Predictive Control Based Cooperative Control of Mobile Robots

Cristian Sava

Thesis to obtain the Master of Science Degree in Electrical and Computer Engineering

Supervisor: Professor João Manuel Lage de Miranda Lemos

Examination Committee
Chairperson: Professor João Fernando Cardoso Silva Sequeira
Supervisor: Professor João Manuel Lage de Miranda Lemos
Member of the Committee: Professora Rita Maria Mendes de Almeida Correia da Cunha

November, 2019
Declaration: I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.
I dedicate this work to my grandfather Vasile Ursu who passed away during my university career. It is the person who inspires me to be who I am.
Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid.

-Einstein
Acknowledgments

The time I have spent at the university was very precious and irreplaceable. If I could do it again, I would do it without any change, making the same choices I made. Although it was necessary to study immensely there were also moments of relaxation, having always been passed on several occasions together with the good friends that I have been doing throughout this academic journey. I participated in different groups of people and in different activities that helped me to evolve both intellectually and personally, helping me to get where I arrived today.

Now that I am finishing my master's degree, I wanted to thank some people who were more important and had more impact on my academic development and education. People who have always been with me, at the worst and the best times.

First, I would like to express my gratitude towards my supervisor and mentor Professor João Miranda Lemos that since day one helped me with everything. Whenever I came across a problem, I did not know how to solve it, the teacher was always available to receive me and to help me with everything that was needed. He was always patient with me, explaining things I did not understand more than once, making me always comfortable and relaxed during our sessions. I always felt motivated to work on the topic since Professor João believed in me since the first day. Professor João is, without any doubt, the person behind all the ideas developed in the dissertation.

I would like to thank INESC-ID for the possibility of using the space and the facilities to work during my dissertation.

Another person that was very important during my last year of university was my good friend João Belfo. He was the person who helped me better understand the meaning of some concepts and how should I apply them in the thesis development. We spent a lot of time talking about interesting subjects of the dissertation and helped me to mentally structure the content of the thesis. Apart from the shared theoretical moments, we had good moments of relaxation during our lunches and breaks were we shared a lot of laughter and talked about life concerns we both had.

I would also like to thank my good friend Diogo Teixeira. We did a lot of work together where we went through very complicated moments and situations of stress. However, with mutual help, we were always able to overcome our difficulties. Almost all my academic background was done in conjunction with him. We did a lot of work and labs and studied together for tests and exams. He was a person to share good times with friends, dinners and socializing and was my gym partner throughout the university.

I would especially like to thank my family, in particular to my mother Mariana, my father Ion, my brother Daniel and my grandmother Ecaterina, who helped me, supported me and were always close to me during these five years of my life. They were the most important people and they will always have an important place in my heart.

I would especially like to thank my mother Mariana who was always present and willing to help me with all that was needed. You are the strongest, most dedicated, caring, and lovable person I have ever met. I will always be grateful for all you have done and all you have abdicated to help me achieve all that I have achieved so far.
Finally, I wanted to thank my girlfriend Isabela Inês Correia e Brito who was always present at
the best and worst moments of my life during university. It helped me overcome my difficulties and made
me a stronger person. Every moment shared beside her was an adventure I will never forget.

Part of this work was performed in the framework of the project HARMONY Distributed Optimal
Control for Cyber-Physical Systems Applications, AAC no 2/SAICT/2017-031411, supported by Portugal
2020 through POR Lisboa-01-0145-FEDER-0 31411.
Abstract

There is an increasing need to find solutions to facilitate the execution of different tasks considered difficult, monotone and that can be performed in environments that are sometimes not favorable to humans. To meet the challenges, the robotics field has experienced a great development regarding remote and cooperative vehicle control in the execution of tasks and obstacle avoidance. This work focuses on the calculation of an optimal path, using MPC, where it is intended to obtain the input value $u$, which corresponds to the optimization variable so that a cost function $J$ is minimized. This can be achieved by designing a controller that, based on the input $u$, current states $x_n$ and final state $Y$, minimizes the total cost of the system.

To achieve this goal, linear and nonlinear models are studied including Single and Double integrator and the Tricycle model. Unconstrained and constrained solvers are used to study the influence of $Q$ and $R$ matrices and the impact of the horizon $H$ on the total cost. The study of the tricycle model can be divided into three main problems. In the first problem, the deviation of dynamic obstacles is approached. In the second problem, the interaction between two vehicles is studied during their journey, while deviating from obstacles. In the third problem, the execution of tasks by the interacting vehicles is studied, in more concrete, the passage of one or more rivers. The optimization process is used with different initial conditions, allowing to understand, based on the value of the cost function, what are the best conditions that minimize the total expenses to reach the target.

Keywords: Cooperative Control, Model Predictive Control, Linear and Nonlinear Optimization, Unconstrained and Constrained Solver, Dynamic Obstacle Avoidance, Vehicle Interaction.
Resumo

Existe uma necessidade crescente de encontrar soluções para facilitar a execução de diferentes tarefas consideradas difíceis, monótonas e que podem ser realizadas em ambientes que às vezes não são favoráveis aos seres humanos. Para enfrentar os desafios, o campo da robótica experimentou um grande desenvolvimento em relação ao controlo remoto e cooperativo de veículos na execução de tarefas e no desvio de obstáculos. Este trabalho concentra-se no cálculo de um caminho ótimo, usando MPC onde pretende-se obter o valor de entrada $u$, que corresponde à variável de otimização, para que uma função de custo $J$ seja minimizada. Isso pode ser alcançado projetando-se um controlador que, com base na entrada $u$, nos estados atuais $x_n$ e no estado final $Y$, minimize o custo total do sistema.

Para atingir este objetivo, modelos lineares e não-lineares são estudados, incluindo o integrador único e duplo e o modelo do triciclo. Solvers irrestritos e restritos são usados para estudar a influência da matriz $Q$ e $R$ e o impacto do horizonte $H$ no custo total. O estudo do modelo de triciclo pode ser dividido em três problemas principais. No primeiro problema, o desvio de obstáculos dinâmicos é abordado. No segundo problema, a interação entre dois veículos é estudada durante o seu percurso, enquanto se desviam de obstáculos. No terceiro problema, a execução de tarefas pelos veículos que interagem entre si é estudada, de forma mais concreta, na passagem de um ou mais rios. O processo de otimização é utilizado com diferentes condições iniciais, permitindo entender, com base no valor da função custo, quais são as melhores condições que minimizam o gasto total para atingir o objetivo.

Palavras-chave: Controlo Cooperativo, Modelo de Controlo Preditivo, Otimização Linear e Não-Linear, Solver Irrestrito e Restrito, Desvio de Obstáculos Dinâmicos, Interação de Veículos.
Contents

Acknowledgments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ix
Abstract . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xii
Resumo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiv
List of Figures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xx
List of Tables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xxiii
Nomenclature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xxv
Glossary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xxviii

1 Introduction 1
1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
1.2 Problem Formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
1.3 Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.4 State machine . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.4.1 Practical example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.5 State of the Art . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.6 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
1.7 Thesis Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2 Model Predictive Control 10
2.1 Model Predictive Control Theory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.2 MPC Principle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.3 MPC Elements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.3.1 Plant model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.3.2 Output equations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
2.3.3 Objective function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.3.4 Constraints . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.4 Nonlinear Model Predictive Control . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3 Optimization 17
3.1 Optimal Control Theory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
3.2 Convex and Non-convex Optimization . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
3.3 Unconstrained solver . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
# Table of Contents

3.4 Constrained solver .................................................. 19
3.5 Optimization Algorithms ........................................... 19
3.6 Constraints for constrained optimization solver .......... 20
3.7 The Q and R matrices ................................................. 22
3.8 The influence of tuning parameters ............................... 22

4 Vehicle models .......................................................... 26
4.1 Single integrator ....................................................... 26
4.2 Double integrator ...................................................... 26
4.3 Tricycle ................................................................. 27
4.4 Euler method .......................................................... 29
4.5 Runge-Kutta method .................................................. 30

5 MPC for path planning .................................................. 31
5.1 Cooperative Control .................................................. 31
5.2 Minimizing the trajectory cost .................................... 32
5.3 Influence of horizon on optimization ............................ 34
  5.3.1 Multi obstacle problem ......................................... 37
5.4 Application of MPC to the tricycle model ..................... 39
5.5 Dynamic obstacle avoidance ....................................... 41
  5.5.1 Horizon effect on dynamic obstacle avoidance .......... 43
5.6 Vehicle interaction .................................................... 44
  5.6.1 Vehicle interaction issues and resolution ................. 46
5.7 Task execution using two vehicles ............................... 51
5.8 Path optimization using different surfaces .................... 53
  5.8.1 Quick Review of Snell’s Law ................................. 53
  5.8.2 Influence of viscosity on path optimization ............... 54
5.9 Task execution and obstacle avoidance ........................ 55
5.10 Task execution and obstacle avoidance using two rivers ... 58

6 Results and Discussion ................................................. 61
6.1 External Supervisor ................................................... 61

7 Conclusion .................................................................. 64
  7.1 Future work ............................................................ 68

Bibliography ................................................................. 69

A Mathematical Equations for Prediction Model ................... A1
  A.1 State and output equations ........................................ A1
  A.2 Cost function ........................................................ A2
List of Figures

1.1 Block Diagram of the overall system, including all the stages necessary to implement the tasks and obtain the trajectory in the linear model. ................................................. 4
1.2 Initial problem representation. ........................................................................... 5
1.3 Representation of state machine for the problem described. ............................. 5
1.4 The state machine that expands crossing the river movement. ............................... 6
1.5 Number of papers published on the subject of Model Predictive Control [25]. ........ 8

2.1 The ‘Moving horizon’ in predictive control. .......................................................... 12
2.2 MPC structure. ................................................................................................... 13

3.1 Examples of convex (left) and non-convex (right) functions. ................................. 18
3.2 Three types of nonlinear constraints with different shapes. .................................... 21
3.3 Evolution of the final state error by changing the value of $R$ from 0 to 50. .......... 23
3.4 State evolution for different values of $R$. .............................................................. 23
3.5 Evolution of the final state error by changing the value of $Q$ from 1 to 50. .......... 24
3.6 Evolution of state for different values of $Q$. ......................................................... 24
3.7 System behaviour with different values of $R$ and $Q$. .......................................... 25

4.1 Tricycle model. ................................................................................................... 28
4.2 Curvature movement of a differential drive mobile vehicle. ................................. 28
4.3 Approximation made using the Euler method. ...................................................... 29
4.4 Approximation made by second order Runge-Kutta method. ............................... 30

5.1 Input $u$ variation for both cost functions. ............................................................. 34
5.2 Path calculation using each one of the previous cost function presented. ............ 35
5.3 Path calculation using different values of horizon. ............................................. 35
5.4 Evolution of the cost $J$ with the increase of the horizon $H$. ............................... 36
5.5 Evolution of the time $t$ in seconds with the increase of the horizon $H$. ............... 36
5.6 Influence of Horizon $H$ on the cost $J$ and on the computational time $t$ in seconds. 38
5.7 Influence of Horizon $H$ on the calculated trajectory. .......................................... 39
5.8 Tricycle model dynamics with different initial orientation. ................................. 41
5.9 Path obtained using a dynamic obstacle for a horizon of 10 (Part 1). .................... 42
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Differences between Interior-point and SQP algorithms [30], [62]</td>
<td>20</td>
</tr>
<tr>
<td>5.1</td>
<td>Simulation parameters</td>
<td>33</td>
</tr>
<tr>
<td>5.2</td>
<td>Coordinates of the obstacles</td>
<td>37</td>
</tr>
<tr>
<td>5.3</td>
<td>Simulation parameters</td>
<td>40</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulation parameters</td>
<td>43</td>
</tr>
<tr>
<td>5.5</td>
<td>Simulation parameters</td>
<td>45</td>
</tr>
<tr>
<td>5.6</td>
<td>Simulation parameters</td>
<td>57</td>
</tr>
<tr>
<td>5.7</td>
<td>Coordinates of the obstacles</td>
<td>57</td>
</tr>
<tr>
<td>5.8</td>
<td>Coordinates of the obstacles</td>
<td>59</td>
</tr>
</tbody>
</table>
Nomenclature

\( \alpha \) Inverse of viscosity \( \eta \).

\( \eta \) Viscosity parameter.

\( \Gamma \) Distance between vehicles with an angle orientation.

\( \Phi \) \((u(i - 1))^T R(u(i - 1))\) or \(R||u(i - 1)||\).

\( \tau \) Weight parameter.

\( \theta \) Angle deviation.

\( \theta_1 \) Incident angle.

\( \theta_2 \) Refraction angle.

\( \theta_{\text{init}} \) Initial and final angle between vehicles.

\( \theta_{\text{ref}} \) Reference angle to be maintained.

\( \theta_{\text{vehicles}}(i) \) Angle between vehicles at discrete time \( i \).

\( \Delta T \) Step time.

\( \hat{y} \) Future state prediction.

\( \mathbb{R} \) Set of real numbers.

\( A \) System matrix.

\( a_{uj}, b_{ij} \) Variables that describe the system.

\( B \) Input matrix.

\( C \) Output matrix.

\( c_{xg}^2, c_{yg}^2 \) \( x \) and \( y \) coordinates of the center of an obstacle \( g \).

\( c_{ij}, d_{ij} \) Constants.

\( c_v \) Speed of the light in the vacuum.

\( D \) Direct transmission term.
\( d_{\text{min}} \)  Radius in case of a circumference, length from tip to center in case of diamond, and length from side to center in case of square.

\( h \)  Sampling time.

\( k \)  Instant of time.

\( l \)  Distance between the two main wheels in a vehicle.

\( N \)  Time step in discrete time within the prediction horizon.

\( n_{\text{index}} \)  Refractive index of a medium.

\( q \)  Vehicle posture in base frame.

\( R \)  Input weighting matrix.

\( R1 \)  Robot 1.

\( R2 \)  Robot 2.

\( s \)  Counterclockwise rotation of \( v \).

\( u \)  Input control vector.

\( v \)  Linear velocity.

\( v_{\text{light}} \)  Speed with which light travels in a medium.

\( w \)  Angular velocity.

\( x \)  State equation vector.

\( Y \)  Reference vector.

\( y \)  Output variables column vector.

\( Z \)  Number of iteration in a simulation.

\( z \)  Forward operator.

\( Q \)  State weighting matrix.

\( H \)  Prediction horizon.

\( J \)  Cost function result.
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOCP</td>
<td>Hybrid Optimal Control Problem</td>
<td>2</td>
</tr>
<tr>
<td>IP</td>
<td>Interior-point</td>
<td>xxiii, 8, 20, 65</td>
</tr>
<tr>
<td>LMPC</td>
<td>Linear Model Predictive Control</td>
<td>3, 16</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Programming</td>
<td>16</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear Time-Invariant</td>
<td>14</td>
</tr>
<tr>
<td>MatLab</td>
<td>Matrix Laboratory</td>
<td>18, 19, 65</td>
</tr>
<tr>
<td>MHC</td>
<td>Moving Horizon Control</td>
<td>10</td>
</tr>
<tr>
<td>MILP</td>
<td>Mixed Integer Linear Programs</td>
<td>2</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
<td>xii, xiv, xx, 3, 7-13, 15, 16, 31, 32, 62, 64, 68</td>
</tr>
<tr>
<td>NLP</td>
<td>Nonlinear Programming</td>
<td>3</td>
</tr>
<tr>
<td>NMPC</td>
<td>Nonlinear Model Predictive Control</td>
<td>9, 16, 64</td>
</tr>
<tr>
<td>OC</td>
<td>Optimal Control</td>
<td>7, 9, 13, 17, 65</td>
</tr>
<tr>
<td>PVC</td>
<td>Polyvinyl chloride</td>
<td>10</td>
</tr>
<tr>
<td>QP</td>
<td>Quadratic Programming</td>
<td>16</td>
</tr>
<tr>
<td>RHC</td>
<td>Receding Horizon Control</td>
<td>2</td>
</tr>
<tr>
<td>RRT</td>
<td>Rapidly-Exploring Random Trees</td>
<td>68, 69</td>
</tr>
<tr>
<td>SQP</td>
<td>Sequential Quadratic Programming</td>
<td>xxiii, 8, 20, 33, 65</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

In this dissertation a capable controller is developed, that based on the model of the system, determine the optimal trajectory of a robot, that moves from a starting point to a final goal, while minimizing a global cost defined over the path. Two different models will be studied in detail, the Double Integrator, and the Tricycle. This thesis also addresses the capability of a group of robots to execute certain tasks, during displacement, establishing how the design parameters will influence the optimal path. As we will see, the optimization subject is fundamental both for the study of the control topic and for the task execution.

Cooperative behavior of multiple autonomous vehicles plays a key role in many applications, that are emerging more and more in this century. It is a vast area that is growing every day and it is possible to find this technology in many fields. It is used in the logistic area to transport different heavy loads from one point to another [32, 16], inspection and maintenance to reach areas that are hard to access due to temperature, pressure or safety constraints [44, 7], cleaning [49, 27], security and defense to provide support on executing dangerous missions [36, 34], in agriculture to develop several agriculture applications, improving quality and performance [24, 51], in nuclear plants to avoid the use of people for assignments [55, 28], and in search and rescue missions [46, 23].

To have a good performance in all the tasks mentioned previously it is necessary to perform initially an analysis of all the actions that are going to be executed and it is fundamental to have a good knowledge of the exact location of the vehicles that are being used, in order that they act cooperatively.

Cooperative control comprises two important tasks: the determination of trajectories and the situation-based allocation of individual roles and sub-tasks to the team members. Almost all applications require a robust online control strategy, making it possible to adapt quickly in real-time to a dynamic environment [35]. Considering this, it is possible to understand that the biggest challenges in this area are related to the vehicles interaction and to the amount of time that the optimization problem requires.

Different approaches exist to solve these problems. One of them is in which actions are indicated based on existing heuristic methods or rules that imply decisions based on behavior. However, optimal control strategies can be obtained based on optimization-based approaches. Optimal control strategies
not only can be applied for each vehicle individually but also to the whole team. An optimal solution can be obtained based on the optimization-based approach for all possible system states, including the methods not covered by the heuristic approach. Modeling a cooperative control problem must enable continuous trajectories as well as discrete variables and logical rules. In most cases a nonlinear Hybrid Optimal Control Problem (HOCP) emerges. A class of nonlinear control systems is the nonholonomic systems that correspond to the Tricycle model studied in this dissertation.

Nonholonomic systems are mechanical systems in a finite dimension where non integrable constraints are imposed on their velocities. These constraints are not derivable from the constraints applied to the position [56]. To represent the previous constraints, integrable linear velocity relationships can be used. The result obtained from the characterization of a nonholonomic system with control inputs is a nonholonomic control system [43]. Nonholonomic systems are usually defined by mechanical systems, that is, systems that can be characterized by a rolling contact (ex. wheels) without slippage [56].

To tackle these nonlinear control problems, nonlinear approaches are used. Although generally controllable, momentarily the movement of nonholonomic systems is restricted to certain directions. These restrictions are designated by nonholonomic constraints [43].

The goal of the HOCP method is to find the optimal hybrid control solution to minimize the time spent or the energy consumed, subject to the system dynamics and all the constraints [8]. A big computational work is required to solve this kind of problems. One way to obtain a good solution estimate is to simplify the HOCP problem using a discretized linear model. Thus, a large part of optimization-based control strategies uses Mixed Integer Linear Programs (MILP) to provide a closer approximation to the actual system considered. This method is highly recommended for modeling real-world decision, allowing processes or systems to achieve the best result based on a set of requirements [5].

This dissertation will discuss strategies for hybrid linear systems of discrete-time, using Model Predictive Control (MPC). Model Predictive Control (also known as Receding Horizon Control), was initially developed to control the industrial processes of petroleum refineries. Nowadays it is used not only in industry but also in other diversified areas, from automated processes to clinical anesthesia [43]. This approach is based on the resolution of multiple MILPs in a receding horizon. With this, it is possible to attain a stabilizing optimal closed-loop control strategy. The applicability of this approach to multi-vehicle cooperative systems, as well as efficiency and scalability, is analyzed based on its performance.

1.2 Problem Formulation

The main objective of this dissertation is to develop, implement and test a controller, for a multiple robot system, that can find a sequence of optimal control actions for their operation. Based on this optimal sequence, it is possible to calculate the best trajectory. With this approach, it is expected that a robot, or set of robots, can execute certain tasks in a predefined period, and find the best trajectory, always considering the minimization of the total cost. This minimization is related to a cost function, allowing the maximization of overall performance. This control strategy is implemented based on an approach designated by Model Predictive Control. It is necessary to implement a controller that is capable of
accurately tracking a given trajectory and avoiding different obstacles in the environment. The trajectory will be provided by a higher-level planning method and the group of robots will try to follow this trajectory until an obstacle appears. When an obstacle appears in the environment, the MPC automatically avoids it [9]. To reach this goal, the following tools will be developed and used:

- State equations;
- Cost function;
- Nonlinear Programming solver (fmincon);
- State machine.

For the development of the controller, it will be necessary to construct a cost function that, besides considering all possible actions, also needs to know what the impact of each of those actions on the model performance is. During the movement of the vehicles, several tasks will be executed, and a different cost function will be used, changing the weight of the parameters of the initial cost function, influencing the value calculated by the optimization method. Therefore, during the execution of each task, these parameters will have a greater or lesser impact on the minimization process of function $J$.

To choose the best action, it is important to consider the working environment. This may contain obstacles that may obstruct the way, preventing the execution of the various tasks previously implemented. To achieve the final objective, it is also necessary to choose the weight of each parameter individually for the optimization method to respect the different constraints.

