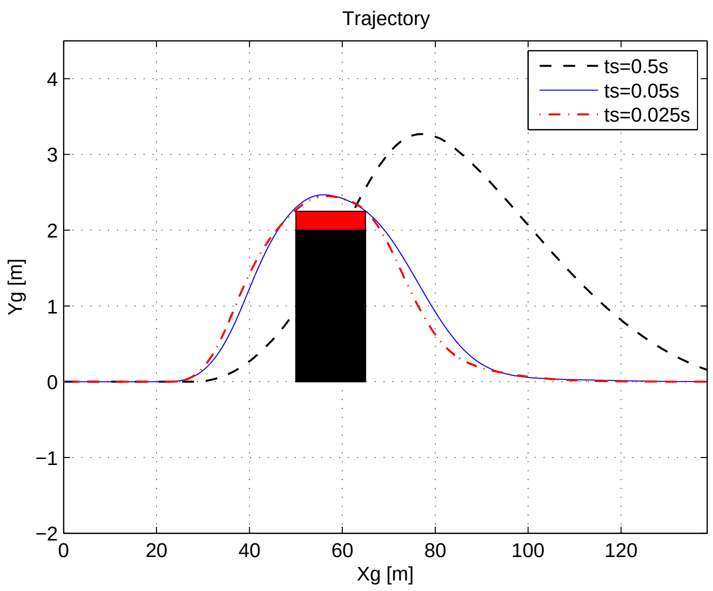

Figure 2.13: *ts* comparison - Trajectory

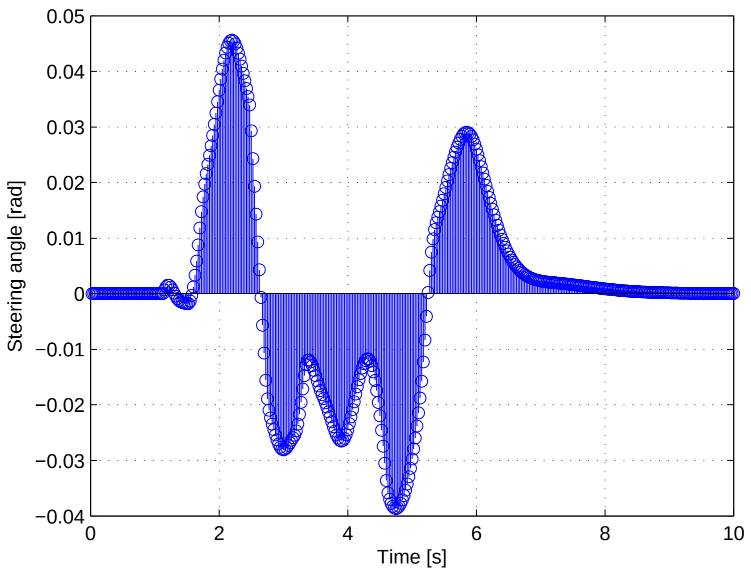

Figure 2.14: $ts = 0.025$ - Steering stems

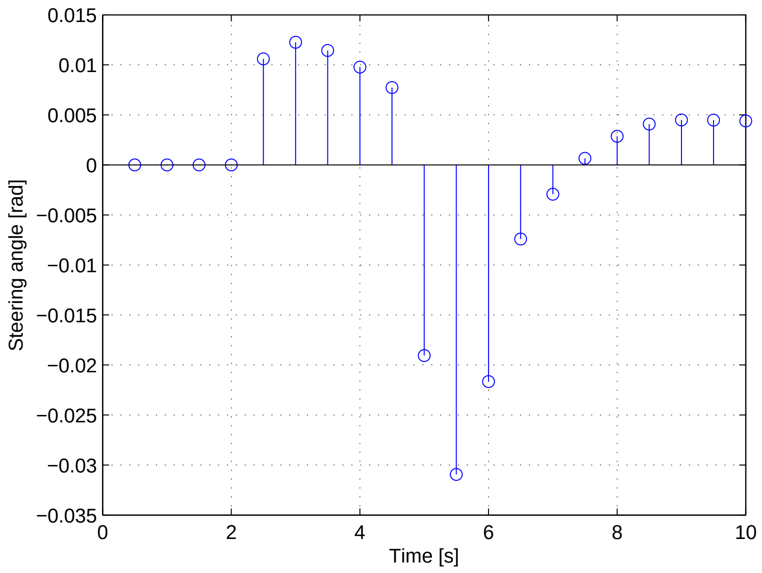

Figure 2.15: $ts = 0.5$ - Steering stems

2.5 Various obstacles

Having now defined the optimal parameters, that we can find in table 2.1, we would like to show that our control is capable of dealing also with different types of obstacles. As an example, in figures 2.16 and 2.17, we can see that our controller is capable of dealing with many obstacles, even if they are placed close one to each other. The obstacles are placed such that they impose lower or upper bound, or even narrow corridors.

In some cases less stable optimizator can cause problems with longer obstacles. As we can see from figures 2.18 and 2.19, there is a period of time in which the output corresponds to the constraints. In many cases it could happen that the optimizator, for numerical reasons, finds the initial position of the prediction to be inside the constrained area, and therefore it stops because no solutions are possible to avoid the obstacle. Thankfully the *quadprog* function in Matlab is able to cope with this problem and therefore no further modifications of the control were required.

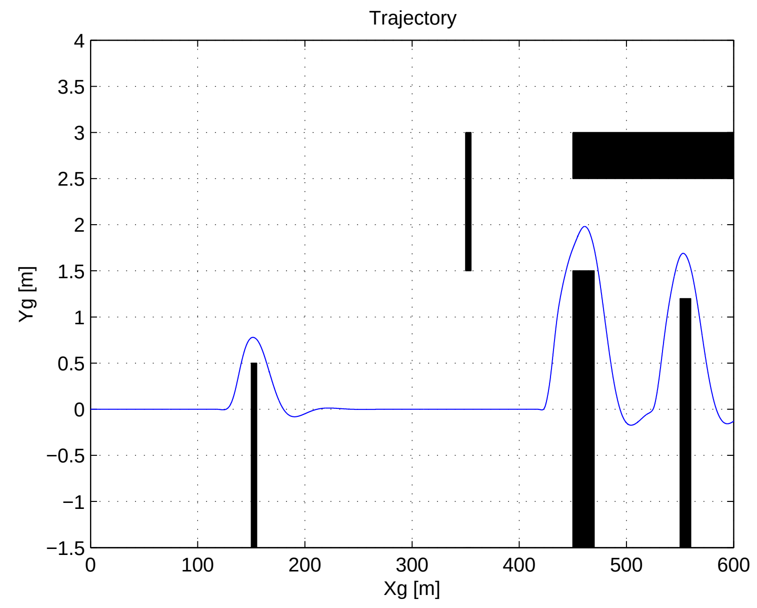

Figure 2.16: Gymkhana - Trajectory

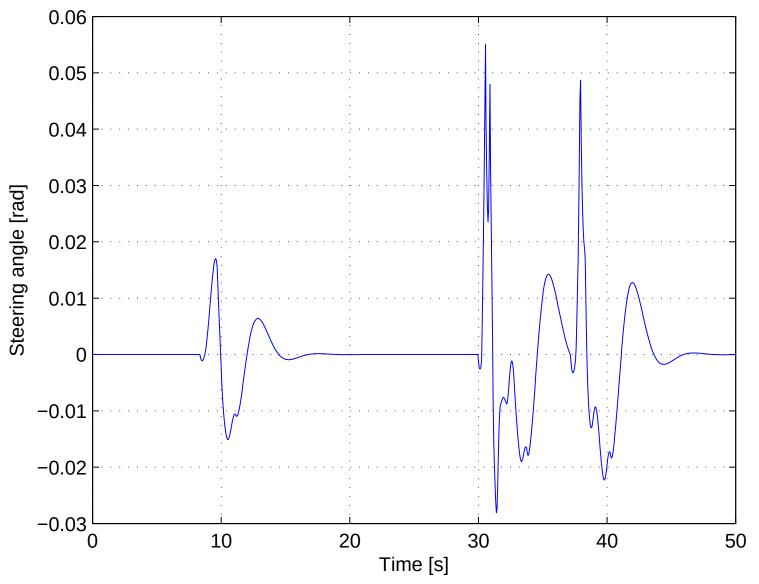

Figure 2.17: Gymkhana - Steering

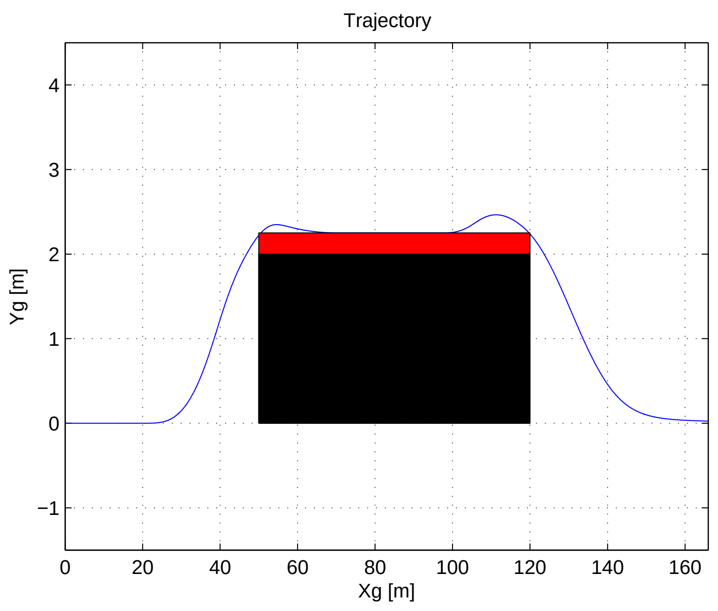

Figure 2.18: Long obstacle - Trajectory

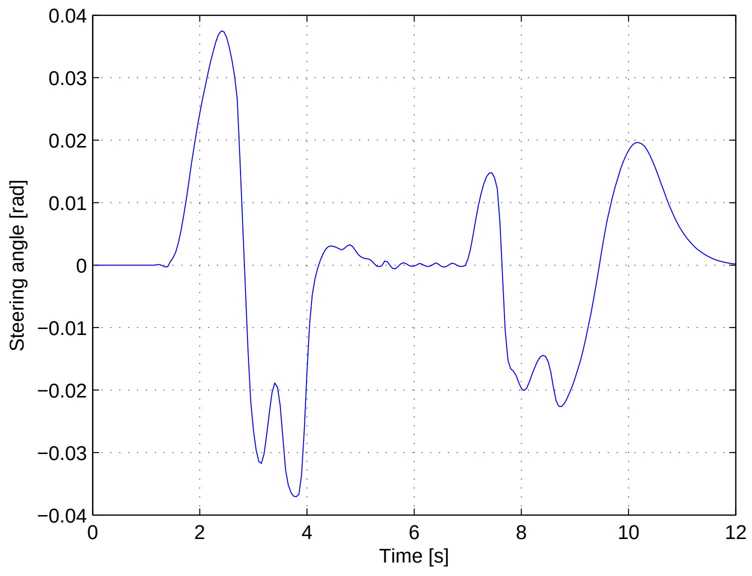

Figure 2.19: Long obstacle - Steering

V_x $[km/h]$	$Dist$ $[m]$ $\dot{\delta} = 0.02$	$Dist$ $[m]$ $\dot{\delta} = 0.04$
30	12	10
45	15	13
50	17	15
60	20	17
70	23	19
90	25	24
100	29	26
110	32	44
130	36	53
150	42	76
170	69	80
210	145	300

Table 2.6: Minimum distance analysis

2.6 Minimum distance analysis

It is now interesting to study how much space is required for the control to avoid the standard obstacle of figure 2.7 at a given initial speed. Even though we know that a good controller must be capable of using also the accelerator and brake pedal to cope with a manoeuvre at the limit, we thought that this study could be interesting. The various trials were made all with the same parameters, only changing the initial speed. Results that showed collision with the obstacle, instability of the controlled system or a maximum lateral position higher than the width of the road of $4m$ were considered failed. Results of these tests are resumed in table 2.6 and figure 2.20.

As we can see two different constraint on the steering rate of change were used and important differences were noticed. In fact for medium low speed the higher rate of change gives back better results, requiring around 15% less space compared to the standard value of $0.02 rad/s$. At higher speed though, the controller showed signs of instability that required the obstacle to be placed much further with respect to the standard case, and it become extremely unreliable. Since the instability starts to appear at speed higher than $150km/h$, we can decide that, not being a legal speed pretty much anywhere in the world, the higher value could be a valuable solutions, but we like the fact that the lower value is capable of driving at any speed that the vehicle is capable of obtain, and for this reason it was used in our work.

Typically the problem is that the control is not able to stay within the required $4m$ in lateral position as we can see from figure 2.21, where an initial

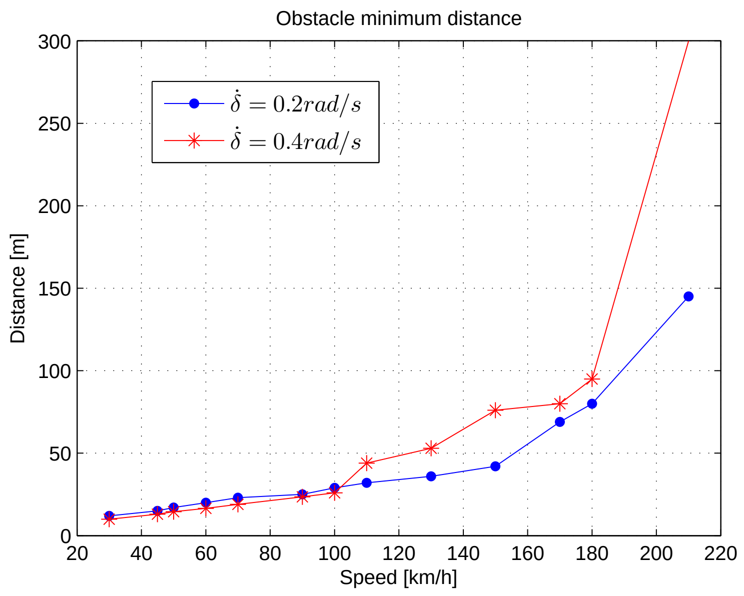

Figure 2.20: Minimum distance analysis

speed of $110km/h$ and a distance of 29.5 meters were used. The car is fast enough to react and the obstacle is lower enough to be avoided pretty much on every trial, but if we push too hard, the control will drive into the ditch. This appears to be the optimal situation since the obstacle could be a person that must be avoided at all costs.

What we have explained is what typically happens below $150km/h$. At higher speeds the problem starts to be instability, as we can notice from figure 2.23 where an initial speed of $220km/h$ and a distance of 295 meters were used. The linearization that we have forced the control to do starts to introduce much bigger errors because the vehicle now is changing behaviour, and therefore matrices, much faster than before. As we can see comparing figures 2.25 and 2.26 the problem is the fact that at higher speed the prediction is absolutely untrustworthy. This is a pure limit of the linear control.

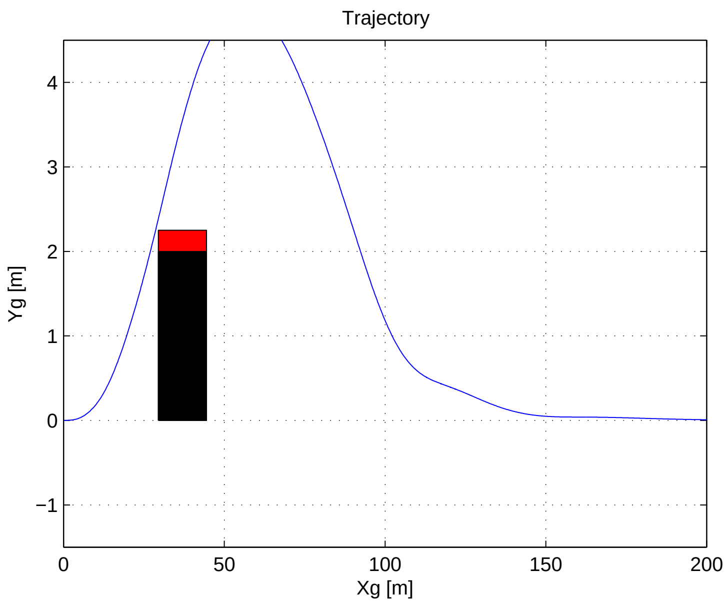

Figure 2.21: 110km/h 29.5m - Trajectory

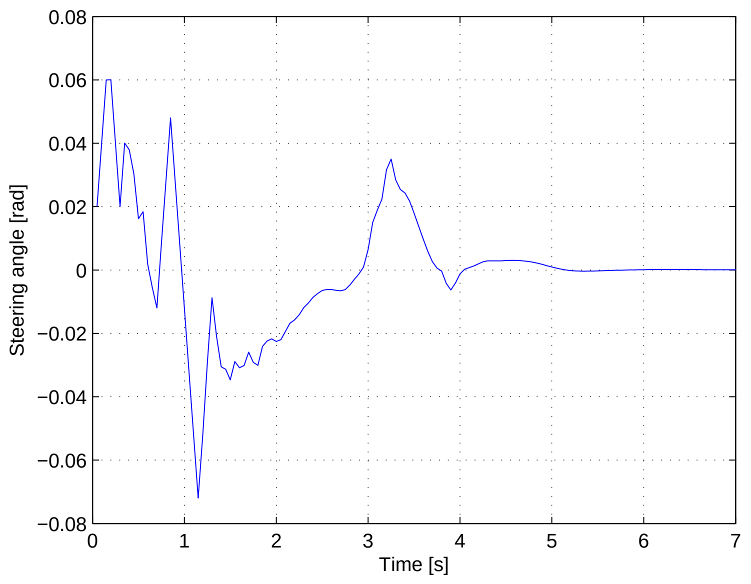

Figure 2.22: 110km/h 29.5m - Steering

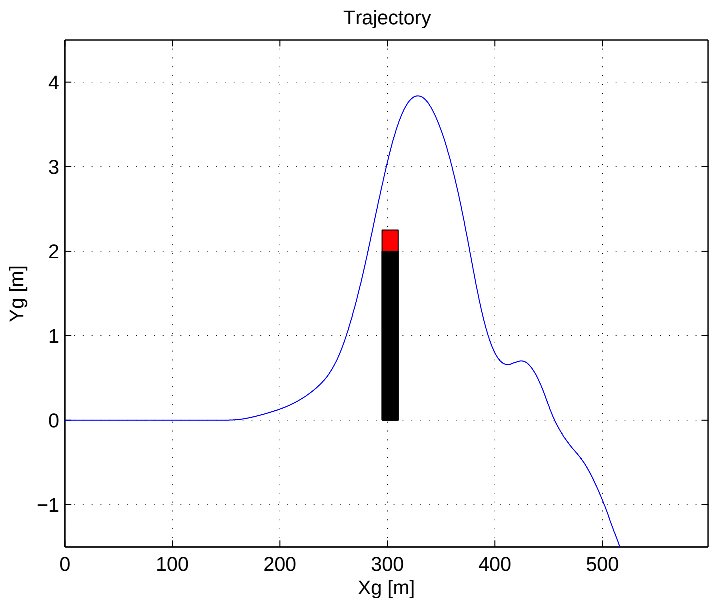

Figure 2.23: $220km/h$ $295m$ - Trajectory

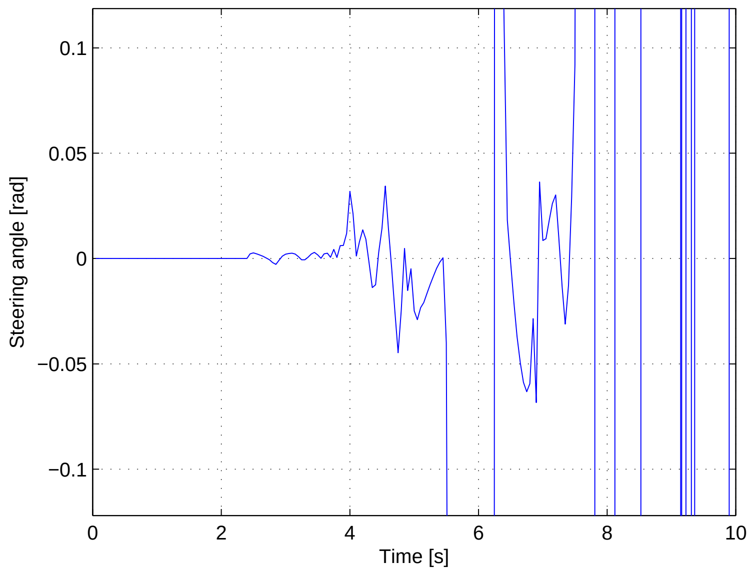

Figure 2.24: $220km/h$ $295m$ - Steering

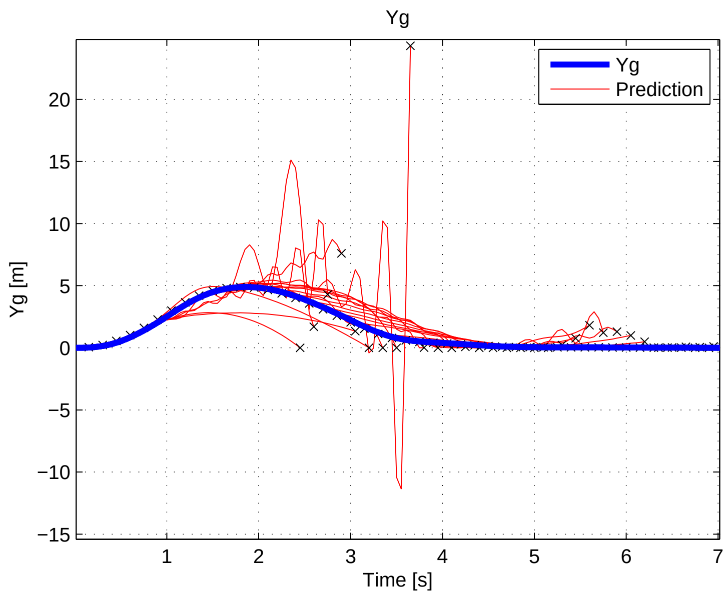

Figure 2.25: $110km/h$ $29.5m$ - Prediction

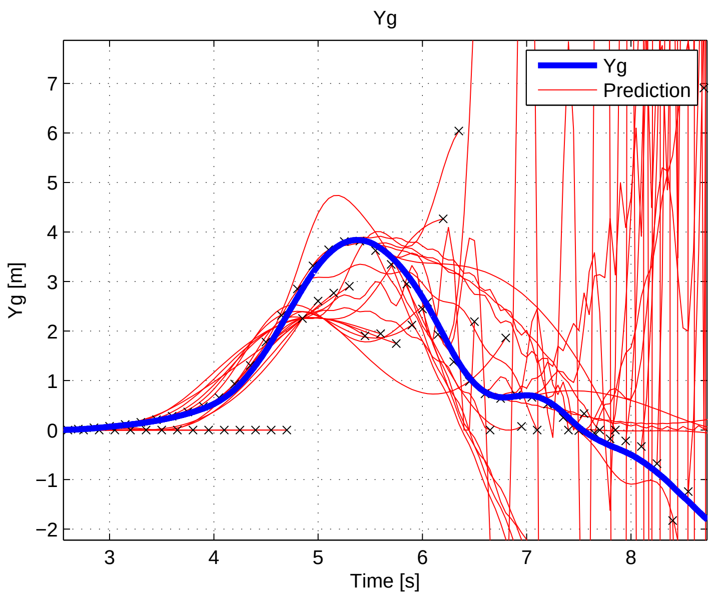

Figure 2.26: $220km/h$ $295m$ - Prediction

2.7 Real time implementation feasibility

Even if the purpose of this work is not to design a real time control, at the end of this chapter few considerations around the control speed can be made. A control can be implemented as a real time control if the time required for calculations does not cause any losses of performances and stability. To have an idea of how long it takes to do every optimization process we have measured many trials. In figure 2.27 we can see what happens during the trial with the standard obstacle. The first operations takes always very long, regardless of the conditions, so we have decided to neglect its importance. This is probably caused by the Matlab logic of allocating memory for calculations. Then we can see that in the straight line the optimization takes always little time, around half the value of ts. We then notice a peak that happens before the obstacle, a little bit after $1s$ and, looking at figure 2.8, we can see that is the moment when the control starts to use the steering. In all the trials only one optimization is not made fast enough to calculate the next steering input before it is too late. To understand if that is a dangerous situation we tried to apply, instead of the first value of steering that the optimization process gives back to us, the second one. This could also be intended as a safety feature for a possible real time implementation of this control. If the calculation requires too much time, the control stops it and set as input the second value of the steering trajectory that it has calculated the step before. If then the calculation fails again, the third value is set, and so on. The results were incredibly similar to the ideal case, and this is because the steps are so close to each other that the difference between them is very small. We can conclude that the most important thing for this type of control is the value of ts because it must evaluate the system dynamic accurately enough, but then we can just skip one step of the inputs without many troubles.

We would also like to show the results of this analysis for the trial with much lower ts with figure 2.14. In that case we required a much thinner discretization and we have obtained comparable results for what regards performances and stability but, as we can see from figure 2.28, the time required for each optimization is always bigger than the discretization step. In this case not even one optimization is made fast enough to be applied in a real time implementation.

At the end we can say that our AFS control, even if it was developed in a programming language that is known to be slow, can be capable of driving a vehicle at normal speed and even in highways. The results shown in 2.27 are in fact comparable with those that we have got from trials at higher speeds, even if the MPC is renown for being an intricate control. We are satisfied with those values since every paper about this control complains about the

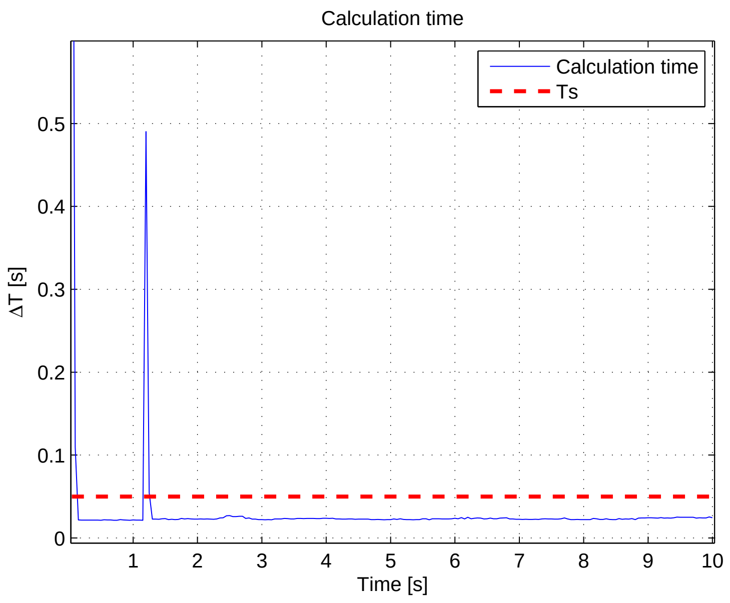

Figure 2.27: Calculation time

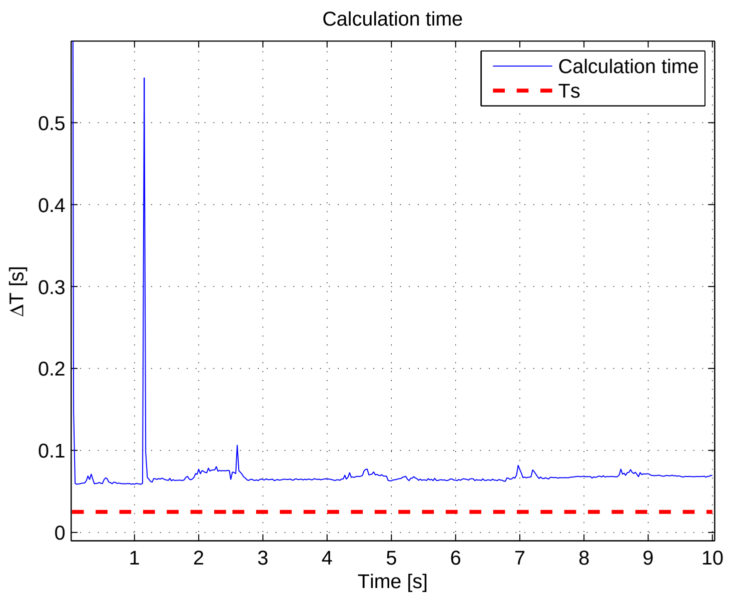

Figure 2.28: Calculation Time $Ts = 0.025$

fact that it can be used only at reduced speed.

Chapter 3

LTV MPC Complete vehicle

This chapter is dedicated to the linear control that drives the whole vehicle. This section could be the motion planner part of a bigger control that is capable of driving a vehicle on every day roads.

