Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(36^{th}cycle)

Real-Time Nonlinear Model Predictive Control: Efficient Methods for Domain Reduction

By

Mattia Boggio

******

Supervisors:
Prof. Carlo Novara
Prof. Michele Taragna

Doctoral Examination Committee:
Prof. Marco Casini, Referee, Università degli Studi di Siena, Italy
Prof. Maria Letizia Corradini, Referee, Università di Camerino, Italy
Prof. Teodoro Rafael Alamo Cantarero, Universidad de Sevilla, Spain
Prof. Laura Menini, Università degli Studi di Roma Tor Vergata, Italy
Prof. Alessandro Rizzo, Politecnico di Torino, Italy

Politecnico di Torino
2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Mattia Boggio
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Transire suum pectus
mundoque potiri

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Carlo Novara, for his support, guidance, and mentorship throughout the course of my research. His expertise and insights have been invaluable to my work. He has been a light of knowledge in the often-dark PhD journey. I would also like to extend my sincere thanks to my co-supervisor, Prof. Michele Taragna, for his precious advice, support and unforgettable experiences at the conferences in Bucharest and Boston.

I sincerely thank also my colleagues and friends of the Automatica Group: Cesare, David, Francesco, Giada, Lorenzo C., Lorenzo Z., and Michele. A special thanks goes to Giada. We navigated this challenging journey together, making the difficult days a little easier. Thank you for all your advice, which has helped me to become a better person.

I owe a debt of gratitude to my family. Thank you for the constant encouragement and for instilling in me the values of hard work and perseverance.

At last, my most special thanks go to to you, Patrizia. Your love, patience, and constant support have given me the strength and motivation to keep moving forward, even in the difficult moments. Looking back on these years, I realize that it was this PhD that brought our paths together and, without a doubt, you are the greatest success of this entire experience. No academic achievement can compare to the fortune of having you by my side. An important chapter of my life closes and an even more significant one begins, with you walking beside me.

This journey would not have been possible without all of you. Thank you.

Abstract

Nonlinear Model Predictive Control (NMPC) is an advanced control method, based on the online solution of a suitable optimal control problem (OCP). However, this operation may require high computational costs, which can compromise the use of NMPC in "fast" real-time applications.

To address this challenge, the thesis introduces two novel NMPC approaches, called Dimensionality Reduction and Side-length Reduction, designed to reduce the computational burden by minimizing the volume of the search domain. Both approaches comprise two phases: (i) offline data generation and preprocessing; (ii) online optimization. In the Dimensionality Reduction approach, the offline phase leverages gradient computation and nonlinear sparse identification techniques. In particular, the gradient computation is used to detect the directions of greatest variability of the optimization problem. This is accomplished by identifying the decision variables that exert the most significant influence on the OCP, called the most relevant decision variables. Subsequently, a nonlinear sparse identification technique is used to approximate from data the NMPC control law. During the online optimization phase, this approximation is exploited to provide a good starting point for the optimization algorithm. Furthermore, the OCP is exclusively performed on the most relevant decision variables, while for the less important ones, the values obtained from the approximation are used. On the other hand, the Side-length Reduction approach employs a Set Membership method to derive tight guaranteed bounds on the relevant decision variables during the offline phase, which are then used to effectively shrink the search space during the online optimization. These two approaches aim to reduce the number of decision variables and the size of the search domain, thereby significantly improving computational efficiency.

A comprehensive theoretical analysis is conducted to investigate the inherent computational complexity of the proposed methods, demonstrating how dimensionality and side-length reductions can decrease the number of mathematical operations required in a nonlinear optimization algorithm and improve its convergence rate. The theoretical findings are complemented by an extensive numerical analysis, where the proposed methods are tested on three realistic autonomous driving scenarios: lane keeping on urban roads, obstacle avoidance on rural roads, and parallel parking. These scenarios were chosen to showcase the versatility of the proposed methods in handling different tasks from simple lane keeping to complex maneuvers involving dynamic obstacles.

Performance evaluations compare the proposed methods with respect to standard NMPC approaches using both Software-In-the-Loop (SIL) and Hardware-In-the-Loop (HIL) simulations. The SIL simulations are conducted using Simulink, employing both Sequential Quadratic Programming (SQP) and Particle Swarm Optimization (PSO) algorithms to demonstrate the effectiveness of the methods across different optimization techniques. The HIL simulations involve implementing the NMPC algorithm on an NVIDIA Jetson Nano board, focusing on real-time performance. The results indicate that the proposed strategies achieve significant reductions in computation time without compromising the quality of the solutions. These methods demonstrate robustness and versatility, being adaptable to a wide range of optimization algorithms and real-time control applications.

List of Abbreviations

AD Algorithm Differentiation

ADAS Advanced Driver Assistance Systems

AQC Adiabatic Quantum Computation

asNMPC advanced-step NMPC

BFGS Broyden-Fletcher-Goldfarb-Shanno

CARLA Car Learning to Act

CLARA Clustering LARge Applications

CoG Center of Gravity

C/GMRES Continuation/Generalized Minimal Residual

DAE Differential-Algebraic Equation

DFP Davidon-Fletcher-Powell

DR-NMPC Dimensionality reduction NMPC

DP Dynamic programming

DSR-NMPC Dimensionality and side-length reduction NMPC

DST Dynamic Single-Track

eMMC embedded MultiMediaCard

FFS Feasible Function Set

GGN Generalized Gauss-Newton

GPU Graphics Processing Unit

HIL Hardware-In-the-Loop

HJB Hamilton-Jacobi-Bellman

IP Interior-Point

IPOPT Interior Point Optimizer

KKT Karush-Kuhn-Tucker

LHS Latin Hypercube Sampling

LICQ Linear Independence Constraint Qualification

LMPC Linear Model Predictive Control

LPDDR4 Low Power Double Data Rate 4

MBD Model-Based Design

MC Monte Carlo

MIMO Multiple-Input-Multiple-Output

MPBVP Multi-Point Boundary Value Problem

MPC Model Predictive Control

m-NMPC multiple shooting NMPC

NLP Nonlinear Programming

NMPC Nonlinear Model Predictive Control

NSI Nonlinear Sparse Identification

OCP Optimal Control Problem

PAM Partitioning Around Medoids

PID Proportional-Integral-Derivative

PSO Particle Swarm Optimization

PWA Piecewise Affine

PWNL Piecewise Nonlinear

QA Quantum Annealing

QC Quantum Computing

QP Quadratic Programming

QUBO Quadratic Unconstrained Binary Optimization

RAM Random Access Memory

RHS Receding Horizon Strategy

RMS Root-Mean-Square

RTI Real-Time Iteration

SIL Software-In-the-Loop

SM Set Membership

SODIMM Small Outline Dual In-line Memory Module

SQP Sequential Quadratic Programming

SR-NMPC Side-length reduction NMPC

St-NMPC Standard NMPC

SVD Singular Value Decomposition

s-NMPC single shooting NMPC

USB Universal Serial Bus

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The concept of automatic control, derived from the ancient Greek term *autòmatos*, meaning "acting of one's own will", has been a cornerstone of technological advancement throughout history. Despite the existence of various definitions, automatic control can be broadly defined as the application of control theory for the regulation of processes without direct human intervention. This concept aligns with the philosophical ideas of Heraclitus and Democritus, which suggest that every form of technology, in its essence, is concerned with the creation of something, called *techne*, that imitates nature. The earliest known control systems date back to 270 B.C., attributed to Ctesibius in Ptolemaic Egypt. He developed a floating regulator for a water clock, i.e., an ingenious mechanism that, using the principles of water pressure and flow, maintained a consistent measure of time without manual adjustments. Since then, our capacity to develop devices that control other machines has significantly improved, enabling us to delegate increasingly complex tasks to automated systems. This evolution is evident in the progression from Ctesibius's water clocks through the early proportional-integral-derivative (PID) controller for ship steering developed by Minorsky in the 1920s up to modern optimal control methods. In today's world, the applications of automatic control are pervasive, ranging from autonomous driving and aerospace to power grid operation and biomedics. These advancements demonstrate our improved ability to create devices that not only imitate but sometimes outperform human decision-making capabilities to achieve specific goals. While many refer to this phenomenon as artificial intelligence, it is essentially an extension of the principles of automatic control.

As we continue to push the boundaries of this field, we can expect to see further advancements that will shape the future of technology and society.

In this context, the thesis focuses on the Model Predictive Control (MPC), an advanced optimal control strategy widely used in industrial and technological domains (see, e.g., [1]). The fundamental principle underlying MPC is the use of a dynamical model of the plant to predict the future behavior of the system under different control inputs or policies [2]. This predicted behavior is then leveraged to select the optimal policy that best achieves the desired control objective. This optimization over predictions is similar to human decision-making, where choices are often based on anticipated future outcomes derived from internal models of the world. Figure 1.1 illustrates the basic principle of the MPC. It involves solving a sequence of optimal control problems (OCPs), potentially non-convex, at each sampling instance t over a prediction horizon T_p, using the current state x_0 of the system as initial state. The iterative resolution of these OCPs determines the predicted state $\hat{x}$ and the input trajectory that minimize a predefined loss function while satisfying state and input constraints. Although the optimization process produces an optimal control sequence, only the first control action u_0 is applied while the rest of the sequence is discarded, resulting in the so-called *receding horizon strategy*. At the next time instant, the horizon is shifted by one sample and the optimization is restarted with the information of the new measurements. Key advantages of MPC include flexibility in formulating objectives for multivariable systems and handling constraints on state, input, and output variables.

1.1 Motivations

Depending on the use of linear or nonlinear dynamics and constraints, MPC can be classified into two main methods: (i) Linear Model Predictive Control (LMPC), and (ii) Nonlinear Model Predictive Control (NMPC). LMPC, a mature technique that leverages constrained linear system models, has become popular in process industry control since the 1970s [3]. It can determine very efficient control strategies, particularly when it is possible to obtain linear models that can accurately represent the dynamics of the systems under control. However, real-world systems often exhibit complex and nonlinear behaviors

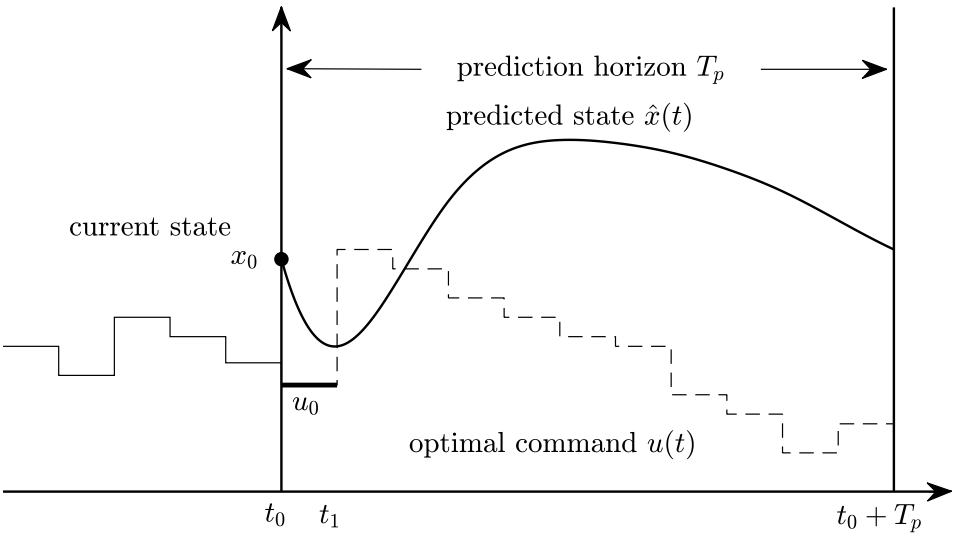

Fig. 1.1 The principle of model predictive control.

that linear models struggle to capture. This has led to a growing interest in NMPC thanks to its ability to handle nonlinear dynamics and constraints while optimizing nonconvex performance indexes [4].

Both LMPC and NMPC rely on suitable Optimal Control Problems (OCPs) to determine the optimal control actions. Although closed-form solutions are available in specific scenarios, solving these optimization problems is generally computationally intensive. As a result, initial applications of MPC were limited to process control systems with slow dynamics [5], allowing for sufficient computation time between control updates (sampling times in hours). However, for real-time systems requiring high sampling rates, such as autonomous driving, it is crucial to generate optimal commands within a short time interval, typically ranging from tens to hundreds of milliseconds. This requires the use of efficient optimization algorithms that can meet the real-time requirements of closed-loop control applications. In the context of linear systems, MPC problems can be formulated as convex optimization problems, e.g., quadratic or linear programs, for which fast and efficient algorithms have been developed [6]. Conversely, Nonlinear Model Predictive Control deals with nonconvex optimization, making it challenging to ensure global optimality (or at least a satisfactory level of suboptimality) within real-time constraints [7].

Despite these limitations, the NMPC algorithms can have the potential to significantly accelerate progress in many industrial sectors, including the development of autonomous driving technologies (see for example Figure 1.2). Considered one of the most transformative advancements in the near future, autonomous driving is expected to revolutionize the transportation sector (see, e.g., [8], [9] and [10]). In this regard, a huge research effort is being spent worldwide by automotive companies and academic institutions for developing vehicles with high levels of autonomy, ranging from advanced driving-assisted systems to fully automated vehicles (see, e.g., [11] and [12]). While traditional control methods like PID and linear state feedback are widely used in low-level industrial automotive applications due to their low computational cost, their performance is limited in complex scenarios requiring real-time decision making, such as emergency collision avoidance. Recent research in autonomous driving has explored Nonlinear Model Predictive Control (NMPC) with promising results [13]. However, the computational demands of NMPC and the limited resources available in real-world applications have hindered its widespread adoption in industrial autonomous driving systems. This underlines the pressing need for continuous development, especially in academia, of fast nonlinear optimization algorithms that can enable the use of NMPC for such applications.

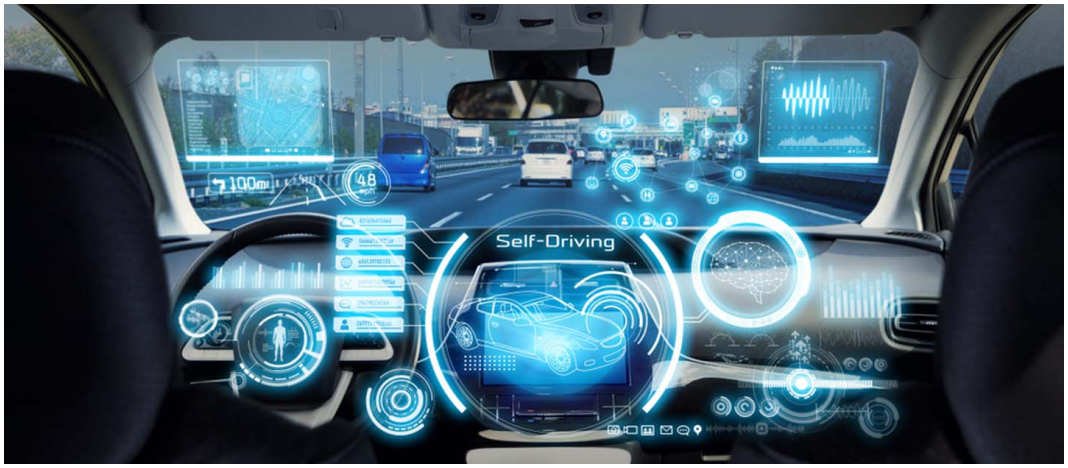

Fig. 1.2 Autonomous driving concept in the near future.

1.2 Contributions

In general, key elements that strongly affect the computation time of nonlinear optimization algorithms are the following:

- the dimensionality d of the search domain (i.e., the number of decision variables): the computational complexity of several algorithms is superlinear in the number of decision variables;
- the side-length s of the search domain (i.e., the region where the decision variables are defined): the larger is the size, the more computationally demanding is its exploration;
- the starting point of the algorithm: wrong choices of this point may result in long times required to find a solution.

In light of these issues, the main contribution of this thesis is to present novel NMPC approaches specifically designed to reduce the computational burden of the underlying optimal control problem by decreasing the volume (or hypervolume if $d > 3$) of the search domain. For the sake of brevity and clarity, we will always use the term "volume" throughout the thesis, even when $d > 3$.

Suppose that s^d is the volume of the search domain, where s represents its side length and d is the number of decision variables of the problem. The proposed approaches comprise two key strategies:

1. *Dimensionality Reduction*: this strategy operates on the parameter d employing gradient computations and approximating the NMPC control law.
2. *Side-length Reduction*: this strategy works on the parameter s using the Set Membership method proposed in [14].