Before implementing the control strategy it is necessary to understand (1) what is the model that characterizes the robot or group of robots, (2) what is the method used to predict the robot trajectory and (3) how it is possible to switch between robot movement (discrete-time), and between the different stipulated actions, which can occur in a non-sequential way. In other words, there is the need to obtain a predicted trajectory taking into consideration the model that characterizes the robot and how the robot’s state variables evolve over some time, depending also on the existing tasks.

To answer question (1) three different methods will be approached for the model of the vehicles. These models will be explained in more detail in chapter 4 corresponding to: 1) Single Integrator, 2) Double Integrator, 3) Tricycle.

Problem number (2) can be subdivided in three different problems: (2.1) what are the parameters necessary to use allowing, at the same time, to find a solution as quickly as possible and to maintain stability on the system (for instance $Q$, $R$, $H$ parameters) - (2.2) what is the cost function and the method that should be used to minimize the error between the predicted value and the reference that is provided - $fminunc$ and $fmincon$ - and (2.3) what are the constrains that must be respected. These constraints correspond to different obstacles that may exist in the environment - circle, diamond, and square shapes.

Problem number (3) can be solved by the help of a state machine that will allow us to jump between two distinct types of behaviors - robot movement and task execution.
1.3 Models

All the problems mentioned above are correlated and their interconnection can be observed in the figure below. If the model that is being studied is the Single or Double Integrator (linear models) the interconnection can be observed in figure [1.1]

Figure 1.1: Block Diagram of the overall system, including all the stages necessary to implement the tasks and obtain the trajectory in the linear model.

Here, the squares represent important phases and methods to develop and circles represent the different parameters and requirements needed to implement the methods.

In figure [1.1] it is possible to observe the total system, including all the fundamental phases to develop the controller for a linear model. Initially, it is necessary to obtain the state equations that define the vehicle motion. The state equations can be calculated based on the transfer function. From the state equations the fundamental parameters \((A, B \text{ and } C)\) can be obtained and together with matrices \(R\) and \(Q\) and considering the reference \(Y\), the function \(J\), shown in equation [A,10], can be minimized. The vehicle state equations are represented by

\[
x(k + 1) = f(x(k), u(k)),
\]

where \(u(k)\) is the control obtained by minimizing, in a receding horizon sense the cost represented by

\[
J = \sum_{i=1}^{H} \left( x(k+i) - Y(k+i)^T Q(x(k+i) - Y(k+i)) + u^T(k+i-1) R u(k+i-1) \right).
\]

1.4 State machine

A major goal of this work is to solve problem number (3). This problem corresponds to the implementation of a state machine that defines the sequence of tasks required to accomplish a given mission, e.g., going from one point to another while crossing a river. To understand how the state machine will have an impact on the project, a practical example will be described, where it will be possible to see, how the transitions occur between the normal movement of the vehicles and the task execution.

1.4.1 Practical example

Imagining that initially there are two robots, both with an initial position. Between them, there is a wooden board and they must carry it until a certain point in time. The main goal of each vehicle is to move to the reference, always considering the lowest possible cost, maximizing the performance. The only obstacle that exists in the environment is a river, dividing the map into two portions, the one that contains the
initial position and the one that contains the final position. The only way to reach the reference is by crossing the river. To cross the river the two robots will have to use the wooden board that they are carrying. In figure 1.2 it is possible to observe an initial illustration of the problem with the two vehicles carrying the wooden board, the river in the middle and the reference represented by a diamond form.

![Figure 1.2: Initial problem representation.](image)

After the previous description of the problem, it is possible to understand that there will be two types of actions that the robots must perform. The first one corresponds to the movement of both robots along homogeneous terrain, and the second one corresponds to the placement of the wooden board and crossing the river through it. The second type of action is also in discrete time, but it is not sequential as the movement. These types of actions can occur in different time steps, for example, the first task occurs in time $k = 3$ and the second task occurs in time $k = 8$.

Having two types of distinct actions there is the need to implement a state machine that will allow the transition between them. The state machine corresponds to a mathematical model of computation that can be on exactly one of a finite number of states at any given time. It can change between different states, considering its response to some external inputs. The behavior of a state machine can be verified in a lot of devices nowadays [60]. Some examples of state machine implementation are vending machines, elevators, traffic lights, and combination locks [19] [66]. In the next figure, it is possible to observe one type of state machine implementation for the problem presented in figure 1.2.

![Figure 1.3: Representation of state machine for the problem described.](image)

From the previous figure, it is possible to observe the two different types of actions: vehicle movements and crossing the river. The vehicles will perform the path calculation until they reach the
river. When their next position is the river, a flag is triggered and the problem transits from the vehicle movement state to the crossing river state. In this state, there is a set of actions that are carried out by both vehicles that allows their crossing. This set of actions can be better understood through the scheme presented in figure 1.4. The crossing movement is performed with the aid of a board of length slightly greater than the width of the river, which is carried by the two vehicles.

After crossing the river, the flag is deactivated and the problem transits again, but this time from crossing river state to vehicle movement state, enabling the vehicles to continue their path calculation until they reach their final goal position. The difference between the two states is the cost function that is minimized for the calculation of the input $u$. In both states, the cost function used is similar to the one presented in equation (1.2). However, what differentiates them are the weights assigned to the matrices $Q$ and $R$. The change of these parameters will allow the vehicles to behave differently in each situation.

### 1.5 State of the Art

The importance of path optimization, calculation and tracking, cost minimization and obstacle avoidance led to an increasing interest in learning and studying more about the cooperative control of mobile robots.

Robots were initially introduced in industry processes back in 1961 and until the 1990s the field of robotic search was mainly dominated by industrial robots. It was a breakthrough that revolutionized the industry, giving the robots the tasks considered more risky, harmful, repetitive and tedious, such as welding, painting, cutting, carrying and assembling. Several industries have taken advantage of this development to improve their assembly lines exponentially, making automated operations faster, reducing waste by achieving optimally controlled production. Therefore, the introduction of robots in industry processes brought some benefits for the business allowing an increase in efficiency, accuracy, cost and time saving, among others. Since then, robots have been evolving continually, originating exciting and diversified areas and allowing the execution of more and more complex tasks.

Robotics is an area that is increasingly developing within our lives. Different areas have emerged that require the use of robots with specific characteristics, such as flexibility. We are constantly surrounded by robots that make life easier for us in many tasks such as building cars, feeding our animals, cleaning houses and others. Thus, it is remarked that the ability to adapt to the environment was one of the most intriguing and challenging problems to solve in this new line of robots. As society evolves new needs and markets emerge outside the usual market of manufacturing and industrial robotics. This leads to a new concept of robot allowing the development of a new sector, with great future, providing services to all human being. This new sector corresponds to mobile robots.
Unlike industrial robots, mobile robots have more degrees of freedom and their movement is unlimited by its physical size. The first mobile robots that were installed in the industry followed a predefined trajectory and had the main objective of transporting heavy materials and tools [21]. As they evolved, they developed the capability of moving and exploring an unknown environment and execute tasks that humans could not execute because of the temperature, pressure or physical limitations. Some tasks and projects that mobile robots are involved in are surveillance, exploring, security, military, industry, domestic cleaners and caretakers, [63]. Mobile robots can be subdivided into three categories according to their operating environment: (i) land robots, (ii) aquatic/underwater robots, and (iii) aerial robots [14].

Most mobile robots operate in an unknown environment, having the necessity to execute four steps: environmental awareness, self-localization, movement planning, and movement generation [21]. To execute all these steps, they need to have higher levels of intelligence capabilities, unlike industrial robots [63]. This field has many challenges yet to overcome. One of them is related to navigation.

There are two types of navigation: (1) reactive navigation: the robot does not know the environment and obtains information about the obstacles from sensors, and (2) path planning: the robot has all the information about the map and can use different algorithms to obtain the optimal trajectory [17].

In robotics, path planning is the ability to calculate an optimal path, based on analyses performed to the environment around the robot, and on a performance criteria, allowing the vehicle to navigate autonomously from the initial point to its final goal [64]. To find a path, i.e., a secure and reliable trajectory, in a complex environment with dynamic and static obstacles, different techniques can be used [40]. In the last years, heuristic methods have gained great importance and are widely used due to their simplicity and computational efficiency, although they do not guarantee to find an optimal solution [22].

The optimal solution depends on the criteria used to evaluate the trajectory. One possible criterion that is to optimize a function that penalizes the deviation of the robot from the desired goal. This cost will be influenced by the complexity of the environment, the quantity of the obstacles and the constraints applied to the system. Therefore, the more complex are the conditions around the moving vehicles, the longer it takes to calculate the path. There are two different methods of path planning that can be used: local path planning and global path planning [15]. What differentiates the two methods is the level of information, about the environment, to which the vehicle has access.

For global path planning, all data and details relating to the environment are considered and the existing obstacles are static [42]. Given that a large amount of information is supplied to the robot, the calculation process becomes slow, not being the most advisable method to use for the determination of the path and the deviation of obstacles, in real-time. For local path planning, only part of the environment information is provided to the robot, most of which is unknown, being necessary to recalculate the path to avoid dynamic obstacles. Therefore, the map is restricted to the zone around the vehicle and it is updated as the robot is moving through the environment. Although making processing less complex and less time consuming, most of the time the solution obtained does not correspond to the optimal solution [15] [64] [42]. To obtain the optimal solution an optimization problem needs to be solved. Different methods can be used to achieve that solution, such as methods from Optimal Control and Model Predictive Control.

The main goal of the methods used in OC and MPC is to find the control law for a dynamical
system in order to achieve a certain optimality criterion, that is, optimize a performance index, satisfying all the constraints applied to the system \[2, 33\]. The differences between **Optimal Control** and **Model Predictive Control** is that OC uses the entire horizon of the problem, while MPC uses a smaller horizon, defined previously. Having a smaller horizon and a feedback control allows to obtain a more accurate and exact path, being able to adapt to the variations that can occur over time, such as dynamic obstacles. This way, a correction of the course at each instant of time is executed.

**Model Predictive Control** is a control algorithm developed in the 1970s and was, in the beginning, based on heuristic control algorithms. It was initially designed to obtain online constraint optimizations. It was used to solve multivariable constrained optimal control problems, especially for chemical and oil industrial processes. Considering the technological, scientific and social developments, MPC had to suffer an improvement, since control requirements were becoming increasingly demanding. In recent years, MPC is expanding rapidly and the demand for model forecasting technology is growing. Therefore, MPC started to be used in various fields such as medical care, power grids, aerospace technology, urban life, water networks, vehicle traffic, diabetes insulin delivery and it can be found in almost every refinery and petrochemical plant \[37, 61, 25\]. Figure 1.5 represents the number of the papers published about **Model Predictive Control**. It is fundamental to notice the increasing interest in the topic, in the past years.

![Figure 1.5: Number of papers published on the subject of Model Predictive Control](image)

**MPC** corresponds to a framework that is used to solve a wide range of control problems and to cope with plant constraints. To obtain the command for the plant, MPC solves an optimization problem, considering the control objectives and the existing constraints. There are several efficient solvers that could be used to cope with the requirements and can be divided in two categories: (1) offline, solves the global path planning problem and then applies that path to the robot, and (2) online solvers, as the path planning problem is solved, in each instant of time, the solution is applied immediately to the robot \[65\].

Online optimization methods can be divided in to two classes: first-order (e.g. gradient methods) and second-order (e.g. active-set, SQP and Interior-point) methods. The difference between these two sets of methods is that the first-order methods have simpler theoretical requirements and converge to a medium-accuracy solution within a few iterations. The second-order methods have more strict theoretical
requirements but are more suitable when a high-accuracy solution is required [65][20].

1.6 Contributions

The main contribution of this thesis is the development of a controller using Model Predictive Control that allows a robot to navigate through a known map, avoiding all the obstacles and executing the tasks required for the group of vehicles. All this is done in conjunction with the minimization of the trajectory cost. This dissertation will approach theoretical concepts of MPC and different solvers and algorithms used for the optimization process. Adjustments and improvements will be done in terms of the cost functions allowing to obtain a better solution, respecting all the constraints applied to the vehicle.

1.7 Thesis Outline

This work is organized in the following chapters:

- In chapter 1, an initial introduction has been done regarding the work developed on the dissertation. It addresses the motivation subject and provides a more detailed explanation of the main problem to be solved, presenting the objectives to be achieved during the development of the project. State machine is introduced and explained using a practical example.

- In chapter 2, a explanation of Model Predictive Control is made. Subjects as Principle and the fundamental elements are presented. The Plant model with the input and output equations is explained and topics as Objective function and Constraints are addressed. Nonlinear Model Predictive Control (NMPC) is approached and a comparison with LMPC is made.

- In chapter 3, a review of Optimal Control theory is made, including a brief introduction to Convex and Non-convex Optimization. Two different optimization methods are studied, the constrained and unconstrained solvers and some optimization algorithms are presented. The influence of some parameters such as matrix $Q$ and $R$ on the optimization process is also demonstrated.

- In chapter 4, vehicle models are presented. All three models are explained allowing to understand the differences between them. In this chapter, two different methods for the approximation of the integral term presented in the tricycle state space equations are also discussed.

- In chapter 5, Cooperative Control is approached. Different aspects are studied, such as the cost function to be used and the influence that the horizon has on the optimization process. The tricycle model is approached, being tested in situations where it is necessary to deviate from dynamic obstacles, it is necessary to have interaction of two vehicles and it is necessary to perform tasks.

- In Chapter 6, an analysis and discussion of the results is performed, presenting a solution for situations where the final solution is inaccurate and unfeasible.

- In chapter 7, it is presented the conclusions made about the work developed and some possibilities for future work to which this project can be applied or improved.
Chapter 2

Model Predictive Control

2.1 Model Predictive Control Theory

Model Predictive Control (MPC), that can also be referred to as Moving Horizon Control, does not designate a specific control strategy. It is a large set of methods which make explicit use of a process model. With this, it is possible to obtain the control signal, allowing the minimization of an objective function.

The MPC concept emerged in the seventies and was developed over the years. Since the beginning several techniques and methods have been implemented to design model-based control systems [58]. It has been used since the 1980s in chemical plants, oil refineries, and process industries. More recently it has been used in power system balancing models and power electronics. All the methods design by MPC provide controllers that have similar construction, presenting adequate degrees of freedom [12]. With the MPC not only it is possible to calculate a control sequence, minimizing an objective function, but it is also used to predict the output of the process in future instants. Using this technique, at each instant the horizon undergoes a shift into the future, involving the application of the first control signal of the sequence calculated in each time step, allowing for a more accurate model [50].

We can encounter different types of MPC algorithms where the model representation of the process, noises and the form of the objective function, as well as the optimizing algorithm, vary. Nevertheless, all the different MPC algorithms have in common the same control idea. The technology behind these algorithms has evolved over the years, making it possible to evolve from a control process based on basic multivariable, to a technology that enables operations of processes within well-known and defined operating restrictions. As referred previously, MPC technology applications are being successfully used in a large number of sectors. In addition to the process industry, MPC is used in robot control processes as well as control of clinical anesthesia. Other applications of the MPC technology are [12]:

- Robot arms;
- Development for distillation columns;
- Cement industry;
- PVC plants;
- Drying towers;
- Steam generators.

Considering the performance of the applications enumerated previously, it is possible to affirm
that MPC can achieve highly efficient control systems that are working for long periods. Although long operating periods, the intervention required for maintenance is very scarce. The increased use of MPC since the eighties is due to the following reasons [12, 58]:

- Existence of limitations in industrial processes related to technological requirements, limitation in valve capacity and the necessity to deliver output products with pre-specified qualities;
- It is a model-based controller design procedure that can be used to handle processes with large time-delays, non-minimum phase, and unstable processes, in an easy way;
- It is an easy-to-tune method. Manly, there are only three basic parameters to be tuned;
- MPC should be able to deal with structural changes such as failures in terms of the sensor and actuator, changes of system structure and the different parameters, embracing a control strategy on a sample-by-sample basis.

However, the main reason that explains why MPC is widely used and it is preferable to other control models is the fact that it has the excellent ability to manipulate constraints. Although it is widely used for the great amount of advantages it presents, this method also presents challenges [58]:

- There is a need to have a well-detailed process model. It is fundamental to have a well-structured understanding of the physical behavior of the plant, being necessary to apply the various methods of identification, allowing the calculation of a good model;
- This procedure is open, this is, there are a lot of variations that allowed the development of many MPC methods;
- Stability and robustness properties are difficult to derive using theoretical analysis, even though, in practice, these properties can be obtained by accurate tuning.

### 2.2 MPC Principle

As mentioned before, there are various number of different algorithms representing the MPC implementation, each one of them with a different model representation, a different objective function or even a different optimizing algorithm. Despite the different aspects, they all have in common the same control idea and the same principles.

The current control signal can be obtained by solving a finite horizon open-loop optimal control online problem at each time step [52]. For this, the current state belonging to the system must be utilized as the initial state for the plant model. With this, it is possible to obtain, at each time step, a sequence of optimal control signals. Although a sequence of signals is generated, only the first one is implemented and applied to the process. Also, before starting a new optimization iteration, the prediction horizon is shifted one-time step, restarting the process with new data for the measurements. The methodology of all controllers which are a part of the MPC family can be represented (in discrete time) in figure 2.1.
Almost every single time the main goal of the model is that the system outputs \( y(k + k\Delta T|k) \), for \( k = 1, \ldots, N \), follows the reference trajectory \( Y(k + k\Delta T) \) for \( k = 1, \ldots, N \), where \( N \) is the number of time steps in discrete-time within the prediction horizon and \( \Delta T \) is the optimization step time. The prediction outputs, \( y(k + k\Delta T) \), not only are dependent on the values that are known up to instant \( T \), that is the past inputs and outputs, but also on the future control signals \( u(k + k\Delta T|k) \), for \( k = 1, \ldots, N \), that are the values calculated and sent to the system [52]. To calculate the set of future control signals it is necessary to optimize a criterion that works to keep the method within the desired values for the path. The set of prediction values are based on dynamical models. Most ordinary models’ representation in MPC correspond to polynomial, impulse response, step response or state-space models [58].

Usually, for the optimizing criterion, it is used a quadratic function of the errors between the predicted output signal and the predicted reference trajectory. If the objective function is quadratic and there are no constraints, the set of future control signals can be calculated directly. If constraints exist, then the process becomes more complex with the need to use iterative optimization algorithms.

After calculating the set of control signals \( u(k + k\Delta T|k) \), only the first value of that set \( u(k|k) \) will be used for the method. The rest will be ignored, and the process will perform an update of the data at the next sampling point. At each time step the variables of the \( u(k + k\Delta T|k) \) are updated since advancing in time allows the model to obtain new important information characterizing the environment. This new information will change the way to implement the tracking of the reference trajectory.

The usual and basic structure of MPC can be observed in the figure 2.2. The MPC model, based on the current state information, that is collected from measurements, calculates, based on the process model, the future output which then can be compared with the reference. The objective function has a member that penalizes the existing error between the reference trajectory and the current state. The optimizer tries to find the optimum of the objective function, always considering the constrains applied to the model. After the work of the optimizer, the first control signal is applied to the method [52].
2.3 MPC Elements

Usually, to determine the design parameters of a system, it is required to use a process of attempted error in which different methods of analysis are used repeatedly. As mentioned before, finding the best control signal, that allows the satisfaction of all physical constraints and the maximization of system performance, is the biggest goal of Optimal Control Theory. Depending on the MPC algorithm, different options can be chosen for the common elements which constitute these algorithms. Therefore, the requirements that are needed for the formulation of an Optimal Control system are: 1) Plant Model, 2) Output Equations, 3) Objective Function, 4) Constraints.

2.3.1 Plant model

The definition of a physical plant can be taken from the book of Naidu [47], as being a system with a set of linear or nonlinear differential or difference equations. In the standard form, the system can be described by a mathematical basis and is expressed as a set of $n$ coupled first-order ordinary differential equations, known as state equations. Each state equation is expressed in terms of discrete-time parameters and it is composed by state variables $x_1(k),...,x_n(k)$ and by the system inputs $u_1(k),...,u_r(k)$. In the general case the form of the $n$ state equations is [57]

\begin{align}
  x_1(k+1) &= f_1(x(k),u(k),k), \\
  x_2(k+1) &= f_2(x(k),u(k),k), \\
  &\vdots \\
  x_n(k+1) &= f_n(x(k),u(k),k),
\end{align}

(2.1)

where each of the functions $f_i(x(k),u(k),k), \ (i = 1,...,n)$ may be a general nonlinear discrete time varying function of the state variables. Normally the state space equations can be expressed in a vector
form and the set of \( n \) state variables can be written as a state vector \( x(k) = [x_1(k), x_2(k), ..., x_n(k)]^T \), and the set of \( r \) inputs can be written as an input vector \( u(k) = [u_1(k), u_2(k), ..., u_r(k)]^T \). The vector representation for the set of \( n \) equations in (2.1) is presented as

\[
x(k) = f(x(k), u(k), k),
\]

(2.2)

where \( f(x(k), u(k), k) \) is a vector function with \( n \) elements \( f_n(x(k), u(k), k) \). There are different classifications of a system. Systems can be described as linear, nonlinear, time-invariant and time-varying. If a system is nonlinear and time-varying, the discrete-time state equations are written as in equation (2.2).