3.1 Vehicle models and linearization

As before we are dealing with two different models. The first one is more simple and it is composed by a single track 2 contact model where the relaxation lengths are neglected. The second one is the one used to simulate the vehicle and therefore is the most sophisticated that we have that is the one we explained in the appendix. If we write the system dynamic considering the 2 contact model we obtain the following.

$$\begin{cases} \dot{V_x} = (m\dot{\sigma}V_y + F_{lf}\cos(\delta) - F_{tf}\sin(\delta) + F_{lr})/m \\ \dot{V_y} = (-m\dot{\sigma}V_x + F_{tf}\cos(\delta) + F_{lf}\sin(\delta) + F_{tr})/m \\ \ddot{\sigma} = (-F_{tr}b + F_{tf}a\cos(\delta) + F_{lf}a\sin(\delta))/J_z \\ \dot{\sigma} = \dot{\sigma} \\ \dot{x_g} = V_x\cos(\sigma) - V_y\sin(\sigma) \\ \dot{y_g} = V_x\sin(\sigma) + V_y\cos(\sigma) \end{cases} \tag{3.1}$$

Again we have only F_{lf} because, being a 2 contact model only one force per axes is considered, and again we have neglected the expression of the forces for clarity. The linearization follows the same steps as before. At every sample time k we will have a set state vector around which we are going to linearize our system. From this linearization we will get exactly the same A_m matrix as the AFS control, in fact the state vector is the same also in this control. What changes is the B_m matrix that has got a column more because we have introduced a control variable. Even if we want to deal with both the accelerator and the brake pedal, we have only added one control variable because that is what we are allowed to do because we can introduce only a reference more and we want to keep the control squared. For this reason the new control variable α will vary from -1 to 1 and will represent the engine when positive, and the brakes when negative. With this simplification the control will never be able to brake and accelerate simultaneously, but this is a situation that is never helpful while avoiding obstacles.

$$A_m = \begin{bmatrix} \frac{\vartheta \dot{V}_x}{\vartheta V_x} & \frac{\vartheta \dot{V}_x}{\vartheta V_y} & \frac{\vartheta \dot{V}_x}{\vartheta \dot{\sigma}} & 0 & 0 & 0 \\ \frac{\vartheta \dot{V}_y}{\vartheta V_x} & \frac{\vartheta \dot{V}_y}{\vartheta V_y} & \frac{\vartheta \dot{V}_y}{\vartheta \dot{\sigma}} & 0 & 0 & 0 \\ \frac{\vartheta \ddot{\sigma}}{\vartheta V_x} & \frac{\vartheta \ddot{\sigma}}{\vartheta V_y} & \frac{\vartheta \ddot{\sigma}}{\vartheta \dot{\sigma}} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{\vartheta \dot{x}}{\vartheta V_x} & \frac{\vartheta \dot{x}_g}{\vartheta V_y} & 0 & \frac{\vartheta \dot{x}_g}{\vartheta \sigma} & 0 & 0 \\ \frac{\vartheta \dot{y}_g}{\vartheta V_x} & \frac{\vartheta \dot{y}_g}{\vartheta V_y} & 0 & \frac{\vartheta \dot{y}_g}{\vartheta \sigma} & 0 & 0 \end{bmatrix}; \qquad B_m = \begin{bmatrix} \frac{\vartheta \dot{V}_x}{\vartheta \delta} & \frac{\vartheta \dot{V}_x}{\vartheta \alpha} \\ \frac{\vartheta \dot{V}_y}{\vartheta \delta} & \frac{\vartheta \dot{V}_y}{\vartheta \alpha} \\ \frac{\vartheta \ddot{\sigma}}{\vartheta \delta} & \frac{\vartheta \ddot{\sigma}}{\vartheta \alpha} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad (3.2)$$

Of course the constraints on the steering are the same as before, but we must introduce constraints for what regards the α input. The upper and the lower bounds are set at -1 and 1 (or -100% and 100%), and also a limitation on his rate of change was introduced. Since we have introduced both limits for the positive and for the negative rate of change, we have decided not to allow too fast acceleration therefore we allowed only $200\%/s$ variation when releasing the brake or accelerating. On the other hand we would like the control to be able to act rapidly when using the brake and therefore a limit of $400\%/s$ was set for when releasing the accelerator or braking. These values were chosen while looking at the behaviour of a race driver.

As we can see from table 3.1, we needed to introduce both weights for the reference and for the control. The first reference is as always the lateral position, while the second one is the longitudinal speed of the vehicle, and from now on will be set at the speed limit. The controller will have to separate from the reference to manage the obstacle and since maintaining the speed is not important while avoiding obstacle, its weight is very little. We still need it though because otherwise the vehicle will not get back to the required speed after the obstacle.

3.2 Path follower

In this section we can see our controller following the two reference that we are going to use. A reference is, as for the AFS control, the lateral position,

n_y	n_c	$ts\ [s]$	$\tilde{Q}$	$\tilde{R}$
50	10	0.05	$\begin{bmatrix}1 & 0\\ 0 & 0.1\end{bmatrix}$	$\begin{bmatrix}50000 & 0\\ 0 & 10\end{bmatrix}$

Table 3.1: Complete - Parameters

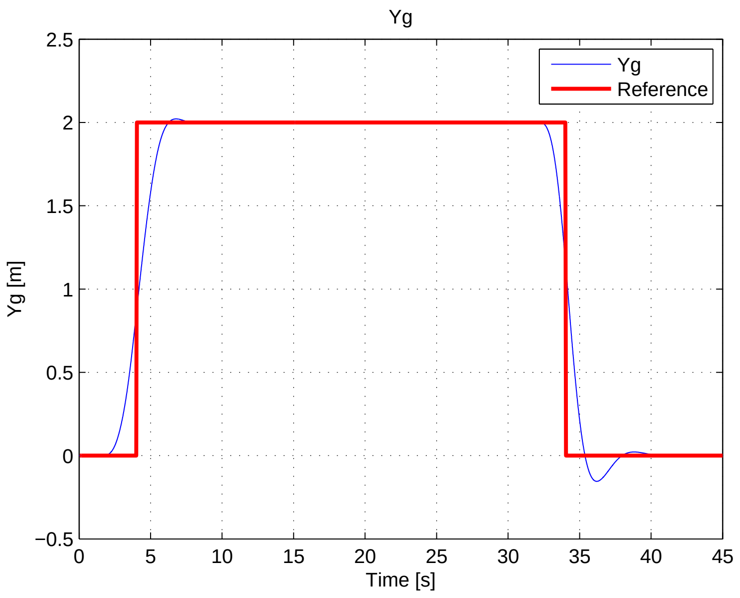

Figure 3.1: Complete path follower - Lateral displacement

while the other one is the longitudinal speed of the vehicle, that is the speed that is displayed by the tachometer. With this trial we want to show that our controller is very good at following references and also that both the model and the linearization are correct. In figure 3.1 and 3.2 we can see that we have imposed two steps in both references and that the controller is capable of following them with no problem. The step in lateral displacement is very similar compared with the step that the AFS did in figure 2.2. The step in speed shows that the vehicle is less responsive if a speed increase is required compared to when a speed decrease is required and that is because the brake maximum torque is much higher with respect to the maximum torque that the engine can provide. Performances for what regards the reference in speed is slightly worse than expected but since the control is useful for avoiding obstacle and it is not programmed to perform speed steps, we have decided that weights and parameters could be kept as in table 3.1.

In figure 3.3 we can see the operation of the engine with the gearbox.

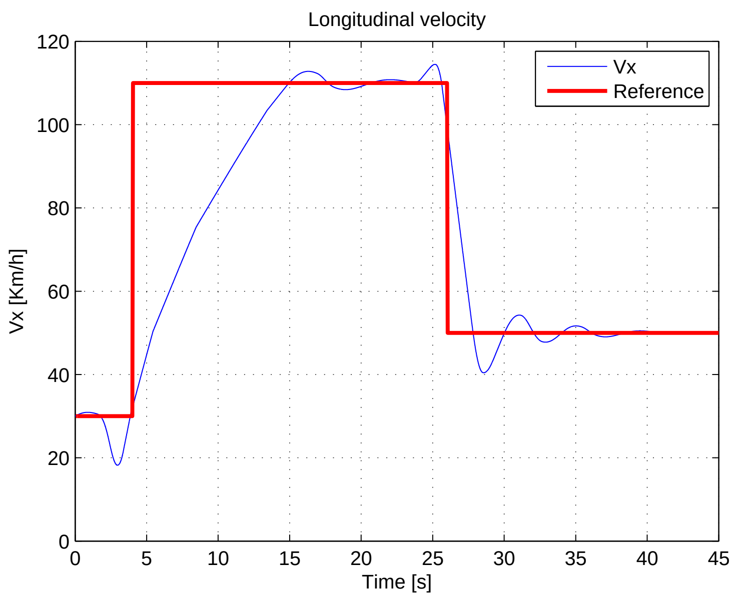

Figure 3.2: Complete path follower - Longitudinal speed

As we can see the automatic gearbox logic is working perfectly and is not creating any problem to the controller that is capable of slowing down or accelerating the car. In figure 3.4 the control usage of the car inputs is shown. We can see that it is never accelerating or braking simultaneously and that the functioning is coherent with the requirements. Even if we can say that we have built an efficient model predictive cruise control, we can see that its performance is not perfect but, as we have stated before, the main purpose of this control is not to be capable of following a speed reference perfectly, but only to guide the vehicle around obstacles. The dotted line represent the value of throttle pedal that cancel the engine brake.

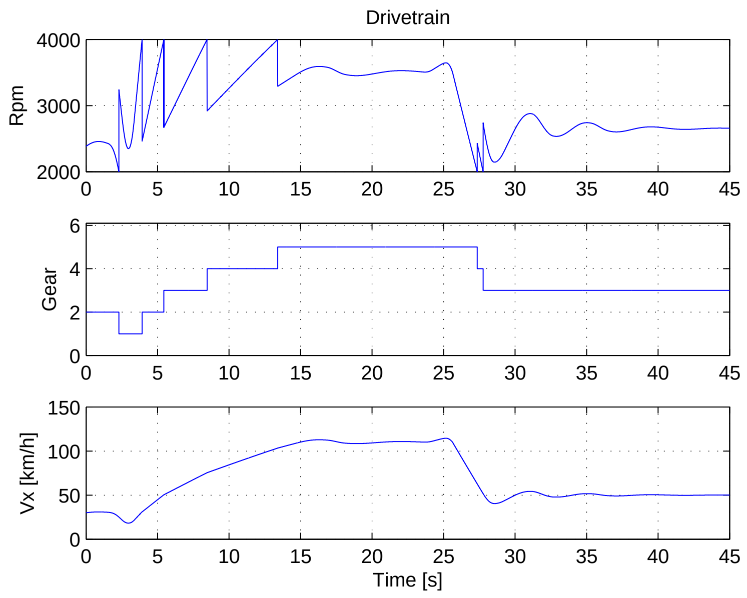

Figure 3.3: Complete path follower - Drivetrain

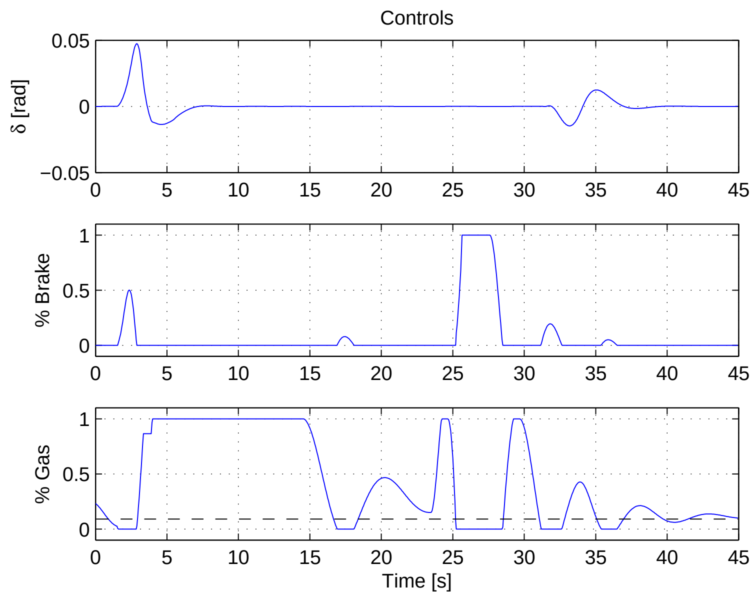

Figure 3.4: Complete path follower - Controls

3.3 Obstacle

Next in our work, the standard obstacle at $50m$ with an initial speed of $50km/h$ is tackled. There is a note that we must add to understand that there are difficulties with this control. The linear MPC can only manage linear system and therefore linear constraints. As we have explained before, linear constraints are constraints that are not dependant on the manipulated variables. This is true for the the AFS control, but not for this one. Here the longitudinal speed of the vehicle is modified by the α input, and therefore the speed of the vehicle is not the initial speed anymore. In the past, since we needed to program the obstacle in the time domain even if it is in reality a constant value in the space domain, we simply calculated the time that corresponded to the space considering the speed as a constant value. We cannot do this anymore, the constraint distance in the time domain is a function of the inputs. In fact if the car brakes it will take more time to get to the obstacle. This effect cannot be considered with the linear control. The optimization process of a linear MPC will never be able to understand that braking the vehicle will mean more available time to manage the obstacle. Still, even if those problems will always be true, we have to find a way to adjust the constraint if the longitudinal speed changes through the trial. At each sample time k we calculate the average speed until that point and we evaluate the distance in meters that there is between the vehicle and the obstacle. Then from those values we obtain the expected time distance and we set that as the new distance to the obstacle.

Figure 3.5 shows us that the trial is a success. The control is capable to consider and avoid the obstacle without any problem. From figure 3.6 we understand that it actually has not used the brake very much.

We would like to show results also for another trial at $110km/h$ with the obstacle set at $50m$. From figure 3.8 we can see that the brake was neglected completely even if the obstacle has to be tackled at higher speed. We added also a plot of the slip angles in figure 3.9 so that we can see that the manoeuvre has still margin (the maximum of the lateral force of the tyre happens at $0.12rad$) but we might want anyway the brake to be used to increase the safety of his operations.

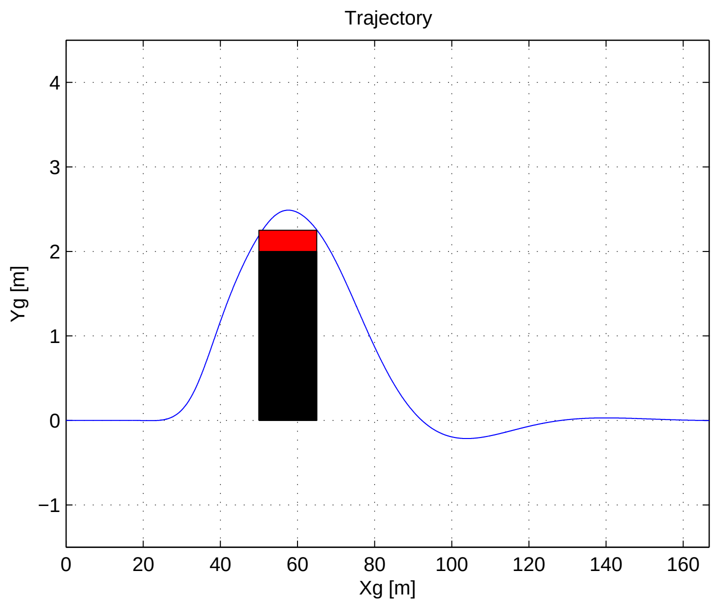

Figure 3.5: Complete obstacle - Trajectory

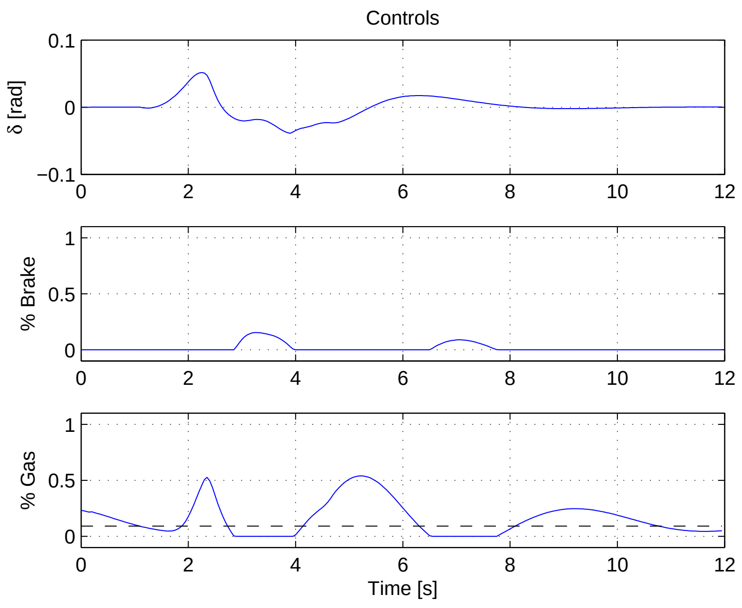

Figure 3.6: Complete obstacle - Controls

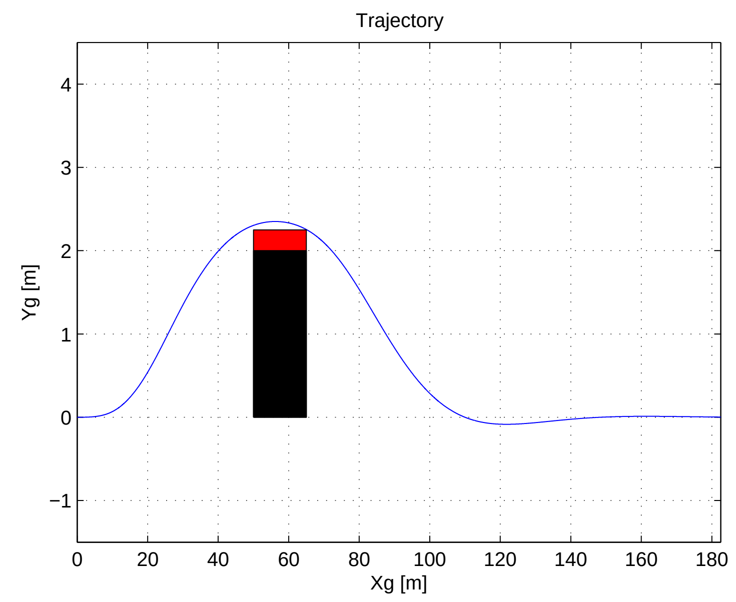

Figure 3.7: Complete obstacle 110km/h - Trajectory

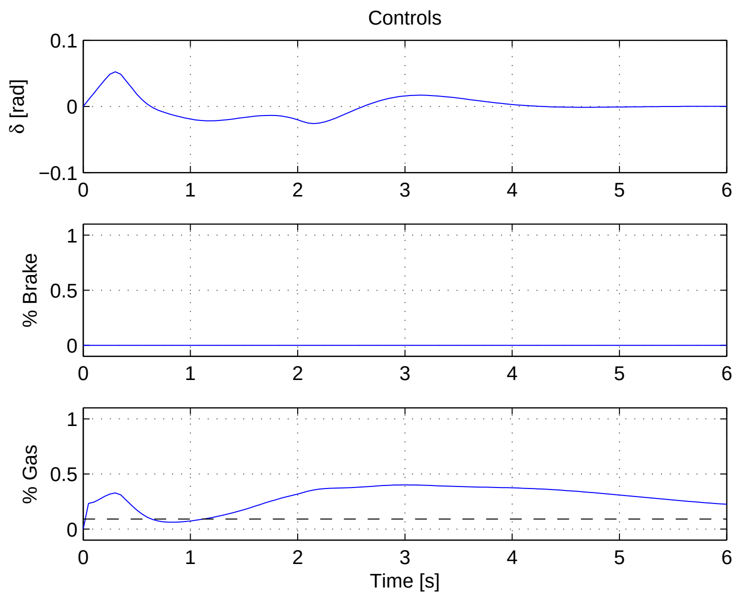

Figure 3.8: Complete obstacle 110km/h - Controls

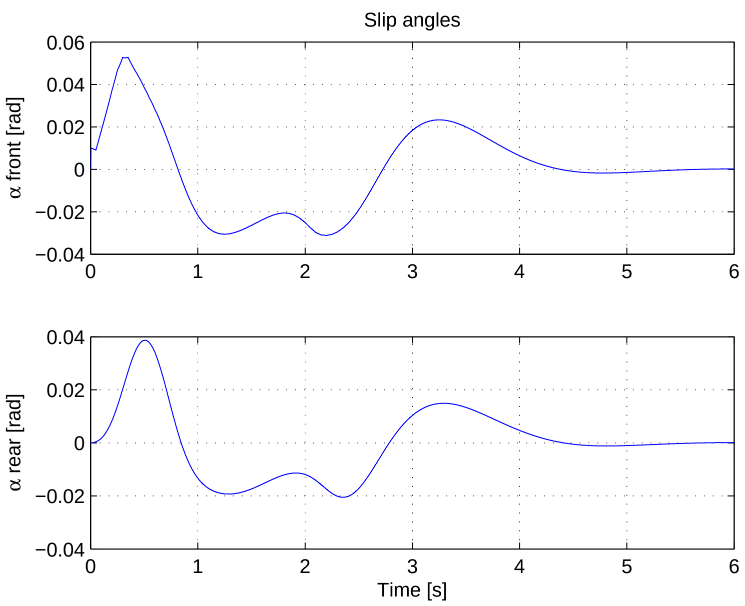

Figure 3.9: Complete obstacle 110km/h - Slip angles

3.4 Obstacle with rear slip angle control

Since we want to force the controller to brake, we tried adding a reference path also for the rear slip angle. This will cause the system not to be squared anymore, so from now on we cannot pretend the control to get to each reference. Our intention was to help him understand that the smoother the manoeuvre the better, and if the slip angles are low, that means that little lateral force is exchanged and therefore is far from the limits. Being a linear control we can impose references only to values that are linear combination of the state of the system. This is not true for the slip angles, and therefore we modified the state vector such that α_r was added.

If we derive the α_r nonlinear expression with respect to time and we neglect the tangent function we can write $\dot{\alpha_r}$:

$$\dot{\alpha_r} = ((\ddot{\sigma}b - \dot{V_y})V_x - \dot{V_x}(\dot{\sigma}b - V_y))/V_x^2 \tag{3.3}$$

We can add this equation to the system:

$$\begin{cases} \dot{V_x} = (m\dot{\sigma}V_y - F_{tf}\sin(\delta))/m \\ \dot{V_y} = (-m\dot{\sigma}V_x + F_{tf}\cos(\delta) + F_{tr})/m \\ \ddot{\sigma} = (-F_{tr}b + F_{tf}a\cos(\delta))/J_z \\ \dot{\sigma} = \dot{\sigma} \\ \dot{x_g} = V_x\cos(\sigma) - V_y\sin(\sigma) \\ \dot{y_g} = V_x\sin(\sigma) + V_y\cos(\sigma) \\ \dot{\alpha_r} = ((\ddot{\sigma}b - \dot{V_y})V_x - \dot{V_x}(\dot{\sigma}b - V_y))/V_x^2 \end{cases} \tag{3.4}$$

and from this equation at each sample step k a linearization is made from which we obtain:

$$A_m = \begin{bmatrix} \frac{\vartheta\dot{V_x}}{\vartheta V_x} & \frac{\vartheta\dot{V_x}}{\vartheta V_y} & \frac{\vartheta\dot{V_x}}{\vartheta\dot{\sigma}} & 0 & 0 & 0 & 0 \\ \frac{\vartheta\dot{V_y}}{\vartheta V_x} & \frac{\vartheta\dot{V_y}}{\vartheta V_y} & \frac{\vartheta\dot{V_y}}{\vartheta\dot{\sigma}} & 0 & 0 & 0 & 0 \\ \frac{\vartheta\ddot{\sigma}}{\vartheta V_x} & \frac{\vartheta\ddot{\sigma}}{\vartheta V_y} & \frac{\vartheta\ddot{\sigma}}{\vartheta\dot{\sigma}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{\vartheta\dot{x_g}}{\vartheta V_x} & \frac{\vartheta\dot{x_g}}{\vartheta V_y} & 0 & \frac{\vartheta\dot{x_g}}{\vartheta\sigma} & 0 & 0 & 0 \\ \frac{\vartheta\dot{y_g}}{\vartheta V_x} & \frac{\vartheta\dot{y_g}}{\vartheta V_y} & 0 & \frac{\vartheta\dot{y_g}}{\vartheta\sigma} & 0 & 0 & 0 \\ \frac{\vartheta\dot{\alpha_r}}{\vartheta V_x} & \frac{\vartheta\dot{\alpha_r}}{\vartheta V_y} & \frac{\vartheta\dot{\alpha_r}}{\vartheta\dot{\sigma}} & 0 & 0 & 0 & 0 \end{bmatrix}; \qquad B_m = \begin{bmatrix} \frac{\vartheta\dot{V_x}}{\vartheta\delta} & \frac{\vartheta\dot{V_x}}{\vartheta\alpha} \\ \frac{\vartheta\dot{V_y}}{\vartheta\delta} & \frac{\vartheta\dot{V_y}}{\vartheta\alpha} \\ \frac{\vartheta\ddot{\sigma}}{\vartheta\delta} & \frac{\vartheta\ddot{\sigma}}{\vartheta\alpha} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\vartheta\dot{\alpha_r}}{\vartheta\delta} & 0 \end{bmatrix} \tag{3.5}$$

This new linearized system has α_r on its state vector so we can impose a reference in zero to it. The same derivation cannot be made to α_f and this is the reason why we had to use the rear one.

We have simulated the same trial as before, with the obstacle at $50m$ and initial speed at $110km/h$, and the parameters used for this are shown in table 3.2. We can see the results in figure 3.10. The obstacle was overtaken without many problems, but the control seems to be a little less stable than before. Unfortunately the brake is still not used properly, and only minor values are used after the obstacle. The values of the rear slip angles has not improved and this is due to the fact that the prediction of the said slip angles is very bad. The linearization introduces too many errors and we cannot pretend the prediction to be accurate. From figure 3.13 we find also another problem. Because the prediction is so bad, hard constraints cannot be used on the rear slip angles. If we impose a lower bound of $-0.04rad$, even if in this trial the simulated vehicle has not reached that value, the optimizator will fail to find a suitable solution because the prediction goes well over it and there is no possible manoeuvre to keep it lower.

n_y	n_c	ts $[s]$	$\tilde{Q}$	$\tilde{R}$
50	10	0.05	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1500 \end{bmatrix}$	$\begin{bmatrix} 50000 & 0 \\ 0 & 10 \end{bmatrix}$

Table 3.2: Complete obstacle w α_r control - Parameters

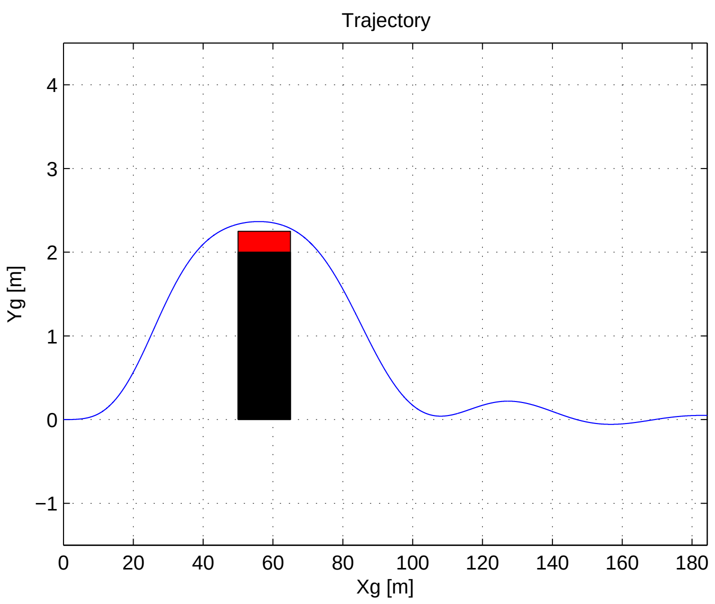

Figure 3.10: Complete obstacle w α_r control - Trajectory

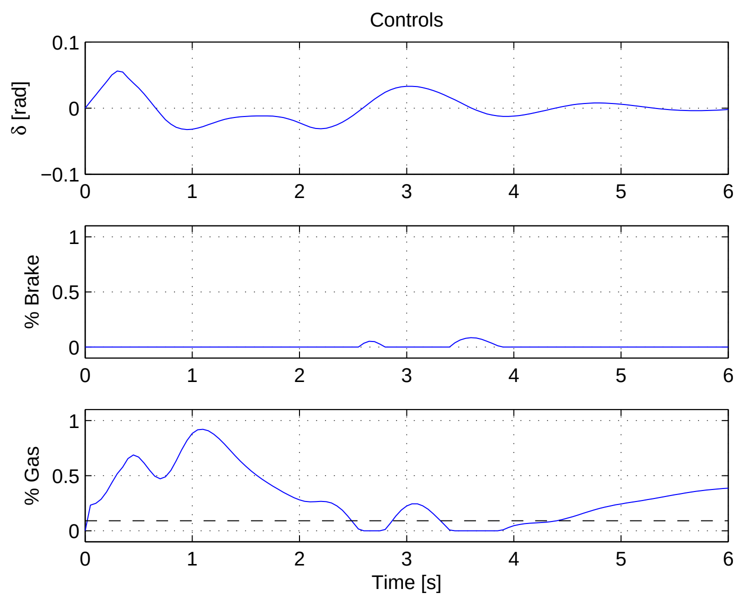

Figure 3.11: Complete obstacle w α_r control - Controls

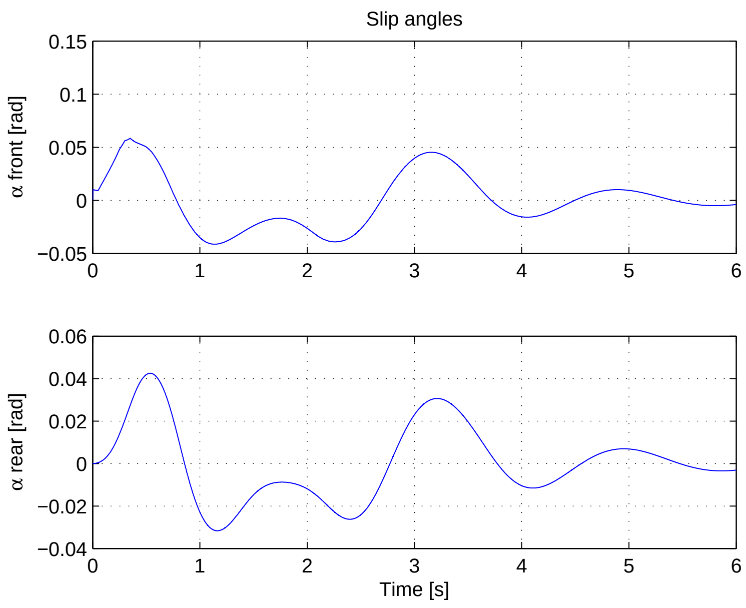

Figure 3.12: Complete obstacle w α_r control - Slip angles

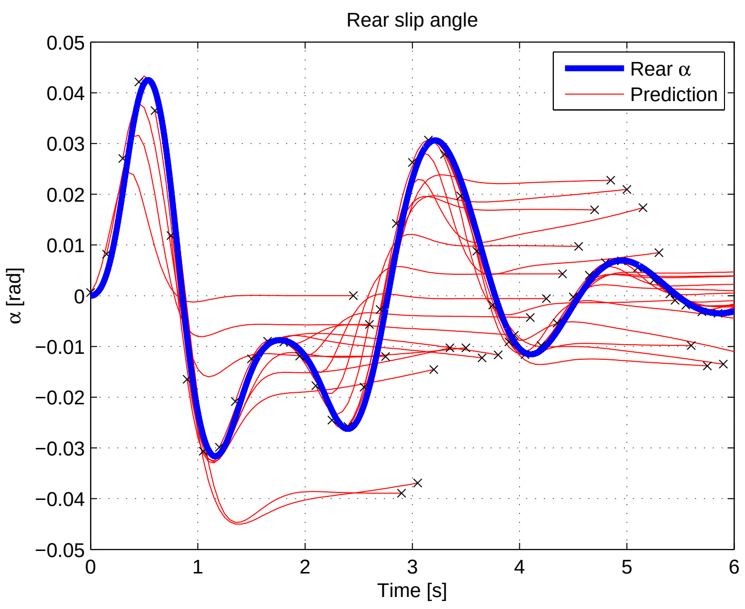

Figure 3.13: Complete obstacle w α_r control - Rear slip angle prediction

3.5 Obstacle with lateral speed or yaw rate control

Other simulations were done with references to other state variables to see if somehow the control was able to understand the usage of the brake pedal. The first one is performed with a reference in zero to the lateral speed of the vehicle. Since it was originally part of the state vector no further modifications are required for both this and the next trial. The weight for the α_r control is substituted with the one for the lateral speed in table 3.3. The trajectory in figure 3.14 is very similar to the previous one, but we can notice that the control used a little bit of brake before the obstacle. Unfortunately, as we can see comparing figure 3.16 with the brake input, the braking peak happens when the vehicle is not straight anymore. Braking in that condition is not optimal. We would like the braking to happen in the straight line and not while cornering since using longitudinal forces reduces the maximum lateral force available.