As a result, both methods aim to decrease the volume of the search domain. However, they achieve this task by focusing on different parameters, requiring the use of distinct techniques. An example of how these two methods work to reduce the volume s^d of the search domain is shown in the Figure 1.3.

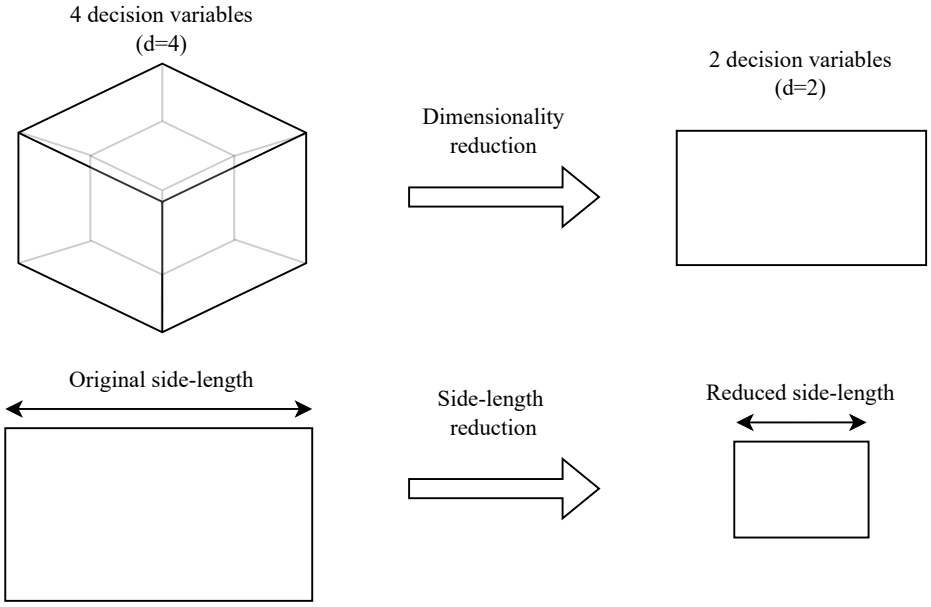

Fig. 1.3 Graphical illustration of how dimensionality and side-length reduction methods reduce the volume s^d.

Let us now examine the two developed strategies in more detail. Each of them comprises two phases: i) offline data generation and preprocessing; ii) online optimization. For the dimensionality reduction strategy, the offline phase leverages the gradient of the cost function, computed using a novel Gradient Computation method, to detect the directions of greatest variability of the optimization problem within the search domain. This is done by identifying the decision variables that exert the most significant influence on the OCP, called the *most relevant variables*. Subsequently, the Nonlinear Sparse Identification (NSI) technique, proposed in [15], is used to approximate from data the NMPC control law, i.e., the nonlinear function that links the state of the system to the optimal command. During the online optimization phase, this approximation is exploited to provide a good starting point for the optimization algorithm. Furthermore, the OCP is exclusively performed on the most relevant decision variables, while for the less important ones, the values obtained from the approximation are used. The second strategy, building upon the works presented in [16–18], applies the Set Membership (SM) method of [14] during the offline phase to derive tight bounds on the most significant decision variables. In

the subsequent online phase, these tight bounds are employed to reduce the side-length of the search domain exclusively for the variables involved in the OCP. These two approaches are aimed at decreasing the number of decision variables and the size of the search domain, thereby significantly improving computational efficiency. It can be observed that these two methods can be used either separately or in combination. Clearly, the combination of them leads to improved performance, enabling a significant reduction in computation time. This has far-reaching implications as it enhances the usability of NMPC strategies in real-time control systems, even when "short" sampling times are necessary.

This thesis makes two other key contributions. First, it demonstrates, through a theoretical analysis, the inherent computational complexity of the proposed methods. Specifically, this analysis provides insights into how volume reduction can both decrease the number of mathematical operations required for solving the numerical optimization and improve the convergence rate. The convergence rate, in this context, refers to the speed at which an iterative algorithm approaches its optimal solution.

Second, the thesis complements these theoretical findings with a numerical analysis. In particular, the proposed methods are tested on three realistic autonomous driving scenarios:

1. Lane keeping on an urban road, where the vehicle has to maintain a designated lane on a road with some curves;
2. Obstacle avoidance of roadworks on a rural road and safely re-entering the lane;
3. Parallel parking, avoiding collisions with other parked cars.

These scenarios were selected to demonstrate the versatility of the developed methods in handling a range of tasks, from simple lane keeping to more complex maneuvers involving obstacle avoidance and parking. The performance of the proposed methods was compared with respect to a standard NMPC, i.e., an NMPC that does not integrate dimensionality and side-length reduction approaches, using both Software-In-the-Loop (SIL) and Hardware-In-the-Loop (HIL) simulations. In the SIL simulation, the entire closed-loop control system

is designed in Simulink, a graphical programming environment for modeling and simulating dynamic systems. Conversely, in a HIL simulation, the NMPC algorithm is implemented and executed on a hardware platform. In this thesis, an NVIDIA Jetson Nano board is used. It should be noted that for the SIL part, two different nonlinear numerical optimization algorithms are used:

- Sequential Quadratic Programming (SQP) [19];
- Particle Swarm Optimization (PSO) [20].

The selection of these two algorithms aims to highlight the versatility of the proposed methods, demonstrating their effectiveness regardless of the specific optimization algorithm employed. Indeed, it is noteworthy that SQP is a local optimization algorithm, while PSO is a global one. For the HIL part, only the SQP algorithm was considered.

The benefits of using the proposed approaches are evident in all the considered scenarios and with the two different algorithms, achieving significantly better results in terms of computation time, without compromising the quality of the solution found. Furthermore, they demonstrate versatility beyond various optimization strategies. These methods can be integrated with a wide range of optimization algorithms, effectively enhancing their numerical efficiency and providing a flexible and adaptable framework for various applications. It is important to underline that the most significant advantages are observed when they are used with the single shooting methods, where the optimization variables are only the command inputs u. Consequently, both the developed methods consider this type of NMPC configuration. However, in recent years, efficient numerical algorithms have been developed for multiple-shooting methods, in which both x and u are considered as optimization variables, reducing their computational burden. In light of this, the results section includes also a comparison with respect to a state-of-the-art multiple-shooting NMPC configuration, showing that the proposed approaches make the single shooting NMPCs computationally more efficient than their multiple-shooting counterparts.

1.3 Comparison with the state-of-the-art

In the literature, different techniques have been proposed to address the problem of reducing the computational complexity of MPC. Among them, we can distinguish strategies based on:

- improvement of the numerical efficiency of the optimization algorithms;
- offline approximation of the control law;
- dimensionality reduction of the optimal control problem.

In the former category, specific online algorithms are developed to reduce the computational burden of the underlying nonlinear programming (NLP) problem. In [21], the multiple shooting algorithm with condensing is proposed. This method involves transforming sparse quadratic programming (QP) problems into dense ones by eliminating state variables from the optimization problem using condensing. Then, the resulting problems are solved with a dense QP solver. To minimize the time between state measurement and control feedback, in [22] an approach, called real-time iteration (RTI), is introduced to solve the NLP approximately. This method performs only one Sequential Quadratic Programming (SQP) iteration per time step and splits the calculations into a preparation and a feedback phase. The implementation of the multi-level version of RTI is described in [23], allowing for further reduction of the computational load. Moreover, the collocation method [24], the multistep Netwon-type controller [25] and the continuation/GMRES method [26] can be mentioned. For a more detailed review of numerical algorithms in the field of NMPC see, e.g., [27, 28] and Chapter 2 of this thesis.

In the second category, approximating functions, computed offline on the basis of a finite number of "exact" control moves, are used to reproduce the MPC/NMPC law f^0. A pioneering contribution along this line was given in [29], using a multilayer feedforward neural network for approximating the control law f^0. However, this approach did not provide any guaranteed approximation error or constraint satisfaction properties. Furthermore, the non-convexity of the functional used in the "learning phase" of the neural network can lead to potential deterioration in the approximation due to entrapment in local

minima. An alternative approach for approximating an NMPC controller has been proposed in [30], which uses an offline approximate multi-parametric programming technique to construct a piecewise affine approximation (PWA). Analogous methods have also been applied to linear systems [31, 32], and extended to the robust min-max case [33], employing a piecewise nonlinear (PWNL) approximation. For these cases, it is possible to derive a bound on the approximation error, expressed as the difference between the exact and approximated optimal cost functions, as well as establish properties related to stability and constraint satisfaction. Finally, a "fast" MPC implementation, based on the off-line nonlinear function approximation by means of the Set Membership approach [14], is presented in [34, 35]. This method allows the derivation of approximate MPC laws with guaranteed stability, constraint satisfaction and state regulation within an arbitrarily small neighbourhood of the origin. However, all these methods become inefficient when the number of system states is large, complex/time-varying constraints must be satisfied, or time-varying references have to be tracked. In this context, the innovative aspect of the proposed strategy, compared to the state-of-the-art and in particular with respect to [34], is that in previous studies the NMPC law was approximated offline and then applied online in an "open-loop" fashion. Instead, in our approach, the SM is used to narrow down the search domain in which the optimal solution can be found during online optimization. Consequently, an additional layer of online optimization is conducted, enhancing the robustness of the system against any potential changes in scenarios (compared to those used for the offline identification).

The final category involves simplifying the Optimal Control Problem (OCP) by employing a low-dimensional input parametrization, which means solving the OCP within a reduced search space. These methods are specifically designed to guide the optimization process effectively, ensuring that computational efforts are focused on the most promising regions of the search domain. By reducing the dimensionality of the search space, these techniques can significantly decrease the computational complexity of NMPC, making it more viable for real-time applications. Various methods for dimensionality reduction have emerged in the context of NMPC, including Laguerre functions, Move Blocking, and Active Subspaces. These methods will be detailed in Chapter 3. Compared to these methods, the strategy developed in the thesis is innovative both in the use of

the gradient for identifying the direction of greatest variability in nonlinear problems, and in the assignment of suitable values to non-relevant decision variables through the use of an approximated NMPC control law. On this last point, in [36], a global sensitivity analysis is used to identify the so-called active and inactive subspaces in the case of NMPC. The active subspace consists of variables that significantly influence the outcome of the optimization problem. These variables are adjusted during each iteration of the optimization process to find the best possible solution. On the other hand, the inactive subspace consists of variables that have a negligible impact on the outcome of the optimization problem. Because these variables do not significantly affect the result, they are not actively optimized during each iteration. Therefore, rather than adjusting these inactive variables at each step, the values of the optimal command computed in the previous optimization step are shifted and used. This approach can lead to problems in scenarios where the optimization problem changes drastically from one sampling time to the next, such as in the presence of obstacles. In our approach, instead, at each optimization step suitable values for these variables are obtained through the use of an approximation model, computed offline, that "emulates" the NMPC control law. This allows the non-relevant decision variables to adapt to possible changes in the scenario.

1.4 Thesis Outline

This thesis contains six chapters, including this introduction and final remarks.

Chapter 2 - Nonlinear Model Predictive Control: A Comprehensive Overview. This chapter offers a comprehensive introduction to Nonlinear Model Predictive Control (NMPC). It begins with the presentation of the NMPC optimization problem considering discrete-time nonlinear models and discusses the numerical methods for its solution, focusing on direct methods like single-shooting and multiple-shooting strategies. The specific NMPC formulation used in the thesis is then introduced. The chapter also provides an overview of Nonlinear Programming (NLP), including the necessary conditions of optimality. Newton-type algorithms, particularly the interior point and sequential quadratic programming algorithms are explained as common methods for solving NLP.

The chapter concludes with an overview of fast NMPC techniques, including the Real-time iteration scheme.

Chapter 3 - Dimensionality Reduction for Fast NMPC. This chapter introduces a new strategy aimed at reducing the computational complexity of Nonlinear Model Predictive Control (NMPC) by decreasing the dimensionality of the optimization problem. It begins with an overview of existing methods for dimensionality reduction in NMPC, followed by a detailed explanation of two foundational strategies of the proposed method: gradient computation of the cost function and the use of a Nonlinear Sparse Identification Method for NMPC control law approximation. The chapter concludes with a presentation of the steps involved in the developed method and a theoretical analysis demonstrating its effectiveness in reducing computational complexity.

Chapter 4 - Side-length Reduction for Fast NMPC. This chapter presents a novel approach that aims to optimally restrict the side-length of the search domain for reducing the computational time of the NMPC. The method uses the Set Membership (SM) approximation, which is thoroughly introduced. Subsequently, the developed method is described in detail, considering its use both with dimensionality reduction and standalone. Finally, an in-depth analysis of computational complexity is conducted to demonstrate how the reduction in side-length results in a faster convergence rate of the optimization algorithm.

Chapter 5 - Simulation results in Autonomous Driving scenarios. This chapter evaluates the two developed methods for trajectory planning and control of autonomous vehicles in real-world scenarios. The performance of these methods is compared with standard NMPCs, demonstrating their effectiveness in reducing computational complexity both theoretically and practically. The evaluation includes three scenarios: lane keeping on urban roads, obstacle avoidance on rural roads, and parallel parking. These scenarios highlight the versatility of the methods in handling various tasks. The performance comparison involves both software-in-the-loop (SIL) and hardware-in-the-loop (HIL) simulations, with the NMPC algorithm implemented and executed using a Matlab Function in SIL and on an Nvidia Jetson Nano board in HIL.

Chapter 6 - Conclusions and future research. In this chapter, the thesis contributions are summarized and possible future research directions are proposed.

1.5 Notation and Definitions

In this thesis, the symbol $\mathbb{R}$ denotes the set of all the real numbers. To simplify the mathematical derivations and expression throughout the work, a **column vector** $x \in \mathbb{R}^{n_x}$ is denoted by $x = (x_1, \ldots, x_{n_x})$, while a **row vector** $x \in \mathbb{R}^{1 \times n_x}$ is $x = [x_1, \ldots, x_{n_x}] = (x_1, \ldots, x_{n_x})^\top$, where $\top$ indicates the transpose. It should be noted carefully that this notation will be used consistently in all the following chapters. Henceforth, the symbol $\doteq$ should be interpreted as "defined by". The ℓ_p norm of a vector $x = (x_1, \ldots, x_{n_x})$ is defined as

$$\|x\|_p \doteq \begin{cases} \left(\sum_{i=1}^{n_x} |x_i|^p\right)^{\frac{1}{p}}, & 1 \leq p < \infty, \\ \max_i |x_i|, & p = \infty. \end{cases}$$

The ℓ_2 matrix norm of a generic matrix $A \in \mathbb{R}^{n_x \times n_x}$ is defined as

$$\|A\|_2 = (\text{maximum eigenvalue of } A^\top A)^{\frac{1}{2}}$$

The ℓ_2 norm both for vectors and matrices is for simplicity of notation defined as $\|\cdot\|$ without the use of the subscript 2. A function is of class $\mathbf{C}^n$ if it is differentiable n times and the n-th derivative is continuous.

We now present some general definitions related to the continuity properties of a function.

Definition 1 (Continuous Function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is continuous in $\mathbb{R}^n$ if for any x_1, $x_2 \in \mathbb{R}^n$ and for all $\epsilon > 0$, there is a value $\delta > 0$ such that $\|x_1 - x_2\| < \delta \Rightarrow \|f(x_1) - f(x_2)\| < \epsilon$.*

Definition 2 (Lipschitz Continuity)**.** *A continuous function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous in $\mathbb{R}^n$ if for any x_1, $x_2 \in \mathbb{R}^n$ there is a constant $L \geq 0$, called **Lipschitz constant**, such that $\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\|$.*

Chapter 2

Nonlinear Model Predictive Control: A Comprehensive Overview

Advances in optimal control theory have led to the development of Model Predictive Control (MPC), a versatile and powerful control strategy that has found widespread application in various industrial and technological domains (see, e.g., [1]). At its core, MPC involves the resolution of a finite-dimensional optimization problem, with the current state as the initial value, in order to generate the optimal control command at each instant. This optimization process is guided by a dynamic model of the system, enabling MPC to predict system behavior over a receding time horizon. As a result, this enhances robustness against disturbances and allows for real-time adaptation to changing conditions. Key advantages of MPC include its flexibility in formulating objectives and process models for multivariable systems, its ability to directly handle equality and inequality constraints on state, input, and output variables, and the potential to quickly respond to unexpected disturbances. In-depth overviews of the theory behind model predictive control are available, e.g., in [37, 38, 2].