For a Linear Time-Invariant (LTI) system of order \( n \), and with \( r \) inputs, equation (2.1) becomes a set of \( n \) coupled first-order linear differential equations with constant coefficients

\[
\begin{align*}
x_1(k+1) &= a_{11}x_1(k) + a_{12}x_2(k) + ... + a_{1n}x_n(k) + b_{11}u_1(k) + ... + b_{1r}u_r(k), \\
&\vdots \\
x_n(k+1) &= a_{n1}x_1(k) + a_{n2}x_2(k) + ... + a_{nn}x_n(k) + b_{n1}u_1(k) + ... + b_{nr}u_r(k),
\end{align*}
\]

(2.3)

where the coefficients \( a_{ij} \) and \( b_{ij} \) are variables that describe the system. Equation (2.3) can be written densely in a matrix form represented in expression (2.4) and summarized in expression (2.5)

\[
\begin{bmatrix}
x_1(k+1) \\
\vdots \\
x_n(k+1)
\end{bmatrix} =
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
\vdots & \vdots & \cdots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
x_1(k) \\
\vdots \\
x_n(k)
\end{bmatrix} +
\begin{bmatrix}
b_{11} & \cdots & b_{1r} \\
\vdots & \cdots & \vdots \\
b_{n1} & \cdots & b_{nr}
\end{bmatrix}
\begin{bmatrix}
u_1(k) \\
\vdots \\
u_r(k)
\end{bmatrix},
\]

(2.4)

\[
x(k+1) = Ax(k) + Bu(k),
\]

(2.5)

where the state vector \( x \) is a column vector of length \( n \), the input vector \( u \) is a column vector of length \( r \), \( A \) is a \( n \times n \) square matrix of the constant coefficients \( a_{ij} \), and \( B \) is a \( n \times r \) matrix of the coefficients \( b_{ij} \) that weight the inputs [47].

### 2.3.2 Output equations

In a plant model, all physical quantities can be observed and measured. Usually, these quantities are designated as outputs and are represented by \( y \). The output of a system is defined to be any variable of the system that is of interest. One fundamental property associated with linear state equation definition is that all system variables can be described by a linear combination of the state variables \( x_i \) and the system inputs \( u_i \) [12]. It is possible to write an arbitrary output equation of order \( n \) having \( r \) inputs as

\[
y(k) = c_1x_1(k) + c_2x_2(k) + ... + c_nx_n + d_1u_1(k) + ... + d_ru_r(k),
\]

(2.6)

where \( c_i \) and \( d_i \) are constants. If a total of \( n \) system variables are defined as outputs, the \( n \) equations are represented by expression (2.7), and written in a more compacted form as shown in equation (2.8)

\[
\begin{align*}
y_1(k+1) &= c_{11}x_1(k) + c_{12}x_2(k) + ... + c_{1n}x_n(k) + d_{11}u_1(k) + ... + d_{1r}u_r(k), \\
&\vdots \\
y_n(k+1) &= c_{n1}x_1(k) + c_{n2}x_2(k) + ... + c_{nn}x_n(k) + d_{n1}u_1(k) + ... + d_{nr}u_r(k).
\end{align*}
\]

(2.7)
\[ y(k) = Cx(k) + Du(k), \]  

(2.8)

where \( \mathbf{y} \) is a column vector of the output variables \( y_i(k) \), \( \mathbf{C} \) is a \( m \times n \) matrix of the constant coefficients \( c_{ij} \) that weight the state variables, and \( \mathbf{D} \) is a \( m \times r \) matrix of the constant coefficients \( d_{ij} \) that weight the system outputs \[12\]. For many physical systems the matrix \( \mathbf{D} \) is the null matrix, and the output equation is reduced to a simple weighted combination of the state variables

\[ y(k) = Cx(k). \]  

(2.9)

### 2.3.3 Objective function

Considering the different types of [MPC](#) algorithms it is known that each one of them have different cost function, being possible to obtain the control law. The main goal is that the future output of the system, represented by \( \mathbf{y} \) on the considered horizon, follows a reference represented by \( \mathbf{Y} \).

The cost function is used to measure the performance of a system quantitatively by minimizing an optimal control system. Finding the perfect control function is a hard-working process. It is necessary to define a mathematical expression that will give the most desirable performance of the system \[12\]. To minimize the error of the output, considering the reference, the general expression of the objective function, adding two extra weight factors, \( Q \) and \( R \), can be represented as in equation 2.10

\[
J = \sum_{i=1}^{H} \left( (\hat{y}(k+i|k) - Y(k+i|k))^T Q (\hat{y}(k+i|k) - Y(k+i|k)) + (u(k+i-1|k))^T R (u(k+i-1|k)) \right),
\]  

(2.10)

where \( \hat{y}(k+i|k), i = 1, 2, ..., H \) is the prediction of the future state generated by the model \( x(k+1) = Ax(k) + Bu(k) \) and \( \mathbf{Y} \) is the reference. Therefore, to predict \( \hat{y}(k+i|k) \), the information used is based on the past information up until discrete time \( k \), under the action of control sequence \( u \). That is, \( \hat{y}(k+i|k) \) is the prediction of \( y(k+i) \), based on the information up to time \( k \). The future system outputs are predicted by using a process model from the inputs and outputs before the time instant \( k \), and from the predicted future control actions \( u(k+i-1|k) \). The sequence \( u(k+i-1|k), i = 1, 2, ..., H \) are the manipulated variables to be optimized. Finally, \( Q \) and \( R \) are symmetric positive semi-definite and definite weighing matrices, respectively, where \( Q \in \mathbb{R}^{n \times n} \) and \( R \in \mathbb{R}^{m \times m} \[47\].

### 2.3.4 Constraints

Usually, all processes are subject to constraints. There is a limited field of action and a determined slew rate for the actuators. Signals must not violate specified bounds due to safety limitations, consumer specifications, environmental regulations, and physical restrictions. These last restrictions can be in terms of temperature, flow in pipes, level limits in reactor tanks, pressure and slew rates of valves \[58\]. Through careful regularization, it is possible to keep values far beyond the stipulated limits. However, for economic benefits, the control system should generally guide the process as close as possible towards the constraints, but never in violation of these restrictions. This occurs, since, being more and more close to the limits corresponds to having an increasing profit \[58\].
Another approach employed by predictive control, that is considered more direct, corresponds to changing the optimal unconstrained solution to prevent constraint violation. Quadratic or linear programming (QP and LP) are different optimization methods which can be employed for this purpose. Different linear constraints are considered by predictive algorithms, having a great impact on the industry. Constraints can be applied to all kinds of variables, restricting slew rate, amplitude and output limits of the control signals \[12, 47, 52, 58\]. Some constraints considered can be observed in equation (2.11)

\[
\begin{align*}
    u_{\text{min}} & \leq u(k) \leq u_{\text{max}}, \forall k, \\
    \Delta u_{\text{min}} & \leq \Delta u(k) \leq \Delta u_{\text{max}}, \forall k, \\
    x_{\text{min}} & \leq x(k) \leq x_{\text{max}}, \forall k, \\
    y_{\text{min}} & \leq y(k) \leq y_{\text{max}}, \forall k.
\end{align*}
\] (2.11)

### 2.4 Nonlinear Model Predictive Control

Model Predictive Control can be subdivided into two categories: linear and nonlinear Model Predictive Control (MPC). LMPC uses linear models to predict the system dynamics, and the constraints applied to the states and to the inputs are also linear \[3\]. The main reasons why LMPC is widely used are: (1) if the plant is working near the operating point, the results obtained by the linear model have a very small error, and (2) the process of obtaining a linear model from process data becomes easy. In many industrial processes, where the main goal is to maintain the operation near the stationary state, and where there is no need to frequently switch between operation points, it is enough to use a precise linear model. One advantage of using a linear model combined with a quadratic cost function is that the problem that emerges corresponds to a convex problem, which has a result that is well studied.

However, many problems have nonlinear dynamics and, therefore, the result obtained by the linear model does not correspond to the most accurate one. Nonlinear models are widely used and the appropriate situations for its use are: (1) when the nonlinearities are very strict and very important to maintain the closed-loop stability, and (2) when proceedings have ongoing transitions and are maintained far from the steady-state operating region for long periods. In this case, a nonlinear control law should be adopted to obtain a more stable operating system and improve overall performance \[12\].

Although NMPC brings benefits when dealing with nonlinear dynamics, there are still some difficulties when using these models such as \[12\]:

- In some situations, the optimization problem becomes slow, increasing the computation time;
- Identifying nonlinear processes based on experimental data is a difficult problem to be solved;
- Stability and robustness are more complex to be studied when it comes to nonlinear models;
- Problems resulting from these models are nonconvex, and its resolution is more complex than when dealing with QP. Problems regarding stability, local optimum and control quality arise.

As quality and performance requirements increase, NMPC is becoming a more and more studied field, being possible to solve many of the problems previously mentioned.
Chapter 3

Optimization

3.1 Optimal Control Theory

Optimal Control theory is widely applied in different scientific fields as a mathematical tool. It can be used in physics, engineering, machine learning, and even quantum chemistry [29]. The idea behind OCT is to obtain an acceptable control function, that can be applied to the system, to achieve an optimality criterion. To achieve this goal a cost function should be minimized, allowing the development of an evaluation method that shows how close the calculated value is to the final objective [1].

Different optimization methods can be used to minimize the cost function. An initial analysis is required to understand what kind of problem is being studied to choose the best method and solver for the optimization process. The ideal would be to deal with a convex problem, since it is possible, more directly and with weaker conditions, to develop an algorithm to find an optimal solution. However, in some cases, the problems are not convex, being necessary to take a different approach [39].

3.2 Convex and Non-convex Optimization

As mentioned before, there are two types of optimization problems: Convex (with convex cost function, constraints or feasible set) and Non-convex (with non-convex cost function, constraints or feasible set).

Several advantages can be obtained when using convex optimization for problem resolution. With convex optimization, it is possible to foresee how much time it is needed to reach the final solution since it can ensure the theoretical convergence of gradient descent. Also, when a local optimal solution is found the optimization algorithm can stop since in convex problems the local optimal solution corresponds to the global optimal solution [13]. On the contrary, a non-convex problem may have several local minimums, being difficult to find an optimal solution. One method widely used to solve non-convex optimization problems consists in relaxing the initial non-convex problem to a convex problem. This can be achieved by applying convex optimization to calculate, on non-convex problems, lower and upper bounds, being possible to apply these bounds to obtain a globally optimal solution [13]. In the next figure, it is possible to observe a visual representation of a convex and non-convex problem.
For convex or non-convex problems, it is possible to use several optimization methods together to find the optimal value. In the next section two different optimization methods are addressed, such as: Unconstrained (fminunc) and Constrained (fmincon) solvers.

### 3.3 Unconstrained solver

The unconstrained solver (fminunc), has as main goal to find a minimum of a scalar function composed of several variables, beginning its computation at an initial point given by the user. This solver is a nonlinear programming algorithm that finds the minimum of a problem specified by

\[ \min_x f(x), \]  

where \( f(x) \) is a function that returns a scalar value. This solver is generally referred to as unconstrained nonlinear optimization, this is, it does not have constraints that should be respected. The syntax that can be adopted, when using the MatLab tool is represented in equation \( 3.2 \)

\[ x = \text{fminunc}(\text{fun}, x_0, \text{options}, P_1, P_2, \ldots), \]  

where \( \text{fun} \) is the function to be minimized, \( x_0 \) is the initial point given to the solver to start the optimization process, \( \text{options} \) is the structure with the optimization parameters, and \( (P_1, P_2, \ldots) \) are the problem-dependent parameters that are passed directly to the function \( \text{fun} \).

Although the previous method is very important to understand the first steps for the optimization process, it only deals with non-constrained problems. Almost all problems imply restrictions, being necessary to resort to another method that can insert them into its optimization process. In the next section, the constrained solver is explained, and the differences between the two methods are presented.
3.4 Constrained solver

The constrained solver (fmincon) has a similar objective as the unconstrained solver (fminunc), that is, to find a minimum to a specific provided function. In this case, the function corresponds to a constrained nonlinear multivariable function and can be observed in equation (3.3) [54]

\[
\min_x f(x) \text{ such that } \begin{cases} c(x) \leq 0, \\ c_{eq}(x) = 0, \\ A \cdot x \leq b, \\ A_{eq} \cdot x = b_{eq}, \\ x_{lb} \leq x \leq x_{ub}, \end{cases}
\]  

(3.3)

where \( x, b, b_{eq}, x_{lb}, \) and \( x_{ub} \) are vectors, \( A \) and \( A_{eq} \) are matrices, \( c(x) \) and \( c_{eq}(x) \) are functions that return vectors, and \( f(x) \) is a function that returns a scalar. Function \( f(x), c(x), \) and \( c_{eq}(x) \) can be nonlinear functions. The syntax that can be used using the MatLab tool can be represented as [54]

\[
x = \text{fmincon}(\text{fun}, x_0, A, b, A_{eq}, b_{eq}, x_{lb}, x_{ub}, \text{nonlcon}, \text{options}, P_1, P_2, \ldots),
\]  

(3.4)

where \( \text{fun} \) is a constrained nonlinear multivariable function, \( x_0 \) is the initial value provided to the optimization method, \( \text{nonlcon} \) is one or more nonlinear constraints, \( \text{options} \) is the structure with the optimization parameters, and \((P_1, P_2, \ldots)\) are the problem-dependent parameters that are passed directly to function \( \text{fun} \) and \( \text{nonlcon} \). Taking into consideration a problem with an initial point \( x_0 = [0.5, 1] \), with two constraints, \((x(1) + 2x(2) \leq 1)\) and \((2x(1) + x(2) = 1)\) and with lower and upper bound, \([0, 0]\) and \([1, 2]\) respectively, it is possible to represent the fmincon data as shown below

\[
A = [1, 2], \quad b_{eq} = 1, \\
b = 1, \quad u_{lb} = [0, 0], \\
A_{eq} = [2, 1], \quad u_{ub} = [1, 2].
\]  

(3.5)

Nonlinear inequalities and equalities observed in (3.3), represented by \( c(x) \leq 0 \) and \( c_{eq}(x) = 0 \), respectively, are defined in a separate function designated by \( \text{nonlcon} \). This function solves the optimality of the system subjected to the defined nonlinear constraints.

In section 3.8 examples using both fminunc and fmincon are presented to understand how both methods work and how they can be applied to an optimization problem. Although fminunc is an unconstrained solver, it is important to understand the bases of the optimization solvers. In the next section, possible algorithms for fmincon are discussed and their differences are presented.

3.5 Optimization Algorithms

To implement the constrained solver (fmincon) it is necessary to choose an algorithm that will be used to optimize the problem. These algorithms are incorporated in the options structure observed in equation (3.4). In this section, two distinct algorithms that can be used to execute the simulations are introduced, presenting their differences and some situations where each one of them can be used.
As mentioned in the State of the Art, there are two different categories of solvers: offline and online. Both set of solvers are widely used. However, since online solvers are more accurate, being able to solve real-time problems, avoiding dynamic obstacles, there is a more detailed study about how this set of methods are applied. Different online algorithms can be used for the optimization process such as interior-point and SQP. In the next table, the differences between them are presented.

<table>
<thead>
<tr>
<th>Interior-point</th>
<th>SQP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Slower</td>
</tr>
<tr>
<td>Confidence</td>
<td>More stable for calculating the optimal solution</td>
</tr>
<tr>
<td>Large number of active set changes</td>
<td>Faster</td>
</tr>
<tr>
<td>Computational Complexity</td>
<td>Lower</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Less Efficient</td>
</tr>
</tbody>
</table>

Table 3.1: Differences between Interior-point and SQP algorithms [30], [62].

The Interior-point algorithm is mostly used when the problem is sparse or is a small dense problem. In each iteration, all boundaries are respected, and it is a large-scale algorithm, that is, it uses linear algebra and does not need to save or work with full matrices. The SQP, like the Interior-point, respects at each iteration all boundaries, however, it is not a large-scale algorithm. The Interior-point has barrier functions that help the optimization process, that is, in the existence of many constraints the barrier functions allow to maintain the search of optimization within the feasible zone of the function. In contrast, SQP does not have these barrier functions, allowing the algorithm to look in more detail for the optimal solution in the vicinity of the constraints. Sometimes this is favorable, however, in the existence of many restrictions this algorithm may present unfavorable solutions and it is, therefore, preferable to use the Interior-point which, although presenting less accurate solutions, are nevertheless favorable.

During the optimization process, the constrained method considers several restrictions that can be applied to the problem. In the next section, some examples of possible constraints are presented both in terms of equations and in terms of visual aspects.

### 3.6 Constraints for constrained optimization solver

This section focuses on the use of fmincon since this is the solver to which constraints may be applied. Different constraints can be used and some of them are linear, if all the components of the constraint are of first order, and others are non-linear, where, contrary to linear constraints, the terms can be raised to any power, can be divided by another variable, or variables multiplied by other variables. In both cases, it is possible to have equality and inequality constraints. Linear equalities and inequalities have the form represented in equation (3.6) and (3.7), respectively [54]

\[ A_{eq} \cdot x = b_{eq}, \]  

(3.6)
where $A_{eq}$ is a $m$-by-$n$ matrix, where $m$ represents the number of constraints applied to a variable $x$ with $n$ components, and $b_{eq}$ is a $m$-component vector, and

$$ A \cdot x \leq b, \quad (3.7) $$

where $A$ and $b$ have the same meaning as in the previous equation. Nonlinear equalities and inequalities can be represented by equation (3.8) and (3.9), respectively.

$$ c_{eq}(x) = 0, \quad (3.8) $$

$$ c(x) \leq 0, \quad (3.9) $$

where $c_{eq}$ and $c$ are vectors of constraints, that have one component for each one of the constraints.

Linear constraints can be applied at the level of input ($u$), output ($y$) and state variables ($x$) as demonstrated in equation (2.11). In addition to linear, there are also nonlinear constraints that correspond to nonlinear models, such as described by equation (2.2). With nonlinear constraints, it is possible to restrain the final solution to a desired region which can be represented in terms of smooth functions. Nonlinear constraints may correspond to equations that define certain prohibited zones in space with diverse shapes, limiting the values that the future state variables can acquire. Some examples of different shapes are circles, squares and diamonds and are presented in equation (3.10).

$$ \begin{align*}
(x_1 - c_1)^2 + (x_2 + c_2)^2 &< d_{min}^2, \\
|x_1 - c_1| + |x_2 - c_2| &< d_{min}, \\
\max\{|x_1 - c_1|, |x_2 - c_2|\} &< d_{min},
\end{align*} \quad (3.10) $$

where the first inequality defines a circumference, the second defines a diamond, and the last one defines a square. Parameter $d_{min}$ is the radius (first inequality), the length from the tip to the center (second inequality) and the length from the side to the center (third inequality). The visual representation of each one of the inequalities previously presented can be observed in figure 3.2.

![Figure 3.2: Three types of nonlinear constraints with different shapes.](image)

To obtain a viable solution from the optimization solver, all constraints (linear and nonlinear), should be respected. When sometimes this is not possible and becomes difficult it is necessary to
resort to other variables that modify the behavior of the system. These variables are the $Q$ and $R$ matrices from the cost function implemented for the optimizer.

### 3.7 The $Q$ and $R$ matrices

As mentioned before, $Q$ and $R$ matrices have a huge impact on the minimization of the cost function $J$. One example of quadratic cost function can be represented as

$$J = \sum_{i=1}^{H} (x^T Q_r x + u^T R_r u).$$

(3.11)

The cost function entails a compromise between keeping the state low without much activity in control. The diagonal entries of $Q_r$ can tell which entries of the state variable $x$ are more important. The higher the value $Q_{rij}$ the lower the modulus of the entry $x_{ij}$. Increasing the value of $R_r$ the modulus of $u$ decreases, allowing $x$ to have a higher value. With this, the closed-loop system becomes slower $[31]$. These parameters are very important for the optimization process. By modifying them, it is possible to change the dynamics of the system allowing, as mentioned in the previous section, to respect all constraints imposed to the model. For example, by increasing the value of $R_r$ the system becomes slower, allowing a more relaxed analysis of all constraints involving each state. With an improved processing analysis, better values for the state can be selected and a better final solution can be achieved.

Matrix $Q$ is square with a size equal to the horizon $H$, chosen initially for the optimization process, and $R$ is a column vector also the size of the horizon. In the next section, examples using the two solvers, previously introduced, are presented. In these examples, constraints are applied and different values for $Q$ and $R$ are chosen, to understand what are the influences of these values on the optimization process.

### 3.8 The influence of tuning parameters

As stated previously, whenever constraints are not respected, it is necessary to modify the parameters ($Q$ and $R$ matrices) that constitute the cost function presented in equation (3.11). In some cases, another parameter, which has much influence on the behavior of the system is the horizon ($H$) of the solver. Each parameter has a different influence on the evolution of the system. Therefore, to better understand their importance, illustrative examples, using both \textit{fminunc} and \textit{fmincon} solvers, are presented, where parameters $Q$ and $R$ are modified. The model described by equation (3.12) will be used to present each example. To study the solution of the model with more detail, appendix B can be analyzed.

$$y(k) - 1.5y(k - 1) + 0.7y(k - 2) = u(k - 1) + 0.5u(k - 2).$$

(3.12)

To obtain the necessary parameters for the cost function minimization, it is necessary to make additional calculations. These supplementary calculations were performed from equation (B.2) to equation (B.7) demonstrated in the appendix B. Thus, the parameters that were obtained for the model are

$$A = \begin{bmatrix} 0 & 1 \\ -0.7 & 1.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0.5 & 1 \end{bmatrix}.$$
Based on the previous parameters it is possible to observe that the model presented in equation (3.12) is a two-state system. After that, the \textit{fminunc} solver is applied. The value of $R$ was varied to be able to understand how it influences the evolution of the final state. The remaining parameters ($Q$ and $H$) were kept constant. For the examples involving parameter $R$, only one state variable was used since a similar result was obtained for the second one. The result obtained can be observed in figure 3.3.