The parameters for the trial when a reference on the yaw rate was imposed are shown in table 3.4. Also for this example the control brakes while cornering and the control seems to be even less stable than the previous one as we can see in figure 3.17.

n_y	n_c	ts $[s]$	$\tilde{Q}$	$\tilde{R}$
50	10	0.05	$\begin{bmatrix}1 & 0 & 0\\0 & 0.1 & 0\\0 & 0 & 5\end{bmatrix}$	$\begin{bmatrix}50000 & 0\\0 & 10\end{bmatrix}$

Table 3.3: Complete obstacle w V_y control - Parameters

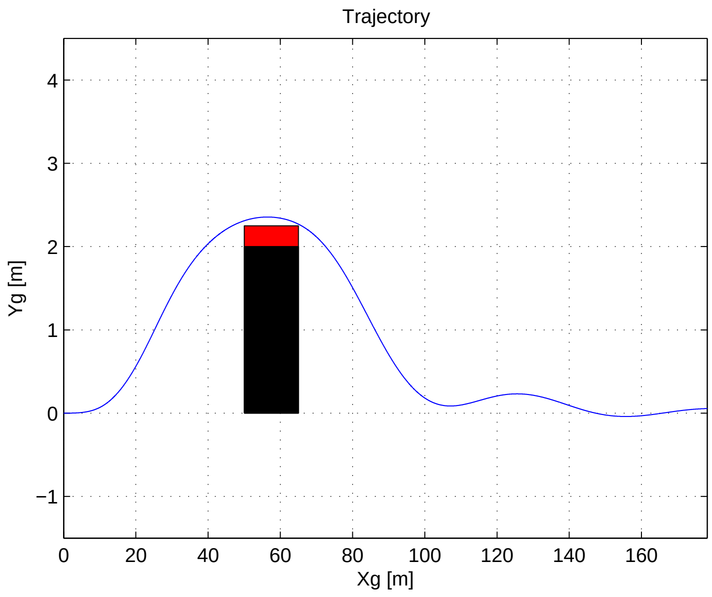

Figure 3.14: Complete obstacle w V_y control - Trajectory

n_y	n_c	ts $[s]$	$\tilde{Q}$	$\tilde{R}$
50	10	0.05	$\begin{bmatrix}1 & 0 & 0\\0 & 0.1 & 0\\0 & 0 & 50\end{bmatrix}$	$\begin{bmatrix}50000 & 0\\0 & 10\end{bmatrix}$

Table 3.4: Complete obstacle w $\dot{\sigma}$ control - Parameters

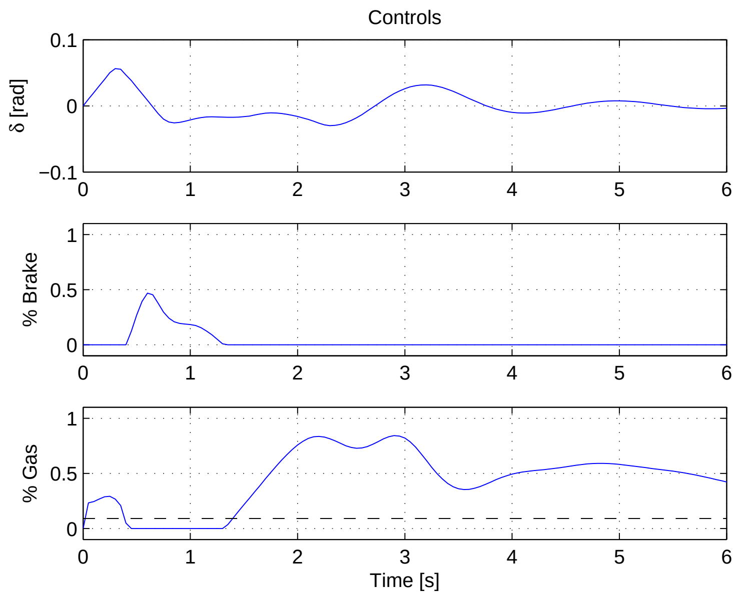

Figure 3.15: Complete obstacle w V_y control - Controls

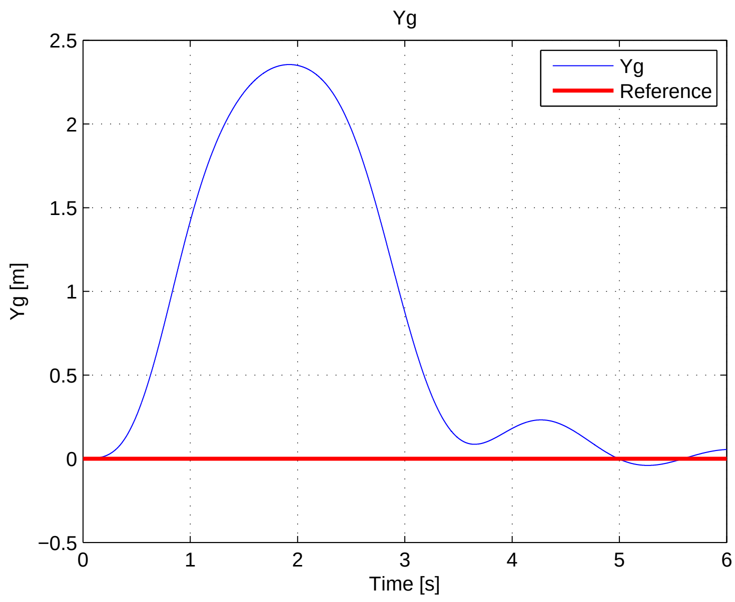

Figure 3.16: Complete obstacle w V_y control - Lateral displacement

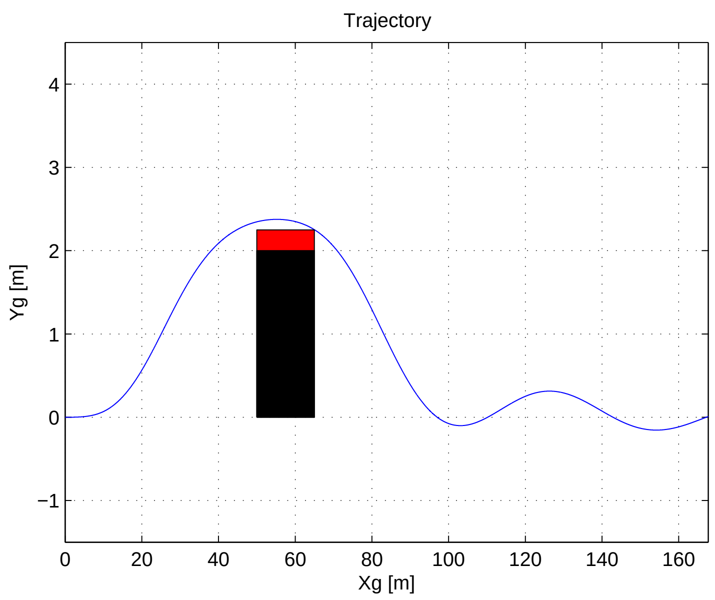

Figure 3.17: Complete obstacle w $\dot{\sigma}$ control - Trajectory

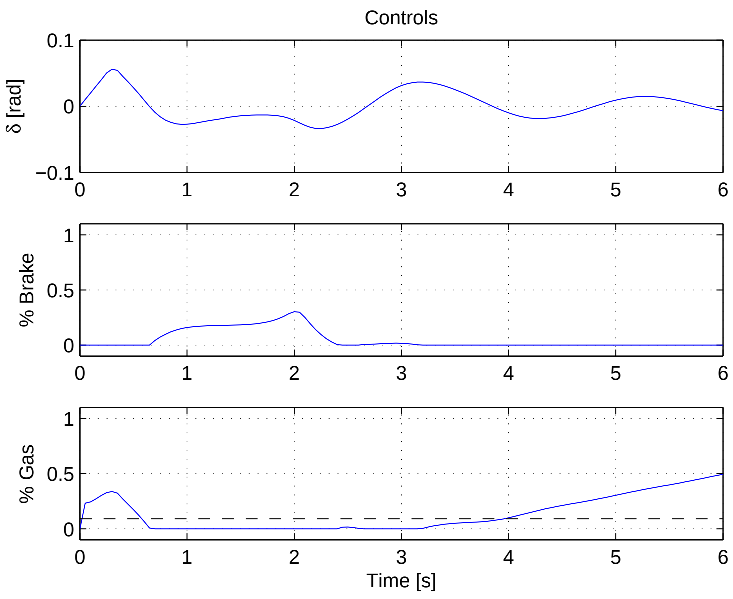

Figure 3.18: Complete obstacle w $\dot{\sigma}$ control - Controls

3.6 Limits of LTV-MPC

In the previous section we have shown how the LTV-MPC was able to follow references. The control is also capable of taking into account obstacles and avoiding them, but it did not show capabilities of slowing down the vehicle in the straight line before the obstacle.

This is a limit of the linear control. To explain this we can show an example of the matrices that the linearization process calculates when the vehicle is in the straight line. These matrices are from the trial with the control also in the rear slip angle.

$$A_m = \begin{bmatrix} -0.0171 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4.5718 & -28.9954 & 0 & 0 & 0 & 0 \\ 0 & 1.0209 & -5.6475 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0 & 0 & -0.0153 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 30.5556 & 0 & 0 & 0 \\ 0 & 0.1984 & 0.6785 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_m = \begin{bmatrix} -0.0582 & 1.5852 \\ 58.2035 & 0 \\ 47.0368 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.3504 & 0 \end{bmatrix}$$

As we can see there is no correlation between the longitudinal speed and the slip ratio variation because the first value of the last row is zero. This means that if the vehicle is straight, the prediction of the slip angle does not depend on the speed, and therefore the control does not see any advantages in slowing down the vehicle. The control in fact decides that braking is useful only when the car has already turned a bit, but that is not the optimal condition for braking. On top of this limit of the linear control, we must also remember that the bad quality of the predictions caused also the impossibility to insert hard constraints on the slip ratios, and in general we can say that controlling very nonlinear functions and nonlinear constraints it has been proven very difficult with this type of control.

The only possibility of controlling the front slip ratio or the lateral acceleration, or using hard constraints on the slip ratios, or managing nonlinear constraints is to use a nonlinear model predictive control.

Chapter 4

NMPC Complete vehicle

A model of a road vehicle, even if we want to consider the steering input only, is a complex nonlinear system that is not supposed to be used with a linear control. The most logical thing to do is to threat it with a nonlinear model predictive control, but at the beginning we wanted to try the linear model predictive control first to thoroughly examine the limits of the linear controller. The linear controller was also preferred because it is much faster in its operations. The MPC controller is renown for being very difficult to implement in real time applications because it is very laborious for the calculator, especially if it has to deal with a big nonlinear system.

In this chapter we will threat the nonlinear control that is supposed to be able to deal with nonlinear or even hybrid systems, with nonlinear constraints. As we can understand from the state of the art section dedicated to this control, the nonlinear controller is of much easier implementation for the programmer, but it leaves all the difficulties to the minimization algorithm, which for us is the fmincon function of Matlab.

4.1 Vehicle models

For this type of controller, since it is able to cope with every type of model, we decided to use the same 4 contact single track model with steering and α as inputs that we used in the linear complete vehicle both in the simulation and in the controller. With this hypothesis we have no variance between the predicted outputs and the simulated outputs, and so we can understand the capabilities of the control to cope with the constraints, that is the first thing we wanted to clarify. The whole system will be written as:

$$\begin{cases} m\dot{V}_x = m\dot{\sigma}V_y - (F_{tfl} + F_{tfr})\sin(\delta) + (F_{lfl} + F_{lfr})\cos(\delta) + (F_{lrl} + F_{lrr}) - F_{aero} \\ m\dot{V}_y = -m\dot{\sigma}V_x + (F_{tfl} + F_{tfr})\cos(\delta) + (F_{trl} + F_{trr}) + (F_{lfl} + F_{lfr})\sin(\delta) \\ J_z\ddot{\sigma} = +(F_{tfl} + F_{tfr})\cos(\delta)a - (F_{trl} + F_{trr})b + (F_{lfl} + F_{lfr})a\sin(\delta) \\ \dot{x} = V_x\cos(\sigma) - V_y\sin(\sigma) \\ \dot{y} = V_x\sin(\sigma) + V_y\cos(\sigma) \end{cases} \tag{4.1}$$

where the expressions of the four tyre transversal forces, functions of the slip angles, have been omitted for clarity. No further modifications were made because this is what directly is used both in the control and in the simulation.

We are going to apply the same constraints that we have used in the linear complete vehicle, but we are going to neglect the constraints on the rate of change of the manipulated variables. This is because the nonlinear optimizator has got unfortunately many limits, among which we find difficulties in managing hard constraints priorities. We have tried to use also the said hard constraints in the rate of change of the manipulated variables, but then the control started to neglect the constraints represented by the obstacle. This was not because the constraints on the rate of change were actually limiting the control, it was simply causing numerical problems that we decided to avoid by neglecting them. On the other hand we managed to insert hard constraints in the front and rear slip angles. With this type of control we are capable of obtaining accurate predictions also for what regards the slip angles, and therefore it was possible to require the control to deal with also limits on them. For what regards the modifications that were required for the linear controller to consider a obstacle with fixed position in the space domain, and therefore floating position in the time domain, here are not needed anymore since the nonlinear controller is able to directly taking into account its presence.

4.2 Path follower

To test the truthfulness of the nonlinear controller we run the same test as we did for the linear controller. Exactly the same references in the lateral position and longitudinal speed were used. The parameters that were used for the controller have exactly the same meaning of the one used for the linear one and are shown in table 4.1. The numbers are slightly different, but the order of magnitude in difference between them is similar. Those number are the results of the normalization with respect to the maximum error possible for each measurement. Again the $\tilde{Q}$ matrix is the matrix of the

n_y	n_c	$ts\ [s]$	$\tilde{Q}$	$\tilde{R}$
50	35	0.05	$\begin{bmatrix} 55 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 3125 & 0 \\ 0 & 0 & 0 & 3125 \end{bmatrix}$	$\begin{bmatrix} 650 & 0 \\ 0 & 20 \end{bmatrix}$

Table 4.1: NMPC - Parameters

outputs weights and $\tilde{R}$ is the matrix of the two controls. As we can see we are weighting four different outputs. The latest two are the front and rear slip angles that we decided to include since they are fundamental values for the vehicle dynamic and must be kept under control. Specifically we want them to be as low as possible since we want the controller to have a safe margin while taking corners. We also imposed hard constraints on the maximum front and rear slip angles, since now with the nonlinear controller we are able to have accurate prediction. We set that limit to $0.08rad$.

As we can see from figure 4.1 and 4.2 the results are very good. The references are obtained with very smooth transitions, much better than the linear controller. From figure 4.3 we see how nicely the controller understands and drives the vehicle. This is absolutely clear as there are no hesitations while using the throttle. The difference between the linear and the nonlinear controller is mostly due to the fact that this one is capable of making accurate prediction and therefore taking accurate control actions as we can see comparing figure 4.4 with figure 4.5. Unfortunately this improvement comes with a high cost. This simulation required more than ten hours of computing, an extreme amount of time if we consider that this should be used in a real time implementation.

Since the purpose of this controller is to drive a vehicle on common roads, the controller must be able to manage corners. Unlike all the trials that we have shown until now, in this case the reference trajectory cannot be defined easily in the time domain. To define the reference it is necessary to express it as function of the curvilinear abscissa s that represent the travelled space of the center of mass of the vehicle. To obtain it, the longitudinal speed of the vehicle needs to be integrated. In this way, sample time by sample time, we are going to be able to calculate the value of s and therefore to impose the references $x_{rif}(s)$ and $y_{rif}(s)$. In this example we impose two corners while asking the controller to stay steady at $50km/h$. The hard constraints on both the front and rear slip angles are lowered at $0.04rad$ in order to consider the fact that while driving normally the safe margin should be high. In figure 4.6 we can see how the reference is followed perfectly. From figure 4.7 we

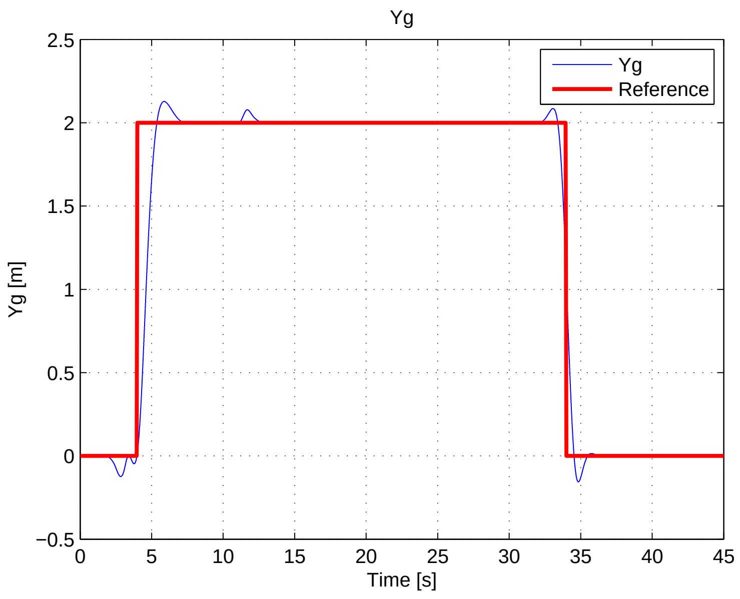

Figure 4.1: NMPC Path follower - Lateral position

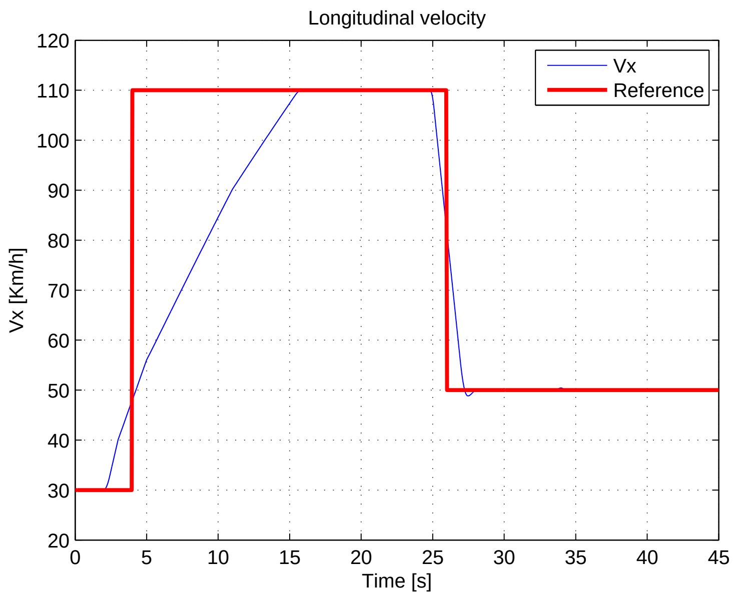

Figure 4.2: NMPC Path follower - Longitudinal speed

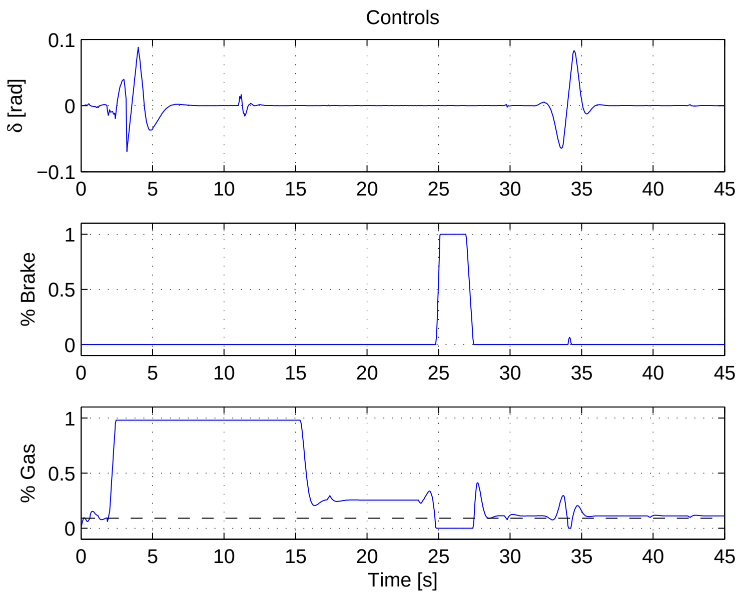

Figure 4.3: NMPC Path follower - Controls

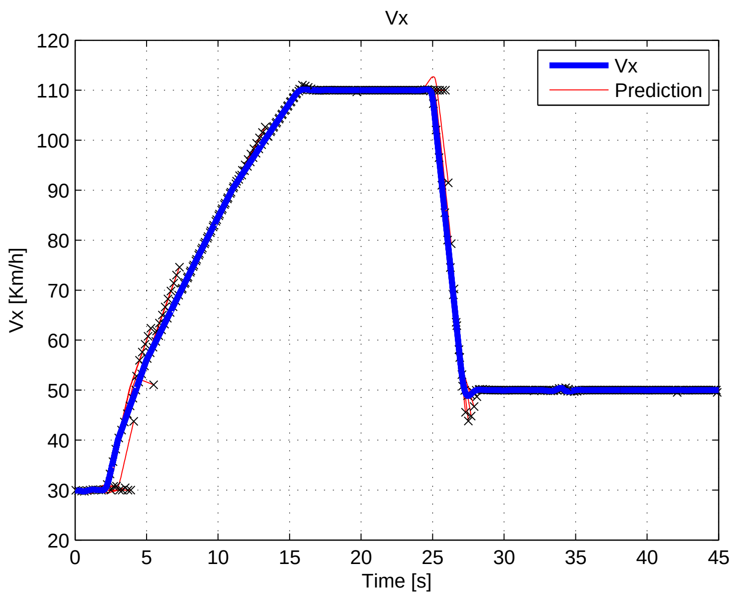

Figure 4.4: NMPC Path follower - Prediction

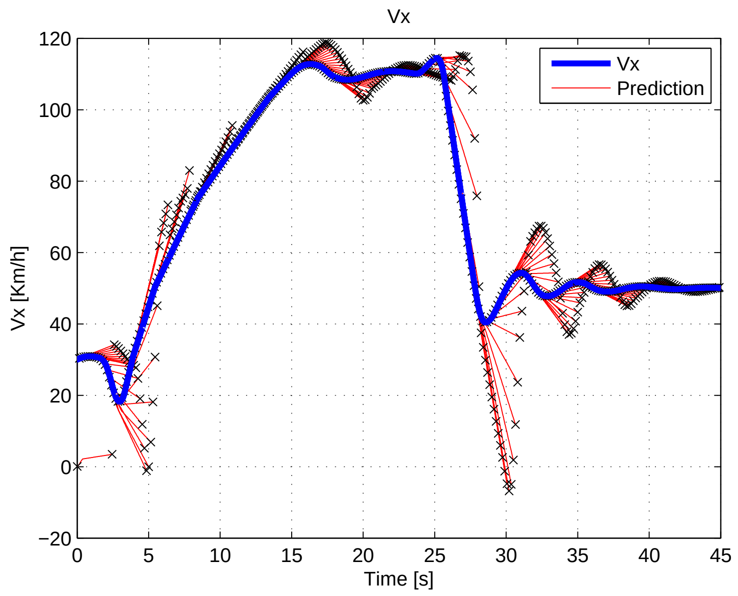

Figure 4.5: Linear Path follower - Prediction

can see that the controller needed to use the brake, especially to engage the second corner that is tighter than the first one. In fact, since the controller is required to keep the slip angles below $0.04 rad$ in figure 4.8, it is forced to reduce its speed before entering the second corner.

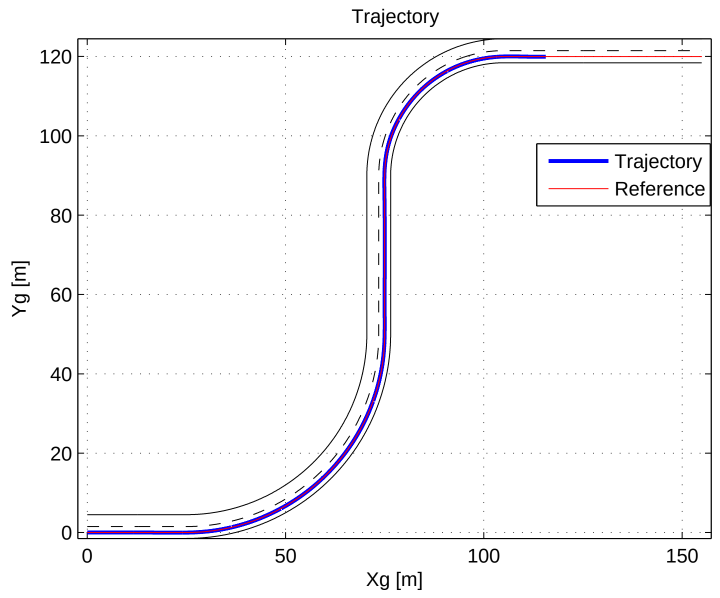

Figure 4.6: NMPC corner - Trajectory

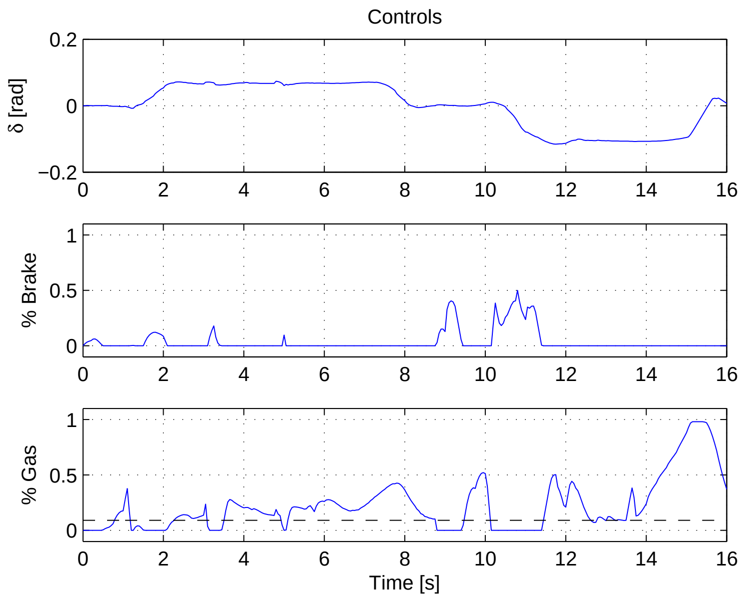

Figure 4.7: NMPC corner - Controls

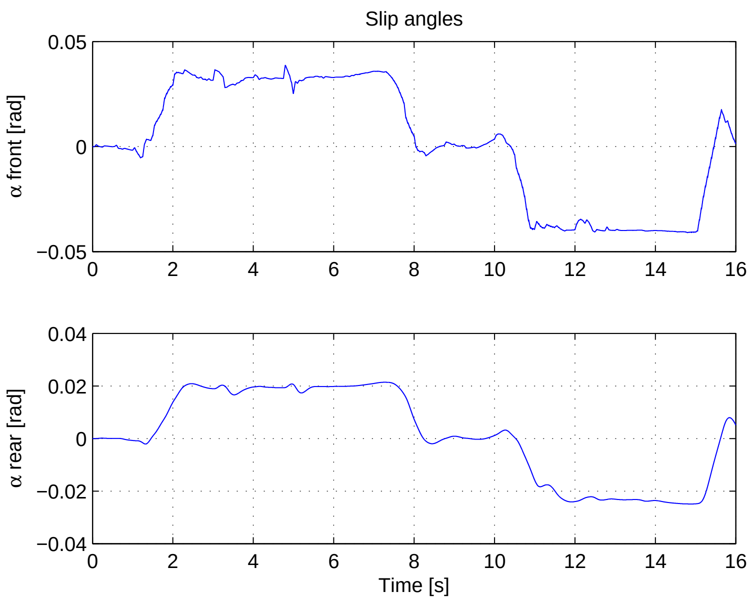

Figure 4.8: NMPC corner - Slip angles

4.3 Fixed obstacle

The real usage of this controller is to avoid obstacles. For the next trial parameters and hard constraints are the same as for the path follower, initial speed is set at $110km/h$ and the distance is the minimum possible, as obtained from a number of trials. Results are in figure 4.9. Its performances have improved compared to the linear controller. Here we are able to avoid the obstacle even if it is placed at 26 meters, while before we had to require at least 32 meters. Also this simulation has required many hours of calculations to simulate this event, and this is the reason why we tried evaluating only the minimum distance for the $110km/h$ initial speed trial instead of doing the complete analysis as we did for the AFS control. Unfortunately here the usage of the brake input is not as we would like to see, since we would like to see the controller to slow down the vehicle while it is in a straight line. We have also to consider that this trial is an extreme event. The space between the vehicle and the obstacle is very small, and the controller could not have any space to actually brake in the straight line. Of course the erratic usage of the accelerator makes is not helpful and it causes also the slip angles to break the hard constraint in figure 4.11. As we can see from figure 4.10 the steering is used at the very beginning of the trial, and in those conditions the accelerator should not be applied.

