Treball de Fi de Màster

Master's Degree in Industrial Engineering (MUEI)

Predictive Control of an Autonomous Vehicle using the RTI method

MEMÒRIA

September 23, 2021

Autor: Martín López García

Director: Dr. Vicenç Puig Cayuela

Convocatòria: October 2021

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Acknowledgments

I would like to express my gratitude to my advisor Dr. Vicenç Puig, who guided me during the development of this thesis and helped me to make this work possible.

I would also like to thank my friends for their emotional support and Núria, for her help and patience throughout the development of this master's thesis.

Finally, special thanks to my family, who always trusted, helped and supported me during this journey and continue to do so, being my inspiration to pursue my goals.

Abstract

With the advance of technology and robotics as well as the increase in automation of factories, new control strategies are needed to fulfill the strict production, time and economic targets.

This master's thesis aims to design and implement a Model Predictive Control based on the Real Time Iteration scheme in a autonomous vehicle. For this purpose, the algorithm was first studied and designed in Matlab and then the controller was implemented in a simulated environment using ROS and a vehicle simulator, where conditions closer to those of a real environment were met.

Both the designing and implementation results were positive, obtaining a MPC controller able to provide the inputs for the vehicle to follow the reference correctly, obtaining a reliable performance. As the simulated environment is meant to be as closer to reality as possible, it is safe to assume that the controller developed and tested in ROS will have a positive result when implemented in a real Radio-Controlled car.

Contents

ETSEIB

List of Figures

List of Tables

Index of Notation

Model notation

Δt	Sampling time
a	Frontal distance to the CoG
A, B, C, D	System matrices
a_{motor}	Rear wheel acceleration
b	Rear distance to the CoG
F_{yf}	Front lateral tire force
F_{yr}	Rear lateral tire force
I	Inertia along z axis
L	Gain matrix
$\hat{x}$	Estimated state vector
I	Identity matrix
u	Control actions
v_x, v_y	Velocities defined with respect to the robots reference frame
x	State vector

Acronym

AGV	Automatic Guided Vehicle
Cf,Cr	Frontal and rear stiffness coefficient

COG Center of Gravity

DMC Dynamic Matrix Control

EKF Extended Kalman filter

ESC Electronic Speed Control

Fyf Lateral frontal forces

Fyr Lateral rear forces

GPC Generalized Predictive Control

LMI Linear Matrix Inequality

LMPC Linear Model Predictive Control

LPV Linear Parameter-Varying

MPC Model Predictive Control

NLP Non Linear Program

NMPC Non-Linear Model Predictive Control

OCP Optimum Control Problem

PWM Pulse Width Modulation

QP Quadratic Programming

RC Radio-Controlled

RTI Real Time Iteration

SQP Sequential Quadratic Programming

Chapter 1

Introduction

This chapter gives a scope of this master's thesis. Section 1.1 presents the context in which this project was developed while Section 1.2 describes the motivation behind it. In Section 1.3, the desired objectives pursued with this thesis are explained. Finally, a short outline of the document's structure along with a summary of each of its chapters is found in Section 1.4.

1.1 Context of the project

This thesis has been carried out with the collaboration of the Institut de Robòtica i Informàtica Industrial (IRI), where the design and implementation of a RTI-MPC control was performed in a simulation environment in ROS. The vehicle simulator is based on the BARC (Berkeley Autonomous Race Car) project, whose copyright is owned by the Model Predictive Lab at the University of California Berkeley.

1.2 Motivation

Autonomous robotics is an state of the art technology revolutionizing the way people and cargo are transported. Although this supposes a great technological advance, it also brings a lot of new difficulties that require to be overcome, since also new control strategies and security measures have to be achieved.

The need to design and successfully implement control loops in these vehicles are a core key in order to achieve a worldwide implementation and use of this means of transportation. The motivations behind choosing this topic in this project are mainly three.

First, the possibility to deepen into the robotics field, particularly in the mobile autonomous robotics, field exponentially growing in the latest years, with new needs for controllers with higher requirements.

Secondly, the opportunity to design an advanced control solution in an application never tested before, using the tools provided by the BARC project, allowing to study and put into practice a Model Predictive Control (MPC) based on the real-time iteration (RTI) scheme.

Finally, the chance to apply the control engineering theory in a real world problem and watch directly the results of this designed control.

It is for those reasons that this project intends to cover the problem from the study of this new technique to its application in a simulated environment in ROS in order to prove its feasiblity as well as the analysis of the results provided by this controller.

1.3 Objectives

The main target of this thesis is to develop a controller based in the Model Predictive Control theory by using the Real-Time Iteration scheme, which is a control algorithm that allows to use the nonlinear model of the system to perform an optimal and predictive control, in a simulated environment. For this purpose, a series of milestones have been proposed:

1. Study of the Model Predictive Control theory.
2. Study of the Real Time Iteration approach to the Non Linear Model Predictive Control problem.
3. Design and development of the RTI control algorithm.
4. Implementation in a simulated environment using the Robot Operating System (ROS).
5. Analysis of the simulation results and verification of the control algorithm.

1.4 Thesis structure

In this section, an outline of the master's thesis is provided in order to give an overview to this project. The sections in which this project is divided are as follows:

- *Chapter 1: Introduction*

  In this chapter, an explanation of the context of the project, motivation and objectives of this master's thesis is provided.

- *Chapter 2: Background and State of the art*

  This chapter gives a look to the level of development of mobile robotics and autonomous driving and its applications, as well as the model predictive control and the real time iteration (RTI) strategy theoretical foundations.

- *Chapter 3: Case study*

  In this chapter, the different control models used in this thesis are explained including control model, simulation model and observer model.

- *Chapter 4: RTI Scheme*

  This chapter focuses on the design of the control scheme and the implementation of this controller in the system studied.

- *Chapter 5: Implementation in ROS*

  This chapter explains the simulation environment and tools used in order to implement and to test correctly the control scheme.

- *Chapter 6: Experimental results*

  This chapter discusses the results of implementation both in Matlab and ROS and its differences and analyses the outcomes of this control.

- *Chapter 7: Economic, social and environmental study*

  This chapter is devoted to analyse the impact of this project in terms of socioeconomic and environmental features for the purpose of analysing its outcomes beyond the technical part.

- *Chapter 8: Project budget and planning*

  This chapter is targeted to provide an indication of the project costs, including materials and workforce. It also shows the time effort and the planning in which the project was carried out.

- *Chapter 9: Conclusions and future work*

  Finally, this chapter explains the conclusions of this thesis extracted from the results and future lines of work to be continued in future projects.

Chapter 2

Background and State of the art

2.1 Mobile Robotics and autonomous driving

In this thesis, an approach to the control of an autonomous vehicle is presented, in terms of design and implementation.

The autonomous mobile robotics field encompasses all those machines with the ability to move around in a working space which means, unlike the first industrial robots, they are not restricted to a single physical location. This working space varies from small areas like houses or small-sized factories to kilometers of car roads or huge tracks. Depending of the environment in which the robots move, we can differentiate three types [1] :

1. Ground mobile systems: This kind of robots include vehicles with wheels, animal-like robots and humanoid robot among others.

2. Aerial mobile systems: The most representative robots of this type are drones (shaped like helicopters, planes, etc) but also satellites.

3. Underwater and water mobile systems: This includes boats, submarines, marine exploration robots, etc.

It is also important to specify that by autonomy in the driving field, two principal different types can be considered. One is a high autonomy, where the robot has the ability to respond and overcome unexpected obstacles through its on board electronics and software, meaning that there is a reasoning implied in the control process of the machine, that is, a need for thinking which path to take. The other kind of autonomy is a low autonomy, in which the robot can guide itself through a track but is not able to cope with unforeseen event. The former type of

autonomy will be the one that will be tackled in this master's thesis.

In the last twenty years, mobile robotics have experimented and increase in use not only with industrial purposes but also with commercial goals. In Figure 2.1, the forecast for the next years show a huge increase in robot product shipments, remarkably the mobile robots suppose the greatest share of this shipments, which show the importance of studying the related technologies.

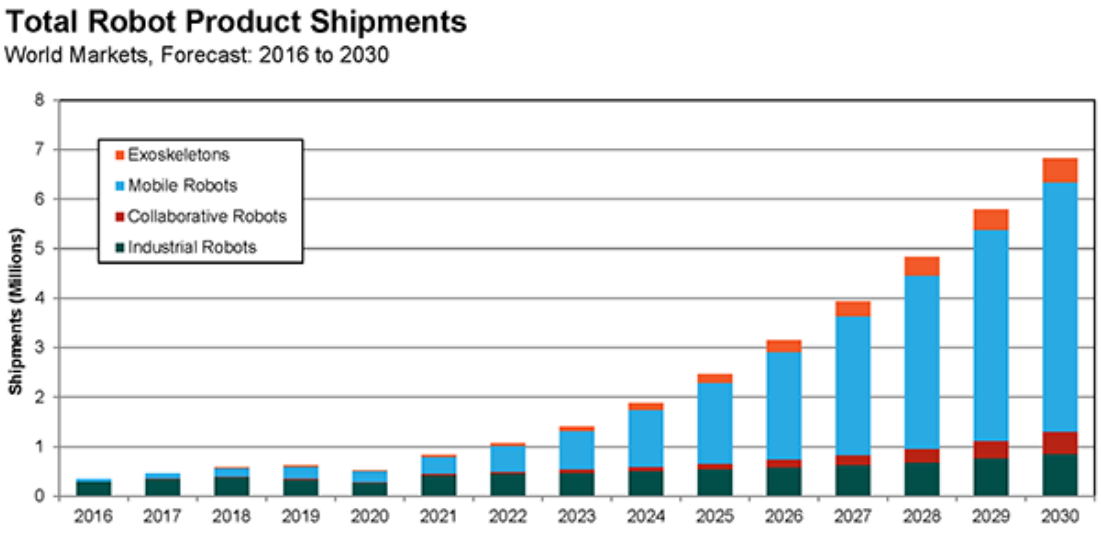

Figure 2.1: Robot shipments forecast

This type of robots are composed basically from sensors, actuators, a power system and a controller. From the combination of sensors and actuators, enough information must be provided to the robot in order to locate itself and interact with the background, while the power system is made of the corresponding electronics and electrical components. A need for a controller is appreciable as this flow of information must be taken into consideration to achieve the goal of the robot, which would be impossible to reach without the correct control scheme. While some of the control problems presented by autonomous movement in robots can be tackled by classical control strategies, as this technologies advances, more sophisticated control schemes are needed, which is why this thesis focuses on a MPC-RTI control scheme.

The range of applications of this robots is numerous. For example, AGVs allow to automate

factories and carry cargo, autonomous passenger vehicles are also being developed and drones are nowadays being more and more produced, which makes mobile robotics an interesting technology to research and develop. In this project, the case of study will be a mobile robot with a predefined track as working space, and will be presented in Section 3.

Figure 2.2: Example of AGV in factories

Figure 2.3: Example of unmanned drone

2.2 Model Predictive Control

The Model Predictive Control theory refers not to a single specific control strategy but to a wide range of advanced controllers with some common features. This type of controllers appeared first in the seventies in the oil and chemistry industry [2] where classic controllers often were not able to achieve the desired requirements in multi-variable constrained control problems.

The main advantages of these controllers [3] are that they permit an easy design and implementation of complex and multi-variable constrained problems, and they can use future references which allows to compensate for dead times. On the contrary, more computational time is needed which gets even worse when using nonlinear multi-variable problems.
In Figure 2.4, a generic view of a MPC controller is shown.