Emerging in the 1970s, Linear Model Predictive Control (LMPC) has become a standard in the process industry, particularly when linear models can accurately depict the dynamics of the systems under control (see, e.g., [39, 40, 3]).

However, given the inherent nonlinearities present in many real-world systems, there has been a growing interest towards Nonlinear Model Predictive Control (NMPC) since the 1980s (see, e.g., [41–43]). This remarkable surge in popularity is due to the ability of NMPC to handle nonlinear dynamics and constraints, and optimize nonconvex performance indexes. More information on the theoretical aspects and industrial applications of NMPC can be found, e.g., in [7, 4, 44, 45]. Both LMPC and NMPC rely on suitable Optimal Control Problems (OCPs) to determine the optimal control actions. The solution to these OCPs must be obtained within a sufficiently short time interval, depending on the application of interest. This requires the availability of efficient optimization algorithms, which can satisfy the real-time requirements of closed-loop control applications. For linear systems, the MPC problem can be formulated as a quadratic or linear program. These formulations belong to a well-studied class of optimization problems, known as convex problems, for which a wide range of fast and efficient algorithms have been developed [6]. On the other hand, the NMPC deals in general with nonconvex optimization, which makes it difficult to guarantee global optimality (or at least a satisfactory level of suboptimality) within real-time constraints. This poses significant limitations for widespread industrial adoption [7], leading to the use of the NMPC mainly in domains characterized by sufficiently long sampling times, such as chemical processing and oil refineries [46, 47]. Recent advancements in numerical methods have enabled the development of efficient classes of nonlinear optimization algorithms, such as Sequential Quadratic Programming (SQP) and Interior-Point (IP) methods. This, combined with the design of fast optimization techniques based on these algorithms, has allowed the use of NMPC even in fast real-time control applications (see, e.g., [48–52]).

In light of these advancements, the present chapter is dedicated to provide a comprehensive understanding of Nonlinear Model Predictive Control (NMPC) by delving into its theoretical aspects, ensuring that readers acquire a deep comprehension of this advanced optimal control technique.

Outline. The chapter is organized as follows. Section 2.1 introduces the general formulation of the NMPC Optimization Problem. In Section 2.2, the numerical methods used for solving Optimal Control Problems are discussed, with a focus on Direct Optimal Control techniques. Section 2.3 presents the

specific formulation of the NMPC used throughout this thesis. In Section 2.4, all the theoretical concepts underlying Nonlinear Programming are considered. Section 2.5 describes the most popular methods for solving NMPC, namely Newton-Type Optimization Methods, with particular emphasis on Sequential Quadratic Programming. Finally, Section 2.6 considers an overview of some of the most popular fast techniques for NMPC, including the Real-Time iteration scheme.

2.1 NMPC Optimization Problem

Throughout this thesis, the Multiple-Input-Multiple-Output (MIMO) time-invariant nonlinear system described by the following discrete-time state equation is considered:

$$x_{t+1} = f(x_t, u_t) \tag{2.1}$$

where $t \in \mathbb{N}_0$ is the discrete time index, $x_t \in \mathbb{R}^{n_x}$ is the system state, $u_t \in \mathbb{R}^{n_u}$ is the command input, and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the dynamical model, which maps the current state x_t and the control input u_t to the subsequent state x_{t+1}. The state is assumed to be measured in real-time. In the case where direct measurement is not available, it can be possibly estimated by means of an observer or eventually replaced with an input-output model.

Suppose that the system state x_t is required to track a desired reference signal r_t. The state and input variables may be subject to constraints and it may be of interest to obtain a suitable trade-off between performance and command effort. To solve this problem, an NMPC approach is proposed. The NMPC has emerged as a versatile and widely adopted real-time control strategy due to its ability to effectively manage nonlinear dynamics and constraints within the optimization framework. This approach is based on the solution of a nonlinear Optimal Control Problem (OCP) at each sampling instant.

Based on the discrete-time system (2.1), the following general optimal control problem is considered:

$$\min_{\boldsymbol{x},\mathbf{u}} \quad \sum_{k=0}^{T-1} L_k(x_k,u_k)+M(x_T) \tag{2.2a}$$

$$\text{subject to:} \qquad \hat{x}_0 - x_0 = 0 \tag{2.2b}$$

$$x_{k+1} - f(x_k,u_k) = 0, \quad k=0,\ldots,T-1 \tag{2.2c}$$

$$h_k(x_k,u_k) \leq 0, \quad k=0,\ldots,T-1 \tag{2.2d}$$

$$r(x_T) \leq 0, \tag{2.2e}$$

where the state vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_{T-1}, x_T) \in \mathbb{R}^{Tn_x}$ and the control vector $\mathbf{u} = (u_0, u_1, \ldots, u_{T-1}) \in \mathbb{R}^{Tn_u}$ are treated as unknown decision variables. As already specified in Section 1.5, column vectors are denoted as $(\ldots)$, while row vectors are represented as $[\ldots] = (\ldots)^\top$. The value T, called *prediction horizon*, denotes the upper limit of the summation and is assumed to be finite and positive. Eq. (2.2a) defines the so-called *cost function*. It consists of L_k and M functions, which typically are referred to as Lagrange and Mayer cost terms, respectively. The optimization problem depends on the parameter value $\hat{x}_0$ through the initial value condition of Eq. (2.2b). Furthermore, Eq. (2.2c) represents the dynamic constraints, while Eqs. (2.2d) and (2.2e) describe the path and terminal constraints, respectively. Given the initial state measurement $\hat{x}_0$, an optimal solution trajectory $(\boldsymbol{x}, \mathbf{u})$ within the prediction horizon T can be obtained by solving (2.2).

Before delving further into the discussion, it is important to note that the use of a state observer for the nonlinear system (2.1) introduces additional assumptions and theoretical considerations that should be carefully addressed to ensure closed-loop stability and convergence. Unlike linear systems, where the convergence of the observer error can be guaranteed if the system is observable and certain conditions, such as the placement of the eigenvalues of the error dynamics, are satisfied, ensuring convergence in nonlinear systems is more complex. This complexity arises from factors such as state-dependent dynamics, nonlinear estimation error dynamics or bifurcations. One of the most widely used tools to analyze and design observers for nonlinear systems is Lyapunov stability theory. In this context, a Lyapunov function is constructed to measure how the estimation error evolves over time. If a Lyapunov function

can be found that decreases over time, then the estimation error is guaranteed to converge to zero, indicating that the observer is stable. The challenge with Lyapunov-based observer design lies in finding an appropriate Lyapunov function for the system. In practice, this is often difficult, especially for highly nonlinear or time-varying systems. Furthermore, it is important to acknowledge the possible presence of non-minimum-phase behavior due to unstable zero dynamics in the system under consideration. A system exhibits non-mimimum-phase (or has unstable zeros) if there exists a (nonlinear) state feedback that can hold the system output identically zero while the internal dynamics become unstable. This phenomenon can introduce several challenges:

- Reduced stability margins: Unstable zero dynamics create difficulties in ensuring that the system can remain stable under feedback control.
- Limitations on achievable performance: Non-minimum phase systems have limitations on the performance criteria that can be met, such as bandwidth, settling time, and transient response.
- Difficulty in achieving accurate tracking: For nonlinear systems with unstable zero dynamics, it is harder to design controllers that can track desired outputs accurately without inducing instability. This can result in scenarios where certain state variables diverge, or the tracking error fails to remain sufficiently small.

All these issues pose a significant challenge for control design and system stability. However, while the problems related to state observers and non-minimum-phase behavior are fundamental in control system design, they are not the focus of this thesis, which is primarily concerned with the computational aspects of NMPC. Consequently, they are not specifically addressed in the current work. For the same reasons, we do not consider modeling errors, disturbances, or issues related to stability and recursive feasibility. All these topics could be the subject of future investigations.

2.2 Numerical Methods for Optimal Control Problems

Numerical methods for solving OCPs are traditionally classified into three main categories: i) Dynamic Programming [53], ii) Indirect Methods [54], and iii) Direct Methods [55]. Before going into the details of each method, let's take a brief historical journey to trace their evolution. Among the three approaches, indirect methods are the earliest ones. Their origins can be dated back to the 17th century, with the advent of calculus of variations, pioneered by Johann Bernoulli to address a novel challenge: finding the time-optimal path for a particle in a gravitational field between any two points. Since then, indirect methods have seen continuous development, both conceptually and algorithmically. Dynamic programming, on the other hand, emerged in the 1930s, thanks to the work of Carathéodory [56], and gained recognition in the 1950s through contributions of Bellman [57]. Both indirect methods and dynamic programming rely on complex mathematical frameworks and analytical approaches. In contrast, direct methods, i.e., the most recent class of methods, emerged alongside the growth of computational power and improvements of numerical methods. One of the pioneering approaches was presented by Bock and Plitt [21]. Their relative simplicity and their use of well-established optimization techniques have made them increasingly popular in recent years.

Dynamic Programming

Dynamic Programming (DP), a mathematical optimization method introduced by Richard Bellman in the 1950s [57], is widely used for multi-stage decision-making problems, where sequential decisions contribute to the minimization of an overall cost [58]. Within the domain of optimal control, dynamic programming stands out as a highly effective method for managing discrete-time systems by leveraging the Bellman's principle of optimality, which states that "*An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision*". Summarizing, it asserts that each subtrajectory of an optimal trajectory is optimal as well, enabling

dynamic programming to efficiently tackle complex problems by decomposing them into smaller, more manageable steps.

Delving deeper into the workings of dynamic programming, it aims to solve the Hamilton-Jacobi-Bellman (HJB) equation [59], which provides the necessary optimality conditions for continuous-time OCPs. The solution is associated with the optimal cost of a value function, or cost-to-go function, that satifies the HJB partial differential equation. However, it requires the enumeration of the continuous state and control spaces for each sampled state. More in detail, the solution is obtained iteratively through a two-step process:

1. Backward Recursion: Starting from the end of the sampled horizon, this step computes all relevant functions, gradually decreasing the time step until reaching the beginning of the horizon.
2. Forward Recursion: Based on the computed functions, this step selects the optimal control action for each state.

This iterative approach enables the reconstruction of the optimal trajectory through forward simulation, starting from the current state estimate, leveraging Bellman's principle of optimality.

Despite its ability to solve optimal control problems globally, perform efficient forward simulations, and derive a closed-form feedback control law, dynamic programming faces significant challenges when applied to high-dimensional systems. The exponential growth of computational complexity and memory requirements associated with state-space tabulation, known as the "*curse of dimensionality*" by Richard Bellman, limits its practical applicability to complex systems.

Indirect Methods

Indirect methods, also known as *first-optimize-then-discretize*, use the concepts of calculus of variations [60] and Pontryagin's maximum principle [61] (also referred to as the minimum principle) to determine the first-order necessary optimality conditions of the original optimal control problem. These conditions result in the formulation of a multi-point boundary value problem (MPBVP), which represents a set of differential-algebraic equations (DAEs)

that are iteratively solved using a Newton method. The optimal trajectory can be reconstructed through forward simulation once the initial value for the adjoint equations is determined. While indirect methods provide high accuracy solutions [62], their practical implementation is challenging due to their computationally demanding and the initial guess sensitivity. Several solutions have been proposed to reduce the computational complexity of these methods. Among them, a novel method employing Pontryagin's principle for Nonlinear Model Predictive Control in spacecraft maneuvering was introduced in [63]. This approach guarantees an explicit control law that can handle nonlinearities without making assumptions about the structure of the input signal. By leveraging the Pontryagin Minimum (or Maximum) Principle, the optimization problem is transformed into a boundary value problem, allowing the explicit computation of the control law while avoiding increases in computational complexity. This approach is specifically designed for proximity operations in rendezvous scenarios using the Clohessy-Wiltshire equations.

Direct Methods

Direct methods, also known as *first-discretize-then-optimize*, are among the most widely used and effective approaches for solving nonlinear optimal control problems. These methods directly convert the original optimization problem into a finite-dimensional nonlinear programming (NLP) problem [64], enabling the formulation of the necessary optimality conditions which can be numerically solved using an NLP solver. They typically use piecewise constant parametrization for control trajectory and adapt the state trajectory based on the employed discretization method. Direct methods have several advantages, including their flexibility in handling different types of dynamical systems and their ability to directly incorporate inequality constraints. However, they also have some drawbacks, comprising the fact that they can only find local optimal solutions, and that they only produce open-loop control policies [62]. There are two main types of discretization approaches used in direct methods: *sequential* and *simultaneous*. In the sequential approaches, the state variables are treated as an implicit function of the control trajectories, where the discrete-time first-order difference equations are addressed as an initial value problem using one of the dedicated integration methods such as Runge-Kutta or Euler algorithms. These

methods can be computationally efficient, but they can become unstable for highly nonlinear systems. On the other hand, in the simultaneous approaches, the state trajectories are also parametrized, and all the parametrized variables are treated as optimization variables in the NLP problem. In other words, these methods split the time horizon into smaller equidistant intervals where each interval represents an independent sequential problem with the addition of continuity constraints. In this case, the discrete-time first-order difference equations are represented as equality constraints. While generally more computationally demanding, simultaneous approaches can offer better stability for highly nonlinear systems. A detailed description of these two methods will be addressed in Section 2.2.1.

2.2.1 Direct Optimal Control

The direct optimal control method are the class of numerical methods underlying the algorithms used in the NMPC presented in this thesis. As defined before, there are two main approaches to direct numerical optimal control:

- The sequential approach, which addresses the optimal control problem (2.2) by numerically solving the constraint (2.2c) during the evaluation of the cost and constraint functions. This approach eliminates the need to include the intermediate states $\boldsymbol{x}$ within the problem formulation, by directly substituting them into the cost and constraint functions. The control command sequence $\mathbf{u}$ is parametrized and treated as an unknown variable. This iterative process of performing the simulation task, by solving numerically the constraint (2.2c) separately from the optimization problem, is often referred to as *Direct Single Shooting* method [65–67].

- The simultaneous approach, which solves the optimal control problem by parametrizing both the state trajectory and the control. In order to guarantee the continuity of the resulting parametrized trajectory along the entire prediction horizon T, additional continuity conditions are introduced as constraints. This approach explicitly represents the intermediate states $\boldsymbol{x}$ as well as the control trajectory parameters $\mathbf{u}$ as unknown variables. As a results, the OCP is addressed directly by a Newton-type optimization algorithm (described in detail in Section

2.5), effectively combining optimization and simulation into a single process. However, this simultaneous treatment results in a larger number of constraints and variables compared to the sequential approach. Two prominent methods within the simultaneous approach are *Direct Multiple Shooting* [21, 22] and *Collocation method* [68, 69].

The following paragraphs will describe two of the three methods mentioned above: i) Direct Single Shooting and ii) Direct Multiple Shooting. Note that, as will be defined in Section 2.3, the NMPC used in this thesis will be based on the concepts related to Direct Single Shooting.

Direct Single Shooting

The Direct Single Shooting method restricts the optimization to the control sequence $\mathbf{u} = (u_0, u_1, \ldots, u_{T-1})$. As a result, given a guessed piecewise-constant control trajectory, the corresponding points on the state trajectory $\boldsymbol{x} = (x_0, x_1, \ldots, x_T)$ can be determined through the following steps: i) set the initial state x_0; ii) for $k = 0, \ldots, T-1$, define $x_{k+1} = f(x_k, u_k)$. Under this definition, the equality constraints (2.2b)-(2.2c) unequivocally determine the variable $\boldsymbol{x}$ if the vector $\mathbf{u}$ is fixed. Consequently, by a system simulation, they can be inverted to yield the implicit function $\bar{\boldsymbol{x}}(\boldsymbol{u})$, which satisfies (2.2b)-(2.2c) for all $\mathbf{u}$. Note that the symbol $\bar{\boldsymbol{x}}$ is used to indicate feasible points, i.e. points that meet all the constraints. For a formal description of *feasible point*, please refer to Definition 4 on page 27. By implementing this method, the optimization problem can be reformulated as

$$\min_{\mathbf{u}} \quad \sum_{k=0}^{T-1} L_k(\bar{x}_k(u), u_k) + M(\bar{x}_T(u)) \tag{2.3a}$$

$$\text{subject to:} \quad h_k(\bar{x}_k(u), u_k) \leq 0, \quad k = 0, \ldots, T-1 \tag{2.3b}$$

$$r(\bar{x}_T(u)) \leq 0. \tag{2.3c}$$

This reduced optimization problem involves $n_u T$ decision variables, which is a significantly smaller set compared to the original problem (2.2). This makes it an attractive choice for optimization procedures. As a result, direct single shooting is frequently used in real-world applications due to its straightforward implementation, intuitive concept, and the limited number of unknowns in the

resulting NLP. The use of single shooting for the NMPC formulation will be detailed in Section 2.3.