To test its capabilities of stopping the vehicle while being on a straight

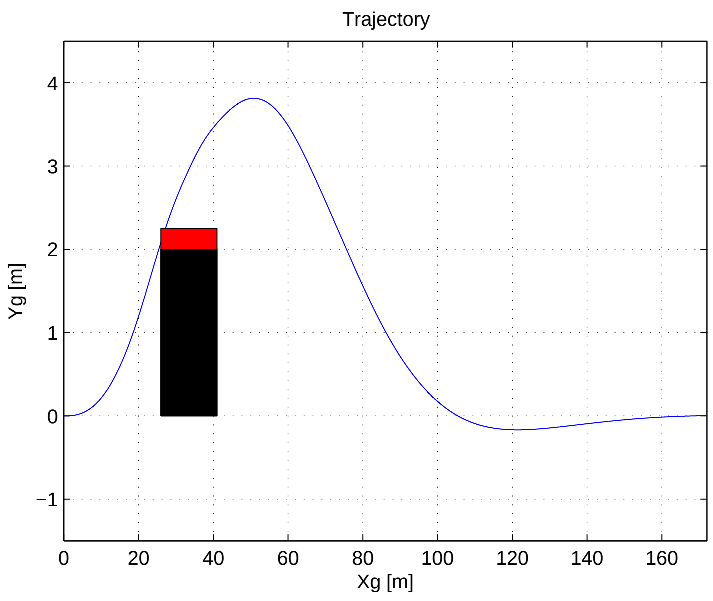

Figure 4.9: NMPC Obstacle $26m$ - Trajectory

line we decided to leave more space between the vehicle and the obstacle. In this way there will be enough space to slow down the vehicle and overtake the obstacle safely. In figure 4.12 we will see the result for the trial where the same initial speed of $110km/h$ was used, but the obstacle was placed at $50m$. As we can see from figure 4.13, the controller is not slowing the vehicle down. It is in fact accelerating until it overtakes the obstacle, then at that point a little bit of brake is applied.

These trials showed that how with weights on the lateral position, on the longitudinal velocity and on the front and rear slip angles it is not possible to impose to the controller to slow down the vehicle. The minimization functions is written in a way that promotes the usage of the brake. Even in there is a weight in the longitudinal speed, taking a corner slowly will reduce a lot more the slip angles, that are weighted more. At this point we considered the cause of our problem to be the obstacle shape.

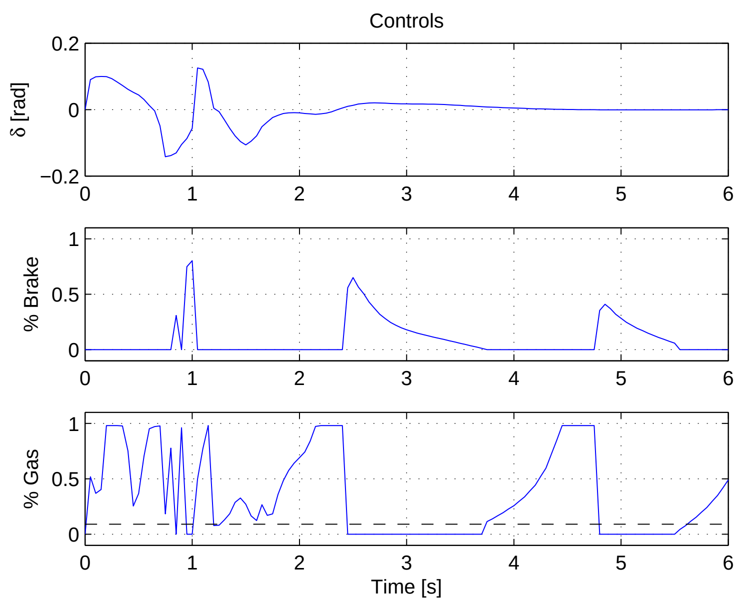

Figure 4.10: NMPC Obstacle 26m - Controls

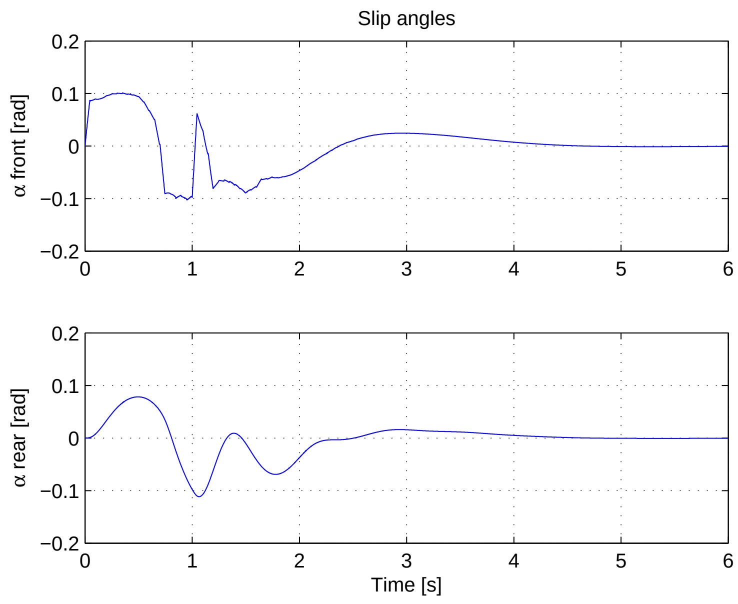

Figure 4.11: NMPC Obstacle 26m - Slip angles

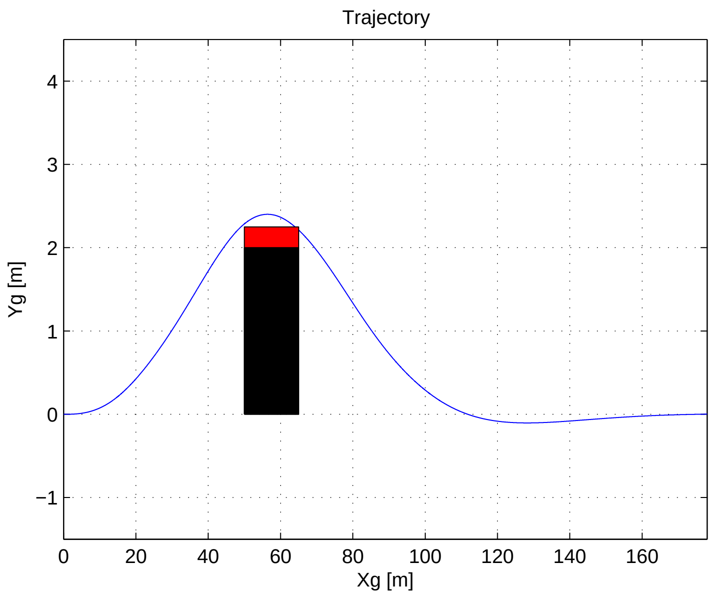

Figure 4.12: NMPC Obstacle 50m - Trajectory

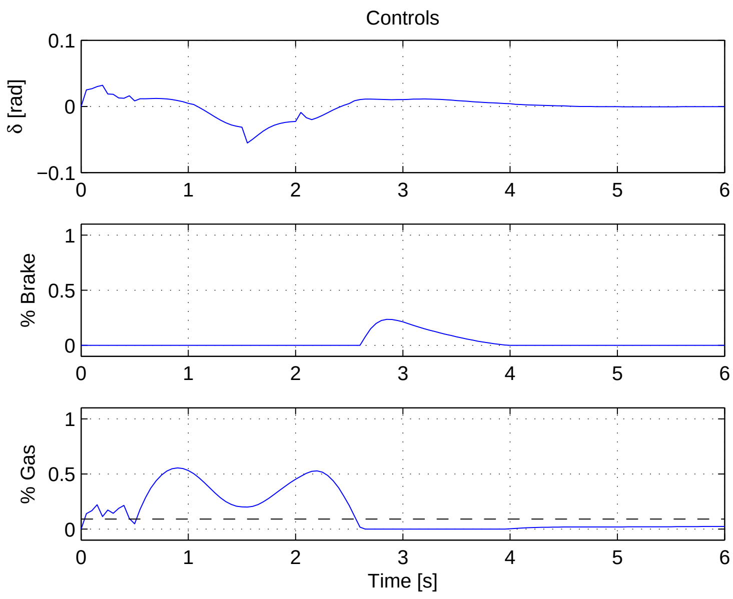

Figure 4.13: NMPC Obstacle 50m - Controls

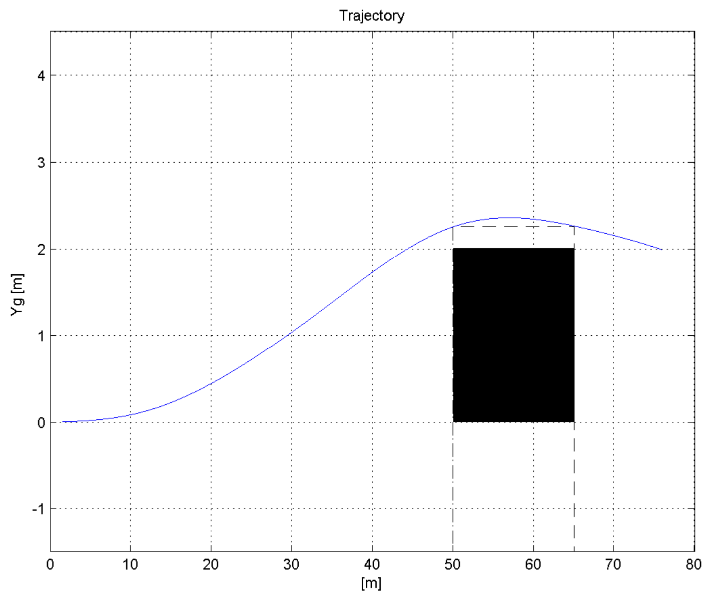

Figure 4.14: Step - Trajectory

4.4 Obstacle shape sensitivity

The nonlinear controller showed sensitivity to the fixed obstacle shape. The obstacle in our work was modelled as a rectangle, but of course we can make the hypothesis to have whichever obstacle shape. The measurement system will detect an obstacle, will determine its sizes and will pass those informations to the controller. If we want, we can confine the real obstacle inside a shape of our choice. Depending on the shape we decided to try, we have experienced great variance with respect to the many trials. The standard event is the fixed obstacle placed at $50m$, with initial speed set at $110km/h$. All those trials, since it is very time consuming to simulate each event, are evaluated only considering the first optimization. What we will see in the graphs are values calculated only for the first time window.

The standard controller with the standard object shape behaves as shown in figure 4.14. As we can see the controller is able to avoid the obstacle, but it is not slowing the vehicle down. It seems like it is trying not to affect at all the longitudinal dynamic of the vehicle.

We then tried to apply a ramp before the obstacle. This ramp was designed to be $25m$ long. As we can see from figure 4.16 the controller is still avoiding the obstacle, but it is now accelerating a lot more than before. A similar result can be obtained using a ramp $8m$ long, while the ramp 4 and 15 meters long

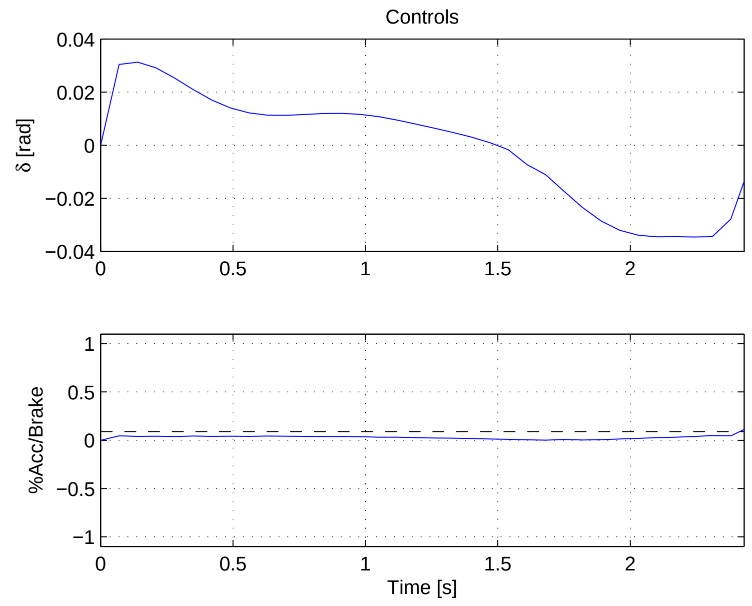

Figure 4.15: Step - Controls

fail to give a result that avoids the obstacle. This shows how the nonlinear optimizator is a very non stable operator.

The next shape we have tried to apply is the circumference. As we can see from figure 4.18, the obstacle is avoided while an intense usage of braking is applied. This is what we want to see from a safe controller. The speed has been reduced from $110km/h$ to $50km/h$ and the manoeuvre is taken with smaller effort compared to the other results.

The circumference uses a lot of space since the obstacle must be included in its shape. To avoid this kind of problem we tried to apply an ellipse. Unfortunately, as we can see from figure 4.20, even if we expect a similar behaviour to the circumference, the accelerator and the brake are used very differently, almost randomly. Again the controller is able to avoid the obstacle, but we can see how it is difficult to understand and setup the optimizator. Of course this trial will not be valid anyway, because the smaller ellipse that is capable of circumscribing the obstacle is higher than the road limit of $4.5m$.

From all these figures we understand the great sensitivity that the controller has with respect to the obstacle shape. The controller is pretty much always able to avoid the obstacle, but its way of doing that, especially for what regards the usage of the brake and throttle pedal, changes a lot. This sensitivity is often non logical as the ramp shaped obstacle trials have shown, and those defects are due to numerical problems that are again the biggest issue for this

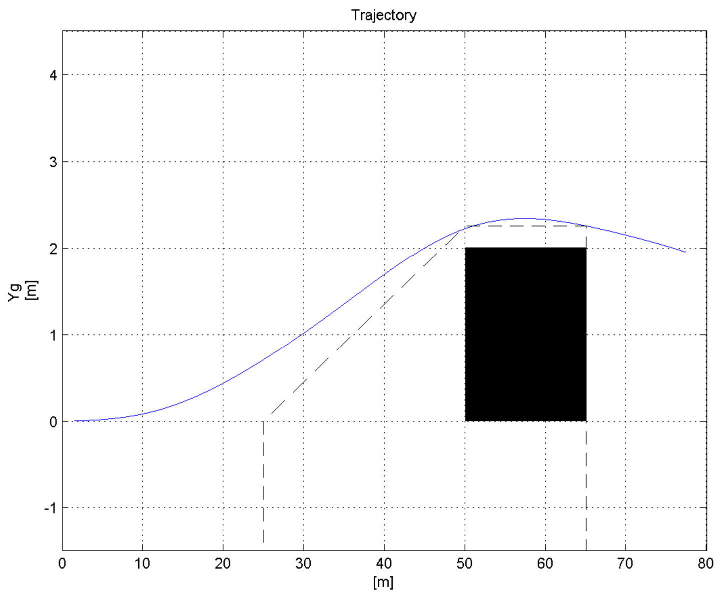

Figure 4.16: 25m ramp - Trajectory

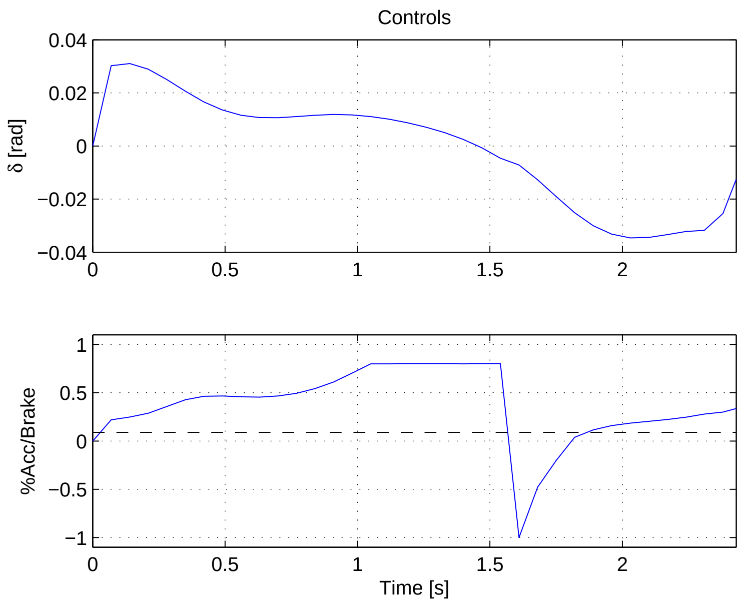

Figure 4.17: 25m ramp - Controls

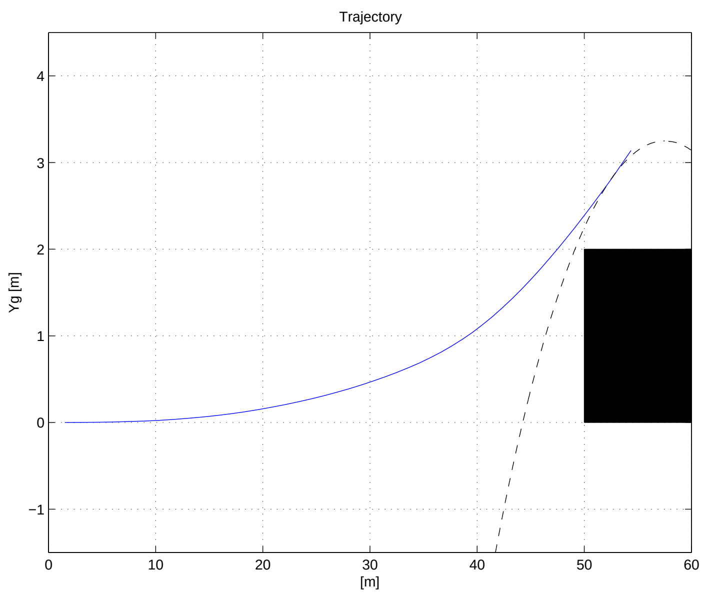

Figure 4.18: Circumference - Trajectory

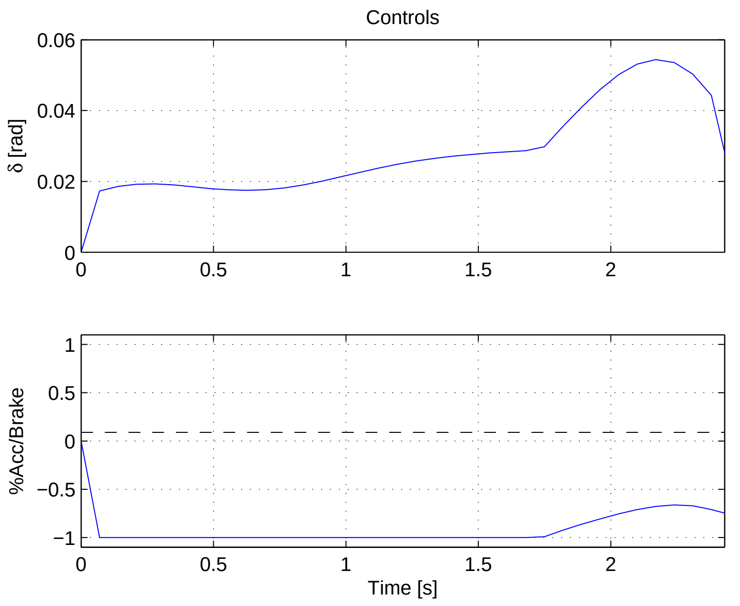

Figure 4.19: Circumference - Controls

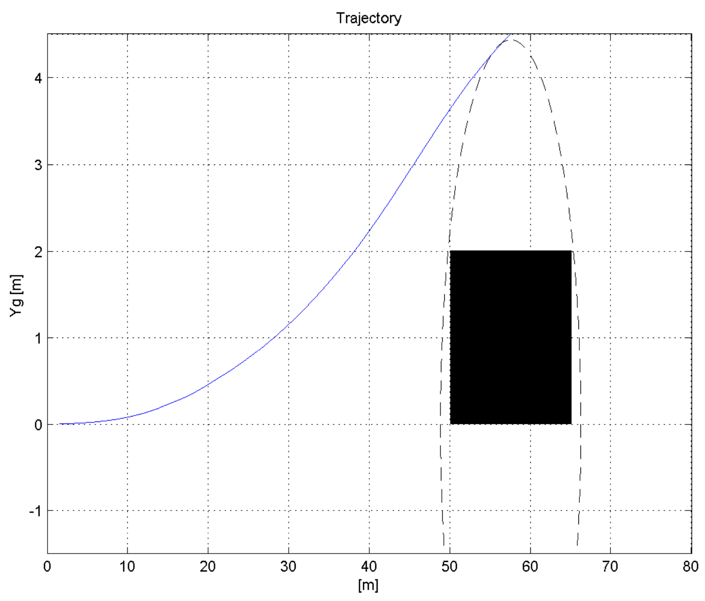

Figure 4.20: Ellipse - Trajectory

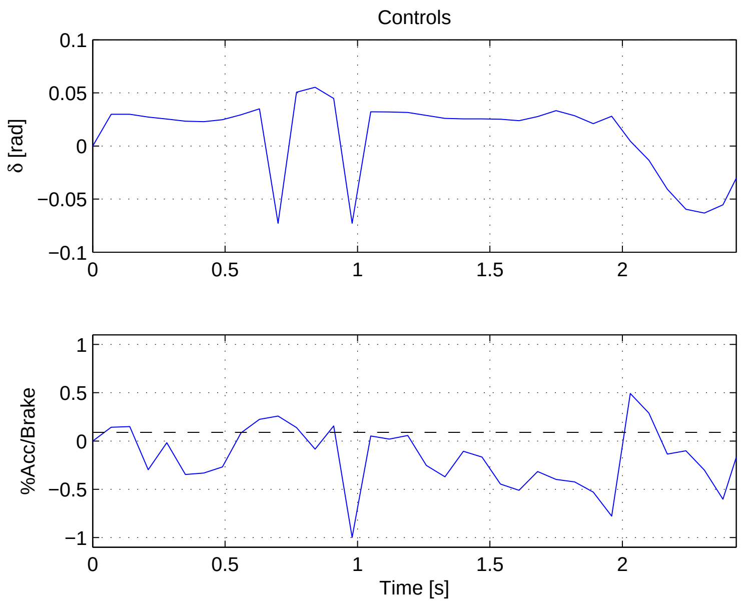

Figure 4.21: Ellipse - Controls

control. Anyhow from all those trials we can conclude that the best one to use is the circumference shaped obstacle because it allows the controller to slow down the vehicle a lot, allowing a safe manoeuvre, but that shape uses a lot of space that will fail to give good results in the case that the obstacle blocks completely the road.

4.5 Optimization algorithm sensitivity

Since now we have never talked about the algorithm used in the optimizator function to solve the minimization problem. Frequently in the other papers dealing with this control we find references on commercial software written specifically for nonlinear optimization. Unfortunately we were not able to use those as they were not available for us. We have also to consider that those are written for other programming languages. For what regards the linear optimization algorithm there were no problems. The function was always stable and able to provide efficient results since it was built to work with linear problems that are a lot simpler compared to the nonlinear ones. On the contrary the nonlinear case can be solved with many different algorithms. The first one we have used, that is the one we used to provide the results above is, as suggested by Matlab, the Interior-Point algorithm. Being very slow at completing calculations, we decided to try also the Sequential Quadratic Programming (SQP), that was declared to be faster. We have noticed a slight improvement in speed, but still we are very far from a real time implementation. Apart from this difference there are bigger diversities if we consider that the controller was challenged with exactly the same obstacle shapes as before. For example the standard obstacle with an SQP algorithm is overcame with a very noisy accelerator pedal signal in figure 4.23, while the Interior Point algorithm was able to obtain the same result with constant zero accelerator. Other odd things must be reported for the ramp shaped obstacle. We have said that the Interior Point algorithm was inexplicably not able to produce a valid result with the ramp 4 and 15 meters long, even if it was capable of doing it with the ramp 8 and 25 meters long. Now, with the SQP algorithm, we can retrieve results for all those trials, but the behaviour for the 8 and 25 meters long ramp are very different. In figure 4.25 we can see that the new algorithm uses a lot less accelerator than before in the $25m$ ramp, and also from figure 4.27 with the $8m$ ramp we can say the same consideration.

The circumference is identical with respect to the case with the Interior Point algorithm, as we can see from figure 4.29, while in the ellipse a major fault has to be reported. The control wants to overtake the obstacle from below in figure 4.30. To solve this problem we tried to add an hard constraint to describe the limit of the road on the right, but unfortunately this brought the control not to be able to return a feasible result, even if the ellipse is perfectly symmetrical with respect to zero.