![Figure 3.3: Evolution of the final state error by changing the value of $R$ from 0 to 50.](image)

Analyzing figure 3.3 it is possible to verify that by increasing $R$, the error between the final state and the reference grows, meaning that the final value departs from the true value that it should acquire.

Considering the cost function from equation (3.11) it is possible to notice that as the value of $R$ increases, the modulus of input $u$ decreases, giving more importance to the state $x$, allowing it to have higher values. Since the goal is to minimize the cost function, by giving a high value to parameter $R$, the first term of equation (3.11), multiplied by matrix $Q$, will be despised, giving greater importance to the minimization of $u$ and therefore, this parameter will have lower values. Having low input values will not be enough to make the system converge to the reference, and so the result will depart from the desired value, as previously mentioned. This behavior can be better understood by looking at figure 3.4.

![Figure 3.4: State evolution for different values of $R$.](image)

From the previous figure, it is concluded that with the increase of $R$, the system becomes slower,
and consequently, stabilizes at a value that becomes lower and lower, depending on the value of $R$, failing to reach the reference. Considering the same model that was previously used, applying the \textit{fminunc} solver, the diagonal values of the $Q$ matrix were modified so that their influence on the evolution of the final state could be studied. The remaining parameters ($R$ and $H$) were kept constant. In this case, as before, only one state variable was used, since the result obtained with the other state variable was the same. In this way, the result obtained for the final state can be observed in figure 3.5.

![Figure 3.5: Evolution of the final state error by changing the value of $Q$ from 1 to 50.](image)

From figure 3.5 it is observed that by increasing the diagonals of $Q$, the error between the final state and the reference decreases, meaning that the final value approaches more and more of the desired goal. Considering the cost function from equation (3.11) it is verified that by increasing the diagonals of $Q$, the modulus of $x$ decreases, giving priority to input $u$, allowing it to have higher values.

In this case, by assigning higher values to the entries of $Q$ will be giving greater importance to the minimization of state $x$, which will acquire lower values. The second term of equation (3.11), multiplied by parameter $R$, will be neglected. Having low values for the states will lead to higher values for the input $u$, causing the system to converge faster. This effect can be verified in figure 3.6.

![Figure 3.6: Evolution of state for different values of $Q$.](image)

Observing figure 3.6 it is possible to confirm that by increasing the value of the entries of $Q$ the
system converges more rapidly, taking less time to stabilize near the desired reference. Contrary to what happens when changing the value of $R$, in this case, the system can progressively approximate to the goal, with an error that decreases as the diagonal entries of $Q$ increase, as shown in figure 3.5.

To reduce the errors obtained previously, both by modifying the value of $R$ and by modifying the diagonal entries of $Q$, an extra term has been added to the cost function presented in equation (A.10) that prevents the output of the system from moving away from the reference. This extra term can be obtained based on the static gain of the system, that can tell the ration of the output and the input under steady-state condition. Using equation (B.2), with $z = 1$ it is possible to obtain

$$Y = \frac{1 + 0.5}{1 - 1.5 + 0.7} \bar{U} = \frac{1.5}{0.2} \bar{U} \Rightarrow \bar{U} = \frac{1}{7.5} Y,$$

(3.13)

where $Y$ corresponds to the reference and $\bar{U}$ corresponds to the optimal input value applied to the system. Therefore, by adding this extra term to equation (A.10) it is possible to obtain a new equation that will allow the system output to converge to the reference. This new equation can be rewritten as

$$J = (WU + \Gamma x - \tilde{Y})^T Q (WU + \Gamma x - \tilde{Y}) + (U - \bar{U})^T R (U - \bar{U}).$$

(3.14)

Considering the extra added term, the effect of the variation of parameter $R$ and the variation of the diagonal entries of matrix $Q$ can be observed in figures 3.7a and 3.7b.

(a) State evolution for different values of $R$.  
(b) State evolution for different values of $Q$.

Figure 3.7: System behaviour with different values of $R$ and $Q$.

Analyzing the previous figures, it is possible to observe that the extra added term totally reduced the error that existed between the final value and the reference, allowing, in this way, in both situations, the system to converge to the desired value.

Another parameter, which is usually changed to reach the goal is the horizon $(H)$. It is necessary to have a sufficiently large horizon, to be able in advance, to make appropriate decisions, allowing the system to converge to the reference. In chapter 5, examples using different vehicle models and different values of the horizon will be presented. These simulations will use constraints that will allow to better understand the influence of parameter $H$. 

25
Chapter 4

Vehicle models

As mentioned before, there are many models to represent and describe the behaviour of a vehicle. In this chapter three models with different dynamics are addressed, allowing to perceive and to understand how each one of them behaves, what are the differences between them, and which are the state equations that allow to calculate the future state variables.

4.1 Single integrator

In this section, the Single Integrator is introduced. This method corresponds to the simplest vehicle model representation expressed in Cartesian coordinates and it can be used for a linear formulation. The motion of the vehicle in a single integrator model can be represented as

\[ \dot{x} = u, \]  

(4.1)

where \( \dot{x} = [x_1, x_2]^T \) corresponds to the states of the system and \( u = [u_1, u_2]^T \) corresponds to the control vector. The discretized equivalent for the method presented in equation (4.1) can be defined as

\[ x_1(k+1) = x_1(k) + h \cdot u_1(k), \]
\[ x_2(k+1) = x_2(k) + h \cdot u_2(k), \]

(4.2)

where \( h \) is the sampling time. Therefore, the state-space representation, that is, the parameters that characterize the single integrator model, in matrix notation, can be represented as

\[ x(k+1) = \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} h & 0 \\ 0 & h \end{bmatrix}, \quad C = A. \]

(4.3)

4.2 Double integrator

The double integrator is a model that can be characterized by the expressions presented in equation (2.5) and (2.9). A example of a double integrator plant is described by the following expressions.
\[ x_i(k+1) = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} x_i(k) + \begin{bmatrix} \frac{h^2}{T} \\ h \end{bmatrix} u(k), \]
\[ y_i(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x_i(k), \]

where \( i = 1, 2 \), and \( h \) is the sampling interval. Taking into consideration the previous state and output equations it is possible to draw the following conclusions
\[
x = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{h^2}{T} \\ h \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix},
\]

where \( q_1 \) and \( q_2 \) are the position and the velocity of a body, respectively. Considering what was previously stated, it is possible to conclude that the double integrator relates the applied force to the velocity, if there is no friction between the two surfaces, and velocity to position. The relation between the applied force and the velocity corresponds to a double integration, and the relation between the velocity and the position corresponds to another double integration. In this way, the vehicle model will have a double integrator \((\frac{1}{s^2})\) in the Cartesian coordinate \( x \) and another double integrator \((\frac{1}{s^2})\) in the Cartesian coordinate \( y \).

Considering the notation presented in equation \( (A.1) \), the state space representation of a vehicle characterized by the double integrator model in a matrix format can be represented by
\[
\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{bmatrix} = \begin{bmatrix} 1 & h & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} + \begin{bmatrix} \frac{h^2}{T} \\ h \\ \frac{h^2}{T} \\ h \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix},
\]

where \( x_1(k+1) \) and \( x_2(k+1) \) are the position and velocity, in the \( x \) coordinate, respectively, and \( x_3(k+1) \) and \( x_4(k+1) \) are the position and velocity in the \( y \) coordinate, respectively.

Assuming the notation described in equation \( (A.2) \), the output representation of a vehicle characterized by the double integrator model in a matrix notation can be represented by
\[
\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix}.
\]

### 4.3 Tricycle

In this section, the tricycle model is described. This model considers that a vehicle is composed of two steerable drive wheels and a fixed wheel. The model of the vehicle, in the Cartesian coordinates, can be observed in figure 4.1. Parameter \( \mathbf{v} \) corresponds to the normalized velocity vector and \( s \) is the counterclockwise rotation of \( \mathbf{v} \). Using tricycle model, it is possible to ignore two distinct questions related to balancing and slippery problems. Disregarding slippery issues, it enables the translation speed to be in the direction of heading, allowing the contact between the wheels and the ground to be pure.

To tackle this model a vehicle state (vehicle posture in base frame) \( \mathbf{q} = [x, y, \theta]^T \), constituted by the position and orientation, is defined being related to a global coordinate system. Another important
information is that the speed of each wheel can be controlled by a controller using a feedback model [45]. The situation in which the differential drive vehicle, with linear velocity $v$ and angular velocity $\omega$, is describing a curve of radius $R$ can be observed in figure 4.2 [45].

If the slippery problem is ignored the following equations can be considered [45]

$$\omega_L = \frac{v_L}{r} (R - \frac{l}{2}) = (v - \frac{\omega l}{2}) \frac{1}{r}, \quad (4.8a)$$
$$\omega_R = \frac{v_R}{r} (R + \frac{l}{2}) = (v + \frac{\omega l}{2}) \frac{1}{r}, \quad (4.8b)$$

where $\omega_R$ and $\omega_L$ are the angular velocities of the right and left wheels, respectively. The parameters $v_R$ and $v_L$ correspond to the linear velocities of each wheel. Parameter $l$ is the distance between the wheels and $r$ is the radius of each wheel. By solving equations (4.8a) and (4.8b) in terms of $\omega$ and $v$, the following equations can be obtained [45]

$$\omega = \frac{v_R - v_L}{l} = \frac{\omega_R - \omega_L}{l} r, \quad (4.9a)$$
$$v = \frac{v_R + v_L}{2} = \frac{\omega_R + \omega_L}{2} r. \quad (4.9b)$$

Based on these equations it is possible to transform $[v, w]^T$ and $[w_L, w_R]^T$. A controller commonly considers an input $u = [\omega_L, \omega_R]^T$, and with this input, the motion kinematics model in the world frame of
a differential drive vehicle can be described using the following nonlinear dynamical system \[6, 45\]
\[
\dot{q}(t) = \begin{bmatrix}
\dot{x}(t) \\
\dot{y}(t) \\
\dot{\theta}(t)
\end{bmatrix} = \begin{bmatrix}
v(t)\cos(\theta(t)) \\
v(t)\sin(\theta(t)) \\
\omega(t)
\end{bmatrix},
\]
(4.10)
where \(q\) corresponds to the robot kinematic state. To apply this model in discrete time, there is a need for a discretization process, allowing the transition from a continuous to a discrete model. The equations that allow this relation can be described as
\[
x((k + 1)h) = x(kh) + \int_{kh}^{(k+1)h} v(\sigma)\cos(\theta(\sigma))d\sigma,
\]
(4.11a)
\[
y((k + 1)h) = y(kh) + \int_{kh}^{(k+1)h} v(\sigma)\sin(\theta(\sigma))d\sigma,
\]
(4.11b)
\[
\theta((k + 1)h) = \theta(kh) + \int_{kh}^{(k+1)h} \omega(\sigma)d\sigma.
\]
(4.11c)

Different methods of approximations can be used to obtain more simplified expressions than the previous ones, by reducing the integral term. Some methods of approximation that can be considered are the Euler and the Runge-Kutta method.

4.4 Euler method

Euler’s method is a numerical process that is used to approximate the solutions to explicit first-order equations with a given initial value. This method is based on making successive linear approximations to the solution [11]. Using Euler method, the local error (error per step) is proportional to the square of the step size, and the global error (error at a given time) is proportional to the step size. This process is usually used as the basis to construct and develop more complex methods.

Figure 4.3 visually shows how the approximation process is done applying the Euler method to a general function \(f(x(\sigma), v(\sigma))\), similar to the one presented inside the integral term in equation (4.11).

![Figure 4.3: Approximation made using the Euler method.](image)

29
From the previous figure, it is possible to conclude that the integral term from equation (4.11a) and (4.11b) can be approximated by
\[
\int_{k \cdot h}^{(k+1) \cdot h} f(x(\sigma), v(\sigma)) \cdot \sigma = h \cdot f(x((k+1) \cdot h), v((k+1) \cdot h)),
\]
where the approximation of the integral is the area of a rectangle and \( h \) is the integration step. Considering Euler approximation, the tricycle model, in discrete time, is described by the following
\[
x(k+1) = x(k) + v(k) \cdot \cos(\theta(k)) \cdot h,
\]
\[
y(k+1) = y(k) + v(k) \cdot \sin(\theta(k)) \cdot h,
\]
\[
\theta(k+1) = \theta(k) + \omega(k) \cdot h.
\]

### 4.5 Runge-Kutta method

The Runge-Kutta algorithm is like the Euler and improved Euler methods but rather than approximating the integral by the area of a rectangle, as does Euler, it approximates by the area of a trapeze. In figure 4.4 the Runge-Kutta approximation for the function \( f(x(\sigma), v(\sigma)) \) can be observed.

![Approximation made by second order Runge-Kutta method.](image)

Analyzing figure 4.4 it is possible to conclude that the approximation of the integral presented in equation (4.11a) and (4.11b) can be simplified as shown in the next equation
\[
\int_{k \cdot h}^{(k+1) \cdot h} f(x(\sigma), v(\sigma)) \cdot \sigma = \frac{1}{2} \left( f(x((k+1) \cdot h), v((k+1) \cdot h)) + f(x(k \cdot h), v(k \cdot h)) \right) \cdot h
\]
where, the approximation of the integral is the area of a trapeze and \( h \) is the integration step. Considering the Runge-Kutta approximation, the tricycle model, in discrete time, is described by the following
\[
x(k+1) = x(k) + \frac{1}{2} \left( v(k+1) \cdot \cos(\theta(k+1)) + v(k) \cdot \cos(\theta(k)) \right) \cdot h,
\]
\[
y(k+1) = y(k) + \frac{1}{2} \left( v(k+1) \cdot \sin(\theta(k+1)) + v(k) \cdot \sin(\theta(k)) \right) \cdot h,
\]
\[
\theta(k+1) = \theta(k) + \frac{1}{2} \left( \omega(k+1) + \omega(k) \right) \cdot h.
\]
Chapter 5

MPC for path planning

In this chapter, predictive control techniques are considered to calculate the sequence of optimal system inputs from which it is possible to obtain a trajectory for the vehicle, minimizing the cost. Before calculating all this, it is necessary to answer some important questions: what cooperative control is, how it is going to be implemented and what are the types of cooperative control used to solve this problem.

5.1 Cooperative Control

Model Predictive Control is comprised of several control methods which are used to minimize a cost function and obtain a control input for the plant model. Considering the model of a vehicle, the MPC methods are applied to obtain a sequence of inputs used to calculate the future positions of the vehicle, allowing to define a trajectory between two points. This process is called path planning and can be done for a single vehicle or for a set of vehicles that move along that path cooperatively.

Cooperative control consists of a rule-enforcing control method that allows a set of dynamic entities to share information to achieve a final common goal. In the past years, this topic has become one of the most discussed control subjects, gaining a great interest in scientific investigation, especially in the field of mobile robotics [10]. Therefore, in robotics, cooperative control consists of the ability of a set of vehicles to perform common and shared tasks, each one of them being dependent on the location and information transmitted by the others, at each moment. Cooperative control is widely used in different fields and for many purposes as air traffic control, collective attacks, cooperative surveillance and transportation, formation flight, underwater exploration, rescue missions, among others [53, 59].

Cooperative control can be achieved using different methods for example game theory or even distributed control that can be divided into centralized and decentralized. In addition to these two methods, others allow establishing a distributed cooperative control method between vehicles.

In decentralized control, it is possible to divide the multi-vehicle system into several subsystems. In this method each vehicle, based on the collected information and considering the decisions of the remaining agents, chooses the best action. A communication protocol is defined to exchange information between them. Since the main problem is divided into several smaller problems, less computational work
is required, and therefore, this method is ideal to be used in online optimization problems [35].

In centralized control, the system composed of all vehicles is considered as a team. In this method, there is a central controller that, based on current system state information, can achieve a global control strategy and then send it to all members. For this method to work properly, it is necessary to ensure a stable communication between the vehicles and the central control system. Considering that all agents are dependent on a central control system it is possible to affirm that the individual vehicles are less autonomous than in the decentralized method [35].

In game theory a game is made between vehicles, where agents exchange their optimal decisions. These decisions are made based on all the information gathered from the environment and on the choices made by the other vehicles. The exchange is done until a decision if found that is considered optimal for all agents. Only then the result is applied to the dynamics of the vehicles.

During the next sections, a different cooperative control method will be used to represent the interactions between agents. In this method, a main vehicle, based on environmental information, makes decisions an transmits them to the others. The remaining vehicles, based on this information and on the data collected from the environment, calculate their optimal choice, respecting all the constraints.

5.2 Minimizing the trajectory cost

In this section, a simple problem is presented. The goal is to calculate, using Model Predictive Control, the optimal trajectory for a vehicle to move from a point to another. This trajectory corresponds to the best states calculated based on the inputs, that minimize the total cost to reach the reference.

To satisfy this requirement, it is necessary to obtain a variable that is used to characterize the evolution of the path cost over time. This variable can be obtained by solving a cost function that is the sum of the difference between the current state and the final state plus the input applied to the vehicle, for a given horizon. Now the main question is: should the input used in the cost function be squared or it should be the absolute value? Assuming a horizon $H$, the considered cost functions are

$$J(u) = \sum_{i=2}^{H} \left( (\hat{y}(i) - Y)^T Q(\hat{y}(i) - Y) + (u(i-1))^T R(u(i-1)) \right),$$  \hspace{1cm} (5.1a)

$$J(u) = \sum_{i=2}^{H} \left( (\hat{y}(i) - Y)^T Q(\hat{y}(i) - Y) + R \| u(i-1) \| \right),$$  \hspace{1cm} (5.1b)

where $\hat{y}(i), i = 1, 2...H$ represent the states for the horizon $H$, $Y$ represents the reference that should be achieved and $u(i-1)$ represents the input applied to the vehicle dynamics in order for the agent to move to state $\hat{y}(i)$. The input $u$ is limited to certain values so that the velocity at $x$ and $y$ coordinates do not exceed $\pm 20$ m/s. The negative velocity represents the situation where the vehicle has moved backward to circumvent a certain obstacle since that is the best trajectory to minimize the total cost.

The main difference distinguishing the two cost functions is that in equation (5.1a), since input $u$ is squared, it gets amplified when its value is greater than one ($u > 1$), and is attenuated when the input is lower that one ($u < 1$). In this way, let’s take into account the double integrator model of the vehicle, with a radius of 2 meters, presented in equation (4.6), for the states and in equation (4.7) for
the output, and add a spatial constraint with circular format as presented in figure 3.2a. Therefore, the problem definition, based on both objective functions, can be written as follows

\[
\begin{align*}
\min_u & \quad \sum_{i=2}^{H} (\hat{y}(i) - Y)^T Q (\hat{y}(i) - Y) + \Phi(u) \\
\text{subject to} & \quad x(1, i + 1) = x(1, i) + h \cdot x(2, i) + \frac{h^2}{2} \cdot u(1, i), \quad x(1, 1) = 1, \\
& \quad x(2, i + 1) = x(2, i) + h \cdot u(1, i), \quad x(2, 1) = 0, \\
& \quad x(3, i + 1) = x(3, i) + h \cdot x(4, i) + \frac{h^2}{2} \cdot u(2, i), \quad x(3, 1) = 2, \\
& \quad x(4, i + 1) = x(4, i) + h \cdot u(2, i), \quad x(4, 1) = 0, \\
& \quad (x(1, i + 1) - c_1)^2 + (x(3, i + 1) + c_2)^2 < d_{\text{min}}^2, \\
& \quad |x(2, i + 1)| \leq 20, \\
& \quad |x(4, i + 1)| \leq -20,
\end{align*}
\]  

(5.2a)

(5.2b)

(5.2c)

(5.2d)

(5.2e)

(5.2f)

(5.2g)

(5.2h)

where \( \Phi(u) \) can either be \((u(i - 1))^T R (u(i - 1)) \) or \( R \|u(i - 1)\|. \) Parameters \( x(1, i) \) and \( x(3, i) \) are the \( x \) and \( y \) coordinates, respectively, \( x(2, i) \) and \( x(4, i) \) are the velocities associated with \( x \) and \( y \) coordinates, respectively, and \( c_1 \) and \( c_2 \) are the \( x \) and \( y \) coordinates of the center of the obstacle with a radius equal to \( d_{\text{min}} \). The radius of the obstacle is equal to 5 meters and since constraints are applied to the problem, \textit{fmincon} solver is used with the \textit{SQP} algorithm. Before proceeding with the simulation, it is necessary to define what are the values used for the horizon \( H \), the sample time \( h \), the number of iterations \( Z \), the matrix \( Q \) and \( R \) and the final state \( Y \) to be achieved. These values can be observed in table 5.1.

<table>
<thead>
<tr>
<th>( H )</th>
<th>( h )</th>
<th>( Z )</th>
<th>( Q )</th>
<th>( R )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1</td>
<td>200</td>
<td>\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}</td>
<td>\begin{bmatrix} 0.1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0.1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0.1 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0.1 \end{bmatrix}</td>
<td>\begin{bmatrix} 50 \ 0 \ 50 \ 0 \end{bmatrix}</td>
</tr>
</tbody>
</table>

Table 5.1: Simulation parameters.