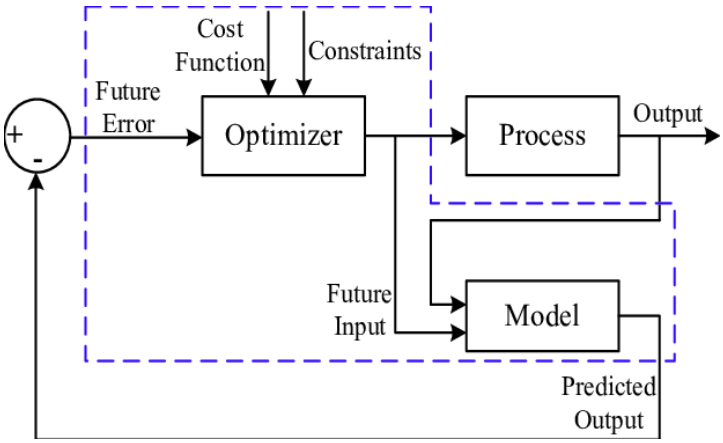

Figure 2.4: Generic MPC controller diagram

Some shared characteristics [4] remarkable in these controllers are:

1. Explicit model of the system in order to predict the process output at future time instants (called prediction horizon).

2. Control sequence in which an objective function is minimized, that provides the control inputs.

3. Receding strategy so that at each sample time, the horizon is shifted towards the future, in which the before mentioned first control inputs are applied to the system.

The model is a cornerstone of a MPC controller. The reason for this is that predictions are calculated with the model, so it is necessary to design the most accurate model possible that reflects the dynamics of the system so that the deviation between real and predicted outputs is minimized.

The cost function may change in the different controllers depending on the target of the controller. Typically, a choice must be made between prioritizing the following of the reference or reducing the control effort due to these two variables being inversely proportional, the more accurately the reference is followed the more control effort needed. The optimizer generates the minimization problem variables and these are included in the control law of the system, which also applies the receding horizon strategy.

The receding horizon strategy (see Figure 2.5) is another key of this type of controllers. As previously stated, the future control signals are calculated using the expected outputs across

the prediction horizon, but only the first control signal is used.

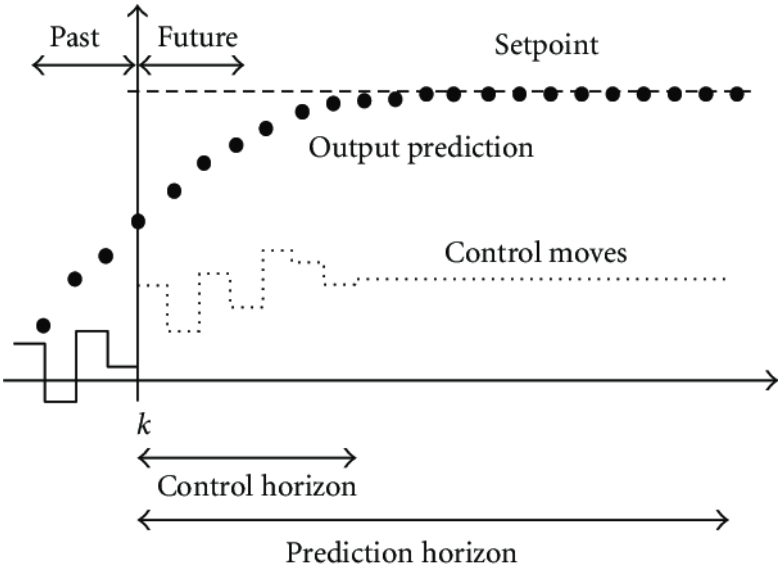

Figure 2.5: Receding horizon strategy

The most popular MPC types include Dynamic Matrix Control, Generalized Predictive Control or Non Linear Model Predictive Control [5].

- DMC: This algorithm was the first Model Predictive Control with industrial applications and nowadays, it is included in a lot of software packages. It works with the first N values of the step response model given by:

$$y(t) = y_0 + \sum_{i=1}^{N} g_i \Delta u(t-1) = y_0 + G(z^{-1})(1-z^{-1})u(t)\Delta \tag{2.1}$$

  where $\Delta u(t) = u(t) - u(t-1)$ and g_i represent discrete output values for the step input. The disturbances are considered to be the difference between the measured output of the system and the estimated output by the model

$$\hat{n}(t|t) = y_m(t) - \hat{y}(t|t) \tag{2.2}$$

This method also supports the addition of constraints which is the reason why it is widely used in industry. Then, a generic optimization problem is solved as the one in Figure 2.3.

$$J = \sum_{i=1}^{P} \delta[y(t+i) - \omega(t+i)]^2 + \sum_{i=1}^{N} \lambda[\Delta u(t+i-1)]^2] \tag{2.3}$$

This technique has the benefit of requiring no prior knowledge of the process, simplifying the identification process while also allowing complex dynamics to be simply described.

- GPC: This algorithm is based on the use of a CARIMA model.

$$A(z^{-1})y(t) = B(z^{-1}z^{-d}u(t-1) + C(z^{-1}\frac{e(t)}{\Delta} \tag{2.4}$$

An efficient recursive algorithm can solve a Diophatine equation in order to find the derivation of the optimal prediction. This is easy to implement using an online estimation algorithm such as recursive least squares. GPC uses a quadratic cost function of the form:

$$J(N_1, N_2, N_u) = \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - \omega(t+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2 \tag{2.5}$$

where the weighting sequences $\delta(j)$ and $\lambda(j)$ are commonly selected constant and the reference trajectory $\omega(t+j)$ can be generated by a simple recursion which starts at the current output and tends exponentially to the setpoint.

- NMPC: Until now different linear MPC have been explained, being these methods much more mature since they have been implemented in industry for more than fifty years. The reason for doing so is that is not always possible to compute the solution to a nonlinear problem sufficiently fast, although system are mostly nonlinear. The dynamic models of these systems are often obtained applying mass, momentum and energy balances to the process resulting in models with the general form:

$$\begin{aligned} \dot{x} &= f(x,u) \\ 0 &= g(x,u) \\ y &= h(x,u) \end{aligned} \tag{2.6}$$

where x is the vector of state variables, u is the vector of input variables and y is the vector of output variables. The following simplified optimum control problem in discrete-time based on this dynamic system form is considered [6]:

$$\begin{array}{lllll} \underset{x,z,u}{minimize} & \sum_{j=N_1}^{N_2} L_i(x_i, z_i, u_i) & + & E(x_N) & \\ \text{subject to} & x_0 - \hat{x_0} & = & 0, & \\ & x_{i+1} - f_i(x_i, z_i, u_i) & = & 0, & i = 0, \ldots, N-1 \\ & g_i(x_i, z_i, u_i) & = & 0 & i = 0, \ldots, N-1 \\ & h_i(x_i, z_i, u_i) & \leq & 0 & i = 0, \ldots, N-1 \\ & r(x_N) & \leq & 0 & \end{array} \tag{2.7}$$

where $x = (x_0^T, x_1^T, \ldots, x_{N-1}^T, x_N^T)^T$ is the state vector, $z = (z_0^T, z_1^T, \ldots, z_{N-1}^T)^T$ is the algebraic vector and $u = (u_0^T, u_1^T, \ldots, u_{N-1}^T)^T$ is the control vector.

In the recent years, technology has improved enough to accurately implement NMPC solutions. A interesting advantage of this approach is that it allows the operating system to work closer to the acceptable working region's edge which also permit to reach greater product quality requirements and productivity needs.

There are several algorithms for solving the NMPC problem. The one that is studied in this thesis is the RTI scheme, that aims to bridge the gap between LMPC and NMPC and will be explained in Section 4.

2.3 Real Time Iteration scheme

In the previous section, it is mentioned that the problem with MPC is the computational power and time spent to perform the control which aggravates with nonlinear model systems, as more complex algorithms need to be deployed. The RTI scheme pretends to solve this problem by performing an online optimization and taking into account that the solutions at adjacent time periods are closely related, i.e., they are very similar. This scheme has been correctly implemented in [7],[8],[9] with good performances which proves the feasibility of these controllers in real life applications.

In the RTI algorithm, a Newton type optimization is carried out, which begins with an initial guess and creates a sequence of iterates that each solve a linearization of the previous system (at the previous iterate). To find a optimal solution to the problem, the Karush-Kuhn-Tucker conditions have to be satisfied [6]. This means that there are multiplier vector λ^* and ν^* that satisfy:

$$\begin{aligned} \nabla_X \mathcal{L}(X^*, \lambda^*, \mu^*) &= 0 \\ G(X^*) &= 0 \\ 0 \geq H(X^*) \perp \mu^* &\geq 0 \end{aligned} \tag{2.8}$$

where:

$$\mathcal{L}(X, \lambda, \mu) = F(X) + G(X)^T \lambda + H(X)^T \mu \tag{2.9}$$

The RTI scheme is based on the SQP approach. Specifically, the constrained Gauss-Newton method is used, and this is supported on approximations of the Hessian. This method can be used when the cost function is a sum of squares:

$$F(X) = \frac{1}{2} \|R(X)\|_2^2 \tag{2.10}$$

where the Hessian is approximated by:

$$A_k = \nabla R(X^k) \nabla R(X^k)^T \tag{2.11}$$

and the QP objective is:

$$F_{QP}^k(X) = \frac{1}{2} \left\| R(X^k) + \nabla R(X^k)^T (X - X^k) \right\|_2^2 \tag{2.12}$$

This scheme executes one SQP iteration with the Gauss-Newton Hessian per sampling time. Each iteration is divided in a preparation phase, where the system is linearized and the QP formed and a feedback phase, where the condensed QP is solved. The preparation phase takes a longer time to be performed, being several times higher that the time of the feedback phase.

Chapter 3

Case study

In this chapter, the different models of the vehicle used are presented. In Section 3.1, the modeling for control is explained. Section 3.2 presents the high precision model used in the simulation to represent the real vehicle dynamics. In Section 3.3, the model used to obtain the states is described.

3.1 Control model

As explained in Section 2.2, a model of the system to be controlled is needed in order to predict future outputs. In order to obtain this and the rest of the models of the following sections, a coordinate system placed in the COG of the vehicle is defined, with the x axis pointing in front of the car, the y axis pointing to the left of the car and the z axis pointing upwards.

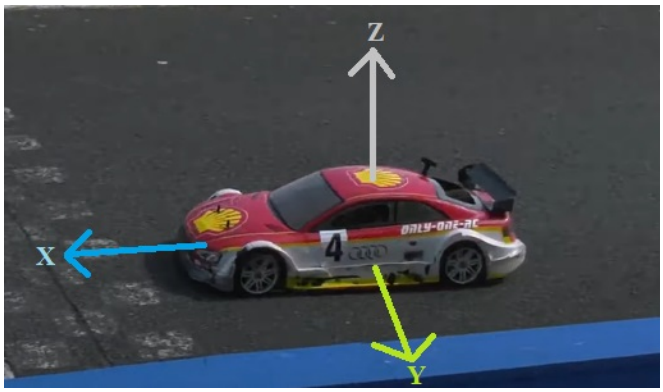

Figure 3.1: Body frame coordinate system

The vehicle model used is based on a bicycle model [10]. In this model, the car is represented as a bicycle where the front wheel is placed in the center of the front axis of the vehicle and the rear wheel in the center of the rear axis and express the velocity dynamics, which takes into account the forces applied to the vehicle (see Figure 3.2).