At the end of this analysis we can conclude that there are behaviours of the optimizator function that are difficult both to understand and to manage. Massive differences for what regards the control behaviour, especially with the throttle and brake pedal, occur when comparing different obstacle shapes

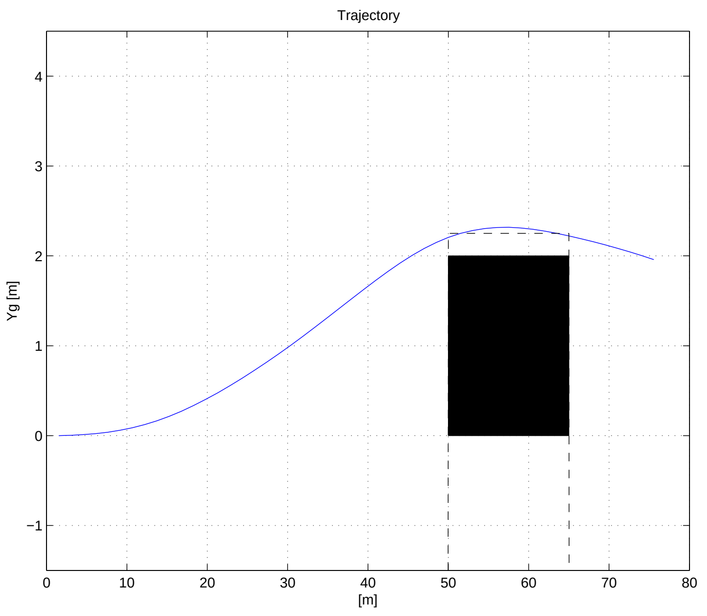

Figure 4.22: Step SQP - Trajectory

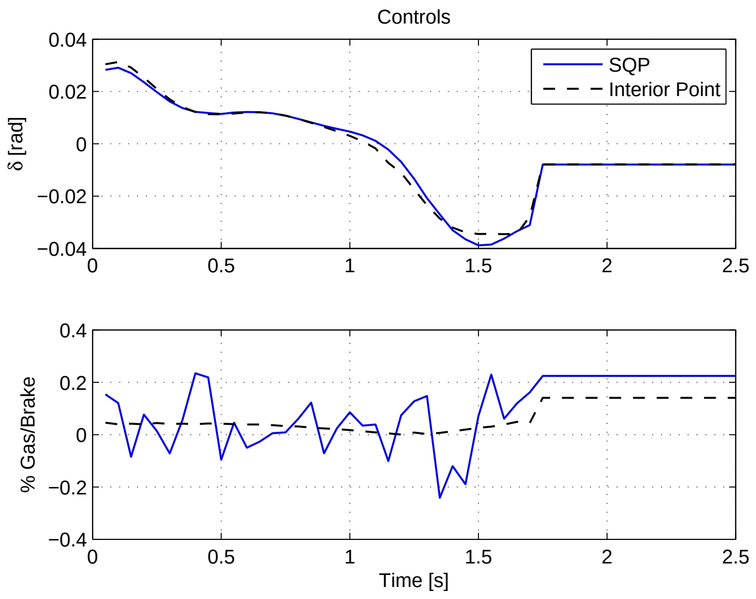

Figure 4.23: Step SQP - Controls

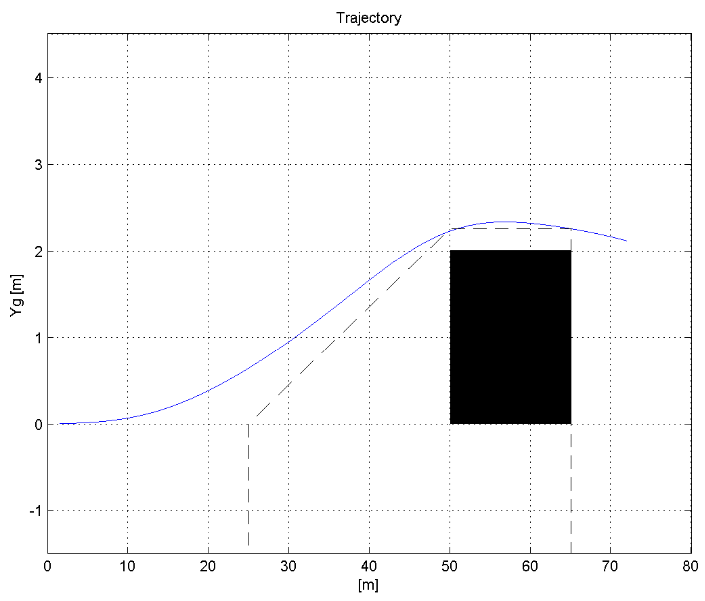

Figure 4.24: 25m ramp SQP - Trajectory

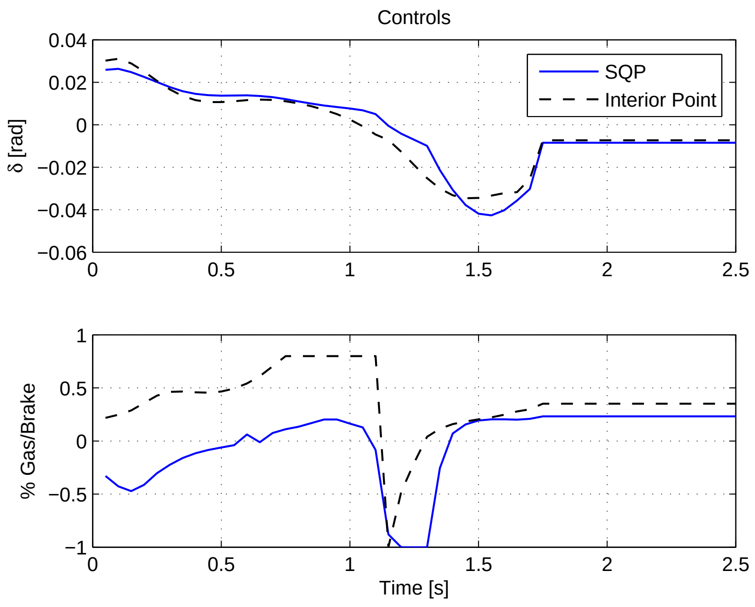

Figure 4.25: 25m ramp SQP - Controls

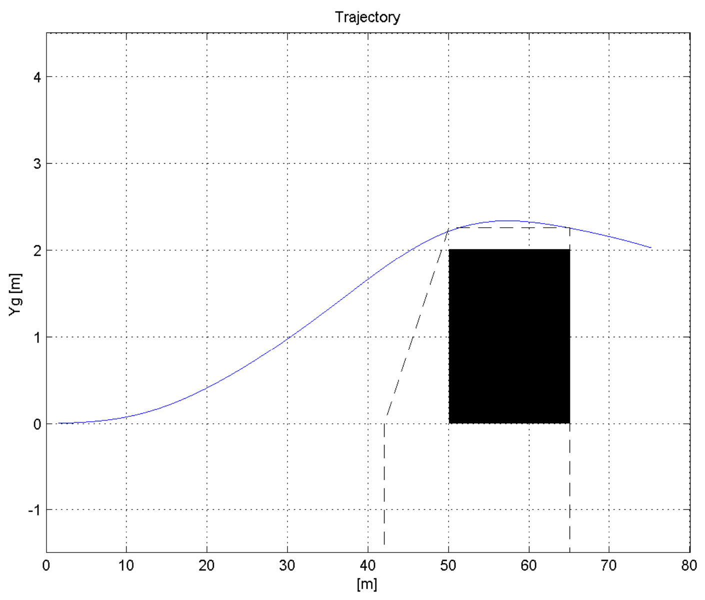

Figure 4.26: $8m$ ramp SQP - Trajectory

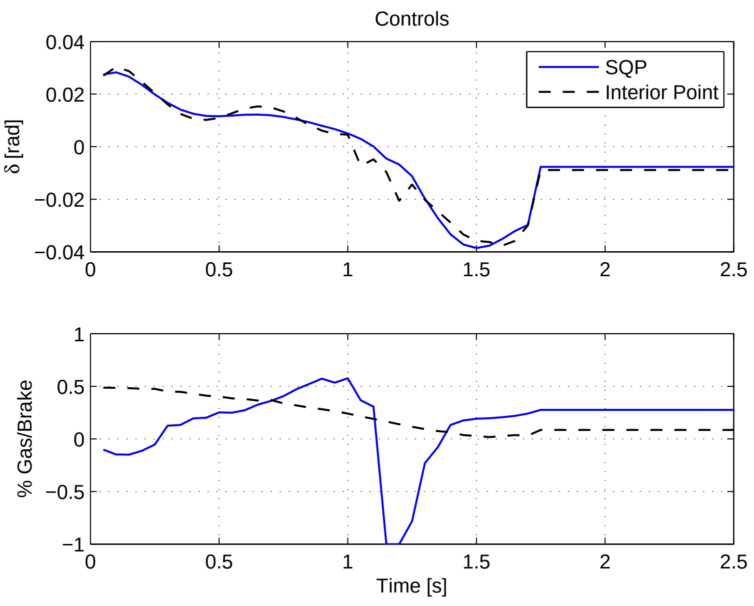

Figure 4.27: $8m$ ramp SQP - Controls

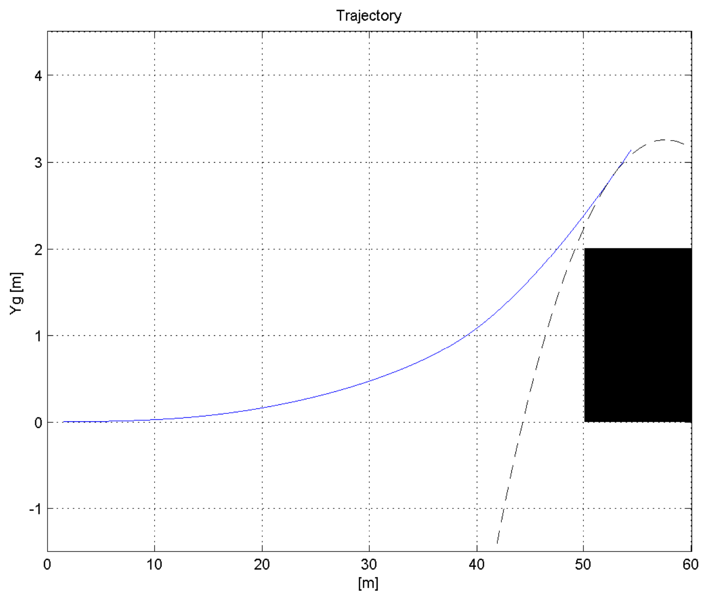

Figure 4.28: Circumference SQP - Trajectory

but also while using different algorithms. The most performing combination of the obstacle shape and algorithm is the circumference used with the SQP algorithm. The circumference is in fact the only shape that allowed to slow the vehicle down to safely overtake the obstacle, while the SQP algorithm is definitively the most stable one. Since, as we have explained earlier, all those trials are made only considering the first optimization step, we would like to show in figure 4.31 and 4.33 the results of a complete simulation using the circumference with the SQP algorithm. As we can see in the first part of the simulation the controller wants to brake, while in the second part we see how the controller wants to accelerate to get back to its reference in speed since it has slowed down the vehicle a lot, as we can see from figure 4.32. This is the behaviour we always expected to see. The slow speed of the vehicle lets the control to be able to use less lateral force from the tyre. From figure 4.34 we can see how the maximum value obtained is $0.03rad$, while without the circumference with the nonlinear control a maximum value of 0.05 was obtained, similar also to the maximum value resulted from the linear control.

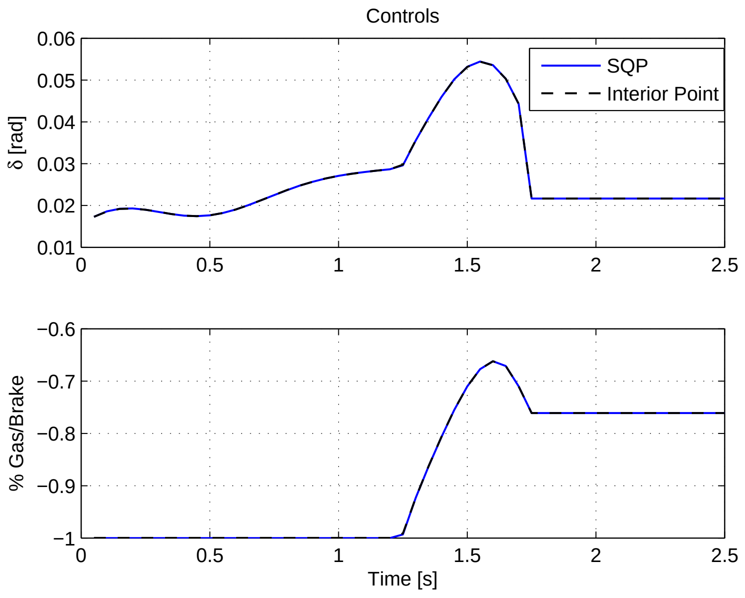

Figure 4.29: Circumference SQP - Controls

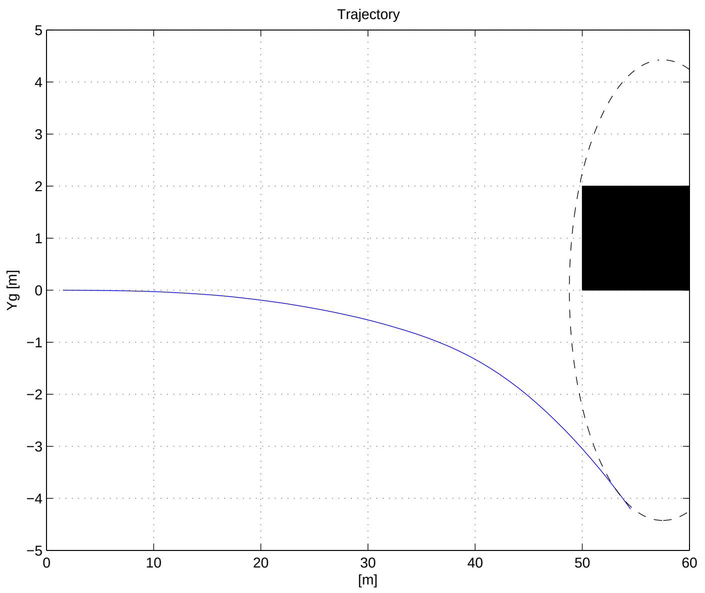

Figure 4.30: Ellipse SQP - Trajectory

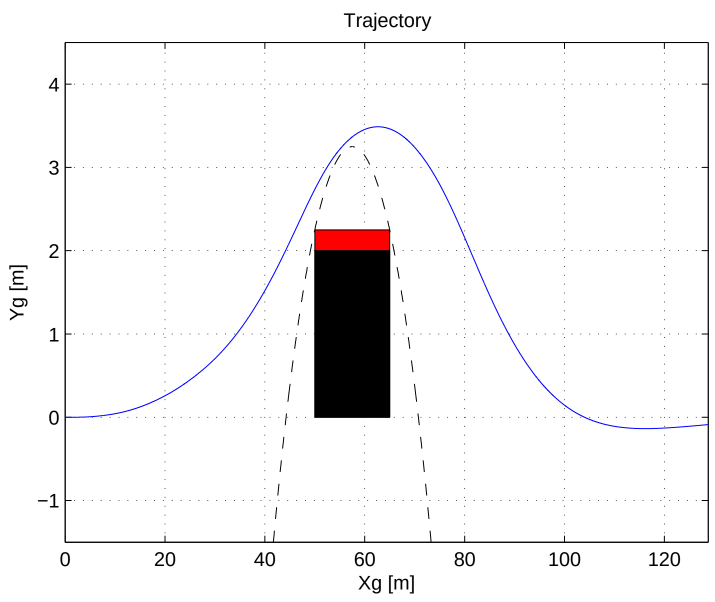

Figure 4.31: Complete Circumference SQP - Trajectory

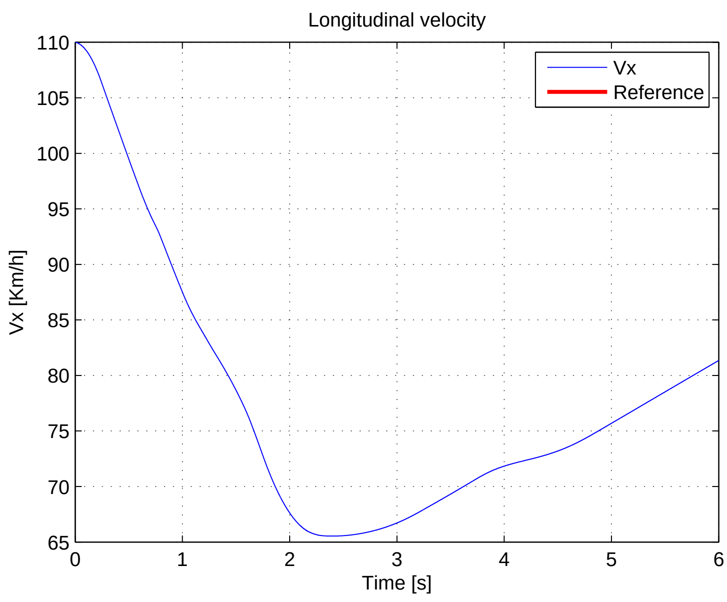

Figure 4.32: Complete Circumference SQP - Speed

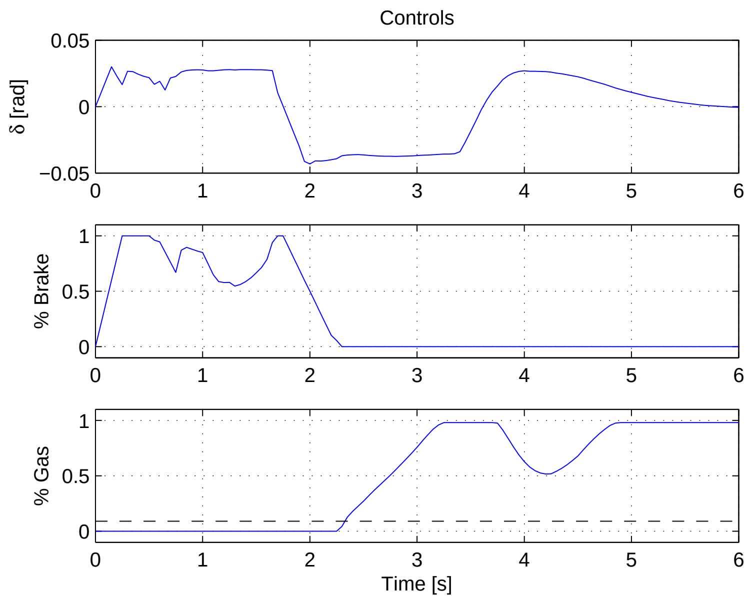

Figure 4.33: Complete Circumference SQP - Controls

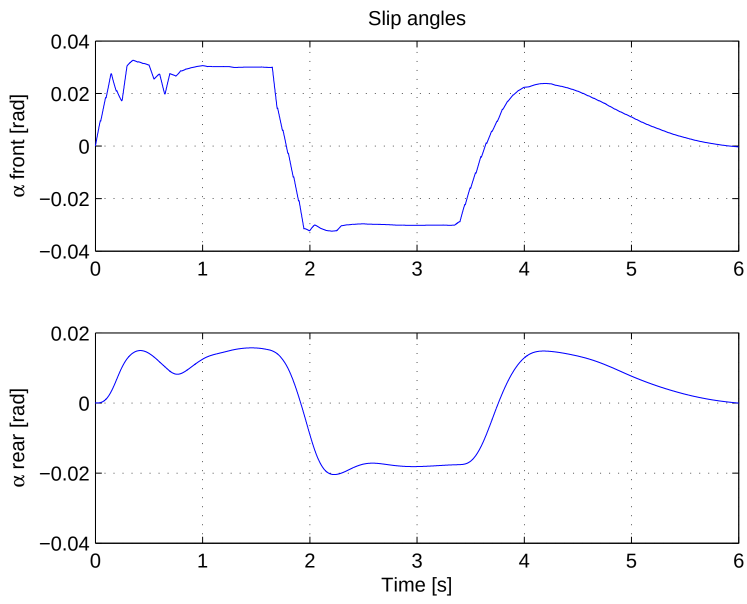

Figure 4.34: Complete Circumference SQP - Slip angles

4.6 Emergency brake test and initial conditions sensitivity

Another important trial that we wanted to test is the emergency brake manoeuvre, as it might happens to find the road completely closed. In those conditions we expect the controller to understand that it is not possible to overtake the obstacle, and therefore it must stop the vehicle before crashing into it. This test was not feasible for the linear controller because it is not capable of managing obstacles in the space domain as we explained in its chapter, and therefore this type of trials were experienced only with the nonlinear control. We have discovered a problematic sensitivity to the initial conditions that we set for each step to the optimizator. The first initial condition that we wanted to try are set, for the whole time window, as the steering in zero and the throttle pedal in zero. The results are shown in figure 4.36, and we can see how the controller is not trying to stop the vehicle as it is trying to accelerate while uselessly steering to avoid the obstacle in figure 4.35.

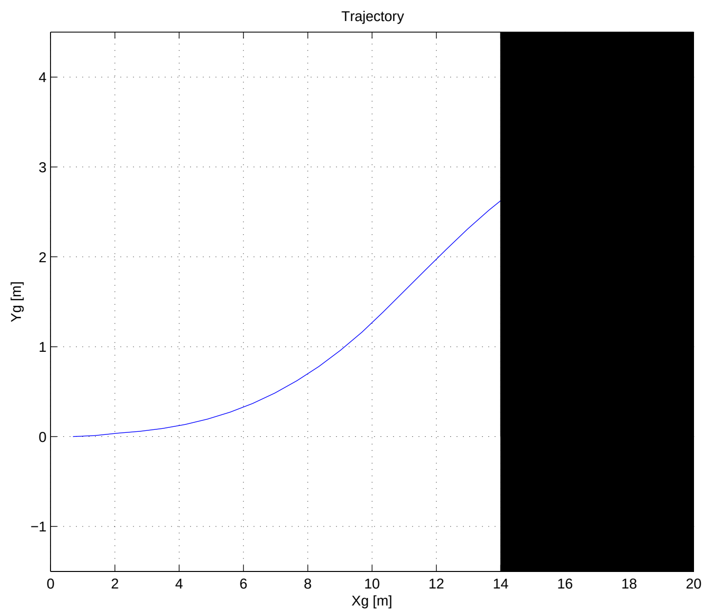

Figure 4.35: Emergency brake Thr= 0 - Trajectory

This bad result pushed us to try to impose different initial conditions. The next test is with the initial conditions set with the brake at 50% for the whole time window. The results in figure 4.38 and 4.39 are again not

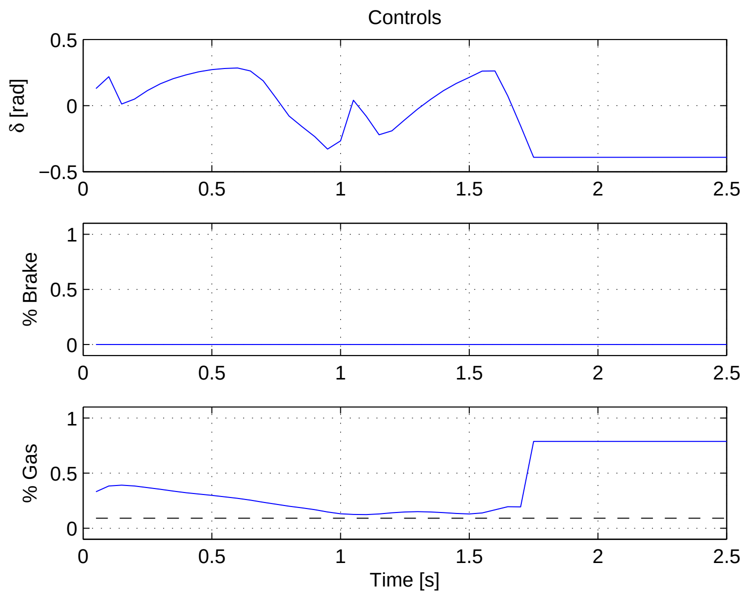

Figure 4.36: Emergency brake Thr= 0 - Controls

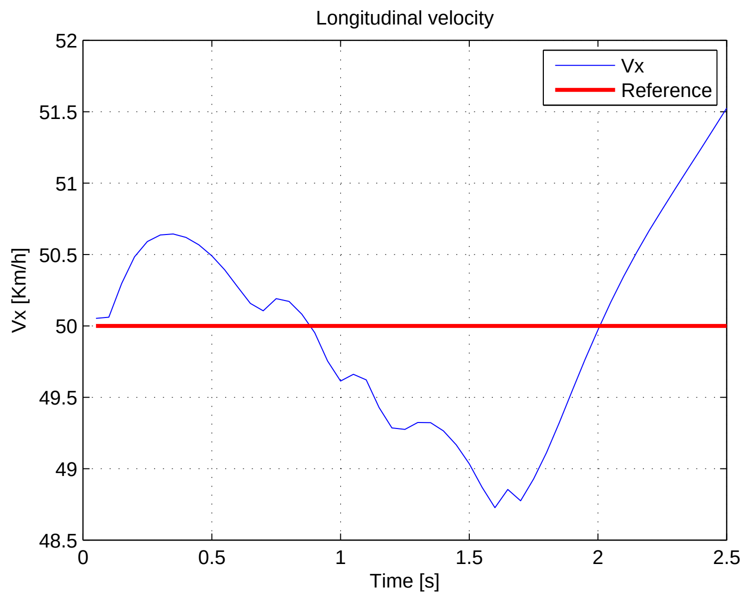

Figure 4.37: Emergency brake Thr= 0 - Speed

satisfactory. The controller starts to understand that the brake is required but it does not use it properly. The car stops within 14 meters if the brake is applied completely, here the brake is not enough to reduce the speed before crashing into the obstacle as we can see from figure 4.39.

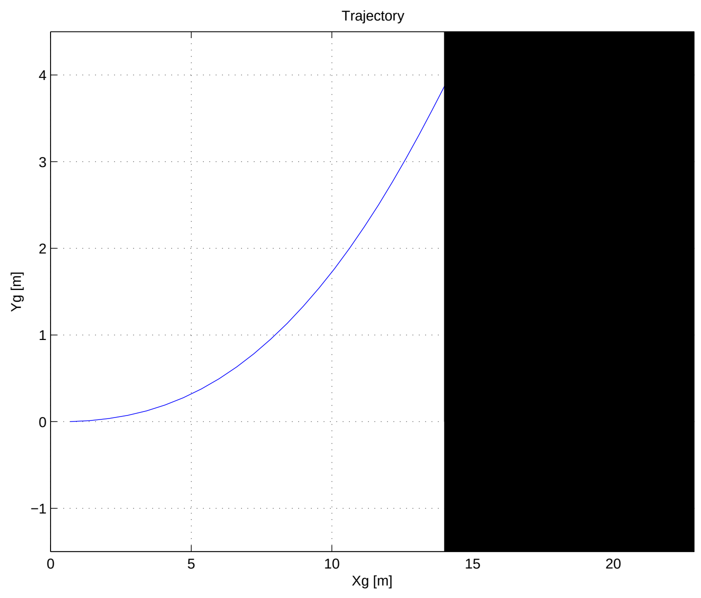

Figure 4.38: Emergency brake Brk= 0.5 - Trajectory

Before trying to impose full braking over the whole time window, we would like to show another trial that prove how the optimization function is difficult to understand and setup. The trial is with initial condition set at full throttle for the whole time window. This condition is obviously not helpful if we consider that we want the car to be stopped, but if we compare figure 4.42 with the first trial that we did with zero throttle in figure 4.36 we can see that at least in this test the controller is not accelerating. This behaviour of the optimizator is obscure.

The last test is the one with full braking set as initial condition. The results in this case are as expected. In figure 4.44 we can see that the vehicle is kept in a straight line and how it is able to stop before the obstacle, while in figure 4.45 we can see how the brake is applied fully almost for the whole simulation.

It is important to understand that for this type of trial we would not need to change the setup of the controller, as it must understand the solution on his own. This is the reason why the speed reference is always set to $50km/h$ as that is the speed that normally it has to follow. Otherwise it means that