Direct Multiple Shooting

The Direct Multiple Shooting addresses the full nonlinear OCP (2.2). This means that, unlike the single shooting method, in this case the state variables $\boldsymbol{x}$ also becomes decision variables. Additionally, the dynamic equations (2.2c) are imposed as equality constraints (called *continuity conditions*), which means that the optimizer must simultaneously satisfy these constraints and minimize the objective function. This can be a challenge, since the $\boldsymbol{x}$ do not usually perfectly meet the requirements. In terms of computational complexity, multiple shooting introduces a larger number of variables and constraints compared to single shooting. With the inclusion of state variables, the optimization problem now involves $(n_x + n_u)T$ decision variables and an additional $n_x T$ equality constraints. This increased complexity can potentially slow down the optimization process. In MATLAB, one of the most powerful tools for implementing the Direct Multiple Shooting method is CasADi [70].

2.3 Direct Single Shooting NMPC Formulation

Consider the discrete-time system (2.1), with the following assumption.

Assumption 1. *Let $f(\cdot)$ be twice continuously differentiable in its arguments.*

The NMPC formulation developed in this thesis is as follows. At each time step t, the system state x_t is measured. During the time interval $[t, t+1)$, an optimal command is computed based on x_t. This command is then applied to the system at time $t+1$. The optimal command is obtained by means of two key operations: prediction and optimization. In particular, the system state is predicted T steps in the future, where the integer $T \geq 1$ is called the *prediction horizon*. For any $k \in [1, T]$, the k-step ahead prediction is obtained by iterating k times the model equation (2.1). The predicted state $\hat{x}_{k+1}$ is thus a function of the "initial" state x_t, the input u_t, and the future input sequence

$\mathfrak{u}_{t,k} \doteq (\mathfrak{u}_{t+1}, \ldots, \mathfrak{u}_{t+k})$:

$$\hat{x}_{k+1} \equiv \hat{x}_{k+1}\left(x_t, u_t, \boldsymbol{\mathfrak{u}_{t,k}}\right).$$

Note that the symbol u_t represents the command input that is "physically" applied to the system (2.1), while $\mathfrak{u}_{t,k}$ in Gotich style denotes the sequences of command inputs used in the prediction interval $[1, T]$. The symbol t indicates the time index characterizing the actual time evolution of the plant (2.1), while k is a "virtual" time index, used in the prediction interval.

The basic idea of NMPC is to compute, at each time step t, a command sequence $\mathfrak{u}^*_{t,k}$ such that the predicted state $\hat{x}_k(x_t, u_t, \mathfrak{u}^*_{t,k})$ has an "optimal behavior". The concept of "optimal behavior" is represented by the objective function

$$J_o(\mathfrak{u}) \doteq \sum_{k=1}^{T} \mathfrak{u}_k^\top R \mathfrak{u}_k + \sum_{k=2}^{T} \tilde{x}_k^\top Q \tilde{x}_k + \tilde{x}_{T+1}^\top P \tilde{x}_{T+1} \tag{2.4}$$

where $\tilde{x}_k \doteq \mathfrak{r}_k - \hat{x}_k$ is the predicted tracking error, $\mathfrak{r}_k \doteq r_{t+k} \in \mathbb{R}^{n_x}$ is the reference to track, $\mathfrak{u} \doteq (\mathfrak{u}_1, \ldots, \mathfrak{u}_T) \in \mathbb{R}^{Tn_u}$ is the applied input sequence, and R, Q, P are diagonal positive-definite weight matrices. The optimal input sequence is chosen as one minimizing the objective function $J_o(\cdot)$, ensuring that the input and state sequences are consistent with the model equation and satisfy possible constraints. The input saturations are described by $\mathfrak{u} \in U_c \subset \mathbb{R}^{Tn_u}$, where U_c is an hyperrectangle with suitable side-lengths. Other general input constraints, as well as state constraints, are treated using penalty functions. In particular, for each constraint, a penalty function $\rho_j \in \mathbf{C}^\beta$, $\beta \geq 1$ is introduced, such that $\rho_j(\hat{x}_k, \mathfrak{u}_k) \cong 0$ when the constraint is satisfied, $\rho_j(\hat{x}_k, \mathfrak{u}_k) \gg J_o(\mathfrak{u})$ when it is not. Then, the penalty functions are added to J_o, giving the "augmented" objective function

$$J(\mathfrak{u}) \doteq J_o(\mathfrak{u}) + \textstyle\sum_{k=1}^{T} \sum_j \rho_j(\hat{x}_k, \mathfrak{u}_k).$$

The optimal input sequence is computed solving, at each time $t \in \mathbb{N}_0$, the following optimization problem:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} J(\mathbf{u}) \tag{2.5a}$$

subject to:

$$\hat{x}_1 = f(x_t, u_t) \tag{2.5b}$$

$$\hat{x}_{k+1} = f(\hat{x}_k, \mathbf{u}_k),\ k = 1:T \tag{2.5c}$$

$$\mathbf{u} \in U_c. \tag{2.5d}$$

The motivations for using penalty functions are that they are relatively simple to implement and avoid the singularity issues proper of barrier functions. Moreover, they are not affected by unfeasibility problems, since the optimization problem (2.5) admits always a solution. This can be useful in real situations where, due to unfeasibility, an approach based on hard-constraints may not find a solution. Instead, an approach based on penalty function provides in any case a solution, which may not satisfy the hard-constraints but can be useful to "limit the damage". It must also be observed that penalty functions do not theoretically guarantee that hard-constraints are satisfied. This issue can be overcome by a proper choice of the penalty functions.

It is worth noting that the OCP (2.5) is similar to the Direct Single Shooting optimization problem (2.3). In fact, the optimization is performed only on the control sequence $\mathbf{u}$. This reduces the number of variables involved in the optimization, making the NMPC suitable for the methodologies developed in Chapters 3 and 4. It is important to note that these methodologies can also be applied to simultaneous approaches, with an inevitable addition of complexity.

Input parametrization

The optimization problem (2.5) is in general non-convex. Moreover, the command sequence $\mathbf{u} \doteq (\mathbf{u}_1, \ldots, \mathbf{u}_T)$ may contain a relatively large number of decision variables. A common technique to reduce the number of variables is to choose a suitable integer $T_c \leq T$, called the *control horizon*, and keep the command $\mathbf{u}_k$ constant for $k \geq T_c$. Here, the more general Move Blocking technique, described

in detail in Section 3.1.2, is used. The command sequence $\mathbf{u}$ is parametrized as

$$\mathbf{u} = \Gamma c \tag{2.6}$$

where $\Gamma \in \mathbb{R}^{Tn_u \times n_c}$ is a full-rank matrix of ones and zeros only, with exactly one non-zero element for each row, and $c \in \mathbb{R}^{n_c}$ is a new command input sequence with reduced dimension $n_c < Tn_u$. This sequence is subject to input saturations, denoted by $c \in \mathcal{C}_c \in \mathbb{R}^{n_c} \subset \mathbb{R}^{Tn_u}$, where $\mathcal{C}_c$ is a subset of U_c defined as

$$\mathcal{C}_c = \prod_j \mathcal{C}_{c_j} \tag{2.7}$$

where $\prod_j$ denotes the Cartesian product, and $\mathcal{C}_{c_j} = [\underline{c}_j, \overline{c}_j]$ for $j = 1, \ldots, n_c$. Note that a sample $\mathfrak{u}_k$ of $\mathbf{u}$ can be obtained using a selection matrix S_k^u such that $\mathfrak{u}_k = S_k^u \mathbf{u} = S_k^u \Gamma c$.

To illustrate how the proposed input parametrization works, an example is reported below.

Example. Suppose that $\mathbf{u} \doteq (\mathfrak{u}_1, \mathfrak{u}_2, \mathfrak{u}_3, \mathfrak{u}_4, \mathfrak{u}_5) \in \mathbb{R}^{5\times 1}$, $\mathfrak{u}_k \in \mathbb{R}, \forall k$. The parametrization

$$\mathbf{u} = \Gamma c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_2 \\ c_3 \\ c_3 \end{bmatrix}$$

allows a reduction of the number of decision variables from 5 to 3. The input is parametrized as a piecewise constant sequence, with three possible values in the prediction interval. The vector S_k^u that selects a sample $\mathfrak{u}_k$ is the k-th row of the identity matrix I_5. □

Under the above input parametrization, the optimization problem becomes:

$$c^* = \arg\min_c J(\Gamma c) \tag{2.8a}$$

subject to:

$$\hat{x}_1 = f(x_t, u_t) \tag{2.8b}$$

$$\hat{x}_{k+1} = f(\hat{x}_k, S_k^u \Gamma c),\ k = 1:T \tag{2.8c}$$

$$c \in \mathcal{C}_c. \tag{2.8d}$$

Reciding Horizon strategy

The NMPC feedback command is computed by solving the optimization problem (2.8) at each time step t, according to a so-called *Receding Horizon Strategy* (RHS):

1. At time t:
   (a) measure x_t;
   (b) compute c^* by solving (2.8);
2. At time $t+1$:
   (a) apply to the system only the first command sample: $u_{t+1} = S_1^u \Gamma c^*$.
3. Repeat steps 1 and 2 for $t+1, t+2, \ldots$ □

The receding horizon strategy is fundamental to have a feedback control action, allowing the NMPC algorithm to stabilize unstable systems, attenuate external disturbances and properly react if sudden changes occur in the scenario where the system of interest is operating. Note that the present NMPC/RHS formulation considers a realistic scenario, where the state is measured at time t, the time interval $[t, t+1)$ is used to compute the optimal command, which is then applied at time $t+1$.

At its core, NMPC is a control strategy that uses a sequence of nonlinear optimization problems to dynamically generate optimal control actions based on the current state of the system and a predictive model of its future behavior.

This approach can be seen as a specially structured case of a general Nonlinear Programming (NLP) problem. The NLP provides the mathematical framework and computational algorithms required for solving these optimization problems efficiently. This allows the NMPC to determine the optimal control inputs that guide the system towards its desired states while satisfying the constraints. Consequently, grasping the theoretical foundations of nonlinear programming is essential for fully understanding the inner workings of NMPC.

2.4 Theoretical Concepts of Nonlinear Programming

A generic NLP problem can be written as follows:

$$\begin{aligned} \min_{y} \quad & \phi(y) \\ \text{subject to:} \quad & g_i(y) = 0, \quad i \in \mathcal{E} \\ & h_j(y) \geq 0, \quad j \in \mathcal{I}, \end{aligned} \tag{2.9}$$

where $y \in \mathbb{R}^{n_y}$ is the vector of decision variables, $\phi : \mathbb{R}^{n_y} \to \mathbb{R}$ is the objective function, $g : \mathbb{R}^{n_y} \to \mathbb{R}^m$ and $h : \mathbb{R}^{n_y} \to \mathbb{R}^q$ represent the equality and inequality constraints, respectively. Please note that $\mathcal{E} = \{1, \ldots, n_e\}$ and $\mathcal{I} = \{1, \ldots, n_i\}$ are two finite sets of indices that indicate the number of equality and inequality constraints.

Assumption 2. *ϕ, g and y are twice continuously differentiable functions.*

In the context of nonlinear programming, it becomes important to determine whether a feasible point $\bar{y} \in \Omega$ satisfies the necessary optimality conditions. If $\bar{y}$ meets these criteria, it can be considered as a potential local minimizer y^* for NLP (2.9). Before introducing these conditions, we need to define some basic concepts.

Definition 3 (Feasible set)**.** *The feasible set Ω of (2.9) is the set of points y that satisfy all the constraints:*

$$\Omega := \{y \in \mathbb{R}^{n_y} \mid g_i(y) = 0,\ i \in \mathcal{E};\ h_j(y) \geq 0,\ j \in \mathcal{I}\}.$$

Definition 4 (Feasible point)**.** *The point $\bar{y} \in \mathbb{R}^{n_y}$ is a feasible point if it satisfies the constraints:*

$$g(\bar{y}) = 0, \quad h(\bar{y}) \geq 0.$$

Definition 5 (Local minimizer)**.** *The point y^* is a local minimizer to (2.9) if it is a feasible point, i.e., $y^* \in \Omega$, and there exists a neighborhood M of y^* so that*

$$\forall y \in M \cap \Omega : \phi(y) \geq \phi(y^*)$$

where $\phi(y^*)$ is a local minimum.

To determine whether the feasible point $\bar{y}$ is a local minimizer y^*, it is necessary to describe the set M, which depends on the active inequality constraints.

Definition 6 (Active constraint)**.** *An inequality constraint h_j is called active at a feasible point $\bar{y} \in \Omega$, if*

$$h_j(\bar{y}) = 0.$$

Definition 7 (Active set)**.** *The active set $\mathcal{A}(\bar{y})$ at any feasible point $\bar{y}$ is defined as*

$$\mathcal{A}(\bar{y}) = \{j \in \mathcal{I} \mid h_j(\bar{y}) = 0\}.$$

This means that the active set is the set of the indices j associated with the inequality constraints that are active.

Note that here the equality constraints are not taken into account, because, by definition, they always result in active constraints. Another key concept necessary for defining optimality conditions is the so-called linear independence constraint qualification (LICQ).

Definition 8 (Linear Independence Constraint Qualification)**.** *The LICQ holds at the feasible point $\bar{y}$ if the set of active constraint gradients $\nabla g_i(\bar{y})$ for $i \in \mathcal{E}$ and $\nabla h_j(\bar{y})$ for $j \in \mathcal{A}(\bar{y})$ is linearly independent. In other words, the LICQ holds if the Jacobian matrix*

$$\nabla G(\bar{y}) = \begin{pmatrix} \nabla g_i(\bar{y}),\, i \in \mathcal{E} \\ \nabla h_j(\bar{y}),\, j \in \mathcal{A}(\bar{y}) \end{pmatrix}$$

is full-row rank.

2.4.1 First-order necessary conditions of optimality

To define the first-order necessary conditions, it is essential to introduce the Lagrangian function.

Definition 9 (Lagrangian function)**.** *The Lagrangian function for the NLP problem (2.9) is defined as:*

$$\mathcal{L}(y, \lambda, \mu) \coloneqq \phi(y) - \lambda^T g(y) - \mu^T h(y) \tag{2.10}$$

where $\lambda \in \mathbb{R}^{n_e}$ and $\mu \in \mathbb{R}^{n_i}$ are the Lagrange multipliers, or dual-variables, of g and h, respectively.

Now we can proceed to state the first-order conditions. These conditions are related to the characteristics of the gradients (first-derivative vectors) of both the objective and constraint functions.

Theorem 1 (First-order necessary conditions)**.** *Consider y^* as a local minimizer of NLP (2.9). Additionally, assume that the functions f, g and h are continuously differentiable, and the LICQ is satisfied at y^*. Then, there exists Lagrange multipliers λ^* and μ^* such that the following conditions hold:*

$$\nabla_y \phi(y^*) + \nabla_y g(y^*)\lambda^* + \nabla_y h(y^*)\mu^* = 0 \tag{2.11a}$$
$$g(y^*) = 0 \tag{2.11b}$$
$$h(y^*) \geq 0 \tag{2.11c}$$
$$\mu^* \geq 0 \tag{2.11d}$$
$$\mu_j^* h_j(y^*) = 0, \quad j \in \mathcal{I}. \tag{2.11e}$$

Proof. See Section 12.3 in [71]. □

The conditions (2.11) are commonly known as *Karush-Kuhn-Tucker (KKT) conditions*. For the reader's curiosity, it is worth noting that the KKT conditions were first introduced in a publication by Kuhn and Tucker in 1951 [72]. These conditions were originally termed as KT conditions until it was discovered that William Karush had already stated them in his unpublished master's thesis in 1939 [73]. Since then, the conditions have been referred to as KKT conditions to acknowledge Karush's contribution.

Resuming the discussion of (2.11) and delving into further detail, each of the Karush-Kuhn-Tucker (KKT) conditions has a well-defined meaning: i) Eq. (2.11a) denotes the *stationary conditions*, which indicates that, given a pair of Lagrange multipliers λ^* and μ^*, the point y^* minimizes the Lagrangian function; ii) Eq. (2.11b) and Eq. (2.11c) represent the *primal feasibility*, which point that y^* must satisfied all the constraints specified in the NLP (2.9); iii) Eq. (2.11d) describes the *dual feasibility*, which states that the Lagrange multiplier μ^* associated with the inequality constraints must be non-negative; iv) Eq. (2.11e) shows the *complementary slackness*, which indicates that for each inequality constraint in the optimization problem, there exists an associated Lagrange multiplier μ_j^* such that the product of the multiplier and the value of the constraint equals zero. This last condition implies that at the optimal solution, either the constraint is active (i.e., $h_j(y^*) = 0$) and the associated multiplier is non-zero, or the constraint is inactive (i.e., $h_j(y^*) < 0$) and the associated multiplier is zero.