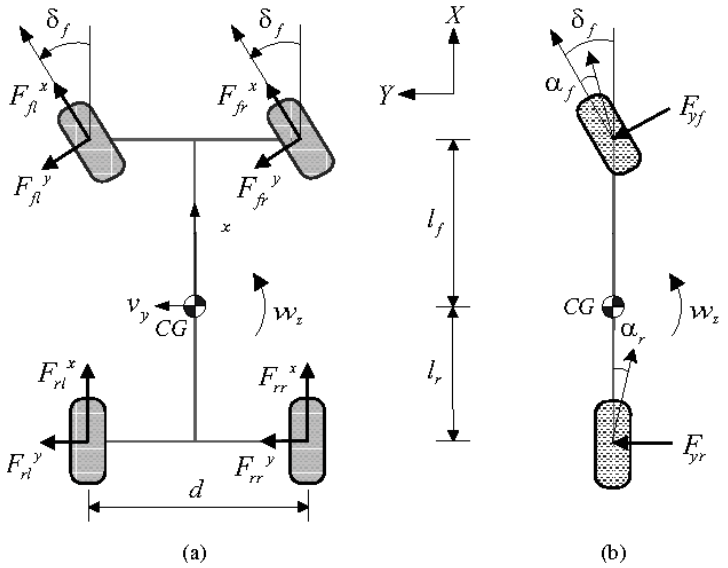

Figure 3.2: Bycicle model of the vehicle

It is also important to take into account that this is a nonlinear model, that as is presented in Section 2.2, requires advanced controllers if a low computational time is pursued. The equations that govern this model are the following:

$$\begin{aligned} \dot{v_x} &= a + \frac{-F_{yf}\sin\delta - \mu m g}{m} + \omega_z v_y \\ \dot{v_y} &= \frac{F_{yf}\cos\delta + F_{yr}}{m} - \omega_z v_x \\ \dot{\omega_z} &= \frac{F_{yf} l_f \cos\delta - F_{yr} l_r}{I} \end{aligned} \tag{3.1}$$

where:

- v_x, v_y are the linear velocities of the vehicle (m/s).
- ω_z is the angular velocity of the vehicle (rad/s).

- a is the acceleration of the rear axis of the vehicle (m^2/s).
- δ is the steering of the front axis (rad).
- F_{yf} is the lateral force applied to the front wheel (N).
- F_{yr} is the lateral force applied to the rear wheel(N).

The lateral tire forces are obtained with the next equations:

$$\begin{aligned} F_{yf} &= C_f \alpha_f \\ F_{yr} &= C_r \alpha_r \\ a_f &= \delta - \arctan\left(\frac{v_y + l_f \omega_z}{v_x}\right) \\ a_r &= \arctan\left(\frac{l_r \omega_z - v_y}{v_x}\right) \end{aligned} \tag{3.2}$$

The rest of the parameters expressed in the equations of the model are described in the Table 3.1.

Parameter	Description	Value
μ	Friction coefficient	0.05
m	Mass of the vehicle	$1.98kg$
g	Gravity acceleration	$9.81m/s^2$
I	Vehicle Inertia	$0.03kgm^2$
l_f	Distance COG-front axis	$0.125m$
l_r	Distance COG-rear axis	$0.125m$
C_f	Front wheel tire stiffness coefficient	$68\frac{N}{rad}$
C_r	Rear wheel tire stiffness coefficient	$71\frac{N}{rad}$

Table 3.1: Model parameters

3.2 Vehicle Simulation Model

In order to test the control designed, it will be implemented in a simulation environment using ROS, as will be explained in Section 5. To achieve an accurate result, similar to the one obtained with a real vehicle, it is necessary a model whose dynamics are close to those of the real vehicle.

For this purpose, a high fidelity model of the vehicle based on the Magic Formula [11] is used, which uses mathematical functions that link the lateral force to lateral slip, the longitudinal force to longitudinal slip, and the aligning moment to lateral slip. This way, the interaction between the tire and the road pavement is accurately described and allows to represent more precisely the vehicle dynamics. With this model, an outcome closer to reality is achieved.

The equations that define this model are the following:

$$\begin{aligned}
\dot{v_x} &= a + \frac{-F_{yf}\sin\delta - \mu m g}{m} + \omega_z v_y \\
\dot{v_y} &= \frac{F_{yf}\cos\delta + F_{yr}}{m} - \omega_z v_x \\
\dot{\omega_z} &= \frac{F_{yf} l_f \cos\delta - F_{yr} l_r}{I} \\
\dot{x} &= \cos\sigma v_x - \sin\sigma v_y \\
\dot{y} &= \sin\sigma v_x + \cos\sigma v_y \\
\dot{\Theta} &= \omega \\
F_{yF} &= d\sin(c\arctan(b\alpha_f)) \\
F_{yR} &= d\sin(c\arctan(b\alpha_r)) \\
\alpha_f &= \delta - \arctan(\frac{v_y}{v_x} - \frac{l_f\omega}{v_x}) \\
\alpha_r &= \delta - \arctan(\frac{v_y}{v_x} - \frac{l_f\omega}{v_x}) \\
F_f &= \mu m g
\end{aligned} \tag{3.3}$$

3.3 Observer model

For the purpose of implementing the RTI controller, it is important to know the values of the state variables that can not be directly measured. To achieve this goal, an state estimator based on a extended Kalman filter with LPV formulation is used (the code used for the state observer has been developed based on [12]).

It is used an LPV observer in the form:

$$\begin{aligned} \hat{x}_{k+1} &= A(\phi)\hat{x}_k + B(\phi)u_k + w_k + L(\phi)(y_k - \hat{y}_k) \\ \hat{y}_k &= C\hat{x}_k + v_k \end{aligned} \tag{3.4}$$

In which the observed states are obtained as:

$$\hat{x}_{k+1} = (A(\phi) - L(\phi)C)\hat{x}_k + B(\phi)u_k + Ly_k \tag{3.5}$$

The observer gain is designed as in [13], through a polytopic approximation:

$$L(\phi) = \sum_{i=1}^{2^N} \mu_i(\phi) L_i \tag{3.6}$$

where μ_i is given by:

$$\begin{aligned} \mu_i(\phi) &= \prod_{j=1}^{N} \xi(\alpha_j, \beta_j) \\ \alpha_j &= \frac{\bar{\phi_j} - \phi_j(k)}{\bar{\phi_j} - \underline{\phi_j}} \\ \beta_j &= 1 - \alpha_j \end{aligned} \tag{3.7}$$

And L_i is computed by seeking a Y and W_i that satisfy the following LMI and applying $L_i = Y^{-1}W_i$:

$$\begin{bmatrix} YA_i + A_i^T Y - W_i C - C^T W_i^T + Y2\lambda & Y(Q^{1/2})^T & W_i \\ Q^{1/2}Y & -I & 0 \\ W_i^T & 0 & -R^{-1} \end{bmatrix} < 0 \tag{3.8}$$

$$\begin{bmatrix} \gamma I & I \\ I & Y \end{bmatrix} > 0 \qquad i = 1, \ldots, 2^{m_S v}$$

Chapter 4

RTI Scheme

The main target of this master's thesis is to design and implement a RTI scheme in a vehicle, as a strategy of control for nonlinear models, which can address the nonlinear constraints and dynamics in an explicit and direct manner, rather than utilising linear approximations.

The RTI scheme used in this thesis is a Sequential Quadratic Programming approach [14], which involves performing quadratic problems sequentially towards the solution of the problem, and is split in two different phases. On one hand, the preparation phase, that will be explained in Section 4.1 and goes before having a new state estimate. On the other hand, the feedback phase, explained in Section 4.2, that is performed after obtaining this new state estimate. The advantages in using two different phases is that the computations performed in the preparation phase does not require the state estimate, which allows to reduce the total time spent in performing the algorithm.

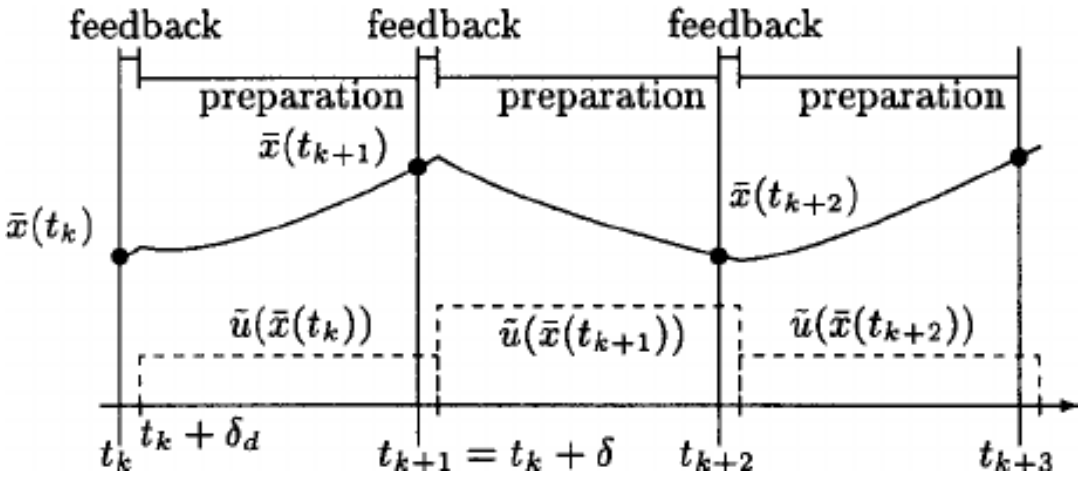

Figure 4.1: Time elapsed in each phase of the algorithm

The total computation time T_t spent until a new control input is obtained is equal to the sum of the time spent in the preparation time t_{pp} and the time spent in the feedback phase t_{fp} which limits the overall sample time (it can not be lower than the total computation time). This is:

$$T_t = t_{pp} + t_{fp}$$

Besides these two different phases of the algorithm and before the first iteration of the algorithm, a series of computations are performed in order to prepare the controller. These computations follow the methods applied in the preparation phase of the algorithm but exploiting the fact that the references are known instead of using the previous NMPC solution, since this solution is not available.

4.1 Preparation phase

This phase as its name suggests is the one that prepares the QP problem and is the one that takes the most time since it performs the shifting procedure, the online linearization and the forming of the QP. As this algorithm performs steps towards the solution of the QP starting from an available guess, it is needed to define these guessed values.

One option is to use the references (as they are known) but this guess can be inaccurate which is the reason why this solution was discarded. Instead, they will be obtained by shifting the previous result of the NMPC in a shifting procedure, that supposes that a fairly accurate

solution to the problem was achieved in the previous instant. As the dynamic model is used the state variables are $x = (v_x, v_y, w_z)$ and the control inputs are $u = (\delta, a)$. The shifted procedure is applied as follows:

$$\begin{aligned} x_{i,k}^{guess} &= x_{i-1,k+1,} k = 0, ..., N-1 \\ u_{i,k}^{guess} &= u_{i-1,k+1,} k = 0, ..., N-2 \\ x_{i,N}^{guess} &= f(x_{i,N-1}^{guess}, u_{i,N-1}^{guess}) \end{aligned} \tag{4.1}$$

with N being the prediction horizon. These guesses are the starting point for the QP problem that will sequentially approximate towards the solution. It is also important to remark that the shifting procedure presupposes that the system's evolution follows closely the trajectory.

When the algorithm starts in the preparation phase, it is also needed to take into account that the NMPC solution for the previous time sample is unavailable. To solve this iteration, the guess is constructed exploiting the fact that the references are known, this way the rest of the algorithm can be performed equally to when the result is available.

After these guesses are obtained, the linearization of the control model is carried out and is the step costs the most computational time. In order to perform this step, a linearization based on the Taylor series expansion is used, retaining only the first derivative term, also called first order Taylor expansion [15]. This linearization is based on the fact that the behaviour of a nonlinear system in the neighbourhood of a given point, $x = x_0$, called operating point, is similar to that of a linear model in that point. As explained, this model will only be valid in a particular range surrounding the operating point, that is why the online approximation in the RTI scheme is necessary in order to obtain an accurate controller. The Taylor series expansion is shown below:

$$f(x_k, u_k) = f(x_{k-1}, u_{k-1}) + \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_0,u_0} (x_k - x_{k-1}) + \left.\frac{\partial f(x,u)}{\partial u}\right|_{x_0,u_0} (u_k - u_{k-1}) \tag{4.2}$$

where x_0, u_0 are the operating points.