The input variation for both cost functions can be observed in figure 5.1a and 5.1b. In these figures, it is presented only the variation of the input \( u \) associated with \( x \) coordinate, since the result for the input associated with \( y \) direction is very similar.

Analyzing both figures it is possible to observe differences in the evolution of \( u \). The input associated with the cost function from equation (5.1b) exhibits a bang-bang behavior. This effect occurs when there are restrictions on the input, limiting the controller to be between a lower and upper bound, and therefore \( u \) changes from one extreme to the other as observed in figure 5.1b. This behavior can sometimes be preferable due to the blunt variation of the control since it can minimize the effort made, improving performance in terms of the vehicle expenses, as opposed to what happens in the quadratic controller, where this variation is less sudden. On the contrary, figure 5.1a shows a smoother variation of the input allowing to eliminate the bang-bang effect that sometimes may not be desirable due to the abrupt input variation, making the controller take actions that do not respect all constraints.
(a) Variation of $u$ with time $k$ using equation (5.1a).
(b) Variation of $u$ with time $k$ using equation (5.1b).

Figure 5.1: Input $u$ variation for both cost functions.

To understand which function minimizes more the system expenses, the total cost was calculated for both cases. Therefore, the results for each function can be observed in equation (5.3) and (5.4)

$$J = \sum_{i=2}^{100} (\hat{y}(k+i) - Y)^T Q (\hat{y}(k+i) - Y) + (u(k+i-1))^T R (u(k+i-1)) = 5.40 \times 10^5. \quad (5.3)$$

$$J = \sum_{i=2}^{100} (\hat{y}(k+i) - Y)^T Q (\hat{y}(k+i) - Y) + R \| u(k+i-1) \| = 3.51 \times 10^5. \quad (5.4)$$

The percentage of margin between both cost functions is presented in equation (5.5)

$$diff = \frac{5.40 \times 10^5 - 3.51 \times 10^5}{5.40 \times 10^5} \times 100 = 35\% . \quad (5.5)$$

Although from the previous result it is possible to see that the cost obtained by the cost function presented in equation (5.1a) is higher than the one obtained by equation (5.1b) by 35%, both results are of the same order of magnitude, and therefore both cost functions can be used for path cost minimization.

To prove the similarity in the minimization process using each function, the path obtained for the vehicle from its starting point to the target, including the circular obstacle, are presented in the next figures.

5.3 Influence of horizon on optimization

The prediction horizon $H$ has a great influence on the optimization process allowing the vehicle to have a smaller or larger forecasting capacity, which consequently helps it make decisions with a greater or lesser anticipation. To better study the influence of this parameter, the problem formulated in equation (5.2) was considered, using for the total path cost calculation the cost function presented in equation (5.1a). Considering the formulated problem, the chosen cost function and the parameters presented in table 5.1 two different values for the horizon were chosen to visually perceive what varies in the state calculation process. The result obtained can be observed in figure 5.3a and 5.3b.
Analyzing the previous figures, it is possible to see that with a horizon of 10 the vehicle identifies the obstacle when it gets very close to it, and therefore the calculated trajectory is near the obstacle border. If the horizon is increased to 20 the obstacle is identified by the vehicle with greater anticipation allowing it to deviate in advance. Visually it is very hard to know what the best horizon is to be chosen. Therefore, a study must be done to understand what is the value of the horizon that minimizes the total cost of the optimization process. In this way, the parameter $H$ was varied so that the evolution of the total cost obtained for each of these values could be analyzed. Therefore, the variation of the cost as a function of the horizon can be observed in figure 5.4.
Analyzing figure 5.4, it is possible to see that for a horizon equal to 4 (red dot) a cost equal to $8.591 \times 10^5$ is obtained. By decreasing $H$ from the value 4 the cost decreases. However, a small horizon prevents the vehicle from making better decisions. Therefore, the total cost obtained, even with small values, does not correspond to a path that allows the vehicle to achieve the reference. In this way, it is necessary to find a sufficiently large horizon, which presents a relatively low cost. Analyzing figure 5.4 is possible to see that there is a horizon range (between 10 and 15) where the the cost, in comparison to other horizon values, is relatively low. However, a new concern regarding computational time emerges.

The computational time corresponds to the duration of the optimization process to find an optimal solution for the problem presented with all the existing constraints. For the problem introduced previously, the computational time is the time required to obtain an optimum trajectory for the vehicle, from an initial point to an endpoint, respecting all constraints imposed on the agent. The computational time cannot be very high since this algorithm aims to find the optimal trajectory in real-time. Therefore, for a better understanding in figure 5.5 is presented the evolution of the computational time as a function of the horizon increase. The values were obtained based on several simulations, using the same value of the horizon several times and at the end, an average of the time obtained for each $H$ was made.
As expected, computational time increases continuously in a linear way with the increase of the horizon. However, it is necessary to find a middle ground between having an optimizer with a relatively large horizon but presenting a relatively low computational time and total cost. Thus, by analyzing figures 5.4 and 5.5 it is possible to conclude that for the problem defined in equation (5.2) and using the cost function presented in equation (5.1a), the horizon, which gives the vehicle a very acceptable field of view, and presents considerable values for the computational time and the total cost, can acquire a value between 10 and 15. Despite what has been concluded previously, it is important to note that these values may all vary since they depend on the problem formulated, on the dynamics of the vehicle, on the constraints imposed, on the cost function used and on the optimization algorithm. Therefore, to better understand the influence of the horizon in the optimization process, a new problem is presented.

### 5.3.1 Multi obstacle problem

In this new situation, the dynamics of the vehicle, the optimization algorithm and the cost function used are the same as in the previous problem. The only thing that changes is the number of space constraints in the environment. Thus, the problem to be solved can be defined by the following

\[
\begin{align*}
\min_u & \quad \sum_{i=2}^{H} (\hat{y}(i) - Y)^T Q (\hat{y}(i) - Y) + (u(i - 1))^T R (u(i - 1)) \\
\text{subject to} & \quad x(1, i + 1) = x(1, i) + h \cdot x(2, i) + \frac{h^2}{2} \cdot u(1, i), \quad x(1, 1) = 1, \\
& \quad x(2, i + 1) = x(2, i) + h \cdot u(1, i), \quad x(2, 1) = 0, \\
& \quad x(3, i + 1) = x(3, i) + h \cdot x(4, i) + \frac{h^2}{2} \cdot u(2, i), \quad x(3, 1) = 2, \\
& \quad x(4, i + 1) = x(4, i) + h \cdot u(2, i), \quad x(4, 1) = 0, \\
& \quad (x(1, i + 1) - c^g_x)^2 + (x(3, i + 1) + c^g_y)^2 < d^2_{\text{min}}, \\
& \quad |x(2, i + 1)| \leq 20, \\
& \quad |x(4, i + 1)| \leq 20,
\end{align*}
\]

where \(c^g_x\) and \(c^g_y\) with \(g = 1, 2, \ldots, 9\) are the \(x\) and \(y\) coordinates of the center of each obstacle, and \(d_{\text{min}}\) is the radius and it is the same for all spatial constraint (5 meters). From the problem definition presented previously it is possible to conclude that there are 9 different geometric constraints that the vehicle will have to consider for calculating the optimum trajectory to the target position. The coordinates of the obstacles can be observed in table 5.2.

<table>
<thead>
<tr>
<th>Obstacles</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate (x) [m]</td>
<td>10</td>
<td>25</td>
<td>40</td>
<td>10</td>
<td>25</td>
<td>40</td>
<td>10</td>
<td>25</td>
<td>40</td>
</tr>
<tr>
<td>Coordinate (y) [m]</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 5.2: Coordinates of the obstacles.

As previously mentioned, it is necessary to have a sufficiently large horizon so that the vehicle has a wide field of view, allowing to identify the obstacles in advance and calculate a better trajectory to
reach the reference. The choice of the horizon must always have in mind the total cost generated and the computational time. The evolution of the total cost and the computational time with the increase of the horizon $H$ can be observed in figure 5.6a and 5.6b.

![Figure 5.6: Influence of Horizon $H$ on the cost $J$ and on the computational time $t$ in seconds.](image)

Analyzing figure 5.6a and 5.6b it is possible to observe that the value of the horizon that at the same time has a lower total cost compared to the others, and an acceptable computational time has a value in range between 8 and 13. Figure 5.7 shows four different cases where the horizon value is varied between 5 and 20. Through these simulations it is possible to observe how the horizon modifies the optimization process, obtaining, in the end, a different trajectory for each $H$ value. For values higher than 20 the simulations at a visual level are very similar to the simulation where $H$ is equal to 20. For each simulation, the values used for the sample time $h$, the number of iterations $Z$, the matrix $Q$ and $R$ and the final state $Y$ to be achieved, are the same as presented in table 5.1.

In figure 5.7a since the horizon of the vehicle is very small (5) the agent identifies the presence of an obstacle when it is very close to it, having a late response to the contour as observed. The late detection of the geometrical constraints causes the speed at which the vehicle moves to be lower to avoid them because if the speed was higher, there would be situations where collision would occur between them. The effects of the low-speed values acquired by the agent can be observed in figure 5.7a as the vehicle cannot reach the target in the number of iterations defined for the problem.

Observing the remaining figures, it is possible to verify that as the horizon increases, the vehicle identifies the presence of an obstacle more quickly, deviating with greater anticipation. This makes the agent choose different trajectories, allowing it to acquire velocity values high enough to help it reach the reference. For values greater than 20 the total cost and the computational time increase a lot, having no advantage to opt for such values.
5.4 Application of MPC to the tricycle model

Let us now consider the tricycle model presented in section 4.3. This model, as mentioned previously, is composed of two main wheels and a third wheel that provides support to the structure. Each of the main wheels has an angular and a linear velocity, from which it is possible to calculate the angular and linear velocity of the vehicle. The equations to calculate the coordinates and orientation of the vehicle in continuous time can be observed in equation (4.10). Passing the expressions to the discrete-time introduces an integral term that makes the calculations difficult. To simplify them, we adopted the Euler approximation method, allowing to obtain the equations for the states presented in equation (4.13).
In this way, using for the vehicle the tricycle model, with a radius of 2 meters, and the cost function presented in equation (5.1a), the problem can be formulated as follows

$$\min_u \sum_{i=2}^{H} (\hat{y}(i) - Y)^T Q(\hat{y}(i) - Y) + (u(i - 1))^T R (u(i - 1))$$  \hfill (5.7a)

subject to

$$x(1, i + 1) = x(1, i) + u(i) \cdot \cos(x(3, i)) \cdot h, \quad x(1, 1) = 1,$$  \hfill (5.7b)

$$x(2, i + 1) = x(2, i) + u(i) \cdot \sin(x(3, i)) \cdot h, \quad x(2, 1) = 2,$$  \hfill (5.7c)

$$x(3, i + 1) = x(3, i) + w(i) \cdot h, \quad x(3, 1) = 45^\circ,$$  \hfill (5.7d)

$$|v(i + 1)| \leq 20,$$  \hfill (5.7e)

$$|w(i + 1) - w(i)| \leq \frac{\pi}{3},$$  \hfill (5.7f)

where $v(i)$ is the linear velocity, at each instant of time, and can be calculated by $\frac{u_R(i) + u_L(i)}{2}$ and $w(i)$ is the angular velocity, at each instant of time, and can be calculated by $\frac{u_R(i) - u_L(i)}{L}$. Parameter $l$, in the previous division, corresponds to the distance between the two main wheels. Parameters $u_R$ and $u_L$ are the inputs applied to the right and the left wheel, respectively. Linear and angular velocities are restricted to certain values. The linear velocity from one moment to the next can only vary from 0 m/s to 20 m/s and the angular velocity from one instant to the next can only vary from 0° to 60°. Parameter $x(1, i)$ and $x(2, i)$ are the $x$ and $y$ coordinates, respectively, and $x(3, i)$ is the orientation of the vehicle. To prevent the vehicle from turning too much, a calculation is made to see if the difference between the angular velocity of state $i + 1$ and the current state $i$ does not exceed 60 degrees. The tricycle model has an interesting characteristic regarding the orientation, making the model quite different from the double integrator. In figure 5.8a and 5.8a it is possible to observe the dynamics of this new model and the characteristic that allows it to distinguish from the previous model.

For the previous simulations the values used for the sample time $h$, the number of iterations $Z$, the matrix $Q$ and $R$, and the final state $Y$ to be achieved, can be observed in table 5.3:

<table>
<thead>
<tr>
<th>H</th>
<th>h</th>
<th>Z</th>
<th>Q</th>
<th>R</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Simulation parameters.

As we can see the amplitude of the values used in the tricycle model simulation are the same as when using the double integrator since there are no spacial constraints or other restrictions that might complicate the problem. The only thing that changes is the size of the $Q$ and $R$ matrices since that depends on the number of states, and the final state $Y$, composed by coordinate $x$, $y$, and the orientation.

By analyzing figure 5.8b it is possible to verify that, when the vehicle has a different orientation, it needs to perform an additional maneuver allowing it to acquire an orientation toward the target. In the double integrator model, this would not happen since this model has no orientation, and in this way, the vehicle would move in a straight line towards the final position.
5.5 Dynamic obstacle avoidance

When using moving vehicles in an environment it is necessary to pay attention to all the existing obstacles, to avoid collision. Most of the time obstacles are static however, there may be situations where they are in motion. Considering that sensors are not used to read the environmental information, it is necessary to provide the optimizer with the coordinates and size of all geometric restrictions and in case they are dynamic it is also necessary to provide their future positions based on their movement.

Using the tricycle model, with a radius of 2 meters, and the cost function described in equation (5.1a), a dynamic obstacle was introduced to study the optimization process in the optimal path calculation. In this way, the problem described can be formulated as follows

\[
\begin{align*}
\min_u & \quad \sum_{i=2}^{H} (\hat{y}(i) - Y)^T Q (\hat{y}(i) - Y) + (u(i - 1))^T R(u(i - 1)) \\
\text{subject to} & \quad x(1, i + 1) = x(1, i) + v(i) \cdot \cos(x(3, i)) \cdot h, \quad x(1, 1) = 1, \\
& \quad x(2, i + 1) = x(2, i) + v(i) \cdot \sin(x(3, i)) \cdot h, \quad x(2, 1) = 2, \\
& \quad x(3, i + 1) = x(3, i) + w(i) \cdot h, \quad x(3, 1) = 45^\circ, \\
& \quad (x(1, i + 1) - c_1(i))^2 + (x(2, i + 1) + c_2(i))^2 < d_{\min}^2, \\
& \quad |v(i + 1)| \leq 20, \\
& \quad |w(i + 1) - w(i)| \leq \frac{\pi}{3},
\end{align*}
\]

where \(c_1(i)\) and \(c_2(i)\) are the coordinates, at each instant of time, of the center of the circular obstacle with a radius \(d_{\min}\) equal to 5 meters. The parameters used for the simulation are the same as those
presented in table 5.3. Therefore, the calculated optimized path can be observed in figure 5.9 and 5.10.

As previously mentioned, the vehicle knows the position of the obstacle at each instant, being able, in advance, to predict the point of intersection between the two. With this, the vehicle can stop its movement at the right moment to avoid contact, letting the obstacle pass as presented in figure 5.9a and 5.9b. The vehicle will resume its march after the complete passage of the obstacle, executing its movement until reaching the reference as observed in figure 5.10a and 5.10b.

(a) Intersection of the vehicle with the obstacle.
(b) Vehicle waiting for the obstacle to pass.

Figure 5.9: Path obtained using a dynamic obstacle for a horizon of 10 (Part 1).

(a) Vehicle resumes its movement.
(b) Vehicle reaching the goal.

Figure 5.10: Path obtained using a dynamic obstacle for a horizon of 10 (Part 2).
5.5.1 Horizon effect on dynamic obstacle avoidance

Considering the situation presented in the previous section, it was decided to complicate the problem, making the obstacle move towards the vehicle. For this specific simulation, it is important not to forget the size of the obstacle, having a radius of 5 meters, and that the obstacle is positioned in such a way that the vehicle is moving towards its mid-point. Two different values of horizon are going to be applied $H = 10$ and $H = 20$.

For this situation, although the characteristics are the same as in the previous problem since the movement direction of the obstacle was varied, the simulation parameters were also changed, allowing the optimal solution to be obtained without disturbance and uneven behavior on the part of the vehicle. Each problem and situation has different characteristics, and it is necessary to change the various parameters of the simulation to obtain, for each case, a solution that ultimately is optimal and also does not exhibit irregular behavior by the agents. Therefore, the values of the parameters used for this simulation can be verified in table 5.4.

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>$Z$</th>
<th>$Q$</th>
<th>$R$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 &amp; 20</td>
<td>0.1</td>
<td>100</td>
<td>3 0 0</td>
<td>1 0 0</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 3 0</td>
<td>0 1 0</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 3</td>
<td>0 0 1</td>
<td>45°</td>
</tr>
</tbody>
</table>

Table 5.4: Simulation parameters.

The result of the calculated path for both values of $H$ can be observed in figure 5.11a and 5.11b for a horizon of 10 and figure 5.12a and 5.12b for a horizon of 20.

Figure 5.11: Path obtained for $H = 10$, using a dynamic obstacle moving in the direction of the vehicle.
Using a $H$ of 10, it is possible to be observed in figure 5.11a and 5.11b that the vehicle when encountering the obstacle stops and starts to retreat backward. This is because the obstacle has a very large size and is positioned in such a way that the vehicle moves exactly towards its middle. Since $H$ is not large enough for the vehicle to predict beyond the obstacle, it begins to move backward so as not to collide, stopping only when the number of iterations ends. In this case, the goal is never achieved.

By increasing the horizon value to 20, the field of view of the vehicle becomes larger, making it possible to predict beyond the obstacle. Observing figure 5.12a and 5.12b it is possible to see that the optimizer, with great anticipation, calculates a trajectory that allows the vehicle to deviate in time to avoid collision and reach the reference. Again, it is possible to note that the horizon plays a fundamental role in the optimization process, helping in the optimal path calculation and achieving the goal.

5.6 Vehicle interaction

As noted above, cooperative control is the ability of a set of vehicles to perform joint functions considering the acquired information that each share with the others. As explained previously, there are different methods by which it is possible to achieve an interaction between agents. In this context, this method consists of a main vehicle that calculates its future positions, based on the defined horizon, geometrical constraints and speed limits, and then sends this information to the second vehicle. The second agent, using the data received as well as the information about existing obstacles, calculates its future positions, always respecting the restrictions applied to its velocity. It is important to note that the first vehicle has no information about the future positions of the second agent. Considering the tricycle model for both vehicles, having both a radius of 2 meters and for the optimization solver the cost function presented in

Figure 5.12: Path obtained for $H = 20$, using a dynamic obstacle moving in the direction of the vehicle.
the problem of vehicle interaction can be formulated as follows

$$\min_u \sum_{i=2}^{H} (\Psi - Y)^T Q (\Psi - Y) + (\Phi(i - 1))^T R (\Phi(i - 1))$$

subject to

$$x_1(1, i + 1) = x_1(1, i) + v_1(i) \cdot \cos(x_1(3, i)) \cdot h, \quad x_1(1, 1) = 1, \quad \text{(5.9a)}$$

$$x_1(2, i + 1) = x_1(2, i) + v_1(i) \cdot \sin(x_1(3, i)) \cdot h, \quad x_1(2, 1) = 2, \quad \text{(5.9b)}$$

$$x_1(3, i + 1) = x_1(3, i) + w_1(i) \cdot h, \quad x_1(3, 1) = 45^\circ, \quad \text{(5.9c)}$$

$$x_2(1, i + 1) = x_2(1, i) + v_2(i) \cdot \cos(x_2(3, i)) \cdot h, \quad x_2(1, 1) = x_1(1, 1) + \Gamma_x, \quad \text{(5.9d)}$$

$$x_2(2, i + 1) = x_2(2, i) + v_2(i) \cdot \sin(x_2(3, i)) \cdot h, \quad x_2(2, 1) = x_1(2, 1) - \Gamma_y, \quad \text{(5.9e)}$$

$$x_2(3, i + 1) = x_2(3, i) + w_2(i) \cdot h, \quad x_2(3, 1) = x_1(3, 1), \quad \text{(5.9f)}$$

$$|v_1(i + 1)| \leq 20, \quad \text{(5.9g)}$$

$$|w_1(i + 1) - w_1(i)| \leq \frac{\pi}{3}, \quad \text{(5.9h)}$$

$$|v_2(i + 1)| \leq 20, \quad \text{(5.9i)}$$

$$|w_2(i + 1) - w_2(i)| \leq \frac{\pi}{3}, \quad \text{(5.9j)}$$

where the previous cost function is the same for both vehicles, changing only the start and end position.

For both vehicles $\Psi$ is the prediction of the future states and it is represented by $\hat{y} = [y^1_1(i); y^1_2(i); y^1_3(i)],$ for the first agent, and by $\hat{y} = [y^2_1(i); y^2_2(i); y^2_3(i)],$ for the second agent. Parameters $y_1(i)$ and $y_2(i)$ are the $x$ and $y$ coordinates of the future positions for each vehicle, respectively, and $y_3(i)$ is the future orientation. The start and end position of the second agent is the same as the first, offset by a $\Gamma$ factor. Parameter $\Gamma$ is the distance between the two vehicles, respecting a certain angle $(\theta_{\text{init}})$ chosen initially, and can be defined as $\Gamma_x = (\text{radius}_1 + \text{radius}_2 + \text{dist}) \cdot \cos(\theta_{\text{init}}),$ for the $x$ coordinate, and $\Gamma_y = (\text{radius}_1 + \text{radius}_2 + \text{dist}) \cdot \sin(\theta_{\text{init}}),$ for the $y$ coordinate. Parameter $\text{radius}_1$ is the radius of the first vehicle, $\text{radius}_2$ is the radius of the second vehicle and $\text{dist}$ is the desired distance to be maintained between them. Parameter $\Phi$ in the cost function is the input applied to the vehicle and it acquires different values for each agent. The input can be defined by $u_1(i),$ for the first vehicle, and by $u_2(i),$ for the second vehicle. The parameters used for the simulation can be observed in table 5.5

<table>
<thead>
<tr>
<th>H</th>
<th>h</th>
<th>Z</th>
<th>Q</th>
<th>R</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1</td>
<td>100</td>
<td>3 0 0</td>
<td>0.1 0 0</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 3 0</td>
<td>0 0.1 0</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 3</td>
<td>0 0 0.1</td>
<td>45°</td>
</tr>
</tbody>
</table>

Table 5.5: Simulation parameters.