Definition 10 (KKT point)**.** *A point (y^*, λ^*, μ^*) that satisfies the conditions in (2.11) and the LICQ is called a* ***KKT point****.*

The combination of the last three KKT conditions, i.e., (2.11c)-(2.11e), yields the concept of *complementarity conditions*. A special case of complementarity is known as *strict complementarity*. Its formal definition is as follows.

Definition 11 (Strict complementarity)**.** *Consider a KKT point (y^*, λ^*, μ^*) of the NLP in (2.9). Strict complementarity is satisfied for this point if all active inequality constraints are strictly active. In other words, if $\mu_j^* > 0$ for all $j \in \mathcal{A}(y^*)$.*

The concept of strict complementarity is fundamental for guaranteeing some desirable properties of numerical methods. Indeed, fulfilling this condition often simplifies the identification of the active set $\mathcal{A}(y^*)$ and speeds up the convergence of the algorithms towards the optimal solution y^*.

2.4.2 Second-order conditions of optimality

The KKT conditions in Theorem 1 serve as both necessary and sufficient conditions for optimality only in the context of convex NLPs, where every local

minimum is also a global solution. However, for non-convex constrained NLPs, they assume a weaker role, becoming merely a set of necessary conditions for a point to be a local optimum. This implies that the existence of a point satisfying the KKT conditions does not necessarily guarantee that it is a local optimum. The reason for this limitation lies in the fact that the KKT conditions only consider first-order information. Indeed, they focus exclusively on the gradient of the Lagrangian function, which represents the first-order derivative of the objective function with respect to both the primal variables and dual variables (i.e., those associated with the constraints). This limited perspective overlooks the curvature of the objective function, which is fundamental in determining the existence and nature of local optima. In non-convex optimization problems, the objective function may have multiple local optima, each corresponding to a distinct point where the gradient vanishes. This means that the KKT conditions may identify multiple KKT points, making it difficult to distinguish between true local optima and saddle points. To address this challenge, additional criteria, such as the second-order sufficient conditions, are employed. These conditions analyze the Hessian matrix of the Lagrangian function, providing insights into the curvature of the objective function around the candidate solution and its neighbouring constraints. With the aid of these second-order conditions, it becomes possible to differentiate between local optima and saddle points, ensuring that identified optimal points are indeed true minima.

In order to formulate the second-order sufficient conditions, the concept of the critical cone at a KKT point is introduced.

Definition 12 (Critical cone)**.** *The critical cone, denoted by* $\mathcal{C}(y^*, \lambda^*, \mu^*)$*, represents a set of directions* d *defined at the KKT point* (y^*, λ^*, μ^*) *where:*

$$d \in \mathcal{C}(y, \nu) \Leftrightarrow \begin{cases} \nabla c_i(y^*)^\top d = 0, & \forall i \in \mathcal{E} \\ \nabla h_j(y^*)^\top d = 0, & \forall j \in \mathcal{A}(y^*) \text{ with } \mu_j^* > 0 \\ \nabla h_j(y^*)^\top d \geq 0, & \forall j \in \mathcal{A}(y^*) \text{ with } \mu_j^* = 0. \end{cases} \tag{2.12}$$

In simpler terms, the critical cone is a set of directions where the gradient fails to provide enough information to determine whether the point is a local minimum. In these directions, the objective function could increase, decrease, or remain constant. This means that first-order information is not enough to

guarantee that a point is a local minimum, as the gradient could potentially point in a direction where the objective function increases. In order to overcome this limitation, the following theorem provides conditions that use the second-order derivatives.

Theorem 2 (Second-order sufficient conditions)**.** *Consider a KKT point* (y^*, λ^*, μ^*) *for the NLP (2.9). If the Hessian of the Lagrangian is strictly positive definite within the critical cone, meaning that the following condition holds*

$$d^T \nabla^2_{yy} \mathcal{L}(y^*, \lambda^*, \mu^*) d > 0, \quad \forall d \in \mathcal{C}(y^*, \lambda^*, \mu^*),\ d \neq 0,$$

then the point y^* *is a local minimizer of the NLP (2.9).*

2.5 Newton-Type Optimization

A comprehensive discussion of nonlinear optimization algorithms can be found in the literature (see, e.g., [71, 74]). In this thesis, the focus is on Newton-type optimization algorithms for solving the NLP (2.9).

The origin of the Newton's method, also known as the Newton-Raphson-Simpson method, can be traced back to several mathematicians, including Francois Vieta, Isaac Newton, Joseph Raphson, and Thomas Simpson [75]. In 1600, Vieta developed a perturbation technique for solving polynomial equations which provided one decimal place of accuracy per step. This method was later simplified by Oughtred in 1647. Newton became familiar with Vieta's method in 1664 and improved it by linearizing the successive polynomials [76]. He demonstrated the efficacy of the method by solving the cubic polynomial $f(x) = x^3 - 2x - 5 = 0$. Raphson eliminated the need for explicit polynomial computation, by iterating directly on the original equation while preserving all decimal places of accuracy in the corrections. Thomas Simpson introduced the concept of derivatives or "fluxiones" in his book [77]. He provided a rigorous formulation of the iterative algorithm for both one equation and a system of two equations in two unknowns, making it applicable to a wider range of problems. Simpson's notation closely resembles the one commonly used today.

For a continuously differentiable function $R : \mathbb{R}^{n_y} \to \mathbb{R}$, Newton's method is an iterative root-finding algorithm used to solve the nonlinear equation $R(y) = 0$

by refining an initial guess until it converges towards the solution. The method starts with an initial guess y_0 and generates a sequence of iterates $\{y_i\}_{i=0}^{\infty}$, each satisfying a linearized version of the original system at the previous iterate. Specifically, for a given y_i, an *exact Newton iteration*

$$R(y_i) + \nabla R(y_i)^T \overbrace{(y_{i+1} - y_i)}^{\Delta y} = 0 \tag{2.13}$$

is defined and solved, using standard linear algebra tools, to obtain y_{i+1}. In other words, the method involves starting with an initial estimate, approximating the function by its tangent line, and then calculating the y-intercept of this line. This y-intercept usually provides a more accurate approximation of the root of the original function than the initial guess, and can be iteratively refined. Newton's method typically exhibits quadratic convergence, leading to rapid error reduction at each iteration. However, exact computation and inversion of the Jacobian $\nabla R(y_i)^T$ can be computationally expensive. If these calculations are approximated rather than performed exactly, the convergence rate may degrade but the iterations become less computationally demanding. This results in a broader class of "Newton-type methods" that provide a trade-off between speed of convergence and computational cost.

In the following, an in-depth analysis of the operational principles of Newton-type optimization algorithms for both equality and inequality constrained optimization is presented.

Equality constrained optimization

Consider the NLP (2.9) with only equality constraint. This means that $\mathcal{I} \in \emptyset$ and $\mathcal{L}(y, \lambda) = \phi(y) + \lambda^T g(y)$. Newton-type optimization algorithms employ a direct approach to solve the first-order necessary conditions

$$\nabla_y \mathcal{L}(y, \lambda) = \nabla_y \phi(y) + \nabla_y g(y) \lambda = 0 \tag{2.14a}$$

$$g(y) = 0 \tag{2.14b}$$

derived from Theorem 1, using the Newton-type root finding method described above. In particular, at each iteration i, the primal-dual iterate $(\Delta y, \lambda_i)$ is considered and Eq. (2.13) is applied to the KKT-conditions (2.14), resulting in

the following exact Newton iteration:

$$\begin{pmatrix} \nabla_y^2 \mathcal{L}(y_i, \lambda_i) & \nabla_y g(y_i) \\ \nabla_y g(y_i)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta \lambda \end{pmatrix} = - \overbrace{\begin{pmatrix} \nabla_y \mathcal{L}(y_i, \lambda_i) \\ g(y_i) \end{pmatrix}}^{R(y_i, \lambda_i)}. \tag{2.15}$$

By employing the definitions presented in Eq. (2.14a), the aforementioned system can be expressed as:

$$\begin{pmatrix} \nabla_y^2 \mathcal{L}(y_i, \lambda_i) & \nabla_y g(y_i) \\ \nabla_y g(y_i)^T & 0 \end{pmatrix} \begin{pmatrix} y_{i+1} - y_i \\ \lambda_{i+1} - \lambda_i \end{pmatrix} = - \begin{pmatrix} \nabla_y \phi(y_i) + \nabla_y g(y_i) \lambda_i \\ g(y_i) \end{pmatrix}. \tag{2.16}$$

Note that the contributions associated with the current multiplier λ_i cancel each other, resulting in

$$\begin{pmatrix} \nabla_y^2 \mathcal{L}(y_i, \lambda_i) & \nabla_y g(y_i) \\ \nabla_y g(y_i)^T & 0 \end{pmatrix} \begin{pmatrix} y_{i+1} - y_i \\ \lambda_{i+1} \end{pmatrix} = - \begin{pmatrix} \nabla_y \phi(y_i) \\ g(y_i) \end{pmatrix}. \tag{2.17}$$

and, consequently,

$$\begin{pmatrix} y_{i+1} \\ \lambda_{i+1} \end{pmatrix} = \begin{pmatrix} y_i \\ 0 \end{pmatrix} - \begin{pmatrix} \nabla_y^2 \mathcal{L}(y_i, \lambda_i) & \nabla_y g(y_i) \\ \nabla_y g(y_i)^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla_y \phi(y_i) \\ g(y_i) \end{pmatrix}. \tag{2.18}$$

It is noteworthy that, while the next multiplier λ_{i+1} can be directly computed, the exact Hessian matrix still depends on λ_i. This observation has led to the development of a wider range of Newton-type optimization techniques known as *Quasi-Newton Methods*. These methods employ strategies to approximate the Hessian matrix using only first-order derivative information. In particular, they aim to find a matrix B_i, such that $B_i \approx \nabla_y^2 \mathcal{L}(y_i, \lambda_i) \nabla_y g(y_i)$. This approximation allows for the derivation of the following Newton-type iteration or search direction:

$$\begin{pmatrix} y_{i+1} \\ \lambda_{i+1} \end{pmatrix} = \begin{pmatrix} y_i \\ 0 \end{pmatrix} - \begin{pmatrix} B_i & \nabla_y g(y_i) \\ \nabla_y g(y_i)^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla_y \phi(y_i) \\ g(y_i) \end{pmatrix}. \tag{2.19}$$

The choice of the Hessian matrix approximation significantly affects the local convergence behavior and computational efficiency of Newton-type optimization algorithms. Therefore, an overview of Hessian approximation schemes within

the context of Sequential Quadratic Programming methods will be presented in Section 2.5.2.

Inequality constrained optimization

Newton-type optimization algorithms can be extended to address optimization problems with inequality constraints by incorporating suitable strategies to handle these constraints within the Karush-Kuhn-Tucker (KKT) conditions. Advancements in optimization strategies have made this extension possible, leading to the development of mature optimization algorithms designed to deal with inequality constraints. Among these algorithms, two main classes stand out: Interior-Point (IP) methods [78] and Sequential Quadratic Programming (SQP) methods [19]. These two classes are discussed below with particular emphasis on Sequential Quadratic Programming, which has become a widely adopted approach for solving constrained nonlinear optimization problems. Furthermore, this algorithm will be used for both theoretical and numerical analyses of the developed methods.

2.5.1 Interior-Point Methods

Interior-point or barrier methods are a popular class of algorithms used for effectively solving nonlinear optimization problems, especially in the context of large-scale programming problems [79, 80]. To address the challenge associated with the KKT system, the key idea involves replacing the non-smooth complementary condition in Eq. (2.11e) with a smooth nonlinear approximation (usually represented as a hyperbola):

$$\mu_j^* h_j(y^*) = \tau, \quad j \in \mathcal{I} = \{1, \ldots, n_i\}, \tag{2.20}$$

where $\tau > 0$ is typically referred to as the *barrier parameter*. The resulting KKT system can be interpreted as a root-finding problem, formulated as $R(y, \tau) = 0$, where the primal-dual solution $(y^*(\tau), \lambda^*(\tau), \mu^*(\tau))$ is iteratively obtained by generating a sequence of positive barrier parameters that converge to zero. This problem can also be viewed as solving the KKT conditions of a barrier problem, commonly known as the *primal barrier method* or *log-barrier method*. As the

terminology implies, the inequality constraints are addressed by incorporating a log-barrier term into the objective function

$$\begin{aligned} \min_{y} \quad & \phi(y) - \tau \sum_{j=1}^{n_i} \log(-h_j(y)) \\ \text{subject to:} \quad & g_i(y) = 0, \quad i \in \mathcal{E} = \{1, \ldots, n_e\}. \end{aligned} \tag{2.21}$$

Note that: i) the inequality constraints are increasingly penalized as they approach the boundaries of the feasible set by the parameter τ; ii) the optimal solution lies within the interior of the constraint set, and its proximity to the true solution increases as the parameter τ decreases. A key characteristic of interior-point methods lies in their ability to progressively reduce the parameter τ without compromising the convergence of Newton's method. This feature enables the achievement of a highly accurate solution to the original NLP problem within a limited number of Newton iterations. Among various implementations of nonlinear interior-point methods, `IPOPT` (Interior Point Optimizer) stands out as a widely adopted and highly regarded open-source software library [81]. This versatile solver excels in handling a broad spectrum of nonlinear optimization problems, from small-scale to large-scale, and from convex to non-convex scenarios.

2.5.2 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) methods iteratively construct and solve Quadratic Programming (QP) subproblems, each of which involves minimizing a quadratic approximation of the objective function subject to linearized constraints. From a theoretical point of view, SQP methods can be seen as Newton-type methods if the strict complementarity holds in a neighbourhood $\mathcal{N}$ of the optimal solution y^*. This implies that when the SQP iteration y_i is sufficiently close to y^*, the solution of the equality-constrained NLP with active constraints also satisfies the sufficient conditions of the corresponding QP [82]. Additionally, SQP methods demonstrate fast local convergence typical of Newton-type methods. In contrast to IP methods, which work by following the central path and then do not show substantial benefits from a good initial

guess [83], SQP methods exhibit strong warm-starting capabilities, making them particularly suitable for embedded numerical optimization problems [28].

More in detail, starting with an initial guess (y_0, λ_0, μ_0), a full step SQP process is executed at each iteration i as follows:

$$y_{i+1} = y_i + \alpha_i \Delta y_i,\ \lambda_{i+1} = \lambda_i + \alpha_i(\lambda_{i_{QP}} - \lambda_i),\ \mu_{i+1} = \mu_i + \alpha_i(\mu_{i_{QP}} - \mu_i) \quad (2.22)$$

where the *step length* $\alpha_i \in (0, 1]$ is determined through an appropriate globalization strategy, which will be explained in detail later, and $(\Delta y_i, \lambda_{i_{QP}}, \mu_{i_{QP}})$ is computed as the solution of the following QP subproblem:

$$\min_{\Delta y} \quad \nabla_y \phi(y_i)^T \Delta y + \frac{1}{2} \Delta y^T H_i \Delta y \quad (2.23a)$$

subject to:

$$g(y_i) + \nabla_y g(y_i)^T \Delta y = 0, \qquad |\lambda_{QP} \quad (2.23b)$$

$$h(y_i) + \nabla_y h(y_i)^T \Delta y \leq 0. \qquad |\mu_{QP} \quad (2.23c)$$

Here, $\nabla_y g(y_i)$ and $\nabla_y h(y_i)$ represent the constraint Jacobians, the matrix H_i denotes either the exact Hessian of the Lagrangian $\nabla_y^2 \mathcal{L}(y_i, \lambda_i, \mu_i)$ or a positive definite quasi-Newton approximation B_i of it and $\nabla_y \phi(y_i)$ is the gradient of the cost function.

Remark (SQP computational complexity)**.** *When using an exact Hessian in SQP methods, the computation of second-order derivatives introduces significant computational overhead. Consider, for instance, the single shooting NMPC defined in Section 2.3 with a decision vector $y \equiv c \in \mathbb{R}^{n_c}$. In this case, the number of operations required to compute the Hessian is proportional to $O(n_c^2)$, while its inversion scales as $O(n_c^3)$.*

These operations can be prohibitively expensive for real-time applications, leading to the development of many popular variants of SQP methods based on Hessian approximation techniques.