With this method, the following linear model in the space state is obtained:

$$\begin{aligned} \Delta x_{i,k+1} &= A_{i,k}\Delta x_{i,k} + B_{i,k}\Delta u_{i,k} + r_{i,k} \\ \Delta y_{i,k+1} &= C_{i,k}\Delta x_{i,k} + D_{i,k}\Delta u_{i,k} + h(x_{i,k}^{ref}, u_{i,k}^{ref}) \end{aligned} \tag{4.3}$$

being:

$$\begin{aligned} r_{i,k} &= f(x_{i,k}^{ref}, u_{i,k}^{ref}) - x_{i,k+1}^{ref} \\ h_{i,k} &= h(x_{i,k}^{guess}, x_{i,k}^{guess}) \end{aligned} \tag{4.4}$$

and:

$$\begin{aligned} A_{i,k} &= \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_{i,k}^{guess}, u_{i,k}^{guess}} \\ B_{i,k} &= \left.\frac{\partial f(x,u)}{\partial u}\right|_{x_{i,k}^{guess}, u_{i,k}^{guess}} \\ C_{i,k} &= \left.\frac{\partial h(x,u)}{\partial x}\right|_{x_{i,k}^{guess}, u_{i,k}^{guess}} \\ D_{i,k} &= \left.\frac{\partial h(x,u)}{\partial u}\right|_{x_{i,k}^{guess}, u_{i,k}^{guess}} \end{aligned} \tag{4.5}$$

Employing this method with the system presented in Section 3.1, the following system matrices are obtained:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & 1 \\ B_{21} & 0 \\ B_{31} & 0 \end{bmatrix} \tag{4.6}$$

where:

$$\begin{aligned} A_{11} &= -\mu \\ A_{12} &= \frac{\sin\delta C_f}{mv_x} \\ A_{13} &= \frac{\sin\delta C_f l_f}{(mv_x) + v_y} \\ A_{22} &= \frac{-C_r + C_f\cos\delta}{mv_x} \\ A_{23} &= \frac{-(l_f C_f \cos\delta) - l_r C_r}{mv_x - v_x} \\ A_{32} &= \frac{-(l_f C_f \cos\delta) - l_r C_r}{Iv_x} \\ A_{33} &= \frac{-(l_f l_f C_f \cos\delta) + l_r l_r C_r}{Iv_x} \end{aligned} \qquad \begin{aligned} B_{11} &= \frac{-C_f \sin\delta}{m} \\ B_{21} &= \frac{C_f \cos\delta}{m} \\ B_{31} &= \frac{l_f C_f \cos\delta}{m} \end{aligned} \tag{4.7}$$

4.2 Feedback phase

The SQP procedure can be initiated after a new state estimate is obtained. In this phase, the solution to the QP is computed and the control inputs of the system at hand are obtained. In this way, the problem is successively approximated towards the solution until reaching convergence taking Newton steps to obtain $\Delta x_i, \Delta u_i$.

The number of optimization variables depends on the prediction horizon N, being $5N$ optimization variables (three states and two inputs in each sample time). The QP defined for this NMPC is shown in Equation 4.8.

$$\begin{aligned} \arg\min_{\Delta x, \Delta u} \quad & \sum_{k=0}^{N-1} \tfrac{1}{2} \begin{bmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{bmatrix} H_{i,k} \begin{bmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{bmatrix} + J_{i,k} \begin{bmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{bmatrix}^T \\ \text{s.t.} \quad & \Delta x_{i,0} = \hat{x}_i - x_{i,0}^{guess}, \\ & \Delta x_{i,k+1} = A_{i,k} \Delta x_{i,k} + B_{i,k} \Delta u_{i,k} + r_{i,k}, \\ & C_{i,k} \Delta x_{i,k} + D_{i,k} \Delta u_{i,k} + h_{i,k} \leq 0 \end{aligned} \tag{4.8}$$

Finally, the full Newton Step (see Eq. 4.9) is applied in order to get the control inputs applied by the RTI Controller to the system:

$$(x_i, u_i) \leftarrow (x_i^{guess}, u_i^{guess}) + (\Delta x_i, \Delta u_i) \tag{4.9}$$

4.3 Design of the algorithm

To design and first testing the RTI scheme the software Matrix Laboratory (MATLAB) was used together with Simulink and MATLAB's toolbox YALMIP. There are several reasons why these software were used:

1. Easiness to work with matrices and big databases.
2. Graphic plotting options to visualize the results of the implementation of the controller
3. Possibility to simulate the control system with Simulink.
4. Yalmip toolbox makes the prototyping of optimization problems fast and reliable.

In order to design the controller, the references of the path to be followed by the vehicle have been provided by a trajectory planner, including both states and inputs references. Also a

Variable limits			
$\bar{x}_1$	1.2 m/s	$\bar{u}_1$	0.267 rad
$\underline{x}_1$	0.1 m/s	$\underline{u}_1$	-0.267 rad
$\bar{x}_2$	0.08 m/s	$\bar{u}_2$	0.6 m/s^2
$\underline{x}_2$	-0.02 m/s	$\underline{u}_2$	0 m/s^2
$\bar{x}_3$	1 rad/s		
$\underline{x}_3$	-1 rad/s		

Table 4.1: States and inputs constraints

series of constraints have been established, which can be seen in the Table 4.3.

In order to analyse the evolution of the system under the control inputs provided by the controller, the high fidelity model explained in Section 3.2 is used. For the purpose of plotting the trajectory of the vehicle, the simulink model in Figure 4.2 is used to obtain the x and y coordinates of the system.

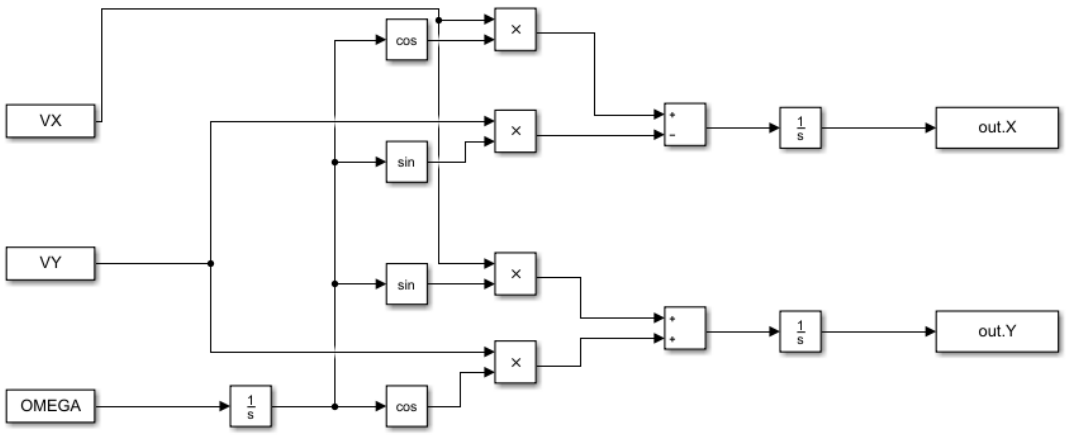

Figure 4.2: Simulink model of the system

In order to perform the first iteration of the control algorithm, since the previous NMPC solution is not available, it is necessary to initialize correctly the control. For this purpose, the first linearization use the references obtain with the trajectory planner, which provides the data needed to start with the feedback phase, and the state estimate is also taken from the reference values.

Once in the feedback phase, the references are updated and introduced to the YALMIP solver, which obtains the optimization variables $\Delta x_i, \Delta u_i$. Then, as explained in the previous section, the full Newton step that gives the values that minimize the objective function are obtained as the sum of these solutions provided by the solver and the guessed values for the problem. These control inputs are then applied to the nonlinear model used to simulate the physical system.

With the feedback phase finished, the preparation phase starts by shifting the previous data obtained with the controller algorithm and then the linearization takes place and the sensitivities are calculated, which allows to continue with the next iteration of the algorithm.

Once the maximum iterations are achieved (when the control arrives to the last reference), the controller stops, and the plotting of the results of the simulation occur, allowing to observe the behaviour of the system with the RTI controller.

Algorithm 4.3 summarizes the workflow of the controller implementation.

Algorithm 1 RTI algorithm at discrete time i

Preparation phase performed in t_{pp}

NMPC solution and reference from the previous instant (x_{i-1}, u_{i-1}),(x_i^{ref}, u_i^{ref})

1: Apply shifting procedure to (x_{i-1}, u_{i-1}) to construct $(x_i^{guess}, u_i^{guess})$.
2: Linealize the system to obtain $A_{i,k}, B_{i,k}, C_{i,k}, D_{i,k}, r_{i,k}, h_{i,k}$
3: Construct the QP omitting the state estimate $\hat{x}_i$ and prepare all possible computations.

QP prepared

Feedback phase performed in time t_{fp} when the state estimate $\hat{x}_i$ is computed.

$\hat{x}_i$, QP from preparation phase

4: Introduce $\hat{x}_i$ into the QP and solve it to obtain $\Delta x_i, \Delta u_i$.
5: Apply the full Newton step. $(x_i, u_i) \leftarrow (x_i^{guess}, u_i^{guess}) + (\Delta x_i, \Delta u_i)$

NMPC solution (x_i, u_i)

Chapter 5

Implementation in ROS

In this section, the implementation of the designed control algorithm is presented. ROS is used in this matter for its characteristics. ROS is a robotic middleware that consists in a set of libraries and tools that helps to create robust robot behaviour in a wide variety of situations. One of the main characteristics that enforces its use in this project is that, as it consists of modular blocks of program, with little effort it can be used in a real mobile robot. The performance of the control algorithm in this framework with less ideal conditions allows to obtain results closer to reality that those obtained in Matlab.

The modularity is the main reason for using ROS as the implementation tool in a simulated environment, together with the simulation of the vehicle provided by the BARC project that allows to obtain results closer to the real system.

Section 5.1 gives an overview of the ROS framework and structure used in this project. After this, the implementation of the controller is explained in 5.2, while the simulation used is detailed in section 5.3 and the tools used to plot and express the results are presented in Section 5.4.

5.1 ROS framework

This section shows the structure of the desidgned controller in the ROS framework. For this purpose, a series of concepts are presented and defined.

ROS is a topic-based system where the communication between the different nodes follows

a publisher-subscriber pattern. This means that an object (called subscriber) subscribes to a particular topic and receive the information via messages, while a sender of messages publishes to a topic, not knowing who is subscribed to that topic. The way this communication system works in the ROS framework is shown in Figure 5.1:

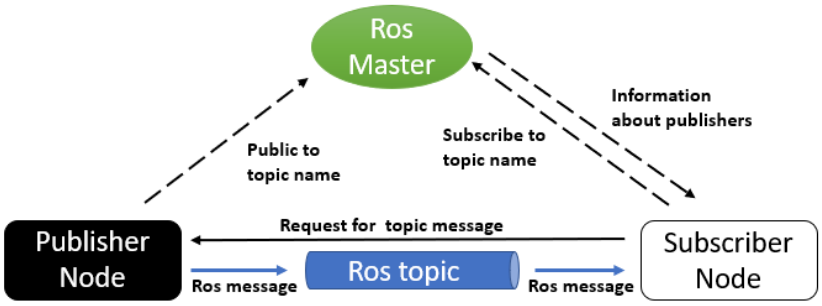

Figure 5.1: Communication in the ROS framework

Nodes: The nodes are processes that perform computations and can be executed simultaneously. For this master's thesis, the following nodes have been implemented:

- *Control*: This node performs the RTI scheme designed, including the optimization of the cost function, exchanging information with the state estimator and providing the control inputs to the simulator of the vehicle.
- *Simulator*: It represents the systems to be controlled and consists in a high fidelity model of the vehicle as it has been explained in Section 3.2. It contains both the high-level control and the low-level control.
- State estimator: This node performs the computations to obtain a new state estimation in each iteration (v_x, v_u, w_z), allowing the feedback phase to start.
- Visualization: It is used to perform all the plotting related to the results of the implementation which is used to observe and interpret the achievements of the RTI-MPC controller.

Messages: The messages are a ROS concept that allows the exchanging of information between the different nodes being executed. In this project, several messages have been used:

- *Control actions*: These are the control inputs of the system that are obtained with the developed algorithm. IT includes the motor signal (acceleration) and servo motor signal (steering). This message is sent from the control node to the simulator
- *Pos info*: This is a message to hold the estimated data with position and trajectory information that is sent to the control and visualization nodes.
- *Simulator states*: This message holds data from the real states computed by the simulator. These data substitute the one from the real vehicle as it is implemented in a simulated environment. The linear velocities are expressed in m/s and angular velocities in rad/s. This message is sent from the simulator both to the state estimator and visualization nodes.
- *Ecu*: This message contains the control inputs provided by the controller and is sent to the simulator. It includes a motor signal which controls the speed of the vehicle through an input torque and a servo signal which controls the steering angle. Both have units of PWM angle (deg) related to the duty cycle.