The entries of matrix $R$ can be decreased allowing input $u$ to have higher values, and therefore the vehicles to have higher angular and linear velocities, making them move faster. The differences between a $R$ matrix with diagonal inputs of 1 or 0.1 can be seen in figures 5.13a and 5.13b.
Observing both figures, it is possible to see that for diagonal values equal to 0.1, the distance between successive states is greater than when using values equal to 1. This property can be better distinguished especially as the vehicle approaches the reference position. With this, it was possible to demonstrate that agents, with diagonal values of $R$ equal to 0.1, move faster, allowing to minimize the total cost of the system. Therefore, for each simulation, it will be necessary to experiment both values and see which one allows to obtain a feasible solution and meets the desired goals.

As mentioned before, the first agent calculates its future states for the defined horizon, always respecting all constraints, both in terms of geometric and vehicle dynamics constraints and transmits this information to the second agent. The second vehicle, having the same restrictions, calculates its future states based on the information received from the first. These states must be calculated such that both vehicles are always at 2 meters from each other. In addition to maintaining the distance, the initial angle formed between the two vehicles, being the same angle as in the final positions, must be maintained during the simulation whenever possible. Since the first vehicle is unaware of the behavior of the second, however, the opposite is true, only the second agent will change its movement based on the states of the first. Therefore, only changes to the dynamics of the second vehicle will be required.

### 5.6.1 Vehicle interaction issues and resolution

Two conditions, mentioned in the previous section, need to be met: maintain 2 meters between the vehicles and the starting angle, concerning the end positions, must be maintained whenever possible. Consider again the model presented in the previous section. To understand more clearly whether the conditions are being met, two spatial restrictions of the format shown in figure 3.2b have been introduced. In this way, the problem studied in this section may be defined as follows.
\[
\begin{align*}
\min_u & \quad \sum_{i=2}^{H} (\Psi - Y)^T Q (\Psi - Y) + (\Phi(i - 1))^T R (\Phi(i - 1)) \\
\text{subject to} & \quad x_1(1, i + 1) = x_1(1, i) + v_1(i) \cdot \cos(x_1(3, i)) \cdot h, \quad x_1(1, 1) = 1, \\
& \quad x_1(2, i + 1) = x_1(2, i) + v_1(i) \cdot \sin(x_1(3, i)) \cdot h, \quad x_1(2, 1) = 2, \\
& \quad x_1(3, i + 1) = x_1(3, i) + w_1(i) \cdot h, \quad x_1(3, 1) = 45^\circ, \\
& \quad x_2(1, i + 1) = x_2(1, i) + v_2(i) \cdot \cos(x_2(3, i)) \cdot h, \quad x_2(1, 1) = x_1(1, 1) + \Gamma, \\
& \quad x_2(2, i + 1) = x_2(2, i) + v_2(i) \cdot \sin(x_2(3, i)) \cdot h, \quad x_2(2, 1) = x_1(2, 1) - \Gamma, \\
& \quad x_2(3, i + 1) = x_2(3, i) + w_2(i) \cdot h, \quad x_2(3, 1) = x_1(3, 1), \\
& \quad |x_1(1, i) - c_{g_x}^g| + |x_2(2, i) - c_{g_y}^g| < d_{\text{min}}, \\
& \quad |x_2(1, i) - c_{g_x}^g| + |x_2(2, 1) - c_{g_y}^g| < d_{\text{min}}, \\
& \quad |v_1(i + 1)| \leq 20, \\
& \quad |w_1(i + 1) - w_1(i)| \leq \frac{\pi}{3}, \\
& \quad |v_2(i + 1)| \leq 20, \\
& \quad |w_2(i + 1) - w_2(i)| \leq \frac{\pi}{3}, \\
\end{align*}
\]

where \( c_{g_x}^g \) and \( c_{g_y}^g \), with \( g = 1, 2 \) are the \( x \) and \( y \) coordinates of the center of each of the obstacles, and \( d_{\text{min}} \) is the distance from the tip to the center. The coordinated movement of both vehicles, using the parameters from table 5.5 with the diagonal entries of \( R \) equal to 1 can be observed in figure 5.14.

![Figure 5.14](image.png)

Figure 5.14: Path obtained using vehicle interaction and a horizon of 10 for both agents.

By looking at the previous figures it is possible to see that the distance between the agents is respected along the calculated path. Although a solution has been found, respecting all the constraints, looking only at the second plot of figure 5.14 it is observed that the angle of the vehicles with the
final positions is only corrected close to reaching the target. As already said, the ideal would be to maintain the initial angle, whenever possible. The changes that need to be made will be implemented at the dynamics of the second vehicle since, as mentioned previously, only the second agent knows the behavior of the first one. Therefore, it will have to adjust in such a way that the constraints applied both to the distance and to the angle between vehicles are respected.

To make this possible it was necessary to add to the cost function of the second vehicle a parameter that penalizes the angular deviation of the second agent from the first concerning the reference angle. This parameter is multiplied by a weight that gives importance to this term. This penalty is made at the cost function level, as it only must be done when possible, being more important to respect the obstacles placed on the environment. Therefore, this angular constraint is considered a soft restriction, while obstacles and the limits applied to the vehicle dynamics are considered hard constraints.

The parameter added to the cost function corresponds to the square of the difference between the reference angle and the angle between the vehicles in every instant of time. Therefore, at each moment it is necessary to calculate the angle that the second agent makes with the first, to be able to obtain this extra term. This parameter can be calculated using equation (5.11)

\[
\theta = \theta_{ref} - \theta_{\text{vehicles}}(i),
\]

(5.11)

where \(\theta_{ref}\) is the reference angle and \(\theta_{\text{vehicles}}\) is the angle between both vehicles at discrete time \(i\). The angle between both vehicles can be calculated based on equation (5.12)

\[
\theta_{\text{vehicles}}(i) = \text{atan2}[x_2(2, i) - x_1(2, i), x_2(1, i) - x_1(1, i)],
\]

(5.12)

where \(\text{atan2}\) can calculate the four-quadrant inverse tangent of any \(Y\) and \(X\) pair of coordinates. In this way, the cost function of the second vehicle from the problem definition presented previously by equation (5.10) can be rewritten as follows

\[
\sum_{i=2}^{H} (x_2(i + 1) - Y)^T Q(x_2(i + 1) - Y) + (u_2(i))^T R(u_2(i)) + \tau \theta^2(i),
\]

(5.13)

where \(\tau\) is the weight that gives importance to the difference between the reference angle and the angle between the vehicles, allowing the second vehicle to correct faster the angle concerning the first.

In addition to the extra parameter added to the cost function of the second vehicle, it was also necessary to change its maximum speed. This is because, in situations where vehicle 1 is moving at full speed and vehicle 2 has, due to space constraints, been forced to move behind vehicle 1, vehicle 2 will never be able to travel with a speed greater than the maximum speed in order to correct the resulting angle with vehicle 1. Thus, the speed constraint of the second vehicle was modified, being increased, allowing vehicle 2 to move at a speed higher than vehicle 1, being able to correct the angular orientation. This change can be observed in equation (5.14)

\[
|v_2(i + 1)| \leq 30.
\]

(5.14)

To know the best value of \(\tau\), it is necessary to perform a study where it is possible to evaluate the influence of this weight on the angular orientation correction. In this way, the deviation between the
reference angle and the angle originated by the second vehicle with respect to the first was calculated for different $\tau$ values. In order to know the value of $\tau$ that allows a faster correction of the angular deviation, observing figure 5.14 it is possible to realize that it is only necessary to analyze the angles obtained between the agents from the moment when both vehicles pass the two obstacles until they reach the reference. Considering that the simulation has 100 iterations, it is only necessary to analyze from iteration 25, which corresponds to the moment when the vehicles pass the obstacles. Since the calculated cost, without introducing the extra term, is of the order of $10^5$, as presented in equation (5.3), and the theta calculated by equation (5.12) is in rad/s, it can be concluded that parameter $\tau$ must have values greater than 100 in order to impact the final value of the cost function. Therefore, the deviation obtained for 100 iterations and for different values of $\tau$ can be observed in figure 5.15.

![Figure 5.15: Percentage deviation of the angle that the second vehicle makes with the first relative to the reference angle, from two perspectives.](image)

Different $\tau$ values were tested from 100 to 1000, increasing each time by 100. Values below 100 have not been found to provide any improvement in the angular correction. Analyzing the previous figure, it is possible to see that for all values of $\tau$ (except for 600, where there are some irregularities), the angular deviation has similar behavior all the way through. At first, the error is very small (almost zero), since the vehicles have not yet reached the obstacles. Between iteration 10 and 25, the angle error relative to the reference increases as agents are in the crossing process and there is a need to change their orientation to comply with spatial constraints. After the vehicles passed the obstacles, the angle was corrected. Testing each of the values individually, it was found that $\tau = 800$ is the one that quickly corrects the angle and makes the agents respect both obstacles since there are values that make the solution unfeasible. Repeating again the simulation presented in figure 5.14 using the new velocity constraint presented in equation (5.14), the modified cost function defined in equation (5.13),
and diagonal entries of $R$ with values equal to 1 it is possible to obtain the result observed in figure 5.16.

Figure 5.16: Path obtained for a horizon of 10, using vehicle interaction and angular correction.

Using for the simulations from figures 5.14 and 5.16 a $R$ equal to 1, allows us to have smaller values of $u$, and therefore a more controlled simulation without any uneven behavior on the part of the vehicles, as verified when using values for $R$ equal to 0.1. Observing figure 5.16 it is possible to see that after the agents pass between the obstacles (figure 5.16c), the second vehicle quickly attempts to correct the angle it makes with the first to decrease the deviation from the reference angle.

The vehicle interaction sometimes implies the execution of several tasks. For this, it is necessary to have a communication system between agents so that they can make decisions based on the choices
of the others as described above. In the next section, the topic of joint task execution will be addressed.

5.7 Task execution using two vehicles

As was said earlier, in robotics, cooperative control is the capacity of a group of vehicles to execute shared and common tasks, interacting with each other. Using two agents, as presented above, to check their behavior when performing a task together, a river was added. The task to be performed is the coordinated crossing of the river. Considering the tricycle model for both vehicles with a radius of 2 meters, the cost function presented in equation \(5.1a\) and the modifications implemented in the cost function and speed of the second vehicle, the task execution problem can be defined as follows \(5.11\)

\[
\begin{align*}
\min_{\mathbf{u}} & \quad \sum_{i=2}^{H} (\mathbf{Y} - \mathbf{Y}')^T Q (\mathbf{Y} - \mathbf{Y}') + (\mathbf{Y}(i-1))^T R (\mathbf{Y}(i-1)) + \tau \cdot \theta^2 \\
\text{subject to} & \quad x_1(1,i+1) = x_1(1,i) + v_1(i) \cdot \cos(x_1(3,i)) \cdot h, \quad x_1(1,1) = 1, \quad (5.15b) \\
& \quad x_1(2,i+1) = x_1(2,i) + v_1(i) \cdot \sin(x_1(3,i)) \cdot h, \quad x_1(2,1) = 2, \quad (5.15c) \\
& \quad x_1(3,i+1) = x_1(3,i) + w_1(i) \cdot h, \quad x_1(3,1) = 45^\circ, \quad (5.15d) \\
& \quad x_2(1,i+1) = x_2(1,i) + v_2(i) \cdot \cos(x_2(3,i)) \cdot h, \quad x_2(1,1) = x_1(1,1) + \Gamma_x, \quad (5.15e) \\
& \quad x_2(2,i+1) = x_2(2,i) + v_2(i) \cdot \sin(x_2(3,i)) \cdot h, \quad x_2(2,1) = x_1(2,1) - \Gamma_y, \quad (5.15f) \\
& \quad x_2(3,i+1) = x_2(3,i) + w_2(i) \cdot h, \quad x_2(3,1) = x_1(3,1), \quad (5.15g) \\
& \quad |v_1(i+1)| \leq 20, \quad (5.15h) \\
& \quad |w_1(i+1) - w_1(i)| \leq \frac{\pi}{3}, \quad (5.15i) \\
& \quad |v_2(i+1)| \leq 30, \quad (5.15j) \\
& \quad |w_2(i+1) - w_2(i)| \leq \frac{\pi}{3}, \quad (5.15k) \\
\end{align*}
\]

\(\text{river, where } x \in \mathbb{R}, \quad y \in [30, 32] \text{ meters},\) \(5.15l\)

where, for the first vehicle, \(\theta\) and \(\tau\) are equal to zero, and for the second vehicle \(\theta\) is equal to the deviation of the angle that the second agent makes with the first relative to the reference angle, and \(\tau\) is equal to 800. As already mentioned, the task corresponds to the crossing of the river by the two vehicles. Imagine that the agents carry a wooden board, which when they find the river, will have to place it so that they can cross. The board must be placed over the width of the river, and before starting the crossing movement, vehicles must try to position themselves in a straight line to pass perpendicularly.

A very important question is where the passing process will take place. In addition to applying the optimization process to calculate the optimal path that minimizes the total system cost, it is necessary to apply an external optimization to analyze what is the best point to cross the river. For each of the crossing points, it is necessary to calculate the optimal path and the total cost associated with that path, being able, in the end, to compare it with the remaining costs obtained for other points of the river. In the end, the optimizer presents the crossing point which, compared to others, has a lower cost. Given that one optimization process is performed within another, computational time is greatly increased. After applying the optimization processes, and using for the simulation the parameters presented in table \(5.5\)
with the diagonal entries of matrix $R$ equal to 0.1, and with a number of iterations $Z$ equal to 200, it was possible to obtain the path shown in figure 5.17.

Figure 5.17: Calculated crossing point and path that minimizes total system cost, using a horizon of 10.

Analyzing the previous figure, it is possible to observe that there are two different movements that correspond to two states of a state machine, as presented in figure 1.3. The first state corresponds to the normal movement of the vehicles and it is executed until a river is detected. After the river detection, a flag is triggered and a state transition occurs, moving to the second movement that corresponds to the river crossing. The river crossing itself has other states to be executed sequentially as presented in figure 1.4. First, the agents place the wooden board on top of the river. Then the first vehicle starts crossing, while the second remains stationary on the board to provide stability. After the first one completes the crossing, the second one begins its passage. When both agents finish the crossing movement, the flag is no longer checked and the state machine returns to the first state of figure 1.3 (normal movement of the vehicles). Both main states can also be differentiated by the color change of the vehicle states, which can be seen in figure 5.17 (normal movement: green and blue, crossing river: red and purple).

The passage from one state to another, depending on the position of the vehicles in the environment, can be seen in figure 5.18. In this figure is shown an auxiliary state machine representing the value of the flag and the state the vehicles are in for each zone of the environment.

Analyzing figure 5.17, it is possible to see that, as expected, the crossing point is in the middle of the river ($x = 30$), since both surfaces (before and after the river) are identical. However, what happens when surfaces are different? In the next section, different surfaces will be used to test the optimization process and the task execution by the agents. To make the surfaces different, viscosity will be assigned to each one, making vehicle movement easier or harder. The theoretical concept used to previously know what the result should be is Snell’s Law.
5.8 Path optimization using different surfaces

5.8.1 Quick Review of Snell’s Law

As previously shown, a set of two vehicles circulates in an environment performing tasks, in this case crossing a river. In the problem shown in figure 5.18, the surfaces, before and after the river, are identical and both vehicles exhibit the same movement behavior, that is, the path taken before the river is the same length as the path taken after the river. However, if the surfaces are different this no longer happens. To know what occurs theoretically, before the simulations, Snell’s law can be used.
Snell’s law is often applied in optics and can be defined by the ratio of the sines of the incident and refraction angles, of a ray of light, and is equivalent to the reciprocal ratio of the indices of refraction or phase velocities in the two media. The equations that define the previous law can be defined as

$$\frac{\sin(\theta_2)}{\sin(\theta_1)} = \frac{n_{\text{index}1}}{n_{\text{index}2}} = \frac{v_{\text{light}2}}{v_{\text{light}1}}, \quad \text{with} \quad n_{\text{index}1} = \frac{c_v}{v_1}, \quad n_{\text{index}2} = \frac{c_v}{v_2}, \quad (5.16)$$

where $\theta_1$ is the incident angle, $\theta_2$ is the refraction angle, $v_{\text{light}}$ is the speed with which light travels in each medium, $c_v$ is the speed of the light in the vacuum and $n_{\text{index}}$ is the refractive index of the medium.

In the context of viscosity’s defined by $\alpha = \frac{1}{\eta}$, where $\eta$ is the general symbol for viscosity. In this case, $\alpha$ and velocity of each medium are related inversely compared to the relation between the refractive indices and velocity in the previous equation. This new relationship can be verified in equation (5.17)

$$\frac{\alpha_1}{\alpha_2} = \frac{v_{\text{light}1}}{v_{\text{light}2}}, \quad (5.17)$$

where $\alpha_1$ and $\alpha_2$ are the inverse of the viscosities of the first and second surfaces, respectively. Therefore, if $\alpha_1 > \alpha_2$ this means that it is easier to move on the first surface than on the second, and therefore the calculated path in the surface before the river must be longer than the calculated path on the surface after the river. If $\alpha_1 < \alpha_2$ then the inverse propriety, mentioned previously, should be verified.

### 5.8.2 Influence of viscosity on path optimization

In this section, we intend to analyze how the optimization method behaves when there are surfaces with different characteristics. The parameters used for the simulations using viscosities are the same as presented in table 5.5 with $R$ equal to 0.1, and the number of iterations $Z$ equal to 200. Considering the tricycle model for both vehicles with a radius of 2 meters, the cost function presented in equation (5.1a), the changes made to the cost function and speed limits of the second vehicle, and the induction of the variables corresponding to the inverse of the viscosity $\eta$, the problem can be presented as follows

$$\min_{u} \sum_{i=2}^{H} (\Psi - Y)^T Q(\Psi - Y) + (\Phi(i - 1))^T R(\Phi(i - 1)) + \tau \cdot \theta^2 \quad (5.18a)$$

subject to

- $x_1(1, i + 1) = x_1(1, i) + v_1(i) \cdot \cos(x_1(3, i)) \cdot h \cdot \alpha_1, \quad x_1(1, 1) = 1, \quad (5.18b)$
- $x_1(2, i + 1) = x_1(2, i) + v_1(i) \cdot \sin(x_1(3, i)) \cdot h \cdot \alpha_1, \quad x_1(2, 1) = 2, \quad (5.18c)$
- $x_2(3, i + 1) = x_2(3, i) + w_1(i) \cdot h \cdot \alpha_2, \quad x_1(3, 1) = 45^\circ, \quad (5.18d)$
- $x_2(1, i + 1) = x_2(1, i) + v_2(i) \cdot \cos(x_2(3, i)) \cdot h \cdot \alpha_2, \quad x_2(1, 1) = x_1(1, 1) + \Gamma_x, \quad (5.18e)$
- $x_2(2, i + 1) = x_2(2, i) + v_2(i) \cdot \sin(x_2(3, i)) \cdot h \cdot \alpha_2, \quad x_2(2, 1) = x_1(2, 1) - \Gamma_y, \quad (5.18f)$
- $x_2(3, i + 1) = x_2(3, i) + w_2(i) \cdot h \cdot \alpha_2, \quad x_2(3, 1) = x_1(3, 1), \quad (5.18g)$

- $|v_1(i + 1)| \leq 20,$
- $|w_1(i + 1) - w_1(i)| \leq \frac{\pi}{3},$ \quad (5.18h)
- $|v_2(i + 1)| \leq 30,$
- $|w_2(i + 1) - w_2(i)| \leq \frac{\pi}{3},$ \quad (5.18i)
The horizon used has a value of 10 for both agents, although it is not very relevant since there are no obstacles to obstruct the route. The horizon has a greater impact when there is a need to have a wider field of vision to be able to calculate in advance a path that will cause vehicles to deviate from obstacles in the environment. In this context, the task to be performed is still the coordinated passage of the river. Let’s assume that the first surface (before the river) has a lower viscosity than the second (after the river) i.e., \( \eta_1 < \eta_2 \), and therefore \( \alpha_1 > \alpha_2 \). The result of the optimization process, in the calculation of the optimal path, can be observed in figure 5.19a.

(a) Calculated path for \( \alpha_1 = 1.5 \) and \( \alpha_2 = 0.5 \).
(b) Calculated path for \( \alpha_1 = 0.5 \) and \( \alpha_2 = 1.5 \).

Figure 5.19: Calculated path for different values of viscosities.