Hessian approximation

The accuracy of the Hessian approximation, that is, its difference with respect to the exact one, significantly influences the local convergence rate of SQP-based

Newton-type optimization algorithms. This means that using the exact Hessian offers the best performance for the convergence rate, leading to a *quadratic convergence rate* in a neighbourhood of the optimal solution y^*.

Definition 13 (Q-quadratic convergence rate)**.** *Consider a sequence* $\{y_i\} \in \mathbb{R}^{n_y}$ *that converges to* y^* *for* $i \to \infty$*. The converge of* $\{y_i\}$ *is said to be Q-quadratic if there exists a positive constant* q *such that*

$$\lim_{i \to \infty} \frac{\|y_{i+1} - y^*\|}{\|y_i - y^*\|^2} \leq q.$$

This condition implies that the distance between the current iterate y_i and y^* decreases at a rate that is proportional to the square of the distance in the previous iteration. This means that the convergence becomes increasingly rapid as the iterates approach y^*.

However, computing exact Hessian requires second-order derivatives, resulting in a substantial computational overhead. This has led to the development of various Hessian approximation techniques for SQP. Although these methods typically exhibit a slower local convergence rate, they significantly reduce the computational cost per iteration, thereby typically reducing the overall computation time. The two most widely used classes of Hessian approximation techniques are *Quasi-Newton* and *Generalized Gauss-Newton* methods.

Quasi-Newton Methods. Quasi-Newton methods are a well-known class of algorithms used for solving nonlinear optimization problems [84]. These techniques, which rely on updating formulas to approximate the Hessian matrix, emerged in the early 1960s. Davidon's original contribution, written firstly on a technical note in 1959 [85] and then published on a journal in 1991 [86], laid the foundation for these methods. Their popularity was further enhanced by Fletcher and Powell in 1963 with the development of the Davidon-Fletcher-Powell (DFP) method [87]. However, the DFP method has gradually fallen out of favour in recent years. In contrast, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating formula, developed independently by Broyden [88], Fletcher [89], Goldfarb [90], and Shanno [91] in 1970, has emerged as a widely adopted approach and has been extensively modified to enhance its performance. Quasi-Newton methods, unlike exact Hessian-based SQP methods, use

exact constraint Jacobians but replace the Hessian matrix with a first-order derivative-based approximation, denoted as B_i. At each iteration, a new Hessian approximation B_{i+1} is computed from the previous approximation B_i using an updated formula that incorporates the difference of Lagrange gradients, $\upsilon_i = \nabla_y \mathcal{L}(y_{i+1}, \lambda_{i+1}, \mu_{i+1}) - \nabla_y \mathcal{L}(y_i, \lambda_i, \mu_i)$ and the step $s_i = y_{i+1} - y_i$. The aim of these Quasi-Newton methods is to effectively capture second-order information in B_{i+1} by ensuring the fulfilment of the secant equation, $B_{i+1} s_i = \upsilon_i$. As previously mentioned, among the various update formulas, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) stands out as the most widely adopted and well-established method. With BFGS formula, the matrix B_i is updated as follows:

$$B_{i+1} = B_i + \frac{\upsilon_i \upsilon_i^T}{\upsilon_i^T s_i} - \frac{B_i s_i s_i^T B_i^T}{s_i^T B_i s_i}. \tag{2.24}$$

By iteratively evaluating the Jacobian $\nabla_y \mathcal{L}(y_i, \lambda_i, \mu_i)$ at sequential points, this approach effectively extracts curvature information, allowing for the estimation of $\nabla_y^2 \mathcal{L}(y_i, \lambda_i, \mu_i)$ with significantly reduced computational overhead. As the iterations progress, the Hessian approximation B_i converges towards the exact Hessian $\nabla_y^2 \mathcal{L}(y_i, \lambda_i, \mu_i)$ in the relevant directions. The BFGS algorithm has the advantage of maintaining a positive Hessian approximation in each update by satisfying the curvature condition $q_i^T s_i > 0$. This is achieved by initializing the approximation B_0 with a positive definite matrix, such as the unit matrix. If $q_i^T s_i$ is not positive, q_i is modified element-wise to ensure that the curvature condition is met [92].

Remark (BFGS-based SQP computational complexity)**.** *Let $y \in \mathbb{R}^{n_c}$ be the decision vector. For a BFGS-based SQP formulation, the number of operations required for updating the solution is proportional to $O(n_c^2)$.*

As can be noted, this method offers a remarkable reduction in computational complexity, achieving an order-of-magnitude improvement over exact Hessian-based SQP methods. However, this computational efficiency comes at the cost of a slightly slower convergence rate. Indeed, the BFGS updating strategy has a *superlinear convergence rate* in a neighbourhood of the optimal solution y^*.

Definition 14 (Q-superlinear convergence rate)**.** *Consider a sequence $\{y_i\} \in \mathbb{R}^{n_y}$ that converges to y^* for $i \to \infty$. The converge of $\{y_i\}$ is said to be Q-*

superlinear if

$$\lim_{i\to\infty} \frac{\|y_{i+1} - y^*\|}{\|y_i - y^*\|} = 0.$$

Generalized Gauss-Newton Method Another successful variant of SQP is the so-called Generalized Gauss-Newton (GGN) method [93]. This technique is applicable to optimization problems in which the cost function is explicitly represented as a nonlinear least squares function:

$$\phi(y) = \frac{1}{2}\|R(y)\|_2^2, \tag{2.25}$$

where $R : \mathbb{R}^{n_y} \to \mathbb{R}^m$ is a smooth function, named *residual*. In the GGN method, the Hessian of the Lagrangian is approximated using the matrix:

$$B_i = \nabla_y R(y) \nabla_y R(y)^T \approx \nabla_y^2 \mathcal{L}(y, \lambda, \mu). \tag{2.26}$$

It is worth noting that the Gauss-Newton Hessian approximation B_i does not explicitly involve the dual variables, making it a multiplier-free method. This feature proves advantageous in real-world applications as it eliminates the need of a good initial guess for the Lagrange multipliers. In light of this approximation, the QP objective in Eq. (2.23a) can be rewritten as:

$$\frac{1}{2}\|R(y) + \nabla_y R(y)^T \Delta y\|_2^2. \tag{2.27}$$

The GGN method relies on the observation that the matrix B_i in Eq. (2.26) provides a reliable approximation of the Hessian assuming that either the residual norm $\|R(y)\|$ is small or the second order derivatives of the problem functions are small. This second assumption means that, given the exact second-order derivative

$$\nabla_y^2 \mathcal{L}(\cdot) = \nabla_y R(y) \nabla_y R(y)^T + \sum_{i=1}^{m} R_i(y) \nabla_y^2 R_i(y) + \sum_{i=1}^{n_e} \lambda_i \nabla_y^2 g_i(y) + \sum_{i=1}^{n_i} \mu_i \nabla_y^2 h_i(y),$$

the terms $\nabla_y^2 R_i(y)$, $\nabla_y^2 g_i(y)$ and $\nabla_y^2 h_i(y)$ are negligibly small. Note that, if at the solution the residual is zero, which indicates a perfect fit, the GGN method exhibits a locally quadratic convergence rate. In all other cases, it typically achieves a *linear convergence rate*.

Definition 15 (Q-linear convergence rate)**.** *Let* $\{y_i\} \in \mathbb{R}^{n_y}$ *be a sequence that converges to* y^* *for* $i \to \infty$*. The converge of* $\{y_i\}$ *is said to be Q-linear if there is a constant* $l_r \in (0,1)$ *such that*

$$\lim_{i \to \infty} \frac{\|y_{i+1} - y^*\|}{\|y_i - y^*\|} \leq l_r.$$

Figure 2.1 illustrates the behavior of the three types of convergence rates described above.

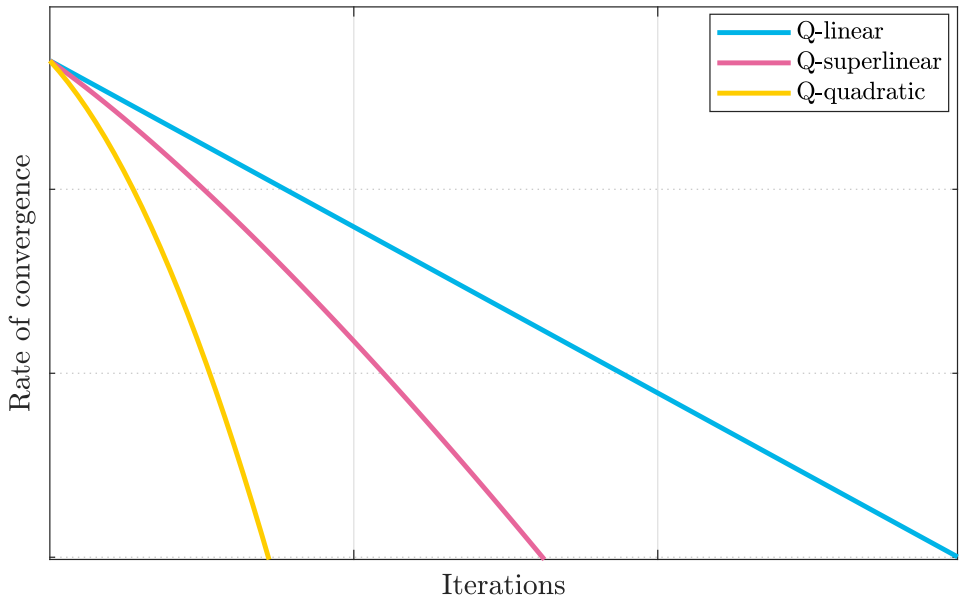

Fig. 2.1 Convergence rates of Newton-type optimization algorithms.

Global convergence strategies

After a thorough analysis of the local convergence properties of the SQP algorithms, i.e. when the sequence y_i is in close proximity to the optimal solution y^*, we will now shift our focus to investigate their global convergence characteristics. This concept is of critical importance in ensuring that SQP algorithms can consistently identify the optimal solution regardless of the initial conditions. The basic algorithm of an SQP can be defined as follows:

Algorithm 1 Basic SQP algorithm

Input: (y_0, λ_0, μ_0), B_0 and $i = 0$.

Output: (y^*, λ^*, μ^*).

1: Form and solve the QP (2.23) to obtain $(\Delta y_i, \lambda_{i_{QP}}, \mu_{i_{QP}})$.

2: Choose the step length α_i.

3: Set:

$$y_{i+1} = y_i + \alpha_i \Delta y_i,\ \lambda_{i+1} = \lambda_i + \alpha_i(\lambda_{i_{QP}} - \lambda_i),\ \mu_{i+1} = \mu_i + \alpha_i(\mu_{i_{QP}} - \mu_i).$$

4: Stop if converged.

5: Compute B_{i+1}.

6: Set $i = i + 1$ and go to Step 1.

Specifically, at each iteration, Steps 1 and 2 are used to generate the new sequence $(y_{i+1}, \lambda_{i+1}, \mu_{i+1})$. Step 1 has been described in detail in the previous paragraphs: it consists in solving the QP subproblem (2.23). Step 2 involves the computation of the step length α_i. Depending on the optimization problem at hand, different strategies can be employed for this purpose.

In real-time optimization scenarios, the assumption of local convergence is typically adopted, given that the previous iteration generally provides a sufficiently good initial guess for the subsequent problem. As a result, a full Newton step is performed, implying that the value of α is always set equal to 1 [28]. This assumption holds significant importance, especially in the context of real-time NMPC, as discussed in Section 2.6.

In the context of generic nonlinear optimization, where optimal solutions can vary significantly between iterations, the implementation of globalization strategies becomes essential to determine the value of α. Among these, line search, trust-region, and filter methods are the most commonly used. In both line search and trust-region approaches, a so-called *merit function* is typically employed to find the value of α. The merit function is essentially a modified version of the cost function that includes metrics for quantifying the violation of constraints. This strategy effectively circumvents poor convergence behavior in iteration regimes where constraints should be relaxed. In methods that use line search, the merit function regulates the size of the step. Conversely, in trust-region techniques, it determines the acceptance or rejection of the step and the potential adjustment of the trust-region radius. A widely used merit

function for the generic NLP (2.9) is the l_1 penalty function, which is defined as follows:

$$\mathcal{M}_1(y;\mu) = \phi(y) + \rho \sum_{i\in\mathcal{E}} |g_i(y)| + \rho \sum_{j\in\mathcal{I}} \max[0, h_j(y)]. \tag{2.28}$$

The penalty parameter, denoted as the positive scalar ρ, establishes the balance between satisfying constraints and minimizing the objective function. A trial step $y^+ = y + \alpha\Delta y$ is considerable acceptable if it results in a sufficient reduction in the merit function $\mathcal{M}_1$. In unconstrained optimization, the concept of "sufficient reduction" in the objective function ϕ is defined by the following inequality:

$$\phi(y_i + \alpha\Delta y_i) \leq \phi(y_i) + c_1\alpha_i\nabla\phi_i^T\Delta y_i \tag{2.29}$$

for a certain constant $c_1 \in (0,1)$. The inequality (2.29), referred to as the *Armijo condition*, suggests that the decrease in the function ϕ should be proportional to both α_i and the directional derivative $\phi_i^T\Delta y_i$. Within the realm of generic NLP, a concept similar to the condition (2.29) can be used. In this context, the directional derivative of the merit function $\mathcal{M}_1(y;\mu)$ in the direction of Δy is defined as $\mathcal{D}(\mathcal{M}_1(y;\mu);\Delta y)$. For line search methods, the "sufficient reduction" condition requires choosing a step length parameter $\alpha > 0$ sufficiently small such that, for a specified constant $\eta \in (0,1)$, the inequality:

$$\mathcal{M}_1(y + \alpha\Delta y;\mu) \leq \mathcal{M}_1(y;\mu) + \eta\alpha\mathcal{D}(\mathcal{M}_1(y;\mu);\Delta y) \tag{2.30}$$

is satisfied. For trust-region methods, a quadratic model, denoted as $q(\Delta y)$, is typically used to approximate the subsequent value of the merit function $\mathcal{M}_1$ after a step Δy. The "sufficient reduction" condition can be expressed in terms of a decrease in this model as follows:

$$\mathcal{M}_1(y + \alpha\Delta y;\mu) \leq \mathcal{M}_1(y;\mu) - \eta(q(0) - q(\Delta y)), \tag{2.31}$$

for some $\eta \in (0,1)$.

Filter methods are step acceptance mechanisms inspired by concepts from multiobjective optimization. Their derivation relies in the understanding that the NLP has two goals: reducing the objective function and fulfilling the

constraints. A measure of infeasibility can be expressed as

$$H(y) = |g_i(y)| + \sum_{j \in \mathcal{I}} \max[0, h_j(y)], \tag{2.32}$$

which allows to write these two goals as

$$\min_y f(y) \qquad \min_y H(y). \tag{2.33}$$

Contrary to merit functions that combine these two objectives into a single minimization problem, filter strategies keep them separate. A trial step y^+ is approved as a new iteration if the pair $(f(y^+), H(y^+))$ is not "dominated" by a previous pair $(f_l, H_l) = (f(y_l), H(y_l))$ produced by the algorithm. When an iteration y_i is acceptable for the filter, typically (f_i, H_i) is incorporated into the filter and any pairs that are dominated by (f_i, H_i) are eliminated. For a more detailed discussion of these globalization strategies, refer to [71, 74].

2.6 An Overview of Fast Optimization Techniques for NMPC

As defined in Section 2.4, the implementation of NMPC involves solving online NLPs, such as the one in (2.9), to determine, at each time instant, the optimal control actions for the system. However, the process of finding online solutions for non-convex optimizations is computationally intensive. It requires high computational times, which often do not meet the fast sampling rate requirements of real-time applications. This challenge is further compounded by the fact that the feasible state and control trajectory are only found at convergence. In this context, during the past decades, significant efforts have been devoted to reduce the computational complexity of NMPC approaches, ensuring real-time feasibility. These advancements are based on the following key ingredients.

- *Approximate Feedback Policies*: NMPC typically requires just a reasonable level of accuracy in the NLP solution. This allows the use of inexact NLP algorithms.

- *Algorithm Initialization*: By sequentially solving a series of closely related optimal control problems, the algorithm can be effectively initialized using the approximate solution from the previous time step. This process is usually referred to as *warm start*.
- *Offline Computations*: To minimize real-time computation demands, many optimization tasks can be performed offline. This includes code optimization, precomputations, and even explicit solutions for simplified and small-scale problems.
- *Preparation and Feedback Phases*: The control problem can be divided into two distinct phases: preparation and feedback. The preparation phase, typically CPU-intensive, can be performed offline, while the feedback phase quickly generates an approximate solution based on the most recent system state.