In Figure 5.2, the diagram with the nodes involved in the simulation and the messages exchanged between them are shown.

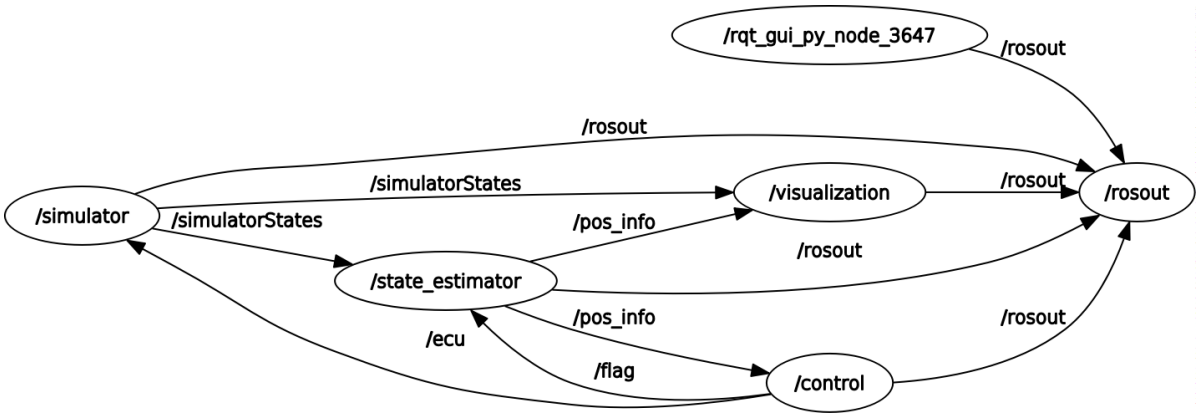

Figure 5.2: Node graph of the full implementation

5.2 Controller

Since the RTI controller has already been designed, the structure applied in ROS is inherited from Matlab. In Figure 5.3, an overview of the control scheme is illustrated.

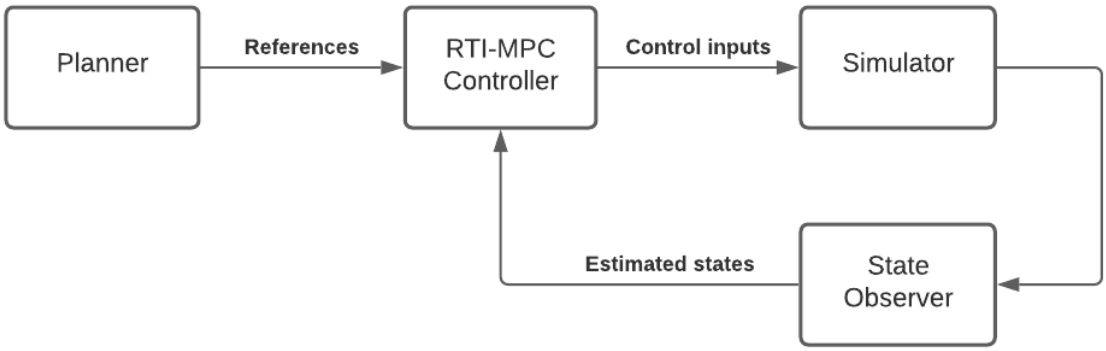

Figure 5.3: Overview of the control algorithm

The main differences in the implementation in this simulated environment are the state estimation and the QP solver.

The solver used in this implementation is the OSQP (Operator Splitting Quadratic Program) of OXFORD University [16]. This numerical optimization package allows to solve QP in the form:

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}x^t Px + q^t x \\ &\text{subject to } l \leq Ax \leq u \end{aligned}$$

where x is the optimization variable or variables and $P \in S_+^n$ is a positive definite matrix. Unlike the YALMIP optimization problem, the OSQP solver is not as explicit when it comes to express the constraints matrices and function cost. That is the reason why rearranging these matrices into the ones that the solver takes for input becomes necessary.

The function cost is expressed as:

$$min \quad \frac{1}{2}x^t Px + q^T x \tag{5.1}$$

where matrices P and q are:

$$P = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_N \\ 0 & 0 & \cdots & R_N \end{bmatrix} \qquad q = P * \begin{pmatrix} x_{guess} - ref_x \\ u_{guess} - ref_u \end{pmatrix} \tag{5.2}$$

The equality constraint matrices are defined by the physics of the model used, that is, the equations that rule the system. In OSQP, the equality constraint matrices are introduced as A and b in the following manner:

$$Ax = b \tag{5.3}$$

It is also necessary to rearrange the equations of the linearized model into those ones for the solver to work. Given the linearized system:

$$\begin{bmatrix} \mathrm{x}_{1,k+1} \\ \mathrm{x}_{2,k+1} \\ \mathrm{x}_{3,k+1} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} \mathrm{x}_{1,k} \\ \mathrm{x}_{2,k} \\ \mathrm{x}_{3,k} \end{bmatrix} + \begin{bmatrix} j & k \\ l & m \\ n & o \end{bmatrix} \begin{bmatrix} \mathrm{u}_{1,k} \\ \mathrm{u}_{2,k} \end{bmatrix}$$

The matrices A and b that need to be delivered to the solver are:

$$A = \underbrace{\begin{bmatrix} -j & -k & 1 & 0 & 0 & & & & & & & & \\ -l & -m & 0 & 1 & 0 & & & & & & & & \\ -n & -o & 0 & 0 & 1 & & & & & & & & \\ & & -a & -b & -c & -j & -k & 1 & 0 & 0 & & & \\ & & -d & -e & -f & -l & -m & 0 & 1 & 0 & & & \\ & & -g & -h & -i & -n & -o & 0 & 0 & 1 & & & \\ & & & & & & & \vdots & & \ddots & & & \\ & & & & & & & & & & & & \\ & & & & & & & -a & -b & -c & -j & -k & 1 & 0 & 0 \\ & & & & & & & -d & -e & -f & -l & -m & 0 & 1 & 0 \\ & & & & & & & -g & -h & -i & -n & -o & 0 & 0 & 1 \end{bmatrix}}_{3N \times 5N} \tag{5.4}$$

$$B = \underbrace{\begin{bmatrix} ax_1 + b_x 2 + cx_3 + r_1 \\ dx_1 + ex_2 + fx_3 + r_2 \\ gx_1 + hx_2 + ix_3 + r_3 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{3N \times 1} \tag{5.5}$$

The inequality constraint matrices are the bounds to which the states and input signals of the system are limited. In OSQP, those matrices are expressed as:

$$Gx \leq h \tag{5.6}$$

The matrices G and h that need to be delivered to the solver are:

$$G = \underbrace{\begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}}_{10 \times 5} \qquad h = \underbrace{\begin{bmatrix} \bar{x_1} - x_{10} \\ x_{10} - \underline{x_1} \\ \bar{x_2} - x_{20} \\ x_{20} - \underline{x_2} \\ \bar{x_3} - x_{30} \\ x_{30} - \underline{x_3} \\ \bar{u_1} - u_{10} \\ u_{10} - \underline{u_1} \\ \bar{u_2} - u_{20} \\ u_{20} - \underline{u_2} \end{bmatrix}}_{10 \times 1} \tag{5.7}$$

The state estimation is also differently implemented in ROS with respect to Matlab. In Matlab, the actual states of the simulator were saved without using a state estimator. This strategy could also be implemented in the ROS simulation, but in order to obtain an accurate result closer to reality, a state estimation is used.

The node *state_ estimator* performs these computations based on the model presented in Sec-

tion 3.3, and provides the control node with the estimated states.

In Algorithm 5.2, the implementation of the control in ROS is presented.

Algorithm 2 RTI algorithm at discrete time i

Offline settings:
Set references
Set upper and lower limits of the inputs and states
Set model parameters

1: Initialize references and QP
2: **while** $iteration < Lastiteration$ **do**
3: Update references from iteration to iteration+N
4: $\hat{x}_i \leftarrow state\_estimator$
5: Optimize (J)$\rightarrow \Delta x_i, \Delta u_i$
6: Apply full Newton step $(x_i, u_i) \leftarrow (x_i^{guess}, u_i^{guess}) + (\Delta x_i, \Delta u_i)$
7: $u_{1,1}, u_{1,2} \rightarrow Simulator$
8: Shift solution as explained in section 4.1
9: Linearize and obtain $A_{i,k}, B_{i,k}, C_{i,k}, D_{i,k}, r_{i,k}, h_{i,k}$
10: $iteration \leftarrow iteration + 1$
11: **end while**

5.3 Simulator

In this implementation, a simulator based in the model presented in Section 3.2 has been used. This simulator has been developed by the University of Berkeley as part of the BARC project[17].

This simulator operates at a rate of $200Hz$, higher than the control, in order to perform the simulation correctly. The simulator receives the control signals from the RTI-MPC controller and sends its position to the visualization node in order to observe its behaviour.

5.4 Visualization

A node to visualize in real time the trajectory of the robot provided by the RTI controller was performed. This node receives the estimated states from the *state_ estimator* node and the simulator states from the simulator node.

For this purpose, a map of the path to be followed by the robot has been implemented.

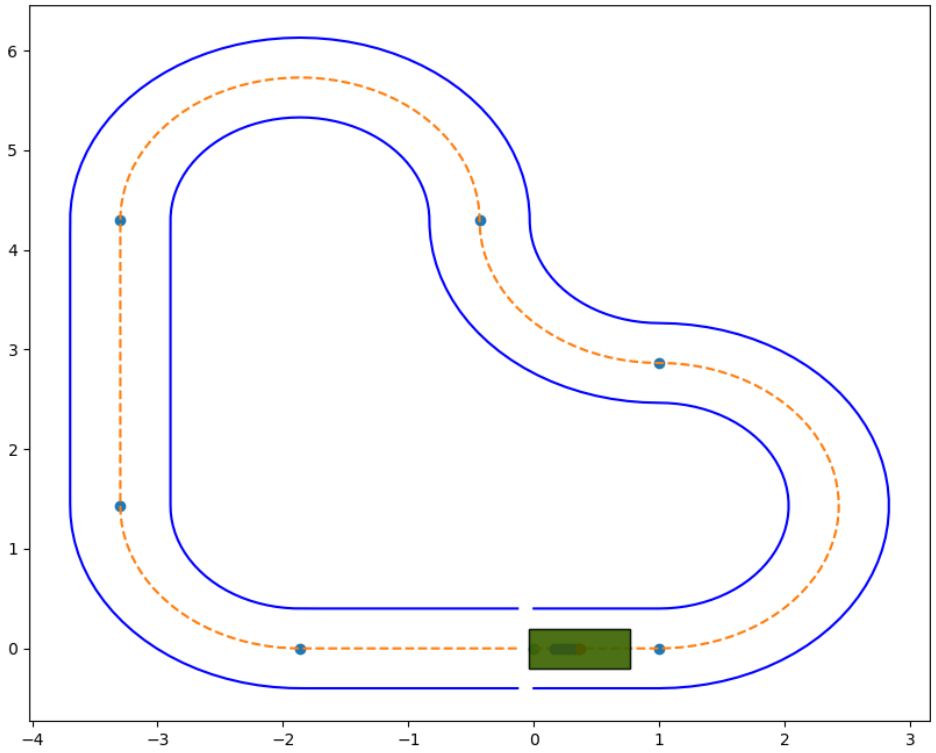

Figure 5.4: Map for results visualization

In this figure, the green rectangle represents the vehicle, the orange dashed line represents the path to be followed by the robot and the blue lines the limits of the track.