Looking at figure 5.19a you can see that the optimization process calculated the optimum path as expected. Since there is greater ease of movement on the first surface to minimize total cost, the optimizer calculated a larger path on the first surface than on the second. Considering now the inverse situation, where the viscosity of the first surface (before the river) is higher than the viscosity of the second (after the river) i.e., \( \eta_1 > \eta_2 \) and therefore \( \alpha_1 < \alpha_2 \). The result of the optimization process, in the calculation of the optimal path, can be observed in figure 5.19b.

Looking at figure 5.19b, it is possible to see that the optimization process calculated the optimal path as expected. By having higher viscosity on the first surface to minimize the total cost, the calculated path on that surface is shorter compared to the calculated path for the second surface.

5.9 Task execution and obstacle avoidance

After analyzing the different problems separately, such as obstacle avoidance and task execution, and making improvements to obtain the desired solutions, we will now present cases where both problems, together with the modifications made, will be used. Let’s consider the same problem formulation as presented in the previous section, with the same model for both agents, the same cost function and the
same modifications and improvements made at the level of the second vehicle. For this example, a river will be used, and some spatial constraints will be added.

The surfaces, before and after the river, are equal and there is no viscosity hindering the movement of vehicles. Again, there is a main optimization process that analyzes the best river crossing point, based on the total cost to travel from the starting position to the river crossing point, plus the cost from the point after crossing the river to the final position.

Each situation has different characteristics, different restrictions, and different initial conditions, being necessary to change the parameters used to obtain the desired result. For this problem, several values of the weight that multiplies the angular difference that allows to maintain the orientation of the vehicles were tested. Therefore, after applying various weight values, it was concluded that having added these new geometric constraints, it was necessary to increase the weight value from 800 to 1000. This is because, as there are several constraints on the agents’ path, their orientation will constantly change, and there is a greater need to control the angular deviation between vehicles at any given time. To understand the difference between the two weight values (800 and 1000) and how they influence the optimization process simulations were performed to visually understand their effect. Initially, it is necessary to define the problem that is going to be studied. Therefore, the formulated problem can be presented as follows

\[
\begin{align*}
\min_{\mathbf{u}} & \quad \sum_{i=2}^{H} (\Psi - \mathbf{Y})^T Q (\Psi - \mathbf{Y}) + (\Phi(i - 1))^T R (\Phi(i - 1)) + \tau \cdot \theta^2 \\
\text{subject to} & \quad x_1(1, i + 1) = x_1(1, i) + v_1(i) \cdot \cos(x_1(3, i)) \cdot h \cdot \alpha_1, \quad x_1(1, 1) = 1, \quad (5.19a) \\
& \quad x_1(2, i + 1) = x_1(2, i) + v_1(i) \cdot \sin(x_1(3, i)) \cdot h \cdot \alpha_1, \quad x_1(2, 1) = 2, \quad (5.19b) \\
& \quad x_1(3, i + 1) = x_1(3, i) + w_1(i) \cdot h \cdot \alpha_1, \quad x_1(3, 1) = 45^\circ, \quad (5.19c) \\
& \quad x_2(1, i + 1) = x_2(1, i) + v_2(i) \cdot \cos(x_2(3, i)) \cdot h \cdot \alpha_2, \quad x_2(1, 1) = x_1(1, 1) + \Gamma_x, \quad (5.19d) \\
& \quad x_2(2, i + 1) = x_2(2, i) + v_2(i) \cdot \sin(x_2(3, i)) \cdot h \cdot \alpha_2, \quad x_2(2, 1) = x_1(2, 1) - \Gamma_y, \quad (5.19e) \\
& \quad x_2(3, i + 1) = x_2(3, i) + w_2(i) \cdot h \cdot \alpha_2, \quad x_2(3, 1) = x_1(3, 1), \quad (5.19f) \\
& \quad (x(1, i + 1) - c_1^2)^2 + (x(2, i + 1) + c_2^2)^2 < d_{min}^2, \quad (5.19g) \\
& \quad |v_1(i + 1)| \leq 20, \quad (5.19h) \\
& \quad |w_1(i + 1) - w_1(i)| \leq \frac{\pi}{3}, \quad (5.19i) \\
& \quad |v_2(i + 1)| \leq 30, \quad (5.19j) \\
& \quad |w_2(i + 1) - w_2(i)| \leq \frac{\pi}{3}, \quad (5.19k) \\
& \quad \text{river, where} \quad x \in \text{Re}, \quad y \in [30 \text{ 32}] \text{ meters}, \quad (5.19l)
\end{align*}
\]

where \( c_1^2 \) and \( c_2^2 \) with \( g = 1, 2, 3 \) are the \( x \) and \( y \) coordinate of the center of each of the obstacles, and \( d_{min}^2 \) is the radius. Based on \( g \) it is possible to conclude that there are three different obstacles that have been placed in such a way as to test all the developed properties and improvements of the vehicles studied so far. The main goal is to obtain a trajectory from an initial point to a final target, deviating from all the obstacles and executing tasks (crossing the river), always respecting the constraints applied to
the dynamics of the vehicles. The parameters used for this situation, where obstacles were added to the environment are very similar to the ones represented in table 5.5. Some changes were made in terms of the number of iterations and on the coordinates of the final target. Therefore, to obtain a valid solution for the situation presented in the problem definition, the parameters presented in table 5.6 are used.

<table>
<thead>
<tr>
<th></th>
<th>H</th>
<th>h</th>
<th>Z</th>
<th>Q</th>
<th>R</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1</td>
<td>200</td>
<td>3 0 0</td>
<td>0.1 0 0</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 3 0</td>
<td>0 0.1 0</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 3</td>
<td>0 0 0.1</td>
<td>45°</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.6: Simulation parameters.

The coordinates and the radius of the obstacles presented in the environment can be observed in table 5.7. After applying the optimization process using both values of the weight \( \tau \), the results obtained from the optimal path calculation can be seen in figures 5.20a and 5.20b.

<table>
<thead>
<tr>
<th>Obstacles</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate x [m]</td>
<td>20</td>
<td>30</td>
<td>50</td>
</tr>
<tr>
<td>Coordinate y [m]</td>
<td>15</td>
<td>50</td>
<td>55</td>
</tr>
<tr>
<td>Radius [m]</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 5.7: Coordinates of the obstacles.

Figure 5.20: Path obtained using different values of \( \tau \).

Analyzing both figures it is possible to see that for each value of \( \tau \), different optimal paths were obtained. The optimal river crossing point is also different in both cases. In both simulations the final target is reached respecting, at each instant, all the constraints applied to the vehicles. Although visually
the path observed in figure 5.20b seems to be the best one, this conclusion cannot be made because, as already mentioned, the best path is the one with the lowest cost. Therefore, considering both weight values, it is necessary to analyze the total cost for the agents to reach the goal.

For the path taken in figure 5.20a, the total cost obtained was $1.7730 \times 10^5$. In contrast, the total cost calculated for the path taken in figure 5.20b was $1.4571 \times 10^5$. Although the values are of the same order of magnitude, the total cost obtained for the path made in figure 5.20b is lower than in figure 5.20a. From this, it can be seen that increasing the weight $\tau$ from 800 to 1000 was an improvement in the optimization process.

Through these simulations, it is possible to conclude that for each situation it is necessary to experiment and analyze different values for all the parameters used since their influence on the optimization process varies from case to case, depending on the environment, on the constraints and the vehicle dynamics.

5.10 Task execution and obstacle avoidance using two rivers

The problem presented in the previous section, where both vehicles had the river crossing task, can be expanded to several rivers. Although it is possible to obtain a solution having many rivers, the time to calculate this solution increases exponentially with the introduction of a new river. Therefore, several tests were made having inserted in the environment only two rivers for the vehicles to cross.

In this situation there is a main optimization method which is applied twice i.e., the $fmincon$ solver is performed twice so that the crossing point on both rivers can be calculated. Within the main optimization method, several secondary optimization methods are used to calculate the best route from one point to another, always considering the crossing point. To better understand this optimization process involving two rivers an example is presented.

Initially, a first $fmincon$ will be applied to calculate the crossing point of the first river. Within this $fmincon$ the route from the starting point of the agents to the crossing point chosen by the main $fmincon$ is calculated. The path calculation is done by applying several secondary $fmincon$ solvers, being possible to obtain future inputs for the defined horizon. Several $fmincon$ solvers are executed because only the first input of the sequence obtained by a single $fmincon$ process is applied to the vehicle dynamics to prevent error propagation to future states. Each crossing point tested will have an associated cost. The main $fmincon$, after testing multiple crossing points chooses the one that ultimately allows the vehicles to spend the least energy i.e., the one that gives the lowest final cost. After crossing the first river, the same process is performed again allowing to obtain the crossing point of the second river that minimizes the expense of the vehicles to make the route. Having crossed the second river, the normal process of optimization takes place, allowing agents to reach the goal.

Let’s consider again the tricycle model for both vehicles, the cost function presented in equation 5.1a and the improvements made to the dynamics of the second vehicle. As already mentioned, compared to the tests performed in the previous section, another river was added to be crossed. Also, new geometric constraints were introduced beyond those that already existed. The problem described
above can be represented as follows

\[
\min_u \sum_{i=2}^{H} (\Psi - Y)^T Q(\Psi - Y) + (\Phi(i - 1))^T R(\Phi(i - 1)) + \tau \cdot \theta^2
\]  

(5.20a)

subject to  
\[
x_{1}(1, i + 1) = x_{1}(1, i) + v_{1}(i) \cdot \cos(x_{1}(3, i)) \cdot h \cdot \alpha_{1}, \quad x_{1}(1, 1) = 1, \quad (5.20b)
\]
\[
x_{1}(2, i + 1) = x_{1}(2, i) + v_{1}(i) \cdot \sin(x_{1}(3, i)) \cdot h \cdot \alpha_{1}, \quad x_{1}(2, 1) = 2, \quad (5.20c)
\]
\[
x_{2}(1, i + 1) = x_{2}(1, i) + w_{1}(i) \cdot h \cdot \alpha_{1}, \quad x_{1}(3, 1) = 45^\circ, \quad (5.20d)
\]
\[
x_{2}(1, i + 1) = x_{2}(1, i) + v_{2}(i) \cdot \cos(x_{2}(3, i)) \cdot h \cdot \alpha_{1}, \quad x_{2}(1, 1) = x_{1}(1, 1) + \Gamma_{x}, \quad (5.20e)
\]
\[
x_{2}(2, i + 1) = x_{2}(2, i) + v_{2}(i) \cdot \sin(x_{2}(3, i)) \cdot h \cdot \alpha_{2}, \quad x_{2}(2, 1) = x_{1}(2, 1) - \Gamma_{y}, \quad (5.20f)
\]
\[
x_{2}(3, i + 1) = x_{2}(3, i) + w_{2}(i) \cdot h \cdot \alpha_{2}, \quad x_{2}(3, 1) = x_{1}(3, 1), \quad (5.20g)
\]
\[
(x_{1}(1, i + 1) - c_{x}^g)^2 + (x_{2}(1, i + 1) + c_{y}^g)^2 < d_{\text{min}}^2, \quad (5.20h)
\]
\[
|v_{1}(i + 1)| \leq 20, \quad (5.20i)
\]
\[
|w_{1}(i + 1) - w_{1}(i)| \leq \frac{\pi}{3}, \quad (5.20j)
\]
\[
|v_{2}(i + 1)| \leq 30, \quad (5.20k)
\]
\[
|w_{2}(i + 1) - w_{2}(i)| \leq \frac{\pi}{3}, \quad (5.20l)
\]
\[
river 1, \text{ where } x \in \text{Re}, \quad y \in [30; 32] \text{ meters}, \quad (5.20m)
\]
\[
river 2, \text{ where } x \in \text{Re}, \quad y \in [60; 62] \text{ meters}, \quad (5.20n)
\]

where \(c_{x}^g\) and \(c_{y}^g\) with \(g = 1, 2, 3, 4\) are the \(x\) and \(y\) coordinate of the center of each of the obstacles, and \(d_{\text{min}}\) is the radius. Based on what was presented, it is possible to conclude that there are four geometrical constraints. The coordinates of the obstacles can be observed in table 5.8.

<table>
<thead>
<tr>
<th>Obstacles</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate x [m]</td>
<td>20</td>
<td>30</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>Coordinate y [m]</td>
<td>15</td>
<td>50</td>
<td>55</td>
<td>75</td>
</tr>
<tr>
<td>Radius [m]</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 5.8: Coordinates of the obstacles.

The parameters used for the situation described in the previous problem definition are very similar to the ones presented in table 5.6. The only changes made was at the level of the number of iterations \(Z\) that was increased to 300, giving vehicles enough time to reach the goal, and at the level of the final state to be reached. In this case the final state is \(Y = [80; 90; 45^\circ]\). Based on what was previously presented and after applying the optimization process using a value of \(\tau = 1000\), the result obtained from the optimal path calculation can be observed in figure 5.21.
Figure 5.21: Calculated path using vehicle interaction, obstacle avoidance, and task execution.
Chapter 6

Results and Discussion

Chapter 5 presented the route that was taken and the improvements that were developed to obtain a valid solution for the case of the two rivers. Initially, simpler cases were presented that served as a support for understanding important concepts used in the most complex examples. As the situation became more complicated, new parameters were chosen and new cost function terms were added to obtain a feasible solution. However, the choice of new parameters and additional terms may not be enough to achieve the goal. Thus, this chapter will present the introduction of a new external supervisor that will help the optimization process in situations where the solution is unfeasible.

6.1 External Supervisor

As mentioned earlier the chosen parameters \((Q, R, Z, T, H)\) may not always make the solution feasible. One of the options to solve this problem would be to modify these parameters and test others. However, this is not the best way to proceed, as this makes the algorithm, developed for the controller, less robust. To avoid this, keeping all parameters constant, it is necessary to develop a supervisor that, based on the information received, changes other characteristics that are not directly related to the parameters defined for the optimizer and the vehicle dynamics. Given what is intended initially, it is necessary to understand how a solution will be identified as feasible or unfeasible.

In the performance of the various simulations, it was observed that, although the vehicles could reach their objectives, during their journeys there were situations where the second vehicle did not respect all the imposed geometrical restrictions. This is because there are many restrictions applied to the vehicle and there is a conflict in the optimal state calculation by the optimizer. Thus, there will be states that cannot be considered for the optimum path as they cause the vehicle to collide with a particular obstacle. These states can be identified by the \textit{fmincon} solver.

It has also been found that in some situations, vehicles could not deviate from obstacles in advance. This causes the agents to stop moving as they approach the edges of an spacial constraint. Although they do not collide and respect all restrictions, the ultimate goal is never achieved.

So far, the syntax used using the Matlab tool has been the same as in equation (3.4). To identify
the state that makes the solution unfeasible a new output variable has been added and will be returned by the \textit{fmincon} solver. Therefore, with the new output variable equation (3.4) can be modified, obtaining the syntax presented in equation (6.1), where $x$ is the minimizer, $fval$ is the value of the objective function $fun$, and $exitflag$ describes the exit condition of \textit{fmincon} \cite{54}.

\[ [x, fval, exitflag] = \texttt{fmincon}(fun, x_0, A, b, A_{eq}, b_{eq}, x_{lb}, x_{ub}, nonlcon, options, P_1, P_2, ...), \]  

(6.1)

Parameter $exitflag$ can acquire different values, but the value that corresponds to an unfeasible solution is -2. The solution that is obtained by \textit{fmincon} solver is the input vector $u$ that has the size of the horizon $H$. If a value of the input vector causes the vehicle not to comply with the imposed restrictions, the $exitflag$ value is immediately -2, even if the remaining values of the array are as intended. This input value will influence future values causing the vehicle to collide with obstacles. Therefore, in each iteration, the supervisor will look at the value of the $exitflag$ variable to see if the input obtained makes the solution feasible or not. If the value of the variable is -2, the optimization process stops and the supervisor makes modifications at the vehicle level, restarting again the simulation. The scheme that represents the interaction between the supervisor, the MPC and the vehicle can be seen in figure 6.1.

![Figure 6.1: Interconnection between the Supervisor, MPC and the Vehicle.](image)

The modifications made by the supervisor at the vehicle level correspond to changes on the initial position, that is, the vehicle is placed slightly beside the previous location, as it is sometimes enough to help the vehicle find another way that makes the solution feasible. Therefore, a vector with positions will be constructed, changing the $x$ and $y$ coordinates. If the supervisor finds an unfeasible solution it will change the initial position of the agents with another pair of coordinates from the constructed vector. If the solution remains unfeasible, other vector positions will be tested until a valid solution is found.

To understand how the supervisor works two practical examples are presented. The environment, the vehicle dynamics and the simulation parameters used are the same as in section 5.6.1. The first example can be seen in figure 6.2.

Analyzing figure 6.2a, it is observed that when starting the simulation at $x = 10$ and $y = 0$ for the first vehicle, the second agent at a given time collides with obstacle 2, preventing the vehicle from reaching the final position. Analyzing figure 6.2b it is possible to see that starting the simulation in the same coordinates ($x = 10$ and $y = 0$ for vehicle 1), vehicle 2 respected all constraints. This is because the supervisor, after analyzing the $exitflag$ value for $x = 10$ and $y = 0$, noted that this corresponded to an unfeasible solution and modified the initial coordinates. Thus, the coordinates of the first agent that make the solution feasible are $x = 12$ and $y = 0$. The second example can be observed in figure 6.3.
Observing figure 6.3a it is possible to see that when the coordinates of the first vehicle are \(x = 20\) and \(y = -1\), both agents stop their movement when they encounter an obstacle and cannot deviate from it in advance. By applying the supervisor it is possible to obtain the result shown in figure 6.3b. In this figure, the initial coordinates of the first vehicle are the same as in figure 6.3a. Nevertheless, it is observed that the simulation started at \(x = 20\) and \(y = 9\). These are the coordinates that the supervisor, after analyzing, found that made the solution feasible. Therefore, by starting the simulation at \(x = 20\) and \(y = 9\) it was possible to obtain an exit flag value always different from -2 during all iterations.
Chapter 7

Conclusion

The goal of this dissertation is to design an optimal controller capable of calculating a sequence of control inputs of a mobile robot, based on its model, that allows the minimization, in a receding horizon sense, of the total cost of the path obtained after applying the calculated inputs. To achieve this goal, other characteristics are taken into account, such as environmental obstacles, number of agents, existing tasks and also all the constraints applied to the vehicles. This dissertation starts with the problem formulation presented in chapter 1, where all the important steps and tools are described as well as their interaction to create the final model.

Chapter 2 starts with an introduction to Model Predictive Control. This chapter presents its history, how it came about, and where it is used. The main characteristics and qualities that led to the increasing use of the various MPC algorithms were referenced. Some points to bear in mind, when making use of this model (considered to be challenges), were also presented. For a better understanding of how MPC works, its principle was described, where the steps necessary to calculate an optimal control sequence were explained to avoid the propagation of disturbances to future states. The structure of MPC was also presented, where it was possible to observe the variables that were considered by the model, the parameters that were used by the optimizer and the way the output was processed to get closer and closer to the desired reference.

Although there are different MPC algorithms, all of them are constituted by the same elements. In chapter 2, each one of these elements was presented. Initially, the concept of plant model was clarified. In this part, state-space equations in discrete time were described for both a nonlinear and time-variant system as well as a linear and time-invariant system. Its representation in matrix format was displayed. After that, the output equations were presented, in matrix and non-matrix format. To reach the reference, MPC algorithms use different performance measurements, and in this chapter, a general expression of the objective function used during the dissertation is presented. Considering this general cost function all its parameters were explained and clarified.

Since all processes are subject to restrictions several examples of general kind are presented. The inclusion of constraints increase the complexity of the minimization process to be performed by the optimizer. Chapter 2 finishes with the introduction of Nonlinear Model Predictive Control where the
advantages of this approach are presented as well as some difficulties that are still present when using nonlinear dynamic systems.

Chapter 3 starts with an introduction to Optimal Control Theory. Different problems can be studied in OCT such as Convex and Non-convex problems. Their differences were presented by defining each problem and also using visual support through two figures obtained using MatLab. The convex and non-convex problems can be solved using unconstrained or constrained solvers. Both solvers were explained and defined, presenting the syntax used for each one of them. In the constrained solver, as there are constraints involved, an example was made where it was possible to understand what each solver variable represented and how it could be written. For each solver, there are several optimization algorithms from which it is possible to obtain different results. In this chapter two algorithms were analyzed, the IP and the SQP, where their differences were enumerated and the purpose of using each one of them was clarified.

Given the constrained method it is possible to have two types of restrictions, linear and nonlinear and each one of them can be subdivided into equalities and inequalities constraints. In this part of chapter 3, it was presented the syntax that was used to define each restriction. At the level of nonlinear inequalities, some examples of constraints defining prohibited zones of space were presented, limiting the values that future states can acquire. After that, the Q and R matrices were introduced, explaining their influence on control calculation.

To better understand the influence that the different tuning parameters exert on the final state calculation, an example was presented. The example was started by changing the value of parameter R keeping constant the others (Q and H). It was possible to conclude that with the increase of R the final state deviates more and more from the desired reference since in this case it is more important to minimize the input u. The calculated input acquires smaller values, not being sufficient for the system to converge to the reference. Then, the values of the diagonal entries of matrix Q were changed, keeping the remaining parameters constant. For this situation it was observed that increasing the diagonal values of Q the system converges faster to the reference, reducing the error between the target and the final state. This happens since, this increase gives greater importance to the minimization of state x, which acquires smaller values. This will enable input u to acquire larger values, allowing the system to converge faster. Finally, to fully reduce the error between the final state and the reference, for both cases (increasing Q or R) an extra parameter was introduced in the cost function that prevents the final state from deviating from the reference. The process of obtaining this extra term was explained.