Existing fast NMPC algorithms employ the aforementioned features to produce suboptimal solutions while preserving a specific level of optimality guarantee. Among several prominent approaches, the following stand out.

Multistep Newton-Type Method

This method, developed by Li and Biegler at the end of the 1980s [25], is one of the first attempts to apply NMPC in real-time. This algorithm considers a sequential optimal control problem, aiming to minimize a cost function that depends on the control inputs $\mathbf{u} = (\mathfrak{u}_1, \mathfrak{u}_2, \ldots, \mathfrak{u}_T)$ over a finite horizon T. Instead of treating state variables as optimization variables, they are considered constraints that must be satisfied by the system dynamics. To solve the optimization problem, an SQP method is employed with an approximation of the Hessian matrix using Gauss-Newton method and the implementation of a line search strategy. The algorithm is designed to work in an online fashion. This means that, instead of solving the optimization problem until the optimality, it performs only one SQP iteration and applies the first element of the updated control input as a feedback for the plant. In this way, the algorithm can cope with the limited computational time and resources available in real-time applications. To prepare the problem for the next sampling time, the

algorithm shifts the control inputs by one step forward and adds a new control input at the end of the horizon: $\mathbf{u}^{\text{next}} = (u_2, \ldots, u_T, u_{T+1}^{\text{next}})$, with $u_{T+1}^{\text{next}} = u_T$. The Multistep Newton-Type Method is a simple and efficient algorithm for NMPC, but it has some drawbacks, such as the possible lack of nonlinear convergence. This means that the algorithm may struggle to find a feasible or optimal solution for the optimization problem, especially if the system dynamics are highly nonlinear or the initial guess is far from the true solution.

Continuation/Generalized Minimal Residual Method

In the early 2000s, Ohtsuka developed an online algorithm for NMPC, called the Continuation/Generalized Minimal Residual (C/GMRES) Method [26], which shares certain similarities with the Multistep Newton-Type Method in its design and functionality. Both algorithms are based on a sequential optimal control problem, where the control inputs are the only optimization variables, and perform just one Newton-type iteration at every sampling time. However, the C/GMRES method differs from the Newton-Type approach in several aspects:

- Inequality constraint handling: It uses an Interior-Point (IP) method with a fixed barrier parameter $\tau > 0$ to effectively address inequality constraints.
- Exact Hessian computation: Instead of using a Gauss-Newton approximation, the C/GMRES method calculates the exact Hessian matrix of the Lagrangian function, enhancing its accuracy.
- Use of GMRES method: For each Newton step, the C/GMRES method employs the GMRES algorithm [94], an efficient iterative technique for solving linear systems.
- Tangential predictor: Instead of solely relying on control input shifts, the C/GMRES method introduces a tangential predictor. This predictor exploits parametric sensitivities, which represent the derivatives of optimal control inputs with respect to system states. By analyzing these sensitivities, the tangential predictor enables the algorithm to predict optimal control inputs for neighbouring states without repeating the

NLP problem. This capability is particularly advantageous in regions near active set changes, where the solution manifold exhibits significant curvature.

- Predictor-corrector pathfollowing approach: The C/GMRES method follows a predictor-corrector pathfollowing strategy, alternating between predicting new points on the solution manifold and correcting them to satisfy constraints.

Closed-loop stability of the C/GMRES method has been established in the literature [95], demonstrating its effectiveness in real-time NMPC applications.

Advanced-Step NMPC Controller

The advanced-step NMPC (asNMPC) controller, proposed by Zavala and Biegler in 2009 [96], is a conservative approach to avoid the convergence issues that arise in predictor-corrector pathfollowing methods. This approach consists of two main steps.

- *Preparation phase*: At each sampling time t, a complete Newton-type IP problem is solved in advance until the convergence (with $\tau \to 0$). The term "advance" is used to denote that the OCP is solved considering an expected state $\hat{x}(t+1)$ as initial value, computed from the last control input $u(t)$ and the last actual measurement $x(t)$. This enables to find a new control action before the next sampling time $t+1$.
- *Feedback phase*: At the sampling time $t+1$, the algorithm does not directly apply the solution obtained from the first step. Instead, it uses the tangential predictor method to correct the discrepancies between the predicted state $\hat{x}(k+1)$ and the actual state $x(k+1)$. This correction enables the adjustment of the control action to the disturbance effects.

One of the peculiar features of this approach is that it performs several Newton iterations in the preparation phase, unlike other online algorithms that use just one iteration per sampling time. Another feature is that it prioritizes system nonlinearity over sudden active set changes, as the IP predictor is not well-suited for handling these types of changes. Therefore, the advanced

step controller is more suitable for nonlinear systems with smooth active set transitions.

Real-Time Iteration Scheme

The Real-time Iteration (RTI) scheme, introduced in [22] at the beginning of the 2000s, has emerged as one of the most successful and widely adopted methods for achieving fast NMPC. This method shares similarities with the Newton-type controller [25] in some aspects of its iterative optimization process, employing a single SQP iteration with Gauss-Newton Hessian approximation at each sampling instant. However, unlike the method developed by Li and Biegler, it implements an *initial value embedding* and uses multiple shooting strategies. As the advanced-step NMPC controller, the online optimization procedure is split into two phases: preparation and feedback.

More precisely, the RTI method works by performing, at each sampling time, one linearization and one SQP iteration to solve the NLP (2.9) parametrized by multiple shooting, resulting in an approximate feedback control policy. This significantly reduces the computational time compared to solving the same problem repeatedly. Additionally, one of the primary motivations behind the RTI scheme is to avoid iterating a problem that becomes outdated as the system evolves, but rather apply an approximate solution computed using the most up-to-date information. This motivation illustrates the so-called *real-time dilemma* [97], which arises due to the dynamic nature of physical systems: as the Sequential Quadratic Programming (SQP) iterations are being executed, the physical system continues to evolve, making the information used to compute the control actions obsolete. In this context, a key component of the RTI scheme is the initial value embedding, which consists in keeping the current state estimate $\hat{x}_1$ as a constrained variable. This characteristic allows for the division of computations into preparation and feedback phases. During the preparation phase, the QP problem is formulated with the predicted state $\hat{x}_1$. Once the actual state x_1 becomes available, the feedback phase quickly produces a solution guess for the new problem by solving the prepared, convex QP problem. Another important aspect of the RTI scheme is the shifting initialization or warm-start technique. It consists in initializing subsequent problems with the solution obtained in the previous iteration. This is based

on the expectation that the actual state estimate does not differ significantly from the first-order correction to the optimal solution of the previous iterate. Efficient implementations of RTI are provided by `ACADO Toolkit` [98, 48] and `acados` [99].

2.7 Chapter summary

This chapter has provided a thorough overview of Nonlinear Model Predictive Control (NMPC), covering its formulation, theoretical underpinnings, and optimization methods. It started by detailing the NMPC optimization problem, including general optimal control problem formulation and numerical methods, with a focus on direct optimal control and direct single shooting NMPC formulation. The chapter also explored theoretical concepts of nonlinear programming, discussing first-order and second-order conditions of optimality, which are crucial for understanding solution criteria in nonlinear optimization. Additionally, it reviewed Newton-type optimization methods, particularly interior-point (IP) methods and sequential quadratic programming (SQP), highlighting their importance in efficiently solving NMPC problems. Lastly, it examined fast optimization techniques tailored for NMPC, emphasizing the need for rapid and reliable algorithms in real-time applications. This chapter sets a solid foundation for the subsequent chapters in which novel methods designed for reducing the computational complexity of the NMPC will be presented.

Chapter 3

Dimensionality Reduction for Fast NMPC

A relevant problem of the MPC optimization process is the high dimensionality of the search space, which can lead to issues like slow convergence, computational inefficiency, and difficulty in exploring the entire search domain. This is especially critical in nonlinear optimization, where a large dimensionality of the problem poses a significant obstacle, requiring extensive computational resources and often results in a combinatorial "explosion" of possible solutions. This problem is often referred to as the *curse of dimensionality*, a term introduced by Richard Bellman [57] to describe a phenomenon in which the overall volume of the search space increases exponentially with the number of variables, leading to a sparse and dispersed sampling of potential solutions. As a result, nonlinear optimization algorithms face an increased computational burden and/or a higher likelihood of getting stuck in "poor" local minima.

In the context of real-time NMPC, this problem becomes even more critical. Indeed, the complexity introduced by the high dimensionality of the search space can pose significant challenges in obtaining an optimal solution within the stringent time constraints typically associated with real-time applications. This may result in poor control performance or even instability. Therefore, effectively addressing the curse of dimensionality is crucial for ensuring the successful application of real-time NMPC.

Dimensionality reduction techniques offer a promising approach to mitigate this problem. Suppose that s^d is the volume of the search domain, where s represents its side length and d is the number of decision variables of the problem, i.e. the dimensionality. The main contribution of this chapter is to present a novel method for reducing the dimensionality d by selecting the decision variables that have the most significant influence on the optimal control problem (OCP) and performing optimization solely on them. This is achieved by combining two strategies:

1. Gradient computation, using a new efficient approach;
2. Nonlinear function approximation, using a Nonlinear Sparse Identification technique.

The effectiveness of this method in reducing computational time in nonlinear optimization algorithms is theoretically demonstrated through an analysis of its computational complexity.

Outline. The chapter is organized as follows. Section 3.1 offers a comprehensive survey of existing dimensionality reduction techniques. In Section 3.2, an overview of the state-of-the-art gradient computation methods is provided, along with a novel and efficient approach developed in this thesis. Section 3.3 describes the Nonlinear Sparse Identification method. Section 3.4 details the novel method based on gradient computation and nonlinear identification. Its computational complexity is analyzed theoretically in Section 3.5.

3.1 Dimensionality Reduction in NMPC: A Review of Existing Methods

Suboptimal solutions are often used in real-time NMPC, as shown, e.g., in [100, 101]. These solutions can result either from stopping the iterative numerical methods before convergence, or from simplifying the Optimal Control Problem (OCP) by using a low-dimensional input parametrization, that is, solving the OCP in a smaller search space. The first approaches have already been outlined in Section 2.6. They consist in fast but inexact NLP algorithms, based on warm

start and offline computations. The second approaches, which aim to solve the optimization problem faster by reducing the dimensionality of the search space, are described in this section. Specifically, these methods are designed to effectively guide the optimization process, ensuring that computational efforts are concentrated in the most promising regions of the search domain. Through the reduction of search space dimensionality, these techniques can significantly reduce the computational complexity of NMPC, making it more feasible for real-time applications. Three methods for dimensionality reduction have emerged in the context of NMPC: i) Laguerre functions, ii) Move Blocking, and ii) Active Subspaces. Other methods, such as Singular Value Decomposition (SVD), are generally used for linear MPC (see, e.g., [102–104]) and therefore they will not be discussed in this thesis.

3.1.1 Laguerre functions

Laguerre functions, named after 19th-century French mathematician Edmond Laguerre, are a family of orthogonal polynomials that have found various applications in mathematics, physics, and engineering. They are characterized by their ability to represent arbitrary signals with varying degrees of accuracy, making them well-suited for modeling and control tasks. In the context of MPC, they can be used to reformulate the online optimization problem, enhancing its computational efficiency and robustness. Notably, the application of Laguerre functions to parametrize the control sequence of the manipulated variables in linear MPC was first introduced in the early 2000s [105]. More recently, Laguerre functions have also been successfully applied to NMPC [106, 107]. In detail, consider $l_{1,t}, \ldots, l_{n_L,t}$ as n_L Laguerre functions at time t. As established by [108], the impulse response of any dynamic system can be represented by using a Laguerre function of order m, represented by a z-transfer function:

$$G_m(z) = \frac{\sqrt{1-a^2}}{z-a}\left(\frac{1-az}{z-a}\right)^{m-1} \tag{3.1}$$

where a represent a scaling factor, commonly referred to as *Laguerre pole*. For stability, the constraint $0 \leq a < 1$ must be satisfied. Laguerre functions are

defined as the inverse Z-transforms of the transfer function $G(z)$, as follows:

$$l_{n,t} = \mathcal{Z}^{-1}(G_n(z)). \tag{3.2}$$

The obtained Laguerre functions can be written in the discrete form as:

$$L_{t+1} = \Omega L_t \tag{3.3}$$

where L_t is

$$L_t = (l_{1,t}, l_{2,t}, \ldots, l_{nL,t})$$

and Ω is a $n_L \times n_L$ matrix defined in terms of a and $\beta = 1 - a^2$. For further mathematical details, please refer, for example, to [106].

Consider the optimization of the mono-dimensional control increments $\Delta\mathbf{u}_t = (\Delta u_{t+1}, \ldots, \Delta u_{t+T})$ at time t, where these increments can be seen as the impulse response of a stable dynamic system. Note that: i) T represents the prediction horizon; ii) $\Delta\mathbf{u}_t$ is a column vector (see Section 1.5). Thus, a set of Laguerre functions along with a set of Laguerre coefficients can be used to depict the dynamic responses of the control increment as follows:

$$\Delta u_{t+i} = \sum_{j=1}^{n_L} l_{j,i} c_{j,t}. \tag{3.4}$$

Here, t denotes the initial time, i represents the future sampling instant, and $c_{j,t}$ are the Laguerre coefficients associated with the Laguerre functions $l_{1,i}, l_{2,i}, \ldots, l_{n_L,i}$. Vectorizing the equation (3.4) yields:

$$\Delta u_{t+i} = L_i^T c_t, \tag{3.5}$$

where $c_t = (c_{1,t}, \ldots, c_{n_L,t})$ is the vector of Laguerre coefficients. Extending to the entire prediction horizon:

$$\Delta\mathbf{u}_t = \mathbf{L} c_t, \tag{3.6}$$

where $\mathbf{L}$ is a $T \times n_L$ matrix defined as

$$\mathbf{L} = \begin{bmatrix} l_{1,1} & l_{2,1} & \dots & l_{n_L,1} \\ l_{1,2} & l_{2,2} & \dots & l_{n_L,2} \\ \vdots & \vdots & \dots & \vdots \\ l_{1,T} & l_{2,T} & \dots & l_{n_L,T} \end{bmatrix}. \tag{3.7}$$

This approach streamlines the optimization process by employing a vector of Laguerre coefficients c_t instead of the vector of control increments $\Delta\mathbf{u}_t$. This transformation is advantageous as $n_L < T$, resulting in a reduction in the number of decision variables and an improved optimization efficiency.

3.1.2 Move Blocking

Move blocking, a concept already introduced in Chapter 2, is a strategy employed both in linear MPC (LMPC) [109, 110] and, more recently, in NMPC [111, 112]. In a typical LMPC/NMPC problem, the optimization algorithm generates a sequence of control inputs, one for each time step within the prediction horizon. This "freedom" of input variation can lead to a large number of decision variables, particularly for systems with long prediction horizons. Since the computational complexity of solving the optimization problem directly correlates with the number of decision variables, a large number of these variables can pose significant computational burdens and hinder the efficient solution of the control problem. Move blocking addresses this issue by forcing the control inputs to be constant over specified periods, known as *blocks*. Instead of making a new decision at each time step, the controller makes a decision only at the beginning of each block. The same control input is then applied throughout the block. This approach significantly reduces the number of decision variables, simplifying the control problem and enhancing the computational efficiency. Furthermore, this method stands out from others parametrized approaches due to its ability to maintain the physical meaning of the input values during the optimization process, while also offering relatively simple hardware implementation. However, it is important to note that this can come at the cost of reduced control performance, as the controller has fewer degrees of freedom to influence the system.

More in detail, without move blocking, the value of the control input is allowed to change at each time step, i.e. each component of the vector $\mathbf{u} \doteq (\mathfrak{u}_1, \ldots, \mathfrak{u}_T) \in \mathbb{R}^{Tn_u}$ is a degree of freedom of the optimization problem. Using the block moving strategy, restrictions on the variability of the control input are introduced. In particular, the control input is fixed to be constant over several sample steps, called blocks. This results in a reduced order control vector $\hat{\mathbf{u}} \doteq (\mathfrak{u}_1, \ldots, \mathfrak{u}_{M_p}) \in \mathbb{R}^{M_p n_u}$, where $M_p < T$ is the number of blocks, and a so-called blocking matrix Γ. This matrix consists of zeros and ones, with exactly one nonzero element in each row. The original control vector $\mathbf{u}$ can then be reconstructed from $\hat{\mathbf{u}}$ using the equation $\mathbf{u} = (\Gamma \otimes I)\hat{\mathbf{u}}$, where $\otimes$ denotes the Kronecker product.