Chapter 6

Application results

This chapter is devoted to the analysis of the performance obtained with the proposed algorithms analysing the results. First, the results of the Matlab implementation are presented with different parameters in Section 6.1. Then, the results of the ROS implementation are shown in Section 6.2

6.1 Results in Matlab

In the first design of the algorithm, the RTI-MPC controller was performed using Matlab. A L-shaped track was used to firstly test the controller, with a prediction horizon of five time samples ($H_p = 5$), obtaining the result presented in Figure 6.1.

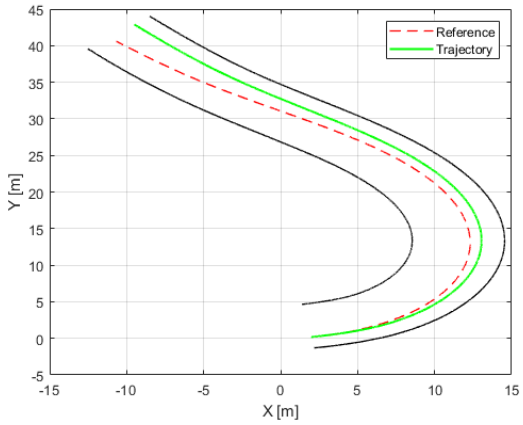

Figure 6.1: Followed trajectory in Matlab simulation

As it can be seen in this figure 6.1, the simulated car follows pretty accurately the reference

in the beginning and it only happens to differ from it after the curve. Despite from it, the robot keeps within the track in all the simulation.

In order to observe that the controllers provides the states accordingly within their bounds, a plot of the states throughout the simulation was computed.

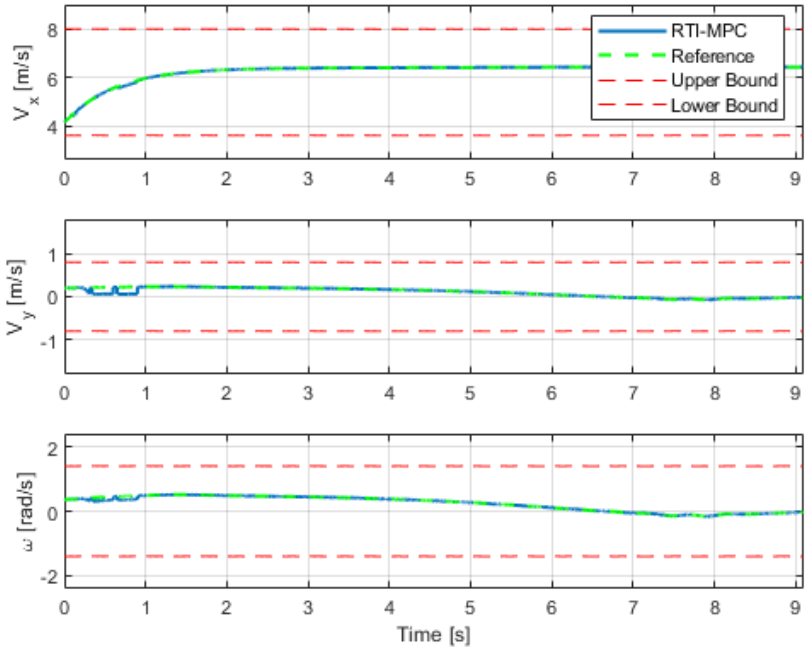

Figure 6.2: States in Matlab simulation

It can bee seen that the three states v_x, v_y, w_z stay within the selected bounds and that they follow faithfully the provided references. The RTI-MPC controller allowed also to constrain the inputs of the system, in which a positive result was also obtained. The control inputs provided by the controller were also plotted, observing that they were located within the limits in all the simulation (see Figure 6.3).

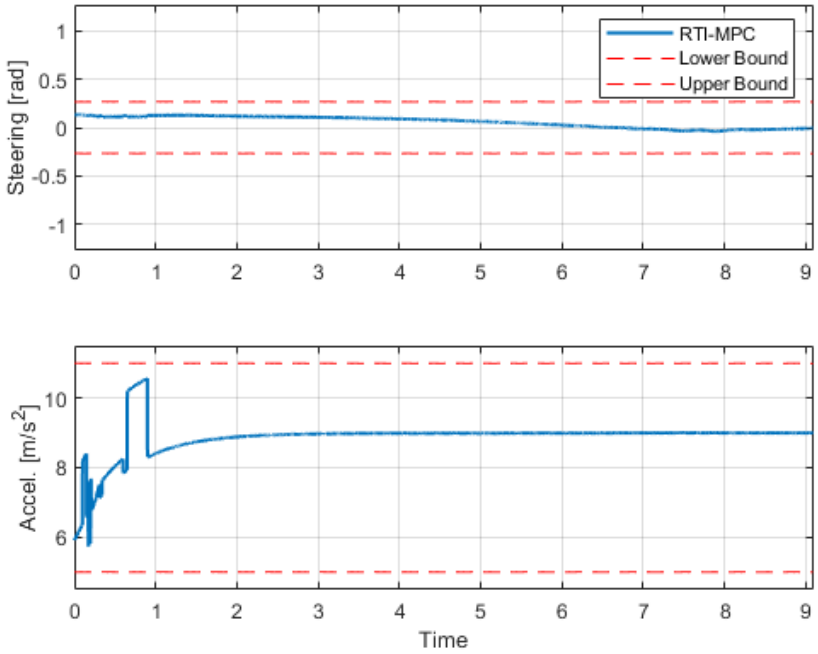

Figure 6.3: Inputs in Matlab simulation

In the Figure 6.4, the errors of the different states in the simulation are shown:

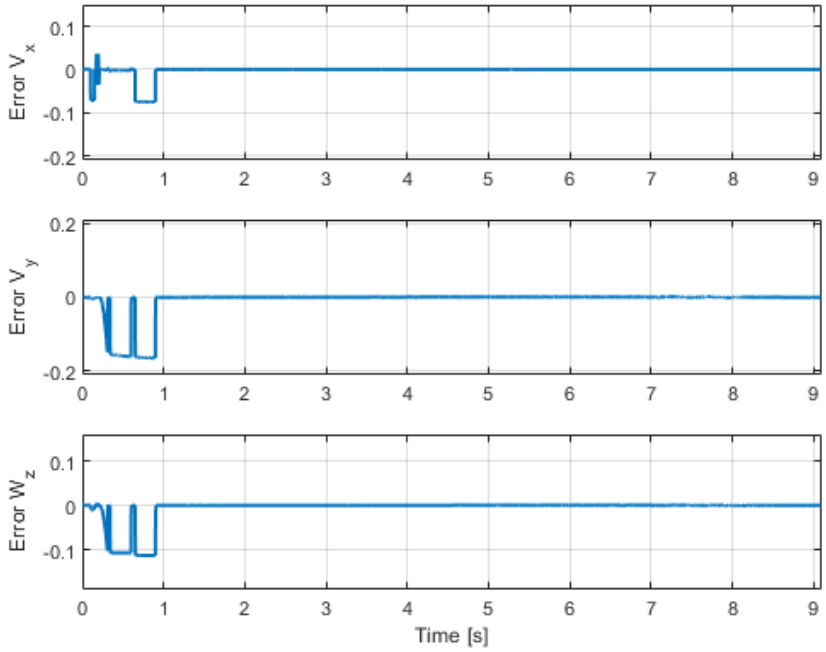

Figure 6.4: State errors in Matlab simulation

Finally, in Figure 6.5, a graphic with the computing time used in each iteration is shown, with almost all the iteration being faster that the sample time of the controller.

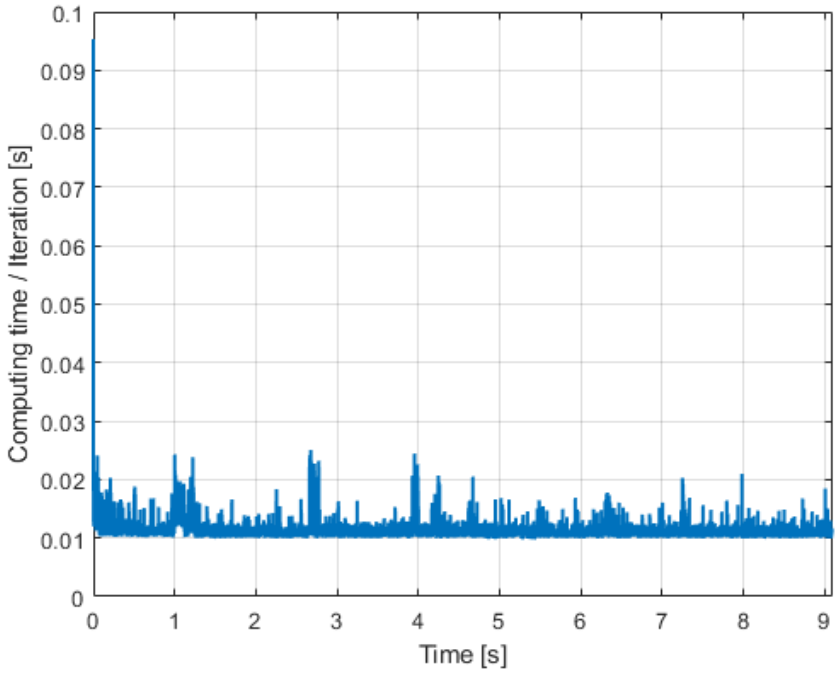

Figure 6.5: Computing time per iteration performed

After obtaining a good performance in the Matlab simulation, the controller was implemented in another simulation environment in ROS software, in order to provide a result more close to real implementation.

6.2 Results in ROS

In the implementation of the control algorithm in a simulated environment using ROS, a different and more complex track was used including several turns. It was also tested with a prediction horizon of five time samples ($H_p = 5$), obtaining the following result on the map:

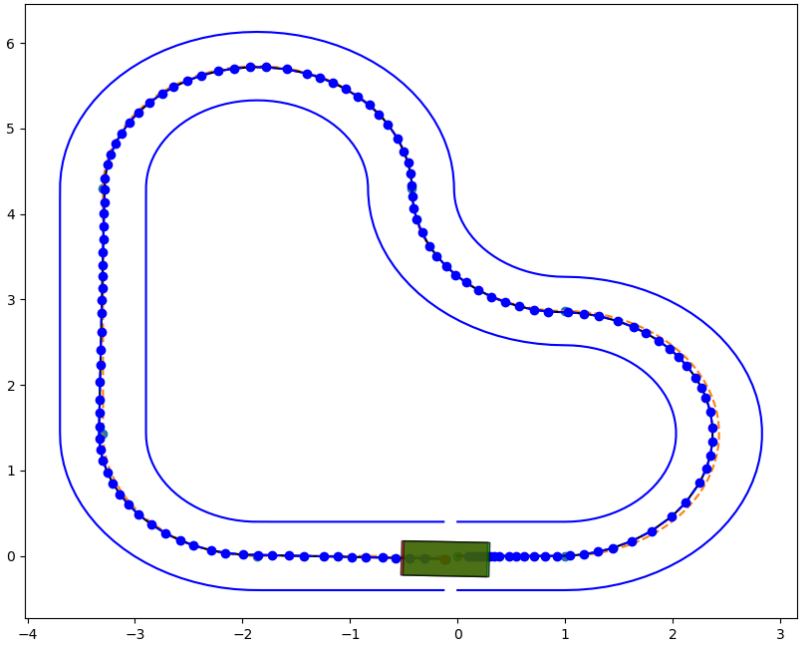

Figure 6.6: Implementation of control algorithm in ROS

It is shown that the vehicle follows the trajectory path accordingly to the references provided by the trajectory planner, proving that the designed RTI-MPC algorithm achieves its goal.

As in the Matlab simulation, the states were plotted in order to watch the trajectory tracking.