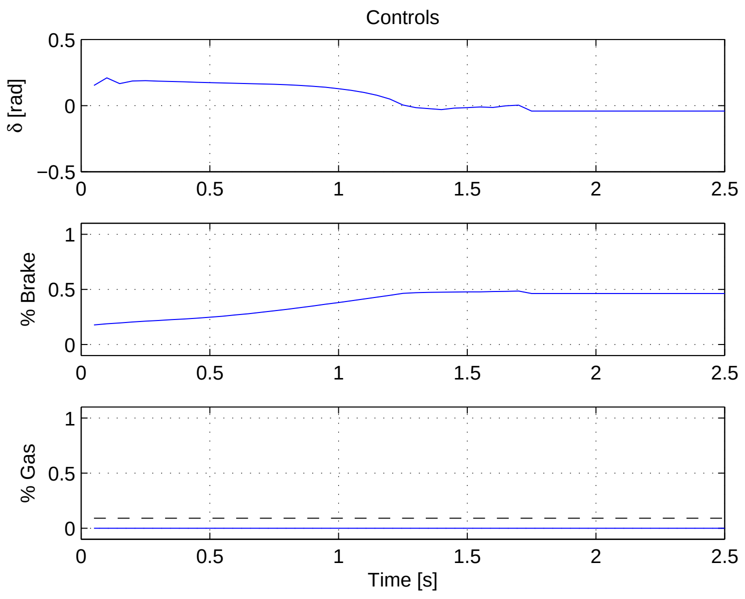

Figure 4.39: Emergency brake Brk= 0.5 - Controls

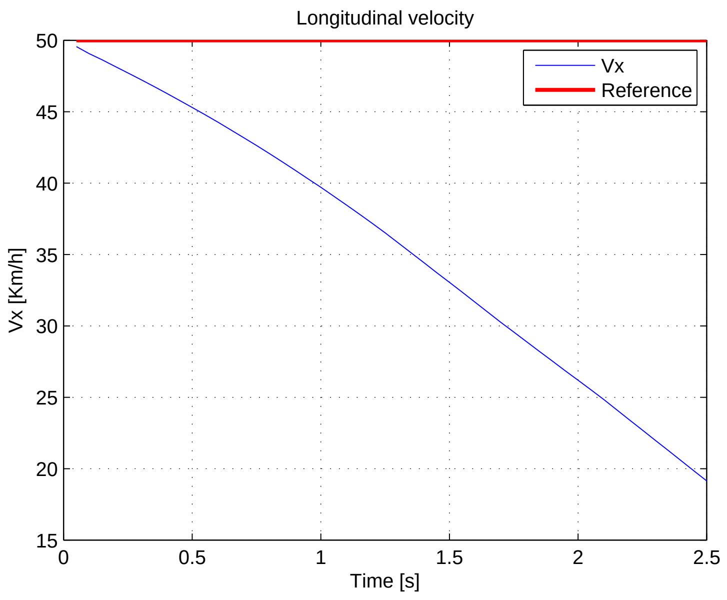

Figure 4.40: Emergency brake Brk= 0.5 - Speed

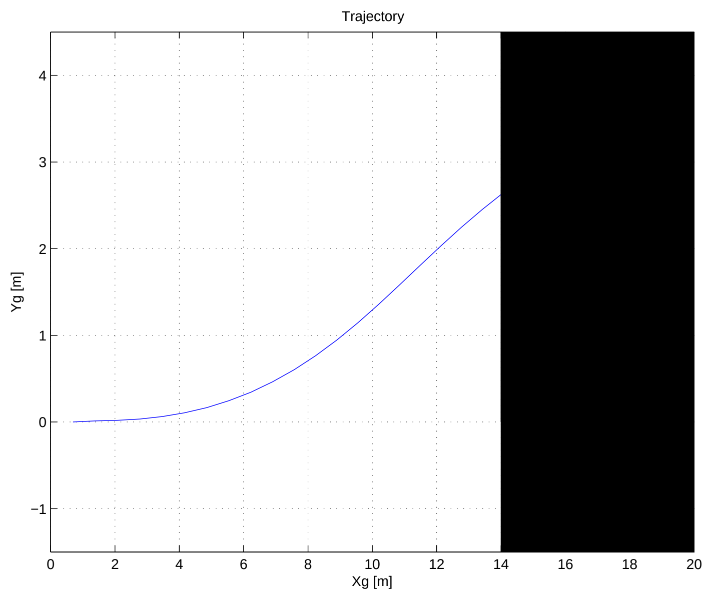

Figure 4.41: Emergency brake Thr= 1 - Trajectory

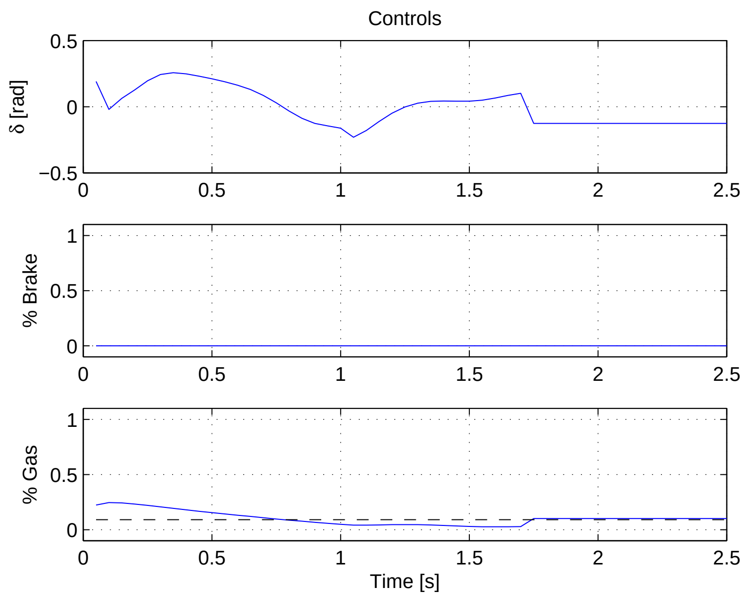

Figure 4.42: Emergency brake Thr= 1 - Controls

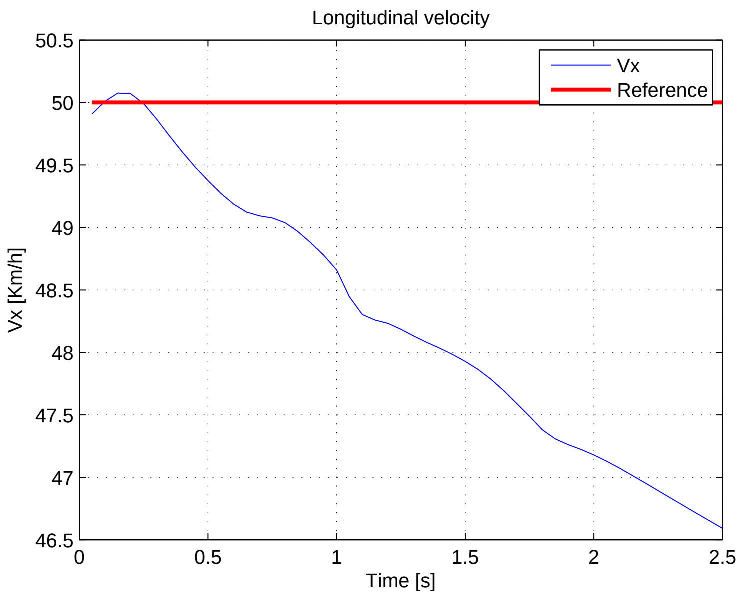

Figure 4.43: Emergency brake Thr= 1 - Speed

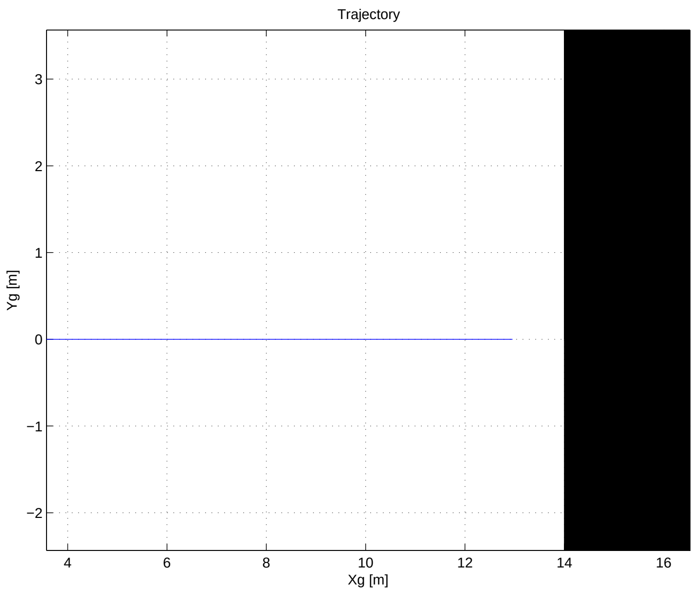

Figure 4.44: Emergency brake Brk= 1 - Trajectory

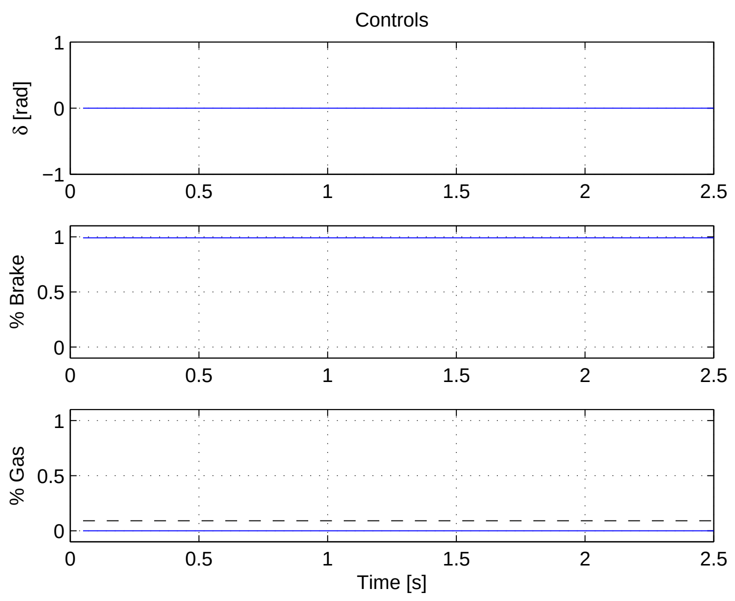

Figure 4.45: Emergency brake Brk= 1 - Controls

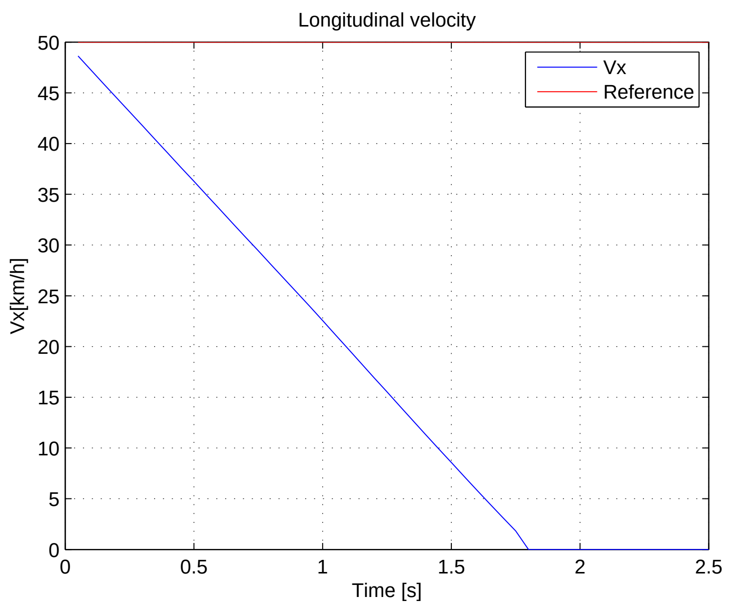

Figure 4.46: Emergency brake Brk= 1 - Speed

we would require an higher level logic unit capable of understanding if the road is blocked, and therefore capable of applying the procedure to stop the vehicle as soon as possible. But with this type of logic the model predictive control is not required, since in those conditions we could simply impose the speed reference to zero, and therefore we could simply use a classic PID controller. Unfortunately for us, as we have seen with those trials, only with the initial conditions set to full braking the controller is able to understand that stopping the vehicle is a feasible solution, but this initial solution is not the one used for every step of a normal driving situation. Normally for each step we impose as initial conditions the solution that the optimizator has obtained in the previous optimization. In this way we guarantee a smooth operation and we increase the speed of calculations. For these reasons the initial conditions are never set to full braking and therefore the controller will never be able to stop the vehicle without the usage of an higher level logic unit.

4.7 Moving obstacle

The nonlinear model predictive control can also consider a moving obstacle in its predictions allowing the vehicle to avoid the obstruction without any issue. This is done directly into the minimization function and it is obvious how this is an advantage on the basic PID path follower that requires necessarily a trajectory to be defined. Of course the trajectory of the obstacle must be known, and therefore somehow we must obtain the future behaviour of the moving obstacle. Even if this can sound tricky, actually the great varieties of obstacle that we find daily on the roads have standard behaviours. For example people or animal crossing the road tend to have a constant velocity across the road, while vehicles tend to remain parallel to it. In case there is a change in velocity of the obstacle, this can be considered thank to the receding strategy of the controller, therefore we are required to know correctly only the first few second of the future behaviour of the obstacle.

The first challenge is an overtaking manoeuvre. Our vehicle is travelling at $110km/h$ and it has to overtake a lorry, which is the standard obstacle that we have used for all our work, driving at $50km/h$. It is difficult to show how the trial was a success but in figure 4.47 we can see how our vehicle is able to be above the lorry encumbrance before getting into it. When the two x coordinates are equal, the vehicle y coordinate is in fact higher than two meters. To clarify we can also watch the series of frames that we took from this trial in figure 4.49, 4.50, 4.51, 4.52, 4.53, 4.54.

To also consider the control ability to avoid moving obstacle going across the road we made another trial. The vehicle is still starting from $110km/h$ while the obstacle crosses the road with a velocity along the x axes of $54km/h$ and along the y axes of $35km/h$ and its starting point is placed so that a collision is imminent. While in the previous example the obstacle shape could be whichever we wanted to, and we have chosen the step since it was the simplest one to apply, here the obstacle must be defined as a general elliptical closed curve, otherwise we cannot describe its shape and movement into the minimization function. A logic choice could be the circumference, but since the object is moving from one side to another of the road, the circumference is taking too much space and therefore an ellipse is used instead. As we can see from the frames in figure 4.55, 4.56, 4.57, 4.58, 4.59, the controller is able to consider the obstacle and to slow down the vehicle to let the obstacle pass as we can see from figure 4.60, without using the steering.

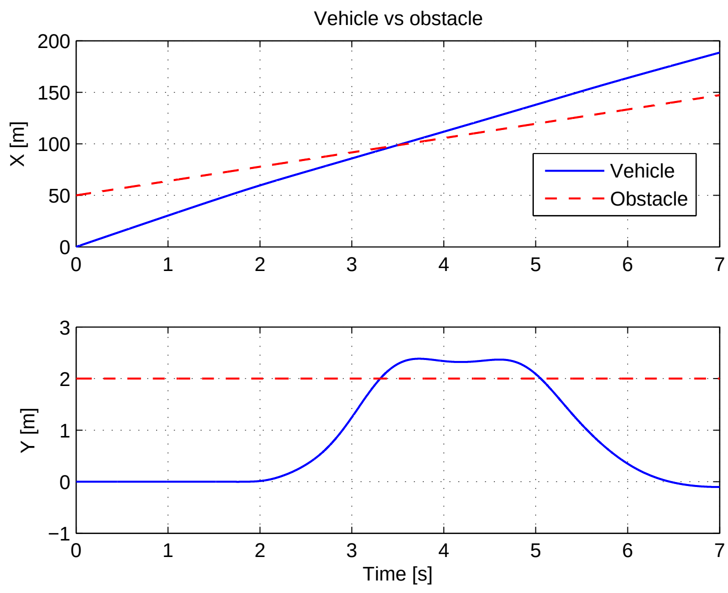

Figure 4.47: Moving obstacle overtaking - Displacements

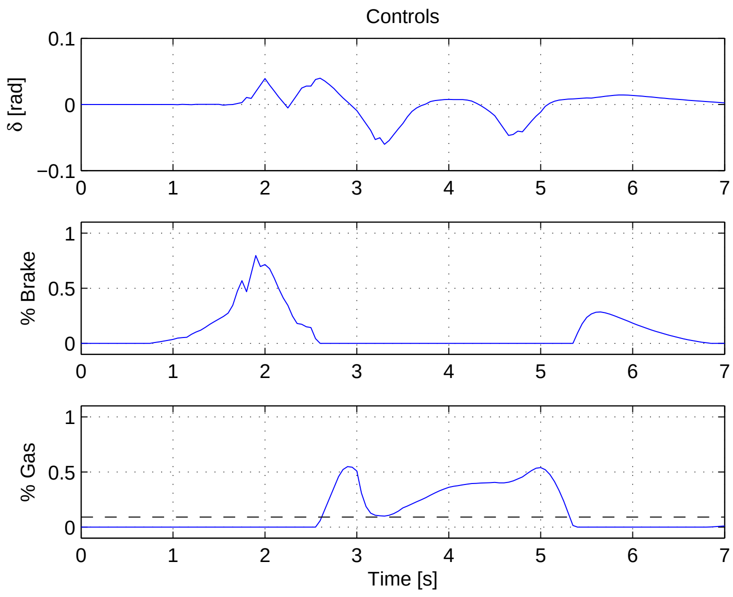

Figure 4.48: Moving obstacle overtaking - Controls

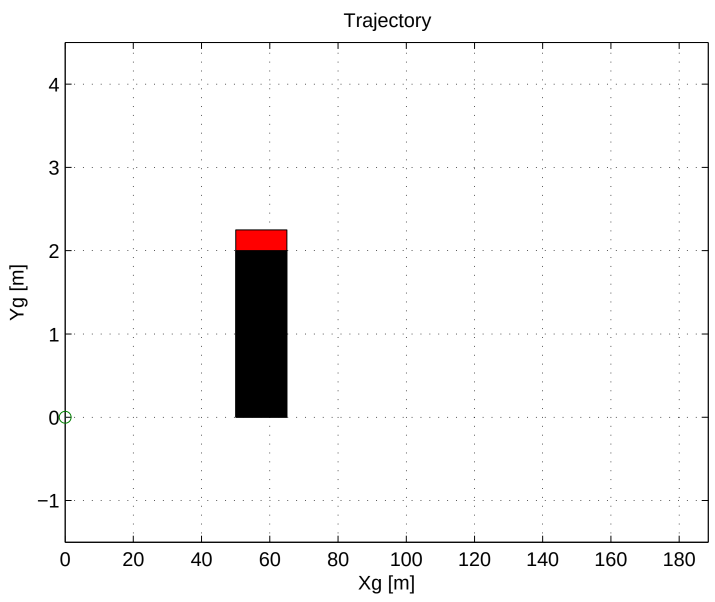

Figure 4.49: Moving obstacle overtaking - Trajectory Frame 1 (0s)

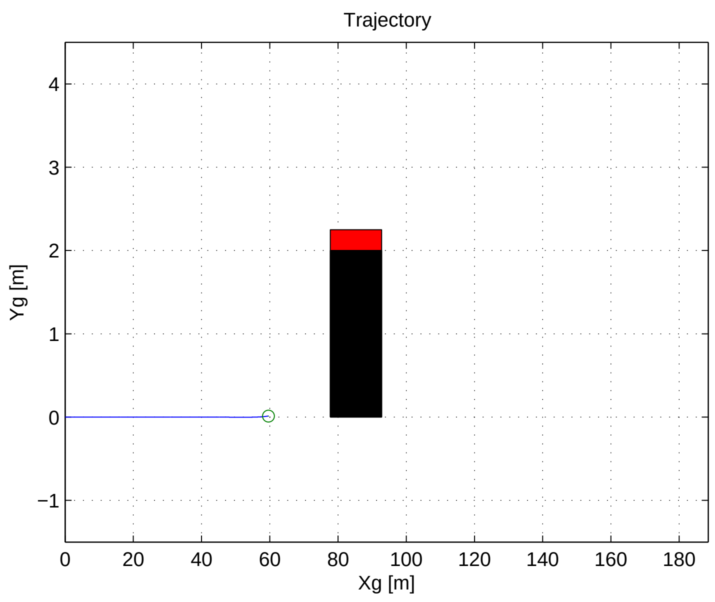

Figure 4.50: Moving obstacle overtaking - Trajectory Frame 2 (2s)

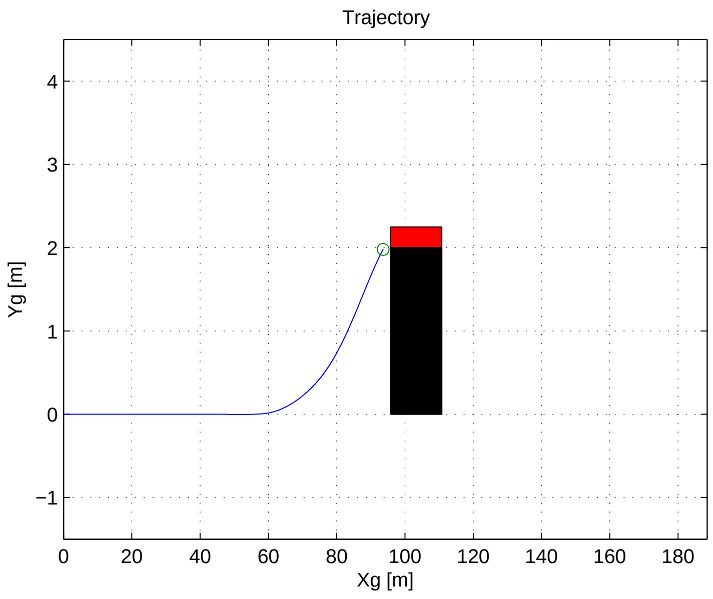

Figure 4.51: Moving obstacle overtaking - Trajectory Frame 3 (3.3s)

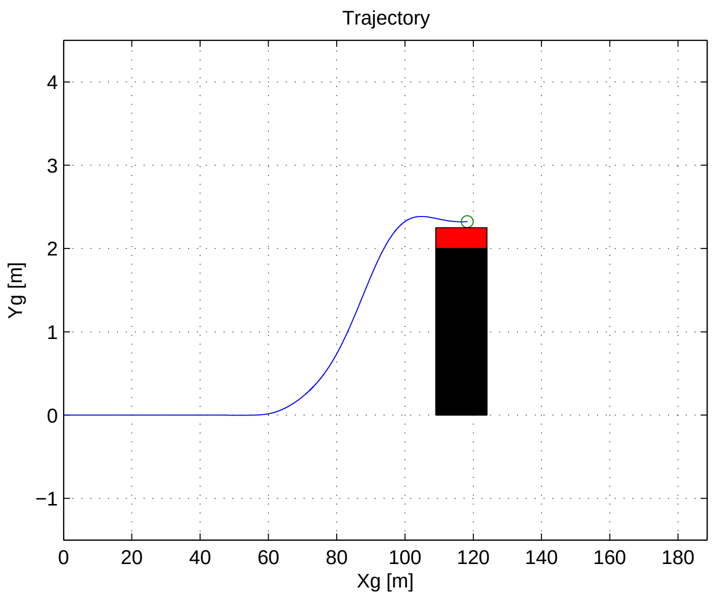

Figure 4.52: Moving obstacle overtaking - Trajectory Frame 4 (4.25s)

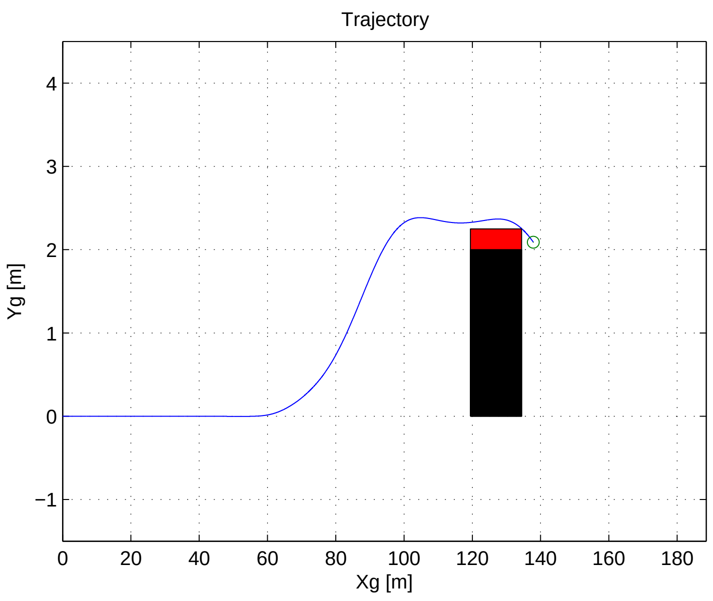

Figure 4.53: Moving obstacle overtaking - Trajectory Frame 5 (5s)

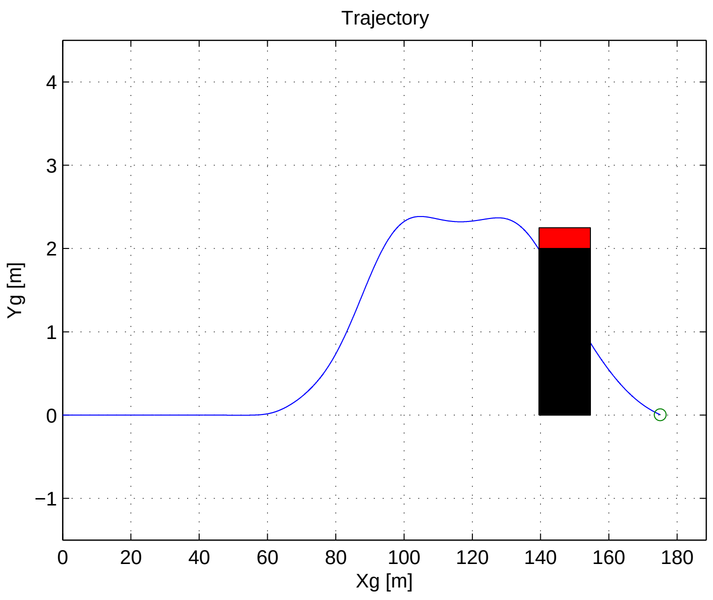

Figure 4.54: Moving obstacle overtaking - Trajectory Frame 6 (6.45s)

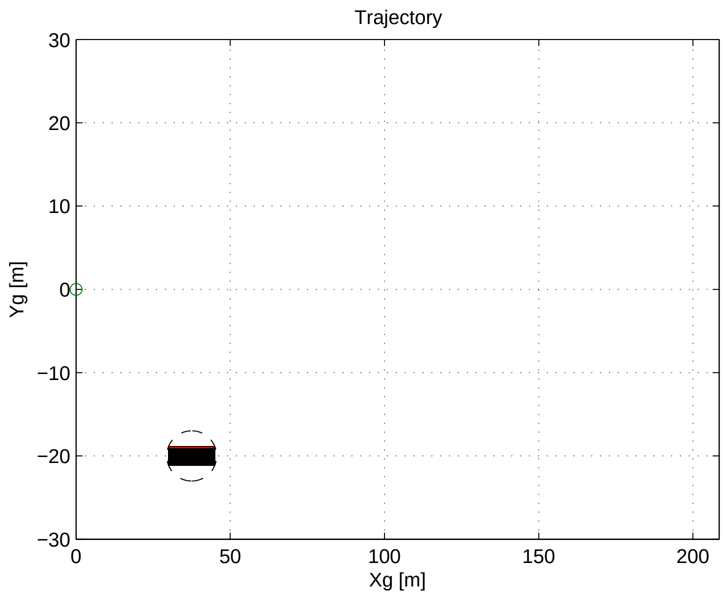

Figure 4.55: Moving obstacle crossing - Trajectory Frame 1 (0s)

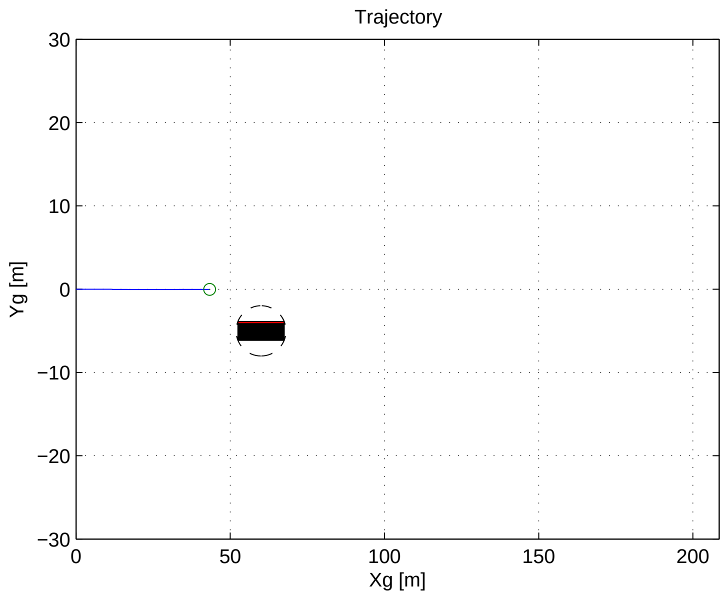

Figure 4.56: Moving obstacle crossing - Trajectory Frame 2 (1.5s)

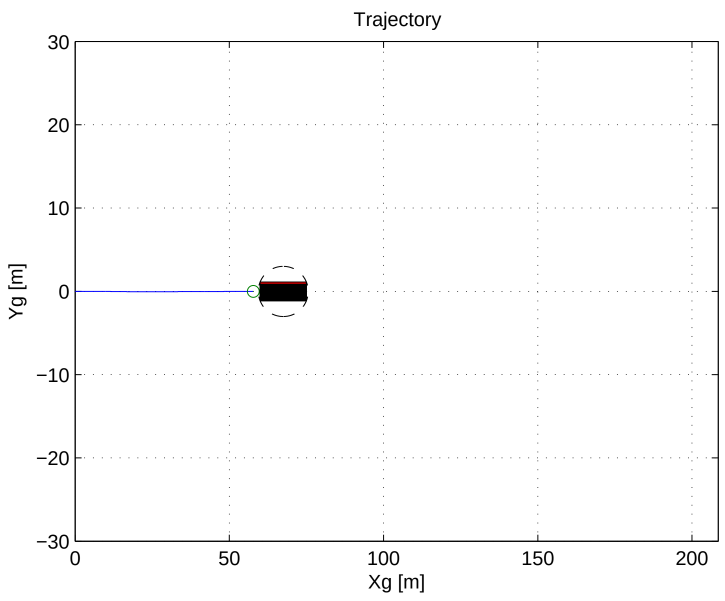

Figure 4.57: Moving obstacle crossing - Trajectory Frame 3 (2s)

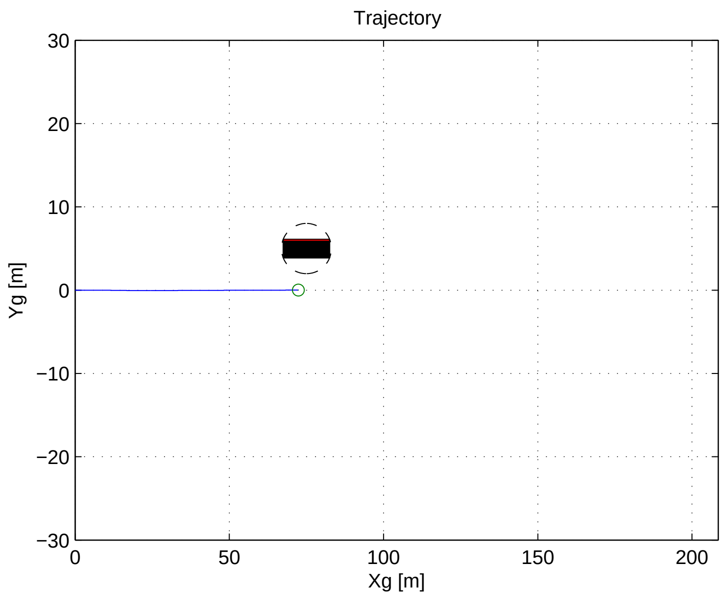

Figure 4.58: Moving obstacle crossing - Trajectory Frame 4 (2.5s)

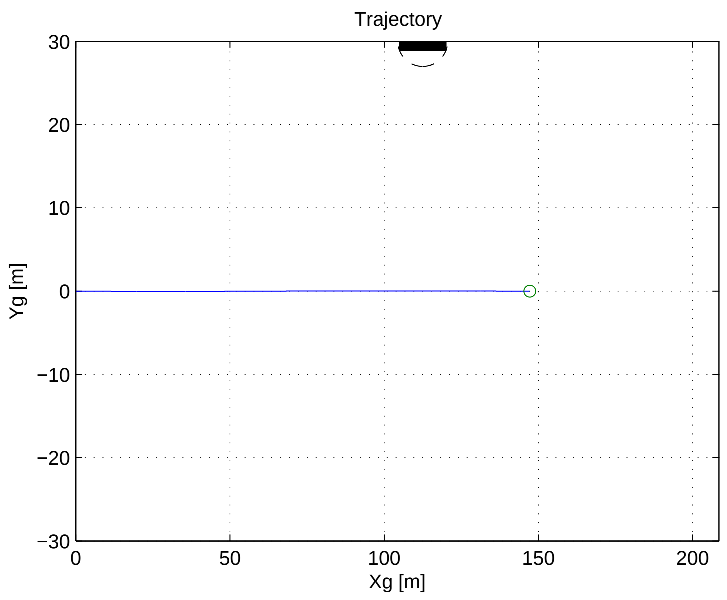

Figure 4.59: Moving obstacle crossing - Trajectory Frame 5 (5s)

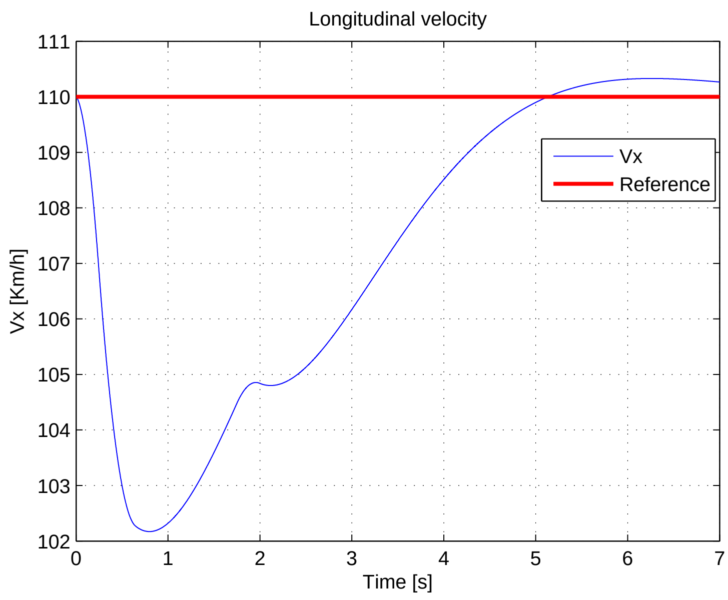

Figure 4.60: Moving obstacle crossing - Speed

Chapter 5

Conclusions

We managed to evaluate the behaviour of a linear Model Predictive Control applied to the steering only, and applied to the steering, to the throttle and to the brake pedals. Moreover, we developed a nonlinear Model Predictive Controller applied to the steering, to the throttle and to the brake pedals. We tried all these controls as path followers and for the purpose of avoiding obstacles.