In chapter 4 three different vehicle models were introduced, allowing us to understand how each one behaves, what are the differences between them and what are the state space equations that allow us to characterize each model. A deeper characterization of the tricycle model was made since it is this model that allows to represent a three-wheeled vehicle having a spatial orientation. As the tricycle model was originally in continuous time, it was necessary to proceed to a temporal discretization. With this, two different methods were introduced that allow to discretize the model, the Euler method and the Runge-Kutta method.

Chapter 5 started with an introduction to Cooperative Control. Its definition was clarified and
different methods of cooperative control were stated. Methods such as decentralized/centralized control and game theory were explained, and the method used during dissertation simulations was introduced. Before proceeding with the simulations, an initial study was developed, to understand how two different cost functions can influence the final result. Therefore, using the double integrator model two distinct objective functions were tested in terms of input $u$ variation over time and total cost of the system. The main difference between the two cost functions is the input term wherein one of them is squared and in the other has the absolute value. After analyzing the value of the input it was possible to observe that when $u$ has the absolute value, the behavior is bang-bang i.e., the input varies from one extreme to the other. If $u$ is squared its variation is more gradual and smooth. In terms of the total cost obtained by each of the functions, it was found that the final value was of the same order of magnitude, and either one could be used to achieve the desired result.

Then, a study was carried out in which the value of the horizon $H$ was varied so that its influence on the prediction of obstacles and the calculation of future states could be verified. This study focused on the variation of the total cost and the computational time that the simulation took, for the situation where there was only one obstacle and for the situation where there were several obstacles. For both cases, it was found that for low $H$ values the cost was high. However, as $H$ increases, the cost decreases reaching an $H$ that makes the cost minimal. If this value of $H$ is exceeded, the total cost increases again. Regarding the computational time, it was possible to verify that as $H$ increases, the simulation time also increases, and the choice of $H$ is mostly made based on the total cost of the system. For the case where there were several obstacles, simulations were presented where the value of $H$ was varied, and it was possible to observe the obtainment of different final paths based on the chosen horizon values.

After the previous studies, the tricycle model was introduced, where the characteristic that makes this model different from the double integrator model was presented. This feature corresponds to the orientation of the agent. Given this model, a dynamic obstacle has been introduced, allowing us to observe how the optimizer reacts in calculating the optimal path. When the movement of the obstacle is perpendicular to the movement of the vehicle, the agent stops its displacement in time, letting the obstacle pass. After its passage, the vehicle resumes its trajectory to the reference, being possible to conclude that the horizon is not a determining parameter to obtain an optimal solution for this case. If the dynamic obstacle moves towards the vehicle it was observed that for a horizon of 10, the agent could not identify the obstacle in advance and was forced to move in the opposite direction to avoid colliding with it, never reaching the reference. Increasing the horizon to 20 it was found that the agent identified the obstacle in advance, calculating a trajectory that allows to circumvent it, avoiding colliding with it and achieving the reference. In this case, it is possible to conclude that the horizon has a very significant influence on the final result.

To study the interaction between vehicles, a new agent was added, also having the dynamics of the tricycle model. Having both agents, two objectives were introduced: to keep the distance between them always equal to $2m$ and try to maintain the initial angle that the second vehicle makes with the first one during the simulation, whenever possible. Two diamond-shaped obstacles were introduced.
to check whether both conditions were being met. After analyzing the simulation it was possible to observe that the distance condition was being respected. However, the orientation was only being corrected close to reaching the reference. Thus, a new parameter was introduced in the cost function that corresponds to the angular deviation obtained from the angle of the agents at each moment and the reference angle (initial and final angle of the simulation). Along with this new parameter was added a weight that multiplies the extra term of the cost function to give it importance. Several weight values were tested and it was concluded that the value which allowed at the same time for the quick correction of the angle between the agents and to respect all constraints imposed on the model was 800. As the second vehicle calculates its future states based on the states of the first agent, it can be concluded that this new term will only be added to the cost function of the second agent. It has also been observed that the maximum speed of the second vehicle needs to be increased since, if the first agent moves at a maximum speed of 20 m/s and the second agent is running behind the first, the speed of the second vehicle needs to acquire values greater than 20 m/s to be able to catch the first one, being possible to correct the angular deviation between them. After applying these new changes to both the cost function and vehicle dynamics of the second agent, the simulation was performed again. In this case, it was observed that the angular correction was done faster, still respecting the distance between the agents.

Another aspect of using two vehicles was the execution of tasks together. The task that was used for the next simulations was crossing a river that was situated in the middle of the map. In this case, it was necessary to proceed with two optimization methods, one within the other. The main one calculates the optimal crossing point of the river, while the second one calculates the trajectory to the reference. Initially, the simplest situation was presented where there were no obstacles in the environment. Since both means, before and after the river, were equal, the crossing point was exactly in the middle of the river’s length. In this situation, it was possible to observe two distinct movements, associated to two different states of a state machine, the normal movement of both vehicles before and after the river, and the crossing of the river. To observe the consistency of the model, viscosities were added to both media. Having different viscosities it was found that the calculated path was longer in the medium where the viscosity was lower. This is in line with the model since the optimizer always tries to decrease total system spend. In this case, having the vehicles a longer path in the less viscous media allows to decrease the total cost of the system.

After testing the simplest case of task execution, it was decided to complicate the problem. Several obstacles were added and the composition of both media (before and after the river) was equal. By applying the optimization process to the problem using a weight of 800, as calculated previously, for the angular deviation, it was possible to verify that the result obtained behaved differently than expected. After testing other values, it was concluded that the weight that yielded a result very close to what was expected was 1000. This is explained by the fact that as several obstacles were introduced in the path of the vehicles, their orientations were always varying. This provided a greater need to control the angular deviation between both agents. However, despite this, both simulations present a valid result, since in both cases the vehicles reach the reference respecting all the imposed restrictions. Thus, to determine which was the best weight value, a study was made, based on the total cost obtained using each weight.
Although both results showed cost values of the same order of magnitude, the cost obtained by the weight of 1000 was lower than the cost obtained by the weight of 800, concluding that increasing the weight value to 1000 was an improvement in the optimization process.

The last problem dealt with in chapter 5 was the crossing of two rivers by two agents, while avoiding several obstacles placed in such a way as to verify all the properties developed throughout this chapter. In this situation, it was again necessary to perform two optimization processes, one within the other. The external process, corresponding to the crossing point calculation, was performed twice, once for each river. The interior process was performed several times and corresponded to the calculation of the path between two points, for example between the starting point and the first crossing point on the first river. Analyzing the result obtained it was possible to conclude that all constraints and obstacles were respected throughout the simulation. It was also possible to conclude that regardless of the number of rivers added, the developed controller can always calculate an optimal solution, allowing the vehicles to reach the final goal. Nevertheless, the problem was tested at most with two rivers, because whenever a new river is added the simulation time increases considerably.

In chapter 6 a supervisor was developed. The need to develop a supervisor appeared, since sometimes when the initial coordinates of the agents were not favorable, the solution obtained presented faults in the calculated states. This supervisor at each moment analyzed the value of a variable returned by the fmincon solver. This variable indicated whether the solution obtained was feasible or unfeasible. If the solution was unfeasible, the supervisor would modify the initial coordinates of the vehicles, restarting the simulation. The coordinates were altered until finding a pair that made the solution feasible. Thus, two distinct cases were presented where it was possible to observe the usefulness of the developed supervisor. In the first case, a situation arose in which the second vehicle did not respect an obstacle, having some of its states situated within it. This is because there are many restrictions applied to the vehicle and there is a conflict in the optimal state calculation by the optimizer. In the second case, a situation was presented where both agents, upon reaching an obstacle, stop their movement, failing to reach the reference. In both problems, the supervisor was applied. It was found that, after the supervisor detected that the current solution was not feasible, the initial coordinates of the vehicles were changed, until a pair was found, which allowed to obtain a trajectory where both agents respected all the restrictions and could achieve the desired reference. With this it was concluded that the developed supervisor made it possible to improve the optimization process, allowing to obtain an optimal solution even when the initial conditions are not the most favorable.

7.1 Future work

For the work developed in this dissertation some changes can be considered in terms of some key points:

- The Rapidly-Exploring Random Trees (RRT) algorithm can be used to help the MPC;
- Using other optimization methods to achieve distributed cooperative control;
• Other vehicle models can be used to test the optimizer in the optimal state calculation;

• Following dynamic references.

In future work, the [Rapidly-Exploring Random Trees (RRT)] algorithm could be used [4, 48]. RRT corresponds to an algorithm that based on a search method finds the path between two points in a high dimensional space. This algorithm allows to generate a set of random points on the map and based on these points and a specific factor of the algorithm allows to join them, making a tree from the start point to the endpoint. This algorithm can be related to the $f_{\text{mincon}}$ solver.

Consider the situation presented in section 5.9 where there are two vehicles with tricycle dynamics, several obstacles, and a river to be crossed. In this situation, it is necessary to apply two optimization processes, a main which allows to find the optimal point to cross the river, and a secondary that allows to calculate all the states of the vehicles. The RRT algorithm can be applied individually to the surface before the river and to the surface after the river, being always necessary to know the starting point and the finishing point. This allows to generate a large set of random states and link them based on the distance between them, the factor specified for the algorithm and also based on the constraints applied to agent velocity. After obtaining the tree that joins the starting point and the ending point, it can be possible to calculate the input vector that gives rise to the states belonging to the built graph. The input vector can be supplied to $f_{\text{mincon}}$ as the initial vector to begin the optimization process. This vector, in principle, should have inputs close to the optimal inputs that can be obtained by the normal optimization process. Providing this vector to $f_{\text{mincon}}$ can work as an aid, decreasing the search time to find the optimal input value, and in the end, decreasing the total computation time.

This work could have been developed using other optimization methods that can achieve a distributed cooperative control between vehicles. Other examples of cooperative control methods that could have been used are distributed control or game theory. In game theory, both vehicles, at each time step, communicate to each other their optimum input. This is done several times until a value is found that is considered optimal for both agents. After finding this optimal value, both vehicles apply the input to their dynamics, advancing to the next state. This process is repeated for each state until they reach the reference. The process of exchanging input information between vehicles can be done many or few times, depending on the purpose of each problem. A disadvantage of this method, when many information exchanges are made, would be the high computational time that the simulation would take to obtain an optimal trajectory.

The third suggestion for future work based on this dissertation would be to apply the developed controller to other vehicle dynamics, such as air vehicles with more states of freedom. For this to work it would be necessary to make minor adjustments to the parts of the algorithm where the dynamics of vehicles come in. However, after making these minor adjustments, it would be possible to verify and validate the proper operation of this controller.

Finally, the last field that can be studied with greater attention in future work is the tracking of a dynamic reference. During the dissertation some tests were done, however there was no major development, since this was not a fundamental problem to be solved for the subject of the dissertation.
Bibliography


Appendix A

Mathematical Equations for Prediction Model

A.1 State and output equations

To address this section it will be necessary to use the equation \textcolor{red}{(2.5)} and also the principle of the Receding Horizon method. Considering a finite prediction horizon $H$, the general form that equation \textcolor{red}{(2.5)} can take can be observed in \textcolor{red}{(A.1)}

\[
\begin{align*}
x(k+1|k) &= Ax(k|k) + Bu(k|k), \\
x(k+2|k) &= A^2x(k|k) + ABu(k|k) + Bu(k+1|k), \\
x(k+3|k) &= A^3x(k|k) + A^2Bu(k|k) + ABu(k+1|k) + Bu(k+2|k), \\
x(k+4|k) &= A^4x(k|k) + A^3Bu(k|k) + A^2Bu(k+1|k) + ABu(k+2|k) + Bu(k+3|k), \\
&\vdots \quad \vdots \\
x(k+H|k) &= A^Hx(k|k) + A^{H-1}Bu(k|k) + A^{H-2}Bu(k+1|k) + \ldots + ABu(k+H-2|k) + \\
&\quad + Bu(k+H-1|k).
\end{align*}
\]  
\textcolor{red}{(A.1)}

Based on equation \textcolor{red}{(2.8)} and the system of equations presented in \textcolor{red}{(A.1)} it is possible to display the system output as follows

\[
\begin{align*}
y(k+1|k) &= CAx(k|k) + CBu(k|k), \\
y(k+2|k) &= CA^2x(k|k) + CABu(k|k) + CBu(k+1|k), \\
y(k+3|k) &= CA^3x(k|k) + CA^2Bu(k|k) + CABu(k+1|k) + CBu(k+2|k), \\
y(k+4|k) &= CA^4x(k|k) + CA^3Bu(k|k) + CA^2Bu(k+1|k) + CABu(k+2|k) + CBu(k+3|k), \\
&\vdots \quad \vdots \\
y(k+H|k) &= CA^Hx(k|k) + CA^{H-1}CBu(k|k) + A^{H-2}CBu(k+1|k) + \ldots + CABu(k+H-2|k) + \\
&\quad + CBu(k+H-1|k).
\end{align*}
\]  
\textcolor{red}{(A.2)}
Using equation (2.9) it is possible to calculate the output of the system for all the prediction horizon \( H \). The output of the system is represented in a matrix form in equation (A.3)

\[
\begin{bmatrix}
  y(k+1|k) \\
  y(k+2|k) \\
  \vdots \\
  y(k+H|k)
\end{bmatrix} =
\begin{bmatrix}
  W_1 & 0 & \cdots & 0 \\
  W_2 & W_1 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  W_H & \cdots & W_2 & W_1
\end{bmatrix}
\begin{bmatrix}
  u(k|k) \\
  u(k+1|k) \\
  \vdots \\
  u(k+H-1|k)
\end{bmatrix} +
\begin{bmatrix}
  \Gamma_1 \\
  \Gamma_2 \\
  \vdots \\
  \Gamma_H
\end{bmatrix}
\begin{bmatrix}
  x(k|k) \\
  x(k+1|k) \\
  \vdots \\
  x(k+H-1|k)
\end{bmatrix}.
\]  

(A.3)

From equation (A.2) it is possible to deduce the expression for the calculation of matrix \( W \) and \( \Gamma \). Therefore, the expressions to calculate matrix \( W \) and \( \Gamma \) can be observed in equation (A.4)

\[
W_1 = CB, \\
W_2 = CAB, \\
\vdots \\
W_H = CA^{H-1}B.
\]  

\[
\Gamma_1 = CA, \\
\Gamma_2 = CA^2, \\
\vdots \\
\Gamma_H = CA^H.
\]  

(A.4)

The representation of the previous equations in the matrix form can be observed in equation (A.5)

\[
W =
\begin{bmatrix}
  CB & 0 & \cdots & 0 \\
  CAB & CB & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  CA^{H-1}B & \cdots & CAB & CB
\end{bmatrix}, \\
\Gamma =
\begin{bmatrix}
  CA \\
  CA^2 \\
  \vdots \\
  CA^H
\end{bmatrix}.
\]  

(A.5)

The matrix form represented in (A.3) can be written in a more simplified format. Therefore, the prediction of the system, at time \( k \), can be expressed by

\[
Y_k = WU_k + \Gamma x(k).
\]  

(A.6)

### A.2 Cost function

There are different types of cost function, and each one of them depend on the problem that is being studied. The cost function that was used for this work, assuming always the same reference, can be written as

\[
J = \sum_{i=1}^{H} \left( \|y(k+i|k) - Y(k+1|k)\|_Q^2 + \|u(k+i-1|k)\|_R^2 \right).
\]  

(A.7)

The last equation can also be written as

\[
J = \sum_{i=1}^{H} \left( x(k+i|k) - Y(k+1|k) \right)^T Q (x(k+i|k) - Y(k+1|k)) + u(k+i-1|k)^T R u(k+i-1|k) \).
\]  

(A.8)

Assuming that \( Q \) corresponds to a square matrix and \( R \) is a scalar with unit value equal to 1, equation (A.8) can be written as

\[
J = (\hat{Y} - \tilde{Y})^T Q (\hat{Y} - \tilde{Y}) + U^T R U,
\]  

(A.9)

where \( \hat{Y} \) corresponds to the prediction of the future state generated by the model and \( \tilde{Y} \) corresponds to the reference to be followed. Parameter \( U \) represents the predicted future control actions.
Using the equation (A.6) to replace \( \hat{Y} \) in equation (A.9) it is possible to obtain equation (A.10)

\[
J = (WU + \Gamma x - \hat{Y})^T Q (WU + \Gamma x - \hat{Y}) + U^T RU. \tag{A.10}
\]

However, what is the optimal value of \( U \)? To answer this question equation (A.10) must be solved. Therefore, the first stage of finding the optimal value of \( U \) is

\[
J = (U^T W^T + x^T \Gamma^T - \hat{Y}^T) Q (WU + \Gamma x - \hat{Y}) + U^T RU =
\]

\[
= U^T (W^T Q W + R) U + 2 U^T W^T Q (\Gamma x - \hat{Y}). \tag{A.11}
\]

To find the optimal value of \( U \) it is necessary to compute the gradient of equation (A.11) with respect to \( U \), as represented in the next equation

\[
\Delta_U J = 2 U^T (W^T Q W + R) + 2 (x^T \Gamma^T - \hat{Y}^T) Q W. \tag{A.12}
\]

Considering \( M = W^T Q W + R \), for simplification reasons, and equate the result from equation (A.12) to zero, it is possible to obtain the following equation

\[
\Delta_U J = \frac{\delta J}{\delta U} = U^T M + (x^T \Gamma^T - \hat{Y}^T) Q W = 0,
\]

\[
MU = -W^T Q (\Gamma x - \hat{Y}), \tag{A.13}
\]

\[
U = - M^{-1} W^T Q (\Gamma x - \hat{Y}).
\]

Therefore, the optimal value for \( U \), represented by \( \hat{U} \), can be obtained by the equation (A.14)

\[
\hat{U} = - M^{-1} W^T Q (\Gamma x - \hat{Y}). \tag{A.14}
\]

Applying equation (A.14), a vector is obtained as represented in equation (A.15)

\[
\hat{U} = \begin{bmatrix}
\hat{u}(k|k) \\
\hat{u}(k + 1|k) \\
\hat{u}(k + 2|k) \\
\vdots \\
\hat{u}(k + H - 1|k)
\end{bmatrix}. \tag{A.15}
\]

Although a vector is obtained, only the first element of \( \hat{U} \) will be used to calculate the current state. Therefore, by doing \( u(k) = \hat{u}(k|k) \) and multiplying equation (A.14) by a vector, as shown in equation (A.16), it is possible to represent the input as

\[
u(k) = -[1 \ 0 \ 0 \ \ldots \ \ldots \ 0] M^{-1} W^T Q (\Gamma x - \hat{Y}). \tag{A.16}
\]

After computing the value of the current state, based only on the first element of \( \hat{U} \), the process is repeated, being necessary to calculate everything again in order to obtain the next state and so on.
Appendix B

Solution of equation in example 2

The initial equation presented in section 3.8 can be described as

\[ y(k) - 1.5y(k-1) + 0.7y(k-2) = u(k-1) + 0.5u(k-2). \]  \hspace{1cm} (B.1)

In order to obtain the fundamental parameters necessary for the optimization method, some prior calculation must be done. Therefore, using equation (B.1) it is possible to obtain the transfer function represented in (B.2)

\[
y(k+2) - 1.5y(k+1) + 0.7y(k) = u(k+1) + 0.5u(k),
\]

\[
h(k) = \frac{y(k)}{u(k)} = \frac{z + 0.5}{z^2 - 1.5z + 0.7},
\]

where \( z \) can be interpreted as a forward operator.

After obtaining the transfer function, a partition property will be applied in order to facilitate the process of obtaining the state space equations. This partition can be observed in the next figure.

![Polynomial division](image)

Figure B.1: Polynomial division.

With the previous division it is possible to develop two separate sets of calculations. The first one corresponds to the determination of the state space equations as represented in equation (B.3)

\[
x_1(z) = \frac{1}{z^2 - 1.5z + 0.7} u(z),
\]

\[
x_1(k + 2) = 1.5x_1(k + 1) - 0.7x_1(k) + u(k).
\]

The second set of calculations corresponds to the output equation that can also be calculated based on the figure B.1. Therefore, \( y(k) \) can be presented as

\[ y(k) = 0.5x_1(k) + x_2(k). \]  \hspace{1cm} (B.4)
Making $x_1(k+1) = x_2(k)$ and $x_2(k+1) = x_1(k+2)$, the resulting equations for $x_1(k+1), x_2(k+1)$ and $y(k)$ can be written as

$$x_1(k+1) = x_2(k), \quad (B.5a)$$

$$x_2(k+1) = -0.7x_1(k) + 1.5x_2(k) + u(k), \quad (B.5b)$$

$$y(k) = 0.5x_1(k) + x_2(k). \quad (B.5c)$$

In equation (B.5) it is possible to observe the state and the output equations. These equations can be described in a matrix form as presented in equation (B.6) and (B.7)

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -0.7 & 1.5 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k), \quad (B.6)$$

$$y(k) = \begin{bmatrix} 0.5 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}. \quad (B.7)$$

Based on the calculations made previously on obtaining the state space equations for $x_1(k+1)$ and $x_2(k+1)$ and also the output system equation $y(k)$, it is possible to conclude that the different parameters needed for the optimization method (minimization of $J$) are

$$A = \begin{bmatrix} 0 & 1 \\ -0.7 & 1.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0.5 & 1 \end{bmatrix}.$$

The process of obtaining the previous matrices could be done by using the function `tf2ss` in `matlab`, however this function will provide a different state representation, changing all the matrices.