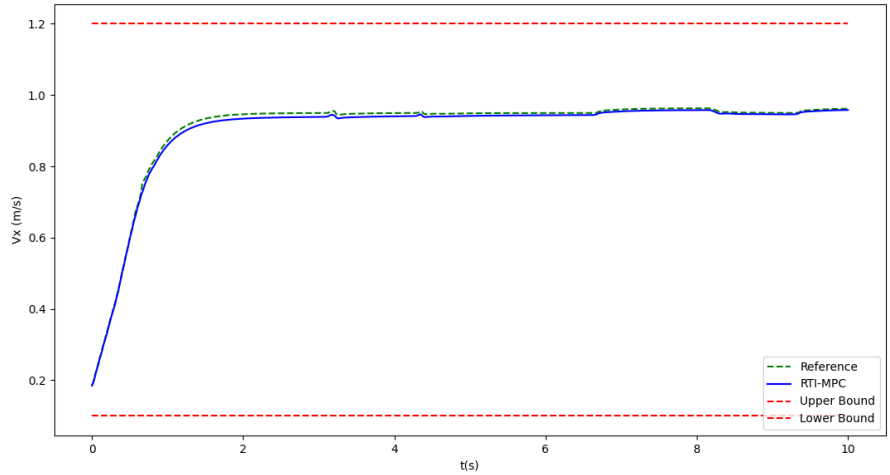

Figure 6.7: V_x state in ROS implementation

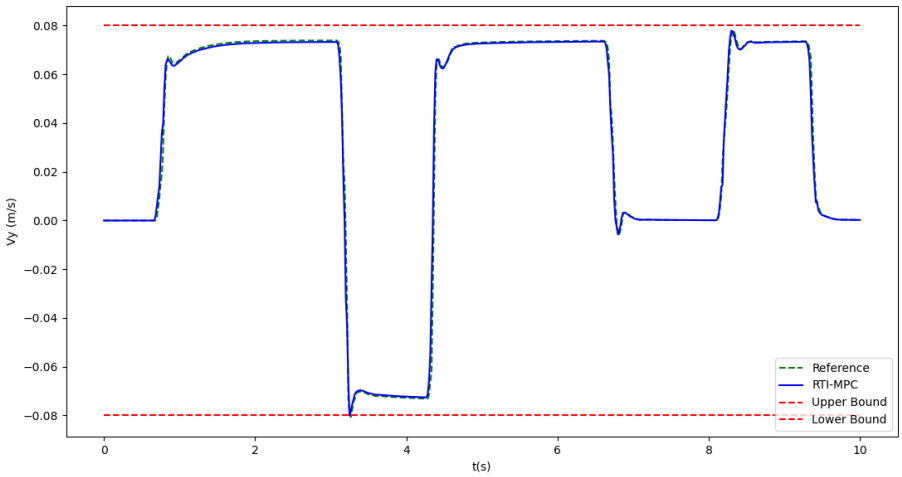

Figure 6.8: V_y state in ROS implementation

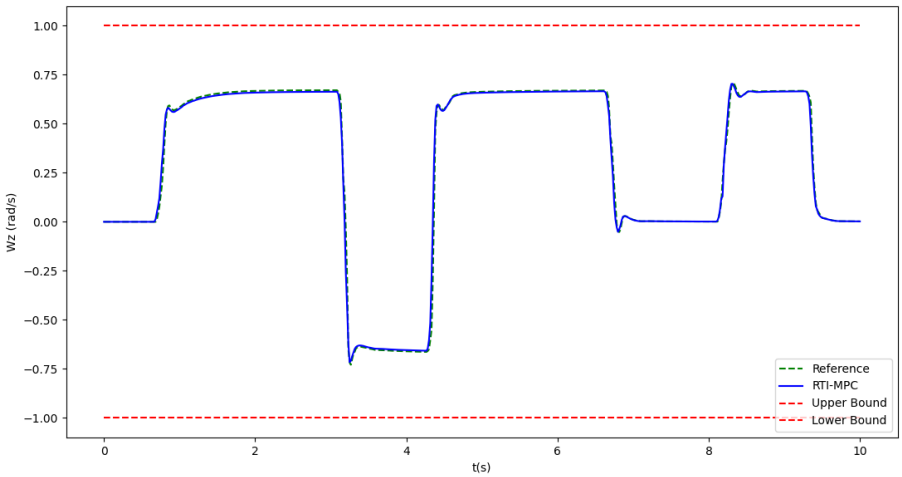

Figure 6.9: W_z state in ROS implementation

It is shown that the states follow closely the reference and it stays within the bound defined. Controller inputs were also plotted and showed equally positive results.

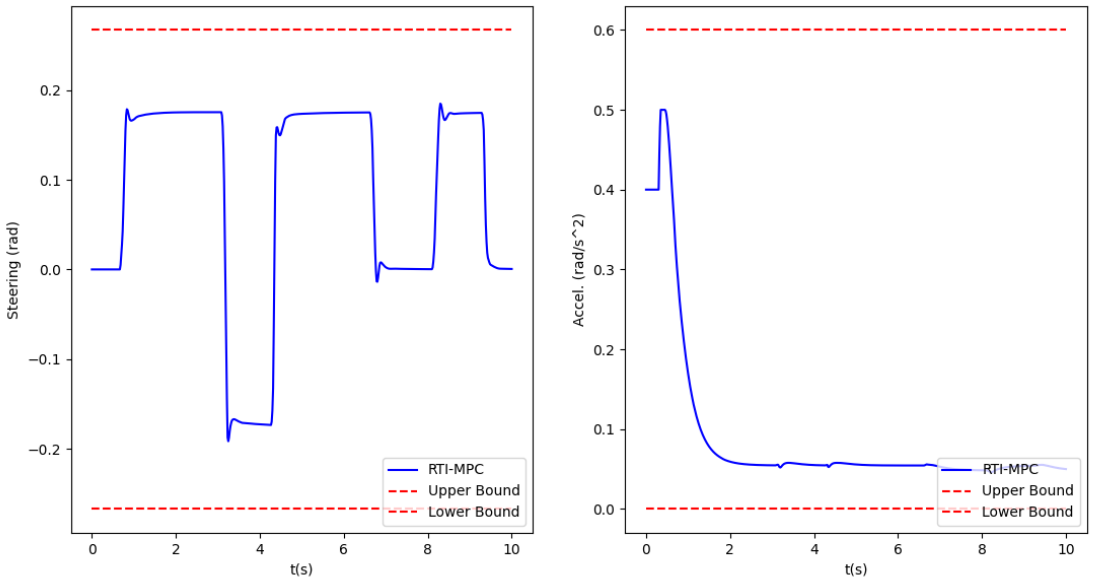

Figure 6.10: Controller inputs in ROS implementation

As in the Matlab simulation, the errors of the different states are shown in the following figures.

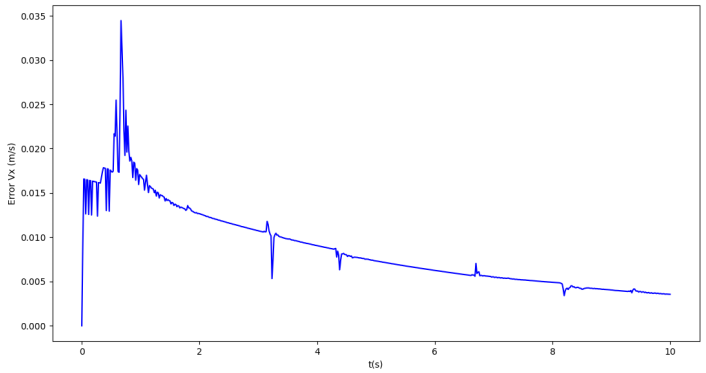

Figure 6.11: Error of V_x state throughout the implementation

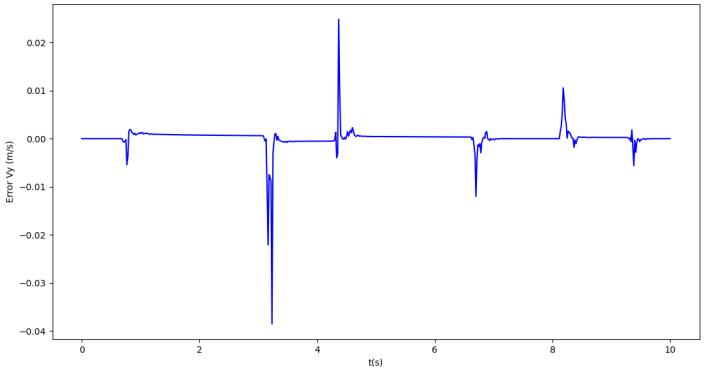

Figure 6.12: Error of V_y state throughout the implementation

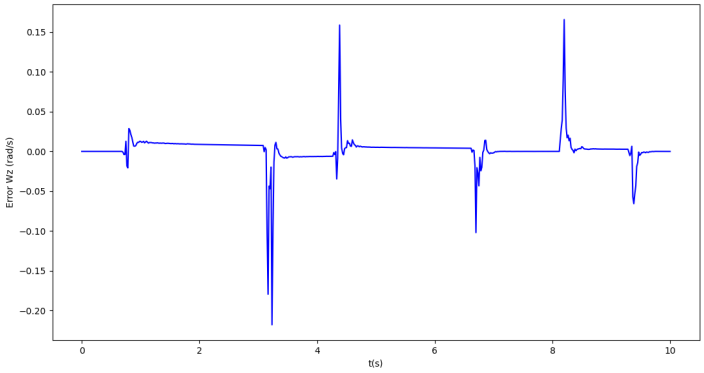

Figure 6.13: Error of W_z state throughout the implementation

Finally, the computing time per iteration is shown in Figure 6.14, where a similar result to the Matlab simulation was obtained, with each iteration performed in less than the sample time.

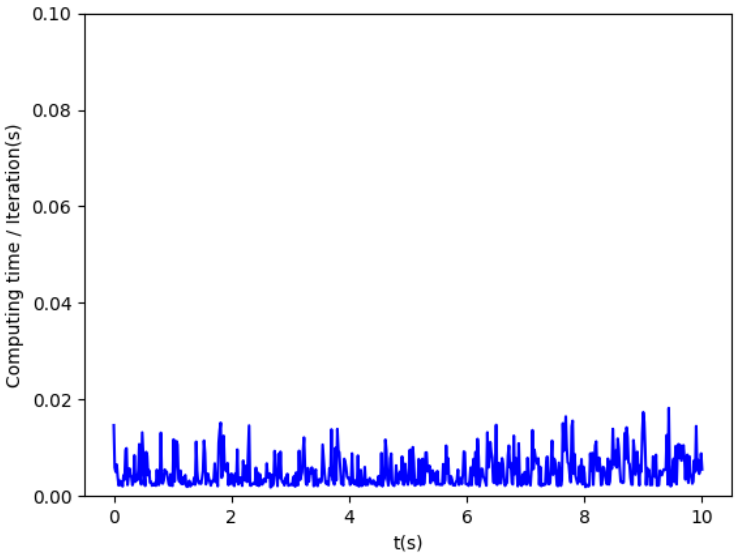

Figure 6.14: Computing time per iteration in ROS implementation

6.3 Discussion of the results

The obtained results show that the controller algorithm designed in the Matlab simulation

achieved a great performance in the ROS simulation, which is an environment closer to reality and proves that this type of controller is appropriate for the task performed in this master's thesis.

Chapter 7

Economic, social and environmental study

This chapter aims to provide an overview to incoming outcomes of the expansion of mobile robotics, autonomous driving and the control of these systems itself in the environment (Section 7.3, the society (Section 7.2 and the economy (Section 7.1) by analysing the situation at the present.

7.1 Economic impact

There is no doubt that autonomic robotic systems are having a huge impact in the way the economy works taking part in the new industry 4.0. In the last decade, all types of robotic systems have been developed in order to increase productivity in factories, reduce the running time and cutting down on costs and expenses, which increase the economic growth and potential of the companies.