The linear controller is easier to setup but it is difficult to implement when it has to cope with nonlinear systems. It has shown good performances, comparable or even better than common PID controllers, and good sensitivity to the various parameters of the controller, i.e. any set of parameters is typically able to cope with much more situations with respect to PIDs. In fact, as can be seen in the appendix, the MPC controller was able to control both a collocated and a non collocated system, while the PID controller required a special setup for these two cases. This is because the setup of the MPC controller concerns the optimizator function rather than the controller itself. For these reasons, even if it requires a little bit of programming effort, we can say that the linear MPC is easier to manage with respect to common controllers. We were pleased by the performance of the linear MPC performing on linear systems.

The active front steering management was a positive result. We were not expecting such a good behaviour with a system linearized every $50ms$ controlled by a completely linear logic. We have shown as the controller is perfectly capable of following any reference trajectory with the vehicle even if we have neglected the load transfer and the relaxation lengths in the model of the controller. It is also capable of avoiding a fixed obstacle if this is seen by the measurement system within a distance that is feasible for the usage on normal roads and in normal traffic conditions. To these performances we have to add the fact that it is has shown high computational speed. Even if we have used Matlab, the controller was able to calculate within $25ms$ for almost every control step and even if the optimizator fails to obtain the value of the future control variable, a valid escape route was identified. However, also a number of problems showed up: at speeds higher than $150km/h$ the controller showed signs of instability caused by the wrong predictions in the lateral dynamics of the vehicle, while predictions of the slip angles were never good enough to be used in hard constraints. These results are satisfactory because they show that it is possible to use a MPC based AFS in order to avoid obstacles. This is an evolution compared to the work that was done in [2],[3] and [4] where an AFS was used to simply follow a reference path on slippery surfaces. This development was also suggested in the conclusions of [5].

In the following step the linear controller was applied over the accelerator and brake pedal, as it was never done before. The vehicle driven by the LTV-MPC was perfectly capable of following references both in the lateral position and in the longitudinal speed of the vehicle and was also able of avoiding obstacles. Nevertheless, it has never shown capabilities of using the brake correctly. The control of the longitudinal dynamics of the vehicle has never been very performing while avoiding obstacles and we have shown that this is due to the limits of the linear controller. In fact the linearized matrices do not show any relation between the brake input and the lateral dynamics of the system. Therefore, the optimization function was never able to understand the purpose of its usage while in straight line. Only when cornering these matrices show relation between longitudinal and lateral dynamics, thus allowing the brake to be applied.

All these limits, among which the impossibility of imposing hard constraints on the front or rear slip angles, pushed us to try to apply the nonlinear controller. The advantages of the nonlinear controller are clear. We showed that the predictions and therefore the control actions are much smoother compared to the one of the linear case. The NMPC is capable of driving on normal roads with corners and we have shown how that is made also taking into account limits on the slip angles. We have also shown how the controller is capable of improving the performances of the linear case. In fact the latter requires 32 meters minimum to avoid the standard obstacle starting from $110km/h$, while the nonlinear controller just needs 26 meters to accomplish the same manoeuvre.

An extensive study of the obstacle shape and the optimizator algorithm was carried out to show that even minor differences in the shape can cause enormous differences in the behaviour of the controller, especially for what regards the usage of the vehicle controls. We managed to understand that the best combination of obstacle shape and optimizator algorithm is the circumference with the SQP algorithm, and we have shown how this combination provide the desired usage of the brake pedal.

The NMPC is the only controller capable of dealing with moving obstacles. We have challenged the controller both with an overtaking manoeuvre of a moving vehicle and with an object crossing the road at constant speed. Both the trials have been success and state the advantage of using this type of control compared to the linear ones.

It is important to remember that there is also an important problem of calculations time. Unfortunately the difficulties with this type of control are all left to the optimization function that find itself doing a large number of calculations and attempts. The calculation time is almost 9000 times the real time. Thus a hard job is expected to obtain a realtime implementation. This slowness is not due to scripts inefficiencies but directly to the fact that some functions are called millions of times, and this, of course, requires a lot of time.

The future of this controller depends largely on how will the speed of calculations improve. In our work we have demonstrated how only the nonlinear Model Predictive Control is capable of taking into account all the challenges that the road proposes, especially for what regards moving obstacles. Though it has the great limit of the calculation speed and therefore its main issue is certainly that. The first necessary future development will be to write the controller in another programming language, such as the much faster C.

The controller also requires some more studies on how to obtain more constant performances. The difficulties we have experienced due to the sensitivity to the obstacle shape can be caused by the fact that, to avoid the obstacle, the controller must move away far from the required reference. This causes the minimization function J to increase a lot and this could bring numerical problems. It is therefore possible to try to create another MPC unit that, knowing the position of the obstacles, generates a feasible reference both for the lateral position and the longitudinal speed. In this way the controller could guide the vehicle following said references and still considering obstacles and limits. This could give good results since our controller has proved great performances while following paths, while the other unit will just need to provide the trajectories.

This thesis only deals with the Motion Planner layer, but also the Mission and Behavioural layer must be developed in order to consider all the tasks an Autonomous vehicle must accomplish. On top of that also a unit that manages measures is required. All our work was developed supposing of knowing exactly the state of the vehicle model, but this is not possible if we are dealing with a real vehicle. The same can be said for the prediction of the future behaviour of the obstacles. In this work we have supposed to know exactly the future positions of the obstacles. Of course this is not true in the real environment and a specific unit must be developed to deal with this problem.

Another interesting future project for a Model Predictive guided vehicle could be its usage in the race car world. In fact it is possible to change the minimization function J to obtain the shortest time as possible. The constraints will not include obstacles anymore, but just the limits of the road that will describe a certain corner or even the complete circuit.

Appendix A

Linear MPC

This chapter was written to help understanding the basic operations that are involved using the linear MPC. Controlling a vehicle is a complex job since it involves dealing with a nonlinear system. Here we want to show how a linear MPC works with a completely linear system. This is also the first control that we have worked with since time and effort were required to both understand and program the logic behind model predictive controls.

A.1 System model

The system that we will use is the classic two degree of freedom bogie system, where the forces are directly applied to each bogie.

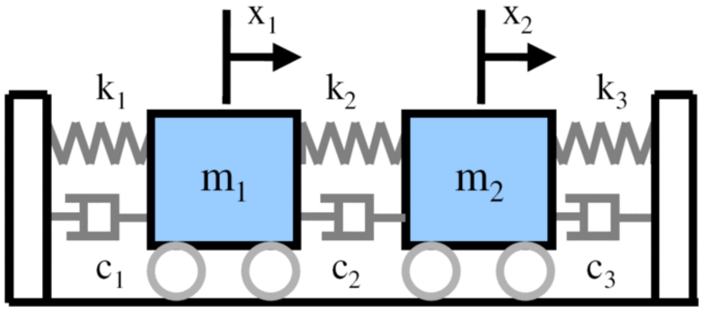

Figure A.1: Linear system model

Its equations are:

$$\begin{cases} m_1\ddot{x_1} + (c_1 + c_2)x_1 - c_2x_2 + (k_1 + k_2)x_1 - k_2x_2 = F_1 \\ m_2\ddot{x_2} + (c_2 + c_3)x_2 - c_2x_1 + (k_2 + k_3)x_2 - k_2x_1 = F_2 \end{cases} \tag{A.1}$$

From these we can obtain the state space form:

$$\underline{\dot{x}} = \begin{bmatrix} -\frac{c_1+c_2}{m_1} & +\frac{c_2}{m_1} & -\frac{k_1+k_2}{m_1} & +\frac{k_2}{m_1} \\ +\frac{c_2}{m_2} & -\frac{c_2+c_3}{m_2} & +\frac{k_2}{m_2} & -\frac{k_2+k_3}{m_2} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \underline{x} + \begin{bmatrix} \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \underline{F} \tag{A.2}$$

where $\underline{x} = (\dot{x_1}, \dot{x_2}, x_1, x_2)'$ is the space vector and $\underline{F} = (F_1, F_2)'$ is the control force vector. Knowing that $k_1 = k_2 = k_3 = 1N/m$, $c_1 = c_2 = c_3 = 0.01Ns/m$ and $m_1 = m_2 = 0.011kg$, we obtain:

$$\underline{\dot{x}} = \begin{bmatrix} -18.2 & 9.1 & -1818.2 & 909.1 \\ 9.1 & -18.2 & 909.1 & -1818.2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \underline{x} + \begin{bmatrix} 909.1 & 0 \\ 0 & 909.1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \underline{F} \tag{A.3}$$

For this system we have decided to use, for all the following tests, the same parameters to show how adaptable is this control compared to the classic PID control. The parameters are shown in table A.1.

n_y	n_c	ts $[s]$
20	10	0.01

Table A.1: Linear parameters

A.2 Collocated control

The first example is a displacement control for the first bogie while acting directly on the bogie, therefore with the force F_1. Starting from zero, we want to impose to the first bogie a displacement of three meters.

As we can see from figure A.2 the control action starts before the step, and therefore the control is capable of minimizing the error with respect to the reference thanks to his prediction capabilities. This is something the PID controller is not capable of doing.

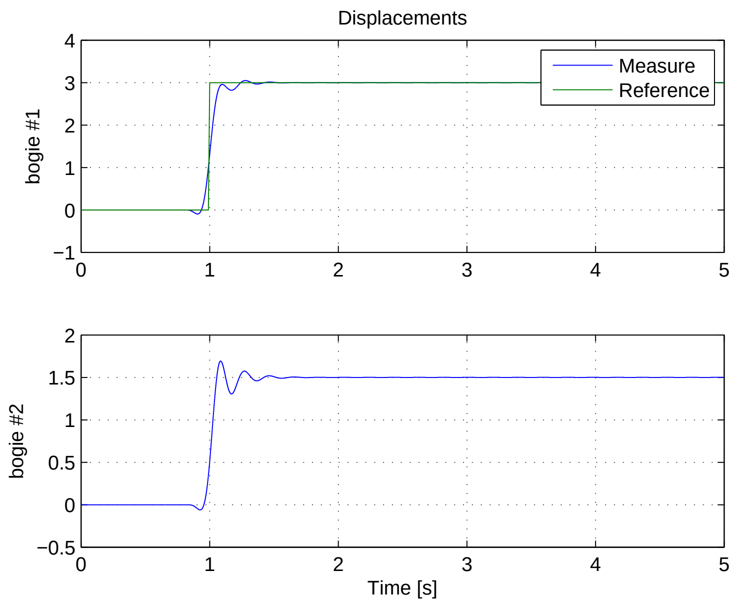

Figure A.2: Collocated - Displacements

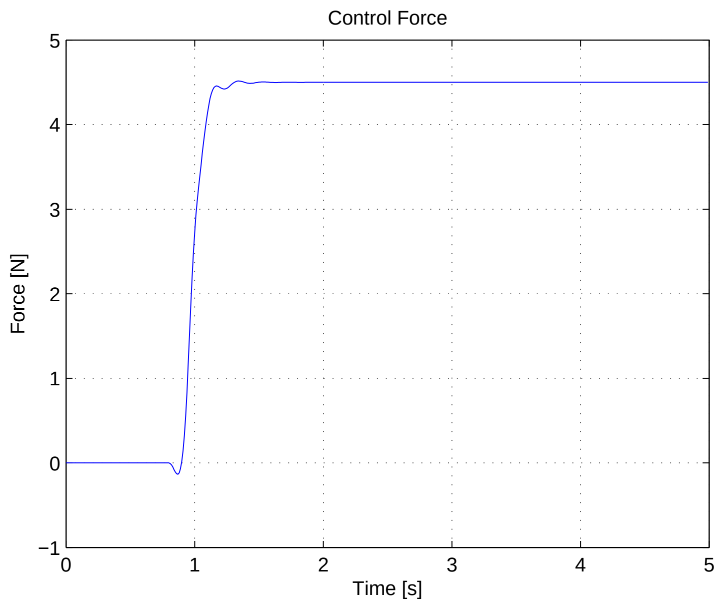

Figure A.3: Collocated - Force

A.3 Non collocated control

Now we want to impose the same three meters step, but to the second bogie. The force will be still on the first bogie and therefore this is defined as non collocated control. We want to show that the model predictive control is capable of taking into account many different situations with the same parameters. The parameters in fact, as explained in the state of the art, are dependant on the system characteristics, and not on the single task the control has to perform. With the classic PID controller we would have changed the gains, but here is not necessary.

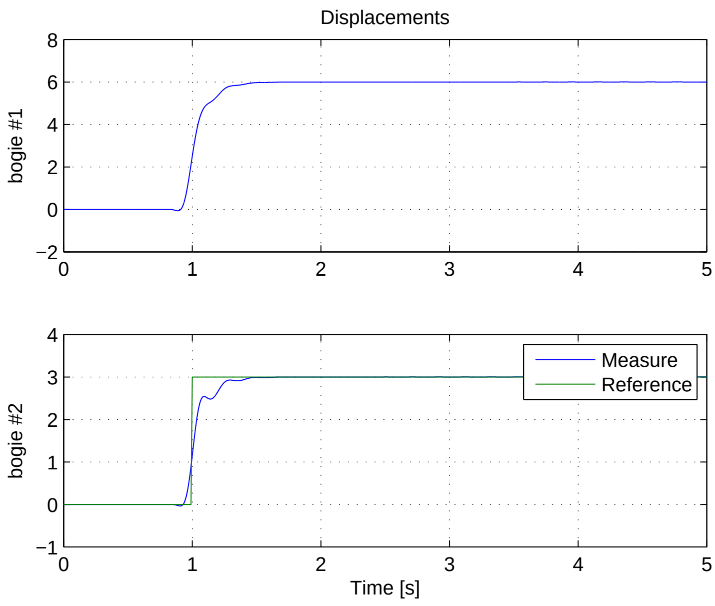

Figure A.4: Non collocated - Displacements

A.4 Multi input multi output

The examples above were single input, single output systems, but the model predictive controller is capable of taking into account whatever requirements. Of course for better performances it is required that we impose one reference per each control force we want to apply, otherwise many different optimal solution could exists and could cause problems to the minimization function. Here we would like to impose a three meters step to the first bogie, while wanting the second one to stand still. This, of course, will require the forces to go in opposite directions. Since we are more interested in having the second

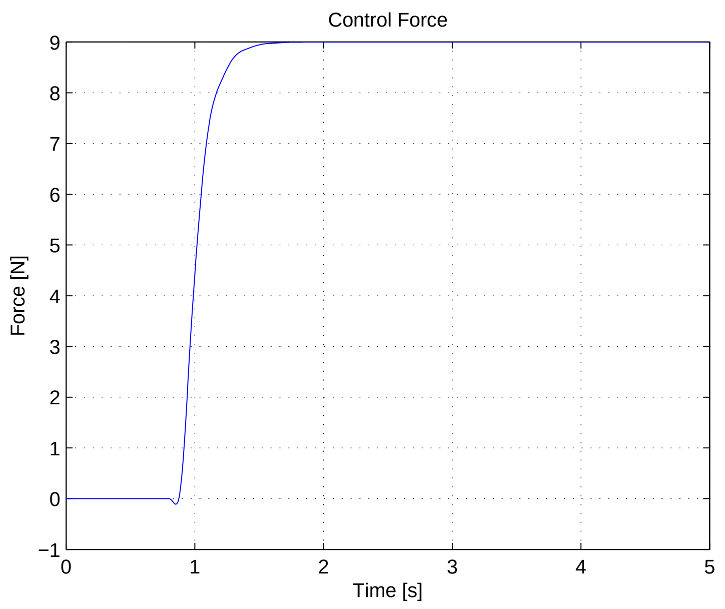

Figure A.5: Non collocated - Force

bogie steady on zero, we weighted differently the errors for both boogies. The weight for the first bogie will be 0.1, while the second one will be set at 1.

As we can see from figure A.6 the controller is capable of dealing with both boogies nicely. Figure A.7 confirms that the forces must go in opposite direction. Now we might want the second bogie steadier, $0.05m$ of displacement could be too much. We are then trying to weight the first bogie even less, for example 0.05.

As we can see from figure A.8 the change in the weights has worked. Now the second bogie is moving a lot less than before, but this has required to lower a little bit the performances for the first bogie. Figure A.9 shows how this is obtained using the same amount of force, just with minor differences difficult to spot in their trajectory.

With this brief part we have showed the basic functions that a model predictive control can manage, and we have done that with a linear model predictive control applied on a linear simple system. We have pointed out that this type of control, opposed to the classic controls that can be used for state space systems, does not require a setup per each situations but only one since its parameters are dependant on the system characteristics. We have also shown its sensitivity to changes in weights.

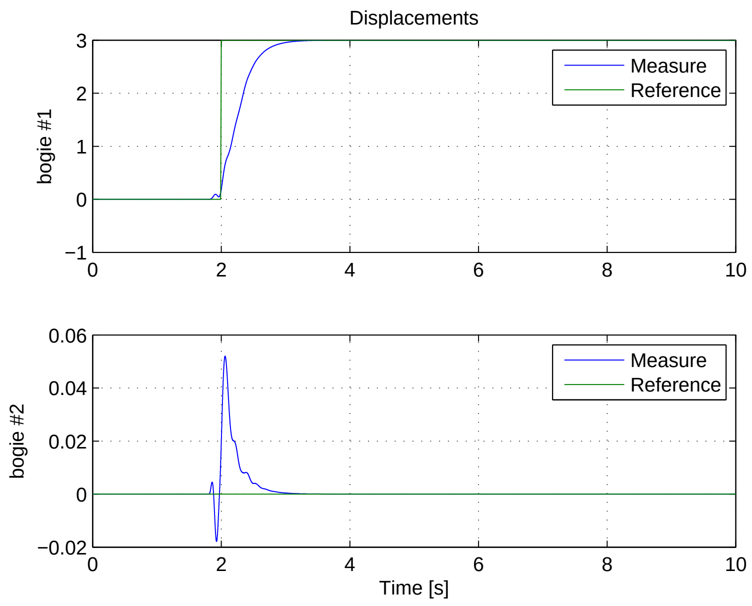

Figure A.6: MIMO - Displacements

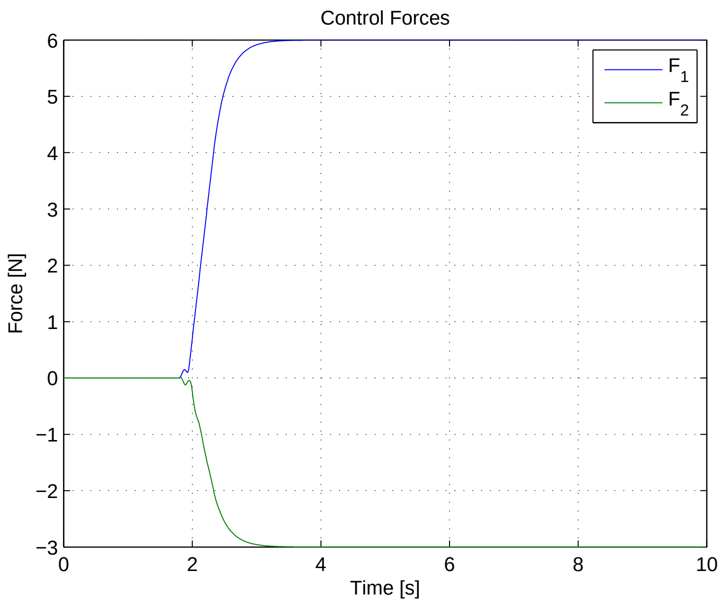

Figure A.7: MIMO - Forces

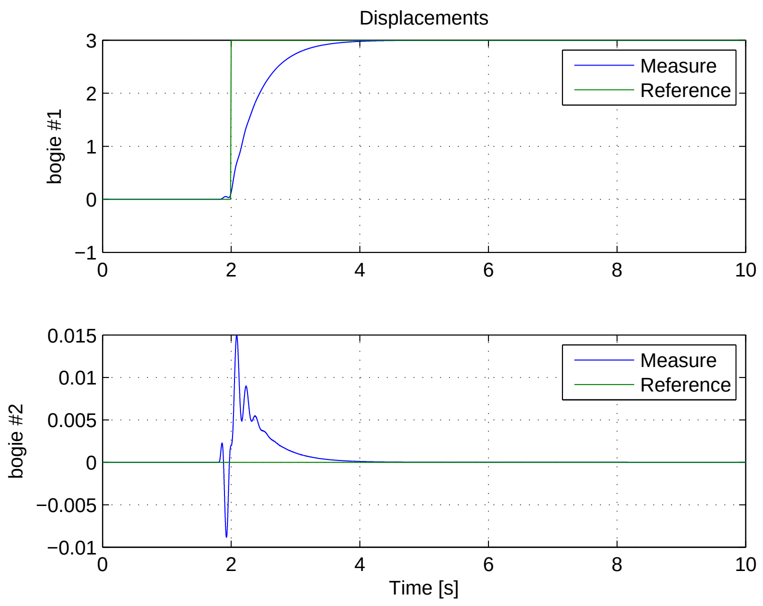

Figure A.8: MIMO - Displacements 2

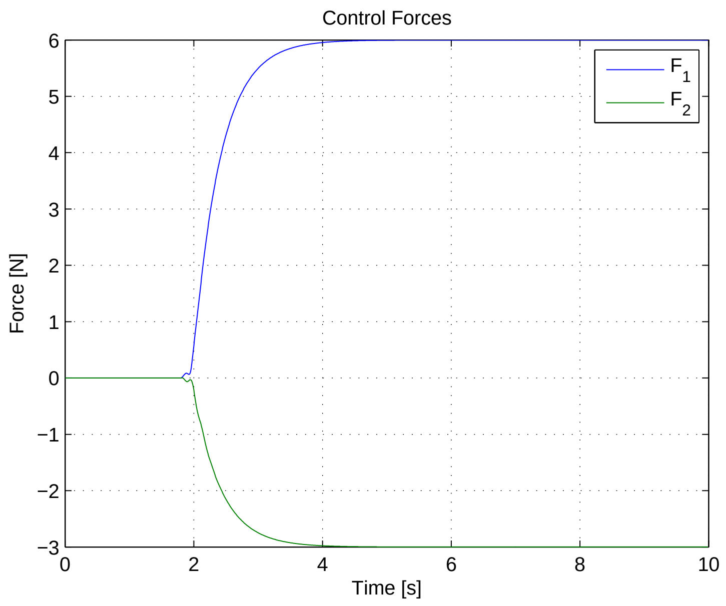

Figure A.9: MIMO - Forces 2

Appendix B

Vehicle Model

Since we are studying an MPC controller we need an accurate model of the system in the state-space form. Although it is important to obtain a trustful model of the system, it is not fundamental for it to be exact since, thanks to the receding strategy, the controller is capable of coping with model variances. Typically, modelling a vehicle do not cause particular problems. The majority of the vehicle parameters tend to remain constant for its whole lifespan, like for example the stiffness of the springs. That value might vary a little bit, but the overall behaviour will remain very close to the initial one. The damping coefficient is one of the few parameters that is capable of varying a lot, but on the common usage of the vehicle the dynamic behaviour remains similar. Only on very hard dynamics, like hard cornering, a difference is noticeable. For these reasons the biggest problem in modelling a vehicle is only due to the difficulties in modelling the tyres. Their parameters are in fact able to change a lot even through a single day thanks to change in weather conditions. We will make for these reasons the strong hypothesis of knowing all the vehicle parameters exactly, included the tyre parameters.

It is important to notice that two different vehicle models are required to simulate our controller. The first one is the model that is used to actually simulate the real vehicle, whom is kept as accurate as possible. We have decided to use a 4 contact single track model to limit the computational effort required for the simulations, but this can be kept as accurate and complicate as needed. The second model is the vehicle model used by the controller to calculate the prediction and therefore obtain the future trajectories of the controls. This model is typically kept intentionally simpler, for example a 2 contact single track model can be used, so that the controller is able to execute the optimization faster. In the end what happens is that the controller takes from the sophisticated model informations on the current state of the system. Then, using the simplified model, it makes the prediction and therefore it obtains the future trajectory of the manipulated variables. Afterwards those control actions are applied, but only for the first step, to the sophisticated model. The possible discrepancies are taken into account thanks to the receding strategy.

B.1 Chassis dynamic

B.1.1 Equations of motion

Here we will obtain the formulation for the 4 contact single model. The simplified model will be obtained easily by reducing this one. The hypothesis under which we are going to develop this model are:

- Rigid body
- Gauge effects neglected
- No elasto-kinematic effect considered
- Trailing arm neglected
- Rolling resistance neglected

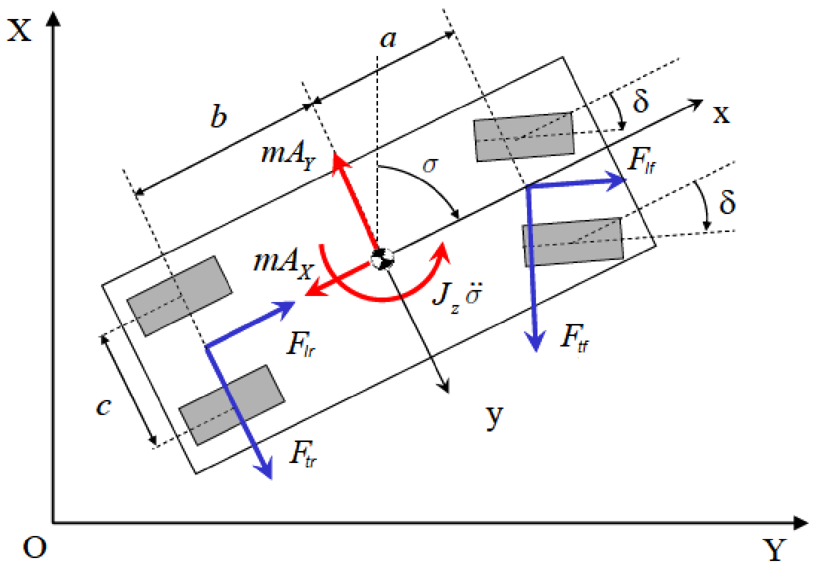

Figure B.1: Vehicle Model

For the state variables considered, that are lateral velocity V_y, longitudinal velocity V_x and yaw angle σ, from simple equilibrium equations from figure B.1 we obtain:

$$\begin{cases} m\dot{V_x} = m\dot{\sigma}V_y - (F_{tfl} + F_{tfr})\sin(\delta) + (F_{lfl} + F_{lfr})\cos(\delta) + (F_{lrl} + F_{lrr}) - F_{aero} \\ m\dot{V_y} = -m\dot{\sigma}V_x + (F_{tfl} + F_{tfr})\cos(\delta) + (F_{trl} + F_{trr}) + (F_{lfl} + F_{lfr})\sin(\delta) \\ J_z\ddot{\sigma} = +(F_{tfl} + F_{tfr})\cos(\delta)a - (F_{trl} + F_{trr})b + (F_{lfl} + F_{lfr})a\sin(\delta) \\ \dot{x_g} = V\cos(\beta + \sigma) \\ \dot{y_g} = V\sin(\beta + \sigma) \end{cases} \tag{B.1}$$

with $\beta = \tan^{-1}(V_y/V_x)$ and $V = \sqrt{V_y^2 + V_x^2}$. δ is the angle imposed to the wheel due to the steering, F_{lfr} is the longitudinal force for the front right tyre, F_{trl} is the transversal force for the rear left tyre and F_{aero} is the drag force expressed as $F_{aero} = \frac{1}{2}\rho V_x^2 C_D A$. Also note that x_g and y_g are the displacements of the center of mass related to the absolute reference system. For what regards the longitudinal tyre forces we have neglected the wheel dynamic and therefore they can be easily expressed knowing the torque C_r and C_f imposed by the transmission and braking system.

$$\begin{cases} F_{lfl} = F_{lfr} = C_f/R \\ F_{lrl} = F_{lrr} = C_r/R \\ F_{tfr} = F_{tfr}(x, \dot{x}, y, \dot{y}, \sigma, \dot{\sigma}, F_{lfr}, N_{fr}) \\ F_{tfl} = F_{tfl}(x, \dot{x}, y, \dot{y}, \sigma, \dot{\sigma}, F_{lfl}, N_{fl}) \\ F_{trr} = F_{trr}(x, \dot{x}, y, \dot{y}, \sigma, \dot{\sigma}, F_{lrr}, N_{rr}) \\ F_{trl} = F_{trl}(x, \dot{x}, y, \dot{y}, \sigma, \dot{\sigma}, F_{lrl}, N_{rl}) \end{cases} \tag{B.2}$$

The longitudinal forces right and left are equal because the braking system impose an equal torque on both wheels and the same can be said for the transmission since we are using an open differential. As we can see the transversal forces depend on the state of the vehicle and on the longitudinal force applied to the same wheel. Another effect that modifies the force that the wheel can apply transversally are the vertical loads like N_{fr}. Since we are dealing with a 4 contact model we must include the load transfer, and this is made considering only a steady-state load transfer.

$$\begin{cases} N_{fl} = \frac{mgb}{2p} + \eta \cdot \Delta N \\ N_{fr} = \frac{mgb}{2p} - \eta \cdot \Delta N \\ N_{rl} = \frac{mga}{2p} + (1-\eta) \cdot \Delta N \\ N_{rr} = \frac{mga}{2p} - (1-\eta) \cdot \Delta N \end{cases} \tag{B.3}$$

where $\frac{mgb}{2p}$ and $\frac{mgb}{2p}$ are the static loads and the rest is the load transfer. $\Delta N = \frac{mA_y h_g}{c}$ is the global load transfer, with $A_y = \dot{V}_y + V_x\dot{\sigma}$, and $\eta = \frac{k_{roll,f}}{k_{roll,f}+k_{roll,r}}$, where $k_{roll,f}$ and $k_{roll,r}$ are the front and rear roll stiffness.

B.1.2 Tyre modelling

The model for lateral forces that a tyre is able to exchange is necessarily a Pacejka formulation.

$$\begin{cases} F_{yfl} = -D_{fl} \sin(C_f \tan^{-1}(B_f\alpha_f - E_f(B_f\alpha_f - \tan^{-1}(B_f\alpha_f)))) \\ F_{yfr} = -D_{fr} \sin(C_f \tan^{-1}(B_f\alpha_f - E_f(B_f\alpha_f - \tan^{-1}(B_f\alpha_f)))) \\ F_{yrl} = -D_{rl} \sin(C_r \tan^{-1}(B_r\alpha_r - E_r(B_r\alpha_r - \tan^{-1}(B_r\alpha_r)))) \\ F_{yrr} = -D_{rr} \sin(C_r \tan^{-1}(B_r\alpha_r - E_r(B_r\alpha_r - \tan^{-1}(B_r\alpha_r)))) \end{cases} \tag{B.4}$$

where everything depend on the front and rear slip angles α_f and αr and the constants B_i, C_i and E_i are characteristics of the tyre. D_{ij} are expressed as follows:

$$\begin{cases} D_{fl} = \mu_{yfl}(q_f + s_f \ df_{fl})N_{fl} \\ D_{fr} = \mu_{yfr}(q_f + s_f \ df_{fr})N_{fr} \\ D_{rl} = \mu_{yrl}(q_r + s_r \ df_{rl})N_{rl} \\ D_{rr} = \mu_{yrr}(q_r + s_r \ df_{rr})N_{rr} \end{cases} \tag{B.5}$$

being $df_{ij} = \frac{N_{ij}-N_0}{N_0}$, and $N_0 = 4000N$. As we can see these are dependant on the load transfer and in this way we can model the correlation that there is between the lateral force and the variation in vertical load. The result of all this brings to the behaviour of the lateral force, function of the slip angle, that can be seen in figure B.2.

To take into account the effect of the combined friction effect, even though we do not have the complete Pacejka model, we decided to make the μ_{yfr}, μ_{yfl}, μ_{yrr} and μ_{yrl} coefficient dependant on the longitudinal forces that are known from the vehicle inputs. Knowing both the longitudinal force and the vertical force acting on the tyre, we can evaluate the engaged friction

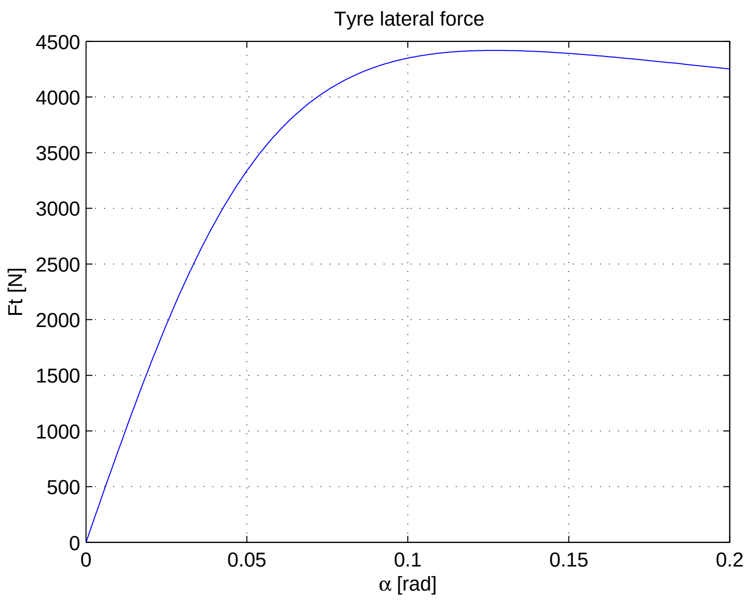

Figure B.2: Lateral force

$\mu_{xij} = F_{xij}/N_{ij}$ and therefore we can reduce the maximum value of friction available for the lateral force μ_{yij}. There is the hypothesis of $\mu_{tot} = 1$.

$$\begin{cases} \mu_{yfl} = \sqrt{\mu_{tot}^2 - \mu_{xfl}^2} \\ \mu_{yfr} = \sqrt{\mu_{tot}^2 - \mu_{xfr}^2} \\ \mu_{yrr} = \sqrt{\mu_{tot}^2 - \mu_{xrr}^2} \\ \mu_{yrl} = \sqrt{\mu_{tot}^2 - \mu_{xrl}^2} \end{cases} \tag{B.6}$$

We must then consider that the forces exchanged by a tyre have their own dynamic and therefore a delay. In our case we are going to consider this effect only for the lateral forces. Typically this behaviour is modelled with a first order dynamic system that is forced by the value of the lateral force that the tyre would exchange if considered in steady-state conditions.

$$\begin{cases} \frac{L_f}{V}\dot{F}_{yfl} + F_{yfl} = \bar{F}_{yfl} \\ \frac{L_f}{V}\dot{F}_{yfr} + F_{yfr} = \bar{F}_{yfr} \\ \frac{L_r}{V}\dot{F}_{yrl} + F_{yrl} = \bar{F}_{yrl} \\ \frac{L_r}{V}\dot{F}_{yrr} + F_{yrr} = \bar{F}_{yrr} \end{cases} \tag{B.7}$$

where $\bar{F}_{yij}$ are drawn from the Pacejka formulation described above, and where L_f and L_r are constant characteristic of the tyre, known as relaxation

ρ $[kg/m^3]$	C_D [ ]	A $[m^2]$	a $[m]$	b $[m]$	p $[m]$
1.225	0.33	2.59	1.235	1.465	2.7
R $[m]$	m $[kg]$	J_z $[kgm^2]$	h_g $[m]$	c $[m]$	η [ ]
0.328	1880	2873	0.6	0.796	0.55
B_f [ ]	B_r [ ]	C_f [ ]	C_r [ ]	E_f [ ]	E_r [ ]
-7.5	-10.078	1.503	1.503	-0.233	-0.059
L_f $[m]$	L_r $[m]$	q_f [ ]	q_r [ ]	s_f [ ]	s_r [ ]
0.5	0.7	1	1.2075	-0.12	-0.12

Table B.1: Vehicle parameters

lengths. At this point we just need to obtain the relations between the slip angles and the vehicle kinematics. The slip angles are considered equals between left and right because we neglect the effect of the wheel track. They are defined as the angle between the lateral V_{ti} and the longitudinal V_{li} speed of the rim.

$$\begin{cases} \alpha_f = \tan^{-1}(\frac{(V_y+\dot{\sigma}a)\cos(\delta)-V_x\sin(\delta)}{V_x\cos(\delta)+(V_y+\dot{\sigma}a)\sin(\delta))} \\ \alpha_r = \tan^{-1}(\frac{V_y-\dot{\sigma}b}{-V_x}) \end{cases} \tag{B.8}$$

B.1.3 Validation

To certify the goodness of the model for what regards the lateral dynamic, we decided to compare some experimental measurements done using an Alfa Romeo 159 with our model. The experimentation consists of simple manoeuvres like the double line change and the slalom. From those trials measurements of the steering input, the side slip angle and the yaw rate were taken. Then the steering trajectory was used as input in our model and results were plotted and compared. The data for the 4 contact single track model of the Alfa Romeo 159 are shown in table B.1.

As we can see from figure B.3, B.4, B.5, B.6, B.7, and B.8, even though there is variance between the real vehicle and the simulated model, we can state with absolute confidence that the model that we have used is a very good representation of a typical road vehicle.

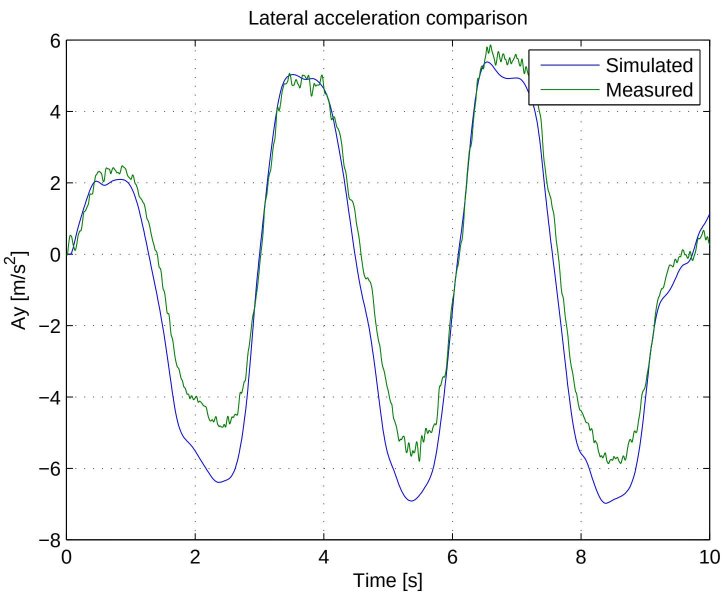

Figure B.3: Slalom - lateral acceleration

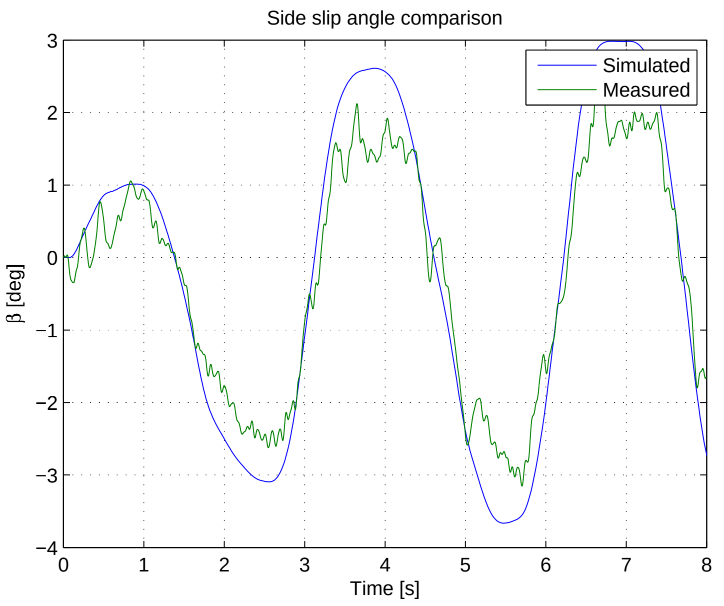

Figure B.4: Slalom - side slip angle

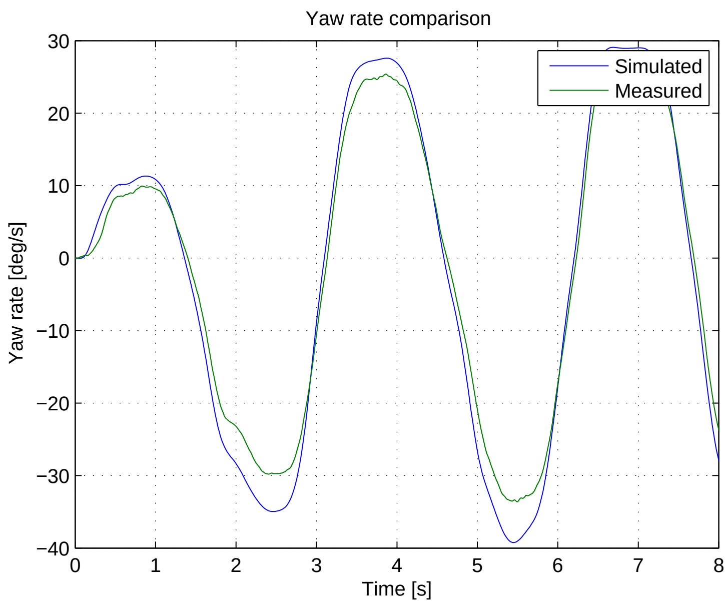

Figure B.5: Slalom - yaw rate

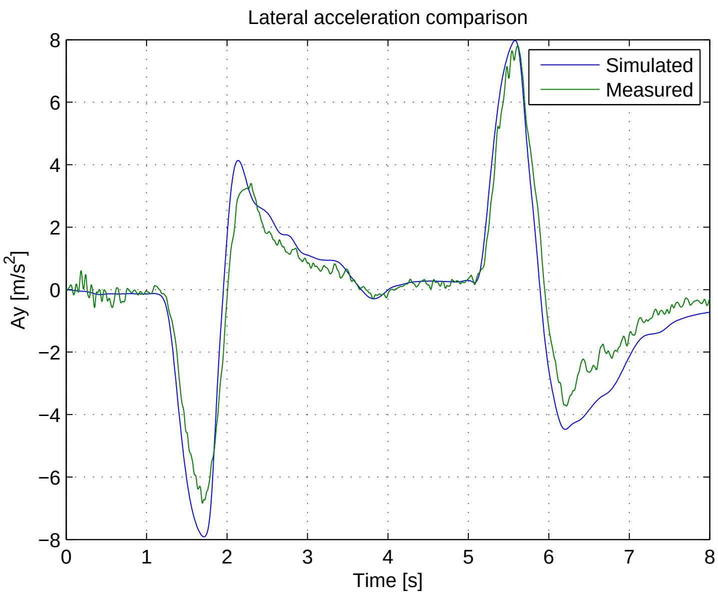

Figure B.6: Double line change - lateral acceleration

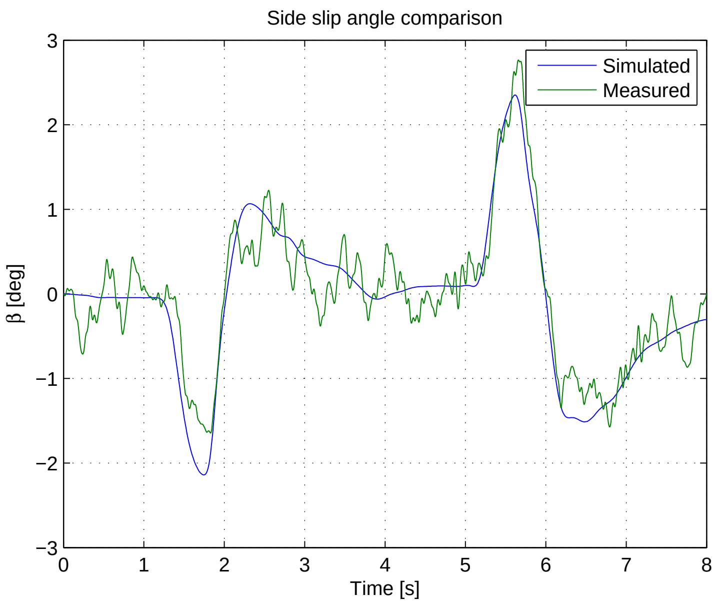

Figure B.7: Double line change - side slip angle

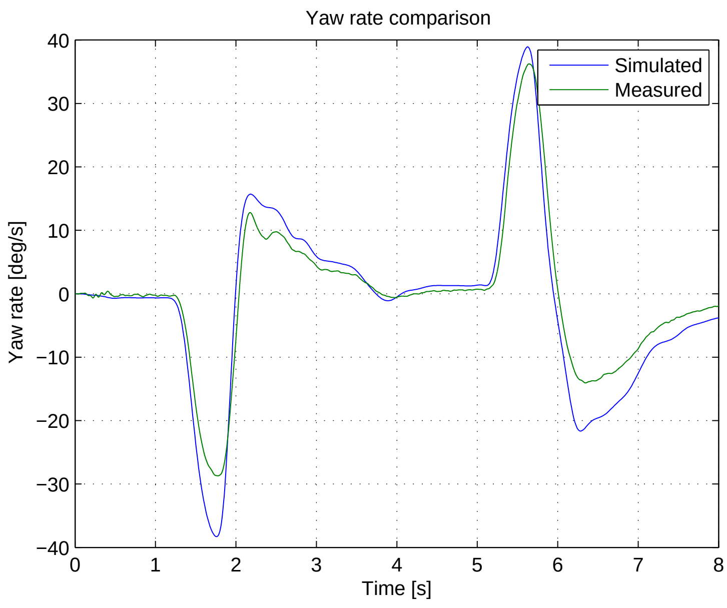

Figure B.8: Double line change - yaw rate

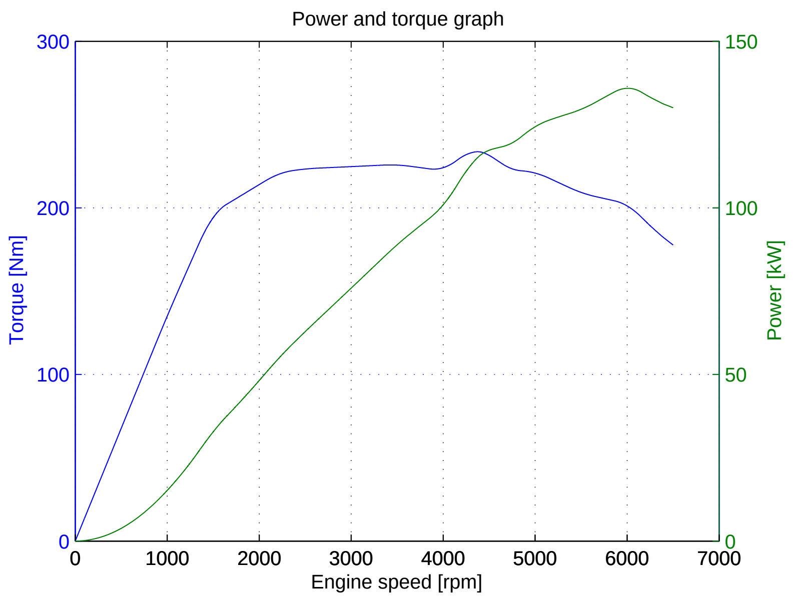

Figure B.9: Engine torque and power

B.2 Engine and drivetrain

B.2.1 Engine modelling

Of course, since we want to consider a controller capable of acting on the vehicle through the common control that it has, it is necessary to model also all the systems that are interested in the longitudinal dynamic. This consider also modelling both the engine of the vehicle, that in our case is the 2.2Jts made by Alfa Romeo, and the transmission with the brakes. Thanks to [14] we managed to obtain an experimental measure of the power and torque that that engine is capable of producing in full throttle. To the torque diagram we added a segment from 0 to 1000rpm to model the usage of a clutch. This is for when the speed of the vehicle is lower than the minimum speed allowed by the idling speed of the engine in a certain gear. To model also the presence of the engine brake, we introduced a value of torque equal to -10% of the torque when full throttle, that brakes the vehicle when the accelerator is not used.

B.2.2 Transmission and brake modelling

Also for what regards the transmission a research for the required parameters was done, and that allowed us to obtain the values of the various gear ratios,

τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_{fin}
1/3.818	1/2.353	1/1.571	1/1.146	1/0.943	1/0.861	1/4.176

Table B.2: Gearbox ratios

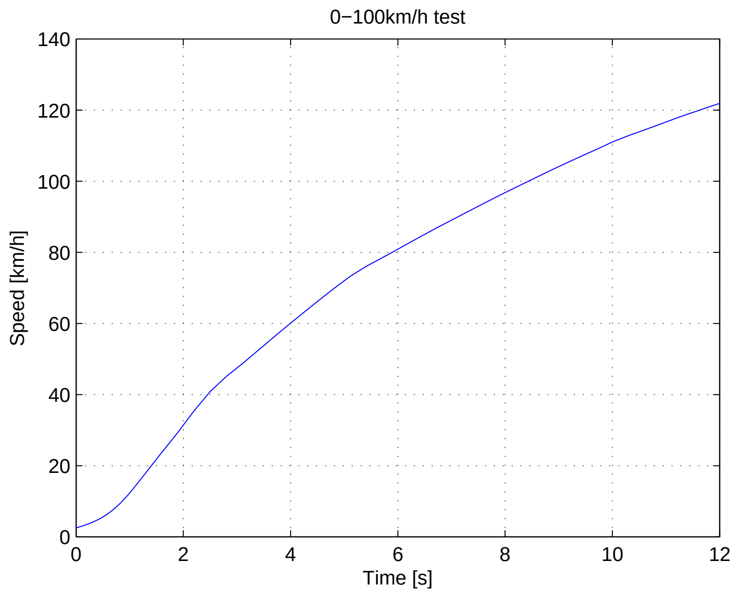

Figure B.10: Acceleration test

final ratio included, that are shown in table B.2. It was also required to program a basic gear change logic such that the vehicle was able on his own to increase or decrease speed for more than what is allowed in one fixed gear. This basic automatic gearbox allows us to not include the gear selection in the outputs of the controller. This choice was made because a linear MPC controller is not capable to deal with hybrid systems, but this can be done while using a nonlinear MPC. The gear change logic is very simple: the vehicle up shifts if the revs are higher than a bound set at $6000rpm$, and it down shifts when the revs are lower than another bound set at $2000rpm$. If the first gear is engaged, lowering the revs under the $2000rpm$ limit will not, of course, produce a down shift, and if the revs will proceed getting even lower than the engine idle speed, the torque diagram will be valid in any case since we have introduced the clutch model. On the other hand, since the drag force is introduced, the engine is not capable of getting to the rev limiter in final gear, and therefore no further modifications are required.

The drag test from 0 to 100 km/h, figure B.10, ends up being totally compliant with the same test made by the real Alfa Romeo 159JTs. The

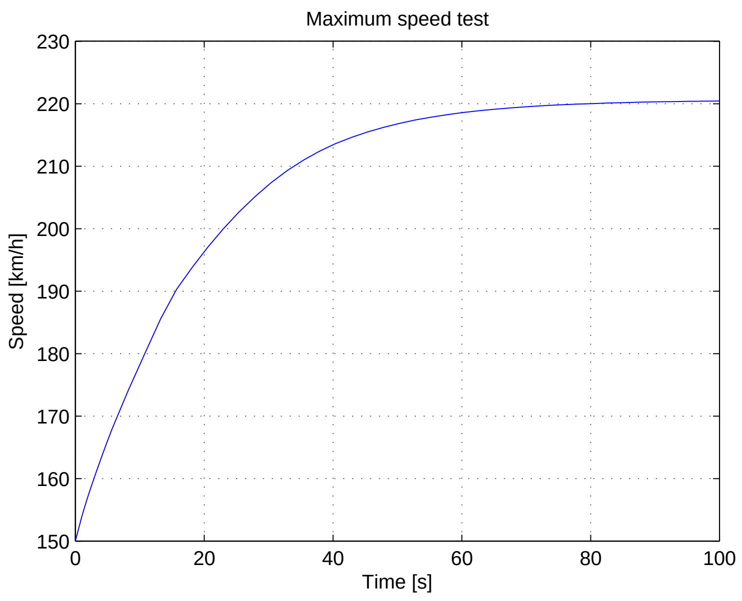

Figure B.11: Maximum speed

proclaimed figures are in fact $8.7s$, absolutely comparable with our $8.5s$.

For what regards the maximum speed obtainable, in figure B.11 we get $220km/h$ from our model, again comparable with the maximum speed of $224km/h$ proclaimed by Alfa Romeo.

The braking system is simply modelled as a torque applied to the wheel that is regulated in amplitude by the brake input. We do not have particular requirements for the braking system, thus the model was kept simple on purpose. Two values of maximum braking torque are present to cope with the common vehicle's biased braking system. The torque for the front axle is $C_{bf} = 2700\,\mathrm{Nm}$ and the torque for the rear axle is $C_{br} = 1800\,\mathrm{Nm}$.

As we can see from figure B.12 with full braking the vehicle is capable of producing approximately $-0.75G$ of longitudinal acceleration that is the value that commonly a mid sized car can achieve. The deceleration is not constant due to non linearity in the torque figure of the engine. We can also see three down shifts.

Finally we can express the longitudinal forces applied at the wheel functions of the accelerator α and of the brake γ.

$$\begin{cases} F_{lfl} = F_{lfr} = \alpha(\frac{C_{max}-C_{min}}{2R\tau_i}) + \frac{C_{min}}{2R\tau_i} - \gamma\frac{C_{bf}}{2R} \\ F_{lrl} = F_{lrr} = -\gamma\frac{C_{bf}}{2R} \end{cases} \tag{B.9}$$

where τ_i indicates the total transmission ratio that also considers the final

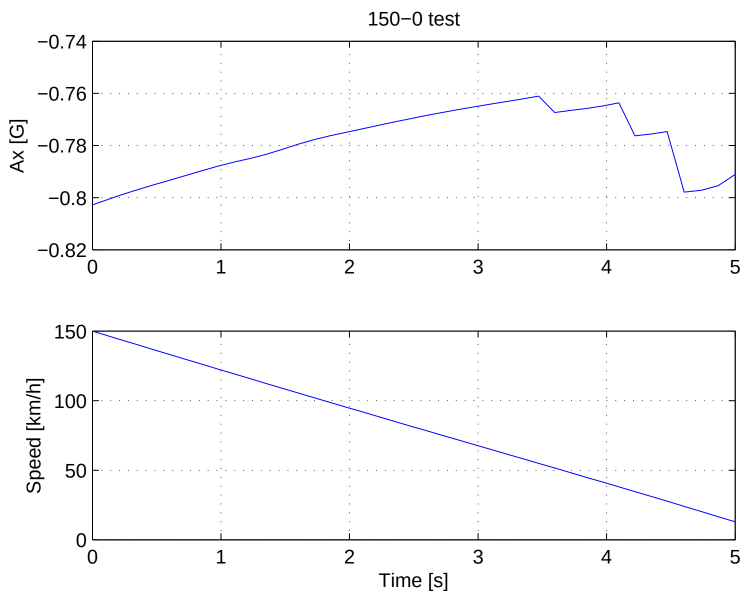

Figure B.12: Deceleration test

ratio. As we have already explained, the longitudinal forces are equal on the axle because the braking system applies the same torque on both wheels and so does also the transmission since we have made the hypothesis of using an open differential. For these reasons both the engine torque and the braking torque are divided by 2.

Acronym

ABS	Antilock Braking System
ESP	Electronic Stability Control
AFS	Active Front Steering
DARPA	Defense Advanced Research Projects Agency
MPC	Model Predictive Control
NMPC	Non linear Model Predictive Control
LTV	Linear Time Variant
SQP	Sequential Quadratic Programming
AGV	Automated Guided Vehicle
MIMO	Multi Input Multi Output

Bibliography

[1] Buehler M., Iagnemma K. and Singh S. "The DARPA Urban Challenge".

[2] Falcone P., Borrelli F., Asgari J., Tseng H. E. and Hrovat D. "Predictive Active Steering Control for Autonomous Vehicle Steering" in *IEEE Transactions on control systems technology* vol. 15, no. 3, pp. 566-580, May 2007.

[3] Keviczky T., Falcone P., Borrelli F., Asgari J. and Hrovat D. "Predictive Control Approach to Autonomous Vehicle Steering" in *American Control Conference 2006*, 2006.

[4] Falcone P., Borrelli F., Tseng H. E., Asgari J. and Hrovat D. "A Hierarchical Model Predictive Control Framework for Autonomous Ground Vehicles" in *American Control Conference 2008*, 2008.

[5] Falcone P., Borrelli F., Tseng H. E., Asgari J. and Hrovat D. "A Model Predictive Control Approach for Combined Braking and Steering in Autonomous Vehicles" in *Mediterranean Conference on Control and Automation 2007*, 2007.

[6] Palmieri G., Barbarisi O., Scala S. and Glielmo L. "A preliminary study to integrate LTV-MPC lateral vehicle dynamics control with slip control" in *28th Chinese Control Conference 2008*, 2008.

[7] Di Cairano S., Tseng H.E., Bernardini D. and Bemporad A. "Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angle domain" in *IEEE Transactions on control systems technology*, vol. 21, no. 4, pp. 1236-1248, July 2013.

[8] Gray A., Gao Y., Lin T., Hedrick J. K., Tseng H. E. and Borrelli F. "Predictive control for agile semi-autonomous ground vehicles using motion primitives" in *American control conference 2012*, 2012.

[9] Naus G. J. L., Ploeg J. , Van de Molengraft M. J. G., W. P. M. H. Heemels and Steinbuch M. "A model predictive control approach to design a parameterized adaptive cruise control"

[10] Yoon Y., Shin J., H. J., Park Y. and Sastry S. "Model predictive active steering and obstacle avoidance for autonomous ground vehicles" in *Control engineering practice*, vol. 17, pp. 741–750, 2009.

[11] IEEE Spectrum website. URL: `http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/toyota-semi-autonomous-lexus-car-will-keep-you-safe`

[12] Rossiter J.A. "Model-based predictive control".

[13] Wang L. "Model predictive control system design and implementation using Matlab".

[14] Rototest Research website. URL:`http://rototest-research.eu/index.php?Visitor=4&DN=29`

[15] Grune L., Pannek J. "Nonlinear model predictive control".

[16] Franklin G.F., Powell D.J., and Workman M.L., "Digital Control of Dynamic Systems" (3rd Edition), 1997.