New advanced controls also allow to work in more strict conditions which is translated in shorter cycle times and cost savings. In Figure 7.1, a study [18] shows the potential economic impact of different disruptive technologies, being autonomous vehicle and advanced robotics two of them with significant impact on the economy:

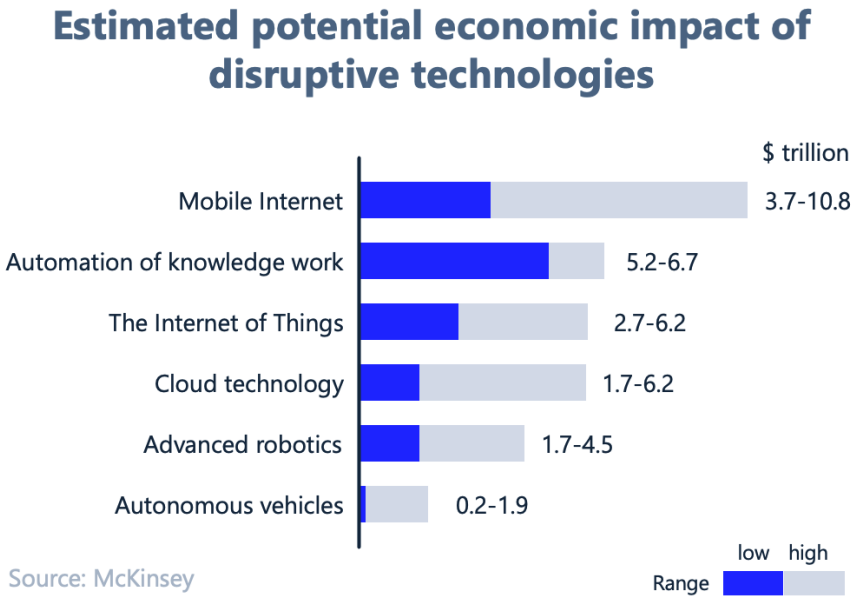

Figure 7.1: Node graph of the full implementation

7.2 Social and legal impact

Autonomous robotic systems are a tool of change in the way people and cargo are transported. Society vision of this technology and automation in general are divided. On one hand, there is a part of society that takes into consideration the benefits of automation and mobile robotics as it allows to reduce hard work labour and increase economic benefits. On the other hand, another part thinks this can lead to the destruction of job positions in a lot of areas. In Figure 7.2, a vision of society on new automation advances is shown:

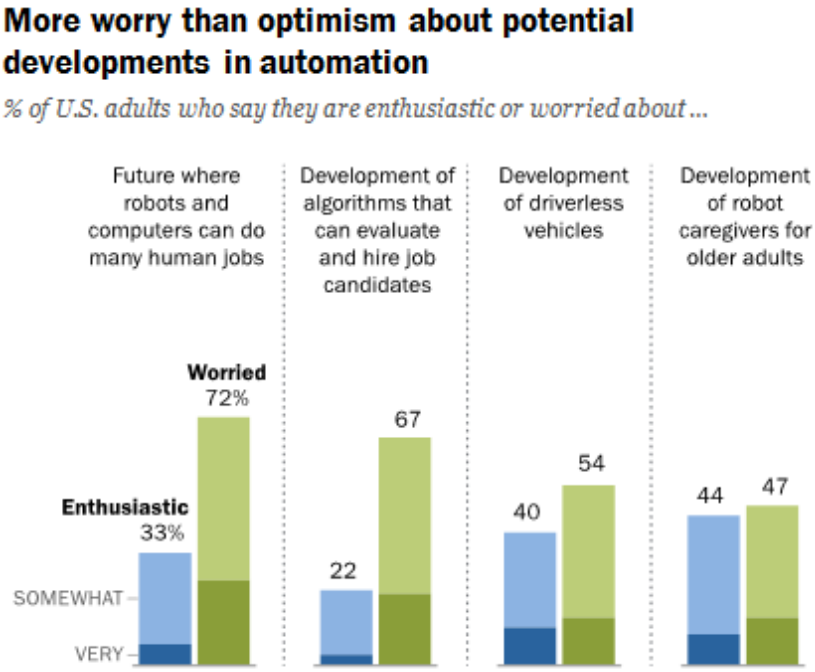

Figure 7.2: Society vision on automation advances

It is visible that technology advances should take into consideration the vision of society in order to progress without obstacles. It should also be considered that the advances in control engineering lead to complex systems with superior safety measures and error handling strategies, which can prevent accidents and damages to people and property.

In the legal matter, technology has been developed way faster than the law with respect to autonomous system. For example, in autonomous passenger vehicles, when the regulatory framework was created, autonomous vehicle were nonexistent and for that reason the legal aspects of autonomous robotic systems need to be reviewed as to not slow down the progress of this technology. Another relevant example are autonomous aircraft, like drones, with recent laws [19] that restrict their use in the European Union.

7.3 Environmental impact

The consequences of a widespread use of mobile robotics are significant. Like all the digital technology, semiconductor are essential components for its functioning, and the manufacturing

of these elements is highly polluting [20].

Also optimum path following autonomous vehicles could decrease the emissions by reducing the kilometers traveled and making efficient trajectories. Advanced control strategies could have a significant result in reducing the environmental impact and polluting emissions.

Several applications of autonomous mobile robotics could also have a direct impact in mitigate climate impact like cleaner robots in the ocean or drones that monitory the atmospheric pollution.

Chapter 8

Project budget and planning

8.1 Project budget

In this section, a brief overview of the cost of the project is presented. This budget is purely indicative, as the real budget would depend notably on the experience and knowledge of the engineer.

The cost is divided in cost of materials and labor cost. In Table 8.1, the cost of materials is explained.

Concept	Unit	Unit cost	Total cost
Laptop	1	1255.36 €	1255.36 €
Matlab License	1	800.00 €	800.00 €
Total cost			2.055,36

Table 8.1: Cost of materials

While the labour cost is explained in Table 8.2

Concept	Unit	Unit cost	Total cost
Design by Industrial Engineer	200 h	15 €/h	3000.00 €
Implementation by Industrial Engineer	300 h	15 €/h	4500.00 €
Total cost			7500.00€

Table 8.2: Cost of labour

If this project was to started from scratch, the total budget needed to carry it out would be 9.555,36€.

8.2 Planning of the project

In order to complete this project it was necessary seven months of work from February to September of 2021. In Figure 8.1, a Gantt diagram with the detailed time plan followed is shown.

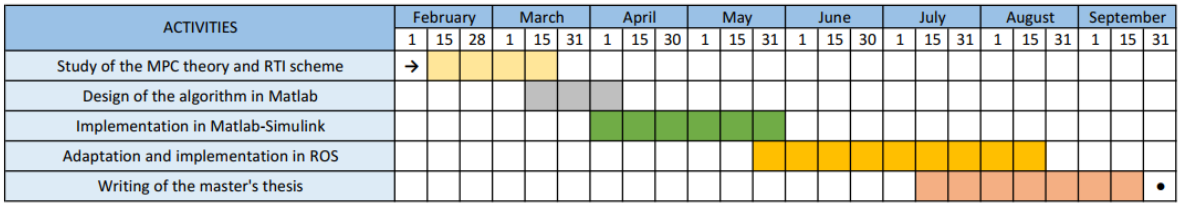

Figure 8.1: Gantt Diagram of the project

Chapter 9

Conclusions and future work

In this chapter a final review of the techniques carried out in the developing of this controller and results obtained is done. In Section 9.1, a series of conclusions are made and in Section 9.2 different possibilities to continue with this thesis are proposed.

9.1 Conclusions

The aim of this master's thesis was to study, design and implement an advanced Model Predictive Control technique based on the Real Time Iteration scheme.

After studying the paper related to this technique, a design was performed in MATLAB-Simulink and subsequently a implementation on a simulated environment in ROS was executed, tested and its results analyzed.

The outcomes of this controller have been positive, obtaining a reliable performance with a reasonable sample time. As the simulated environment is meant to be as closer to reality as possible, it is safe to assume that the controller developed and tested in ROS will have a positive result when implemented in a real Radio-Controlled car.

9.2 Future work

Due to lack of time, several tests, discussions and adaptations have remained to be continued in future projects. The main goal to be completed in this master's thesis is the implementation of the Real Time Iteration Controller in the physical vehicle designed by the BARC project.

Since this project has been developed in ROS, which stands out for its modularity, almost all of the coding can be reused in a RC vehicle, only replacing the simulator node for a node that performs the computations needed to convert the high-level control in a low-level control that can manage the actuators of the car. Also the computations needed to read the data for the on-board sensors have to be included in this future work.

Another possibility left out in this project is to change the control model used for predicting future outputs in the MPC for a more accurate problem with more states, which could result in a even better performance of the controller.

Related to the testing of the controller, new adaptations such as a better tunning of the weighting matrices, changes in the inequality constraints or testing in new tracks can be performed in future work in order to analyze thoroughly the performance of the controller and improve its design.

Bibliography

[1] Saso Blazic Gregor Klancar Andrej Zdesar and Igor Skrjanc. *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*. Elsevier, 2017.

[2] Jay H. Lee Manfred Morari. "Model Predictive Control: Past, present and future". In: *Computers and Chemical Engineering 2* (September 3, 1999).

[3] José Ramón Domínguez Frejo Eduardo Fernández Camacho. "Global Versus Local MPC Algorithms in Freeway Traffic Control With Ramp Metering and Variable Speed Limits". In: *IEEE* 13.4 (December, 2012), pp. 4600–4605.

[4] Carlos Bordons Eduardo Fernández Camacho. *Model Predictive Control*. 2nd ed. Springer Science Business Media, 2013.

[5] Carlos E. Garcia Manfred Morari and David M. Prett. "Model Predictive Control: Theory and Practice". In: *IFAC Proceedings Volumes* 21.4 (1988), pp. 1–12.

[6] Joachim Ferreau Moritz Diehl. *NonLinear Model Predictive Control*. Springer, 2009, pp. 391–417.

[7] Johannes P. Schloder Moritz Diehl Hans Georg Bock. "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control". In: *SIAM Journal on Control and Optimization* 43.5 (January, 2005), pp. 1714–1736.

[8] Sebastian Gros, Rier Quirynen, and Moritz Diehl. "An Improved Real-time Economic NMPC Scheme for Wind Turbine Control Using Spline-Interpolated Aerodynamic Coefficients". In: *Proceedings of the IEEE Conference on Decision and Control*. IEEE. 2014.

[9] Mario Zanon et al. *Model Predictive Control of Autonomous Vehicles*. Springer Science Business Media, 2014.

[10] Manfred Hiller Dieter Schramm and Roberto Bardini. *Vehicle Dynamics Modeling and Simulation*. 3rd ed. Springer, 2014.

[11] Hans Pacejka. *Tire and vehicle dynamics*. 3rd ed. Elsevier, 2005.

[12] Marc Facerías Pelegrí. "State Estimation in SLAM Techniques for Autonomous Vehicles". MA thesis. Spain: Universitat Politècnica de Catalunya, 2020.

[13] Eugenio Alcala et al. "Gain-scheduling lpv control for autonomous vehicles including friction force estimation and compensation mechanism". In: *IET Control Theory Applications* 12.12 (2018), pp. 1683–1693.

[14] Rien Quirynen Sebastien Gros Mario Zanon and Alberto Bemporad. "From linear to nonlinear MPC: bridging the gap via the real-time iteration". In: *Internation Journal of Control* 93.1 (September, 2016), pp. 1–19.

[15] Norman S. Nise. *Control Systems Engineering*. 7th ed. Wiley, 2015.

[16] B. Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: 10.1007/s12532-020-00179-2. URL: https://doi.org/10.1007/s12532-020-00179-2.

[17] J. Gonzales et al. "Autonomous drifting with onboard sensors". In: 2016.

[18] James Manyika et al. *Disruptive technologies: Advances that will transform life, business,and the global economy*. Tech. rep. McKinsey Global Institute, 2013.

[19] Comisión europea. *REGLAMENTO DELEGADO (UE) 2019/945 de 12 de marzo de 2019 sobre los sistemas de aeronaves no tripuladas y los operadores de terceros países de sistemas de aeronaves no tripuladas*. URL: https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32019R0945&from=EN.

[20] Yun-Sung Eom et al. "Emission Factors of Air Toxics from Semiconductor Manufacturing in Korea". In: *Journal of the Air Waste Management Association* 56.11 (2006), pp. 1518–1524.