Until now, no specific informations have been given on how to determine the number of blocks M_p and design the matrix Γ. In the literature, different methods for accomplishing this task have been presented. Among them, the *Optimal Move Blocking* strategy, described in [110], is one of the most widely used. This approach involves defining an optimization problem to obtain an optimal blocking structure that, for example, minimizes both the number of blocks and the overall prediction horizon length.

3.1.3 Active Subspaces

Active subspaces represent a dimensionality reduction technique used for simplifying high-dimensional optimization problems, where the objective function depends on many input variables or parameters (see, e.g., [113, 114]). As shown in [36], this method can be effectively applied in the case of NMPC to reduce the computational complexity without degrading closed-loop performance. The main idea behind active subspaces is to identify the most important directions in the input space that affect the output, and then project the inputs onto a lower-dimensional subspace spanned by these directions. This results in a simpler surrogate model that approximates the original function, enabling more efficient and accurate optimization. Active subspaces can be computed by estimating the gradient of the objective function at different input points, and then performing a singular value decomposition on the matrix of gradient samples for obtaining the corresponding eigenvalues and eigenvectors. Since the eigenvalues represent the relative sensitivity of the objective function along

each direction, the eigenvectors corresponding to the largest eigenvalues can be used to define the active subspace, which encapsulates the primary directions of variability in the objective function.

Mathematically, the active subspace can be defined as follows. Let $J : \mathbb{R}^{Tn_u} \to \mathbb{R}$ be a smooth function, and let $\mathbf{u} \in \mathbb{R}^{Tn_u}$ be an input vector with a probability density function $p(\mathbf{u})$. The gradient of J at $\mathbf{u}$ is denoted by $\nabla_{\mathbf{u}} J(\mathbf{u})$ (for more information about the gradient of a function see Section 3.2). Under the assumption that the products of the partial derivatives are integrable, the $Tn_u \times Tn_u$ matrix $\mathbf{C}$ can be defined as:

$$\mathbf{C} = \int_{\mathbb{R}^{Tn_u}} \nabla_{\mathbf{u}} J(\mathbf{u}) \nabla_{\mathbf{u}} J(\mathbf{u})^\top p(\mathbf{u})\, d\mathbf{u}. \tag{3.8}$$

The matrix $\mathbf{C}$ is a measure of the uncentered covariance between the components of the gradient vector. It is a symmetric and positive semidefinite matrix, which implies that it has a real eigenvalue decomposition, as expressed in the following equation:

$$\mathbf{C} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T, \quad \mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_m), \tag{3.9}$$

where $\mathbf{W}$ is the $Tn_u \times Tn_u$ orthonormal matrix spanned by the eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{Tn_u}$ of $\mathbf{C}$ and $\lambda_1 \geq \cdots \geq \lambda_{Tn_u} \geq 0$ are the corresponding eigenvalues.

The eigenvectors $\mathbf{W}$ transform the original space $\mathbb{R}^{Tn_u}$ into a rotated coordinate system, effectively remapping the domain of the function J. The eigenvalues, arranged in descending order, divide the rotated coordinate system into two distinct subsets: one with a larger average variation and one with a smaller average variation. This division allows for the partitioning of the eigenvalues and eigenvectors into two separate matrices:

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \\ & \mathbf{\Lambda}_2 \end{bmatrix}, \quad \mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2]. \tag{3.10}$$

Here: i) $\mathbf{W}_1$ is a $Tn_u \times m$ matrix representing the *active subspace*, and $\mathbf{\Lambda}_1 = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$ with $m < Tn_u$ is the matrix containing the corresponding eigenvalues; ii) $\mathbf{W}_2$ is a $(Tn_u - m) \times m$ matrix representing the *inactive subspace*, and $\mathbf{\Lambda}_2 = \operatorname{diag}(\lambda_{m+1}, \ldots, \lambda_{Tn_u})$ contains the corresponding eigenvalues. Note that the choice of the dimension of the active subspace, i.e., the selection of the parameter m, is specific to the problem at hand and depends on the spectrum

of $\mathbf{C}$. The active subspace captures the directions in the input space that exert the most influence on the output. The eigenvalues λ_i quantify the amount of variation in the output due to changes in the direction $\mathbf{w}_i$. Therefore, if there is a gap between λ_m and λ_{m+1}, it means that the output is relatively insensitive to the directions in the inactive subspace, and the active subspace can be used to approximate the function.

The rotated coordinates $\mathfrak{u}_a \in \mathbb{R}^m$ and $\mathfrak{u}_{in} \in \mathbb{R}^{Tnu-m}$ can be defined as:

$$\mathfrak{u}_a = \mathbf{W}_1^T \mathfrak{u}, \qquad \mathfrak{u}_{in} = \mathbf{W}_2^T \mathfrak{u}, \tag{3.11}$$

where $\mathfrak{u}_a$ and $\mathfrak{u}_{in}$ denote the active and inactive variables, respectively. The active variable $\mathfrak{u}_a$ is a lower-dimensional representation of the input $\mathfrak{u}$ that preserves the most information about the output $J(\mathfrak{u})$. One can then fit a surrogate model $J_g : \mathbb{R}^m \to \mathbb{R}$ to the data $(\mathfrak{u}_{a_i}, J(\mathfrak{u}_i))$, where $\mathfrak{u}_i$ and $\mathfrak{u}_{a_i}$ are sampled from the input space and the active subspace, respectively. The surrogate model J_g can be used to approximate the original function J by

$$J(\mathfrak{u}) \approx J_g(\mathbf{W}_1^\top \mathfrak{u}).$$

The surrogate model J_g can also be used to perform optimization on the active subspace, by finding the optimal active variable $\mathfrak{u}_a^* = \arg\min_{\mathfrak{u}_a \in \mathbb{R}^m} J_g(\mathfrak{u}_a)$, and then mapping it back to the input space by $\mathfrak{u}^* = \mathbf{W}_1 \mathfrak{u}_a^*$. This can be much faster and more accurate than optimizing the original function J on the full input space.

3.2 Efficient Gradient Computation Method

This thesis proposes a novel method to address dimensionality reduction in NMPC by using gradient information to identify the directions of greatest variability within the search domain of the optimization problem. To achieve this, a novel and efficient approach for gradient computation has been developed. Before discussing this approach, an overview of the state-of-the-art gradient computation methods is presented below.

3.2.1 Overview of Gradient computation methods

The gradient of a multivariable function $f(x, y, \dots)$, denoted as ∇f, represents the collection of all its partial derivatives in a vector form. Formally, it can be expressed as:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \dots \right)$$

In this representation, each component of the vector corresponds to the partial derivative of the function with respect to a particular variable. The efficient computation of these derivatives is a fundamental task in various applications of computer science, particularly in optimization. Indeed, most optimization schemes rely on derivative information when minimizing a function. However, manually computing derivatives is error-prone and can be time-consuming, especially when dealing with complex functions. Hence, methods for automating this computation have been designed. The three major approaches that can be found in the literature are: i) Finite difference [115], ii) Symbolic differentiation [116], and iii) Algorithm differentiation [117]. Each method offers distinct characteristics in terms of accuracy and computational complexity, which will be discussed in detail in the following.

Finite difference method

The finite difference approximations for derivatives are among the simplest and oldest methods used to solve differential equations. They were already known by L. Euler around 1768, in one-dimensional space, and were probably extended to two dimensions by C. Runge around 1908. The advent of finite difference techniques in numerical applications began in the early 1950s, stimulated by the emergence of computers that offered a convenient framework for dealing with complex problems in science and technology. Theoretical results regarding the accuracy, stability, and convergence of the finite difference method for partial differential equations have been obtained over the last five decades [118].

This method approximates the derivatives using values of the original function calculated at some points. In its simplest form, it is based on the limit definition of a derivative. For example, for a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$,

one can approximate the gradient $\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)$ using

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + h\mathbf{e}_i) - f(x)}{h}, \tag{3.12}$$

where $\mathbf{e}_i$ is the i-th unit vector and $h > 0$ is a small step size.

The finite difference method, while uncomplicated to implement, has two limitations. Firstly, numerical approximations of derivatives are inherently ill-conditioned and unstable. This is due to the introduction of truncation and round-off errors caused by the limited precision of computations and the chosen value of the step size h. Truncation error tends to zero as $h \to 0$. However, as h decreases, round-off error increases becoming dominant. Secondly, it requires performing $O(n)$ evaluations of f for a gradient in n dimensions, which can be computationally expensive for large values of n. Therefore although simple, the finite difference method is inexact and slow.

Symbolic differentiation method

Symbolic differentiation involves the automatic manipulation of expressions for obtaining other symbolic expressions representing the derivatives. This is carried out by applying transformations that represent rules of differentiation, such as the sum and the product rules, which are defined below:

$$\frac{d}{dx}(f(x) + g(x)) \to \frac{d}{dx}f(x) + \frac{d}{dx}g(x), \tag{3.13}$$

$$\frac{d}{dx}(f(x)g(x)) \to \left(\frac{d}{dx}f(x)\right)g(x) + f(x)\left(\frac{d}{dx}g(x)\right). \tag{3.14}$$

Symbolic differentiation offers exact results and avoids the numerical errors associated with finite precision calculations. Furthermore, especially in optimization, symbolic derivatives offer valuable insights into the structure of the problem domain and, in some cases, can even provide analytical solutions for extrema (e.g., $\frac{d}{dx}f(x) = 0$), thus eliminating the need for derivative calculation. However, it is worth noting that symbolic derivatives may not be efficient for the runtime calculation of derivative values due to their potential exponential growth compared to the original expression.

Algorithm differentiation method

Algorithm Differentiation (AD), also called Automatic Differentiation [119], exploits the concept that all computer computations, regardless of complexity, involve a series of basic arithmetic operations (such as addition, subtraction, multiplication, and division) and elementary functions (like exp, log, sin, cos). Through repeated applications of the chain rule to these operations, it becomes possible to automatically compute partial derivatives of any order with precision, using only a minimal increase in the number of arithmetic operations compared to the original program. In other words, the AD method involves using the chain rule for breaking down complex functions into elementary ones for which it is possible to exactly compute the derivative.

Consider for example the following function:

$$f(x_1, x_2) = x_2 \sin(x_1^2 + x_2^2). \tag{3.15}$$

The equation above can be expressed in terms of elementary functions using the *three-part* notation, defined in [117]. In detail, this notation involves constructing a function $f : \mathbb{R}^n \to \mathbb{R}^m$ by using intermediate variable w_i such that:

- variables $w_{i-n} = x_i$, for $i = 1, \ldots, n$, are the input variables,
- variables w_i, for $i = 1, \ldots, l$, with l the numbers of operations in which f can be decomposed, are the working variables,
- variables $y_{m-i} = w_{l-i}$, for $i = m-1, \ldots, 0$, are the output variables.

Using this notation, the Eq. (3.15) can be represented as an *evaluation trace* of elementary operations:

$$y = w_0 \sin(w_{-1}^2 + w_0^2) = w_0 \sin(w_1 + w_2) = w_0 \sin(w_3) = w_0 w_4 = w_5, \tag{3.16}$$

with the variables w_i defined in Table 3.1, representing the the so-called *forward primal trace*. Note that the arrow indicates the direction of calculation of the function.

Table 3.1 Forward primal trace of the function in Eq. (3.15).

Forward Primal Trace
$w_{-1} = x_1$
$w_0 = x_2$
$w_1 = w_{-1}^2$
$w_2 = w_0^2$
$w_3 = w_1 + w_2$
$w_4 = \sin(w_3)$
$w_5 = w_0\, w_4$
$y = w_5$

The trace of elementary operations can be represented as a *computational graph*, as shown in Figure 3.1. This visualization can be helpful for understanding the dependency relations between intermediate variables.

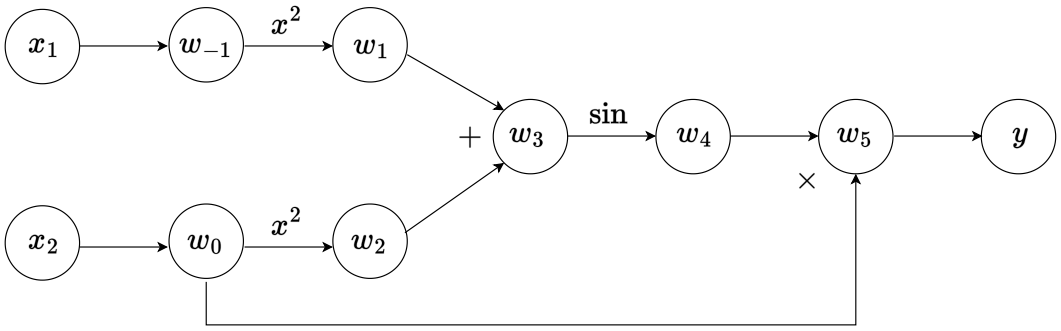

Fig. 3.1 Computational graph of the function in Eq. (3.15).

Typically, two distinct methods are used for automatic differentiation based on how the graph is traversed for the calculation of derivatives: i) forward mode, and ii) reverse mode.

Forward mode The forward mode evaluates a numerical derivative by performing elementary derivative operations together with the evaluation of the function itself. This process relies on the following recursive relation:

$$\frac{\partial w_i}{\partial x_i} = \frac{\partial w_i}{\partial w_{i-1}} \frac{\partial w_{i-1}}{\partial x_i}. \tag{3.17}$$

In this equation, the term $\frac{\partial w_i}{\partial w_{i-1}}$ captures the intermediate derivative information during the function evaluation. As an example, consider the function in Eq. (3.15). To compute the derivative of the function f with respect to x_1, the initial step involves associating each intermediate variable, denoted as w_i, with the corresponding derivative:

$$\dot{w}_i = \frac{\partial w_i}{\partial x_1}. \tag{3.18}$$

Applying the chain rule to each elementary operation in the forward primal trace generates the corresponding *forward derivative trace*, as shown in Table 3.2.

Table 3.2 Forward derivative trace of the function in Eq. (3.15).

Forward Derivative Trace
$\dot{w}_{-1} = \dot{x}_1$
$\dot{w}_0 = \dot{x}_2$
$\dot{w}_1 = 2\, w_{-1}\, \dot{w}_{-1}$
$\dot{w}_2 = 2\, w_0\, \dot{w}_0$
$\dot{w}_3 = \dot{w}_1 + \dot{w}_2$
$\dot{w}_4 = \dot{w}_3 \cos(w_3)$
$\dot{w}_5 = w_4\, \dot{w}_0 + w_0\, \dot{w}_4$
$\dot{y} = \dot{w}_5$

Note that in this case, the computation of the derivative is the same as the forward primal trace.

Reverse mode In the reverse mode, the computation of derivatives is not propagated concurrently with the evaluation of the function. Instead, the derivatives are computed in a backward manner, from the output to the input. This is done using the following recursive relation:

$$\frac{\partial y_j}{\partial w_i} = \frac{\partial y_j}{\partial w_{i+1}} \frac{\partial w_{i+1}}{\partial w_i}. \tag{3.19}$$

In this equation, $\frac{\partial y_i}{\partial w_{i+1}}$ represents the sensitivity of the output y_j with respect to changes in w_{i+1}. Introducing the adjoint variable $\overline{w}_{i+1}$ defined as

$$\overline{w}_i = \frac{\partial y_j}{\partial w_i}, \tag{3.20}$$

the Eq. (3.19) can be rewritten as follows:

$$\overline{w}_i = \overline{w}_{i+1} \frac{\partial w_{i+1}}{\partial w_i}. \tag{3.21}$$

More in detail, the reverse mode employs a two-phase process. In the first phase, the original function is traversed forward, storing intermediate variables w_i and tracking their dependencies through a computational graph. In the second phase, the derivatives are computed by propagating the adjoint variables $\overline{w}_i$ in reverse, from the outputs to the inputs. Considering the example in Eq. (3.15), with the reverse mode we can compute the contribution $\overline{w}_i$, which represents how much a change in each variable w_i affects the output y, as delineated in Table 3.3.

Table 3.3 Reverse derivative trace of the function in Eq. (3.15).

Reverse Derivative Trace
$\overline{x}_1 = \overline{w}_{-1}$
$\overline{x}_2 = \overline{w}_0$
$\overline{w}_{-1} = \overline{w}_1 \frac{\partial w_1}{\partial w_{-1}} = 2\,\overline{w}_1\, w_2$
$\overline{w}_0 = \overline{w}_0 + \overline{w}_2 \frac{\partial w_2}{\partial w_0} = \overline{w}_0 + 2\,\overline{w}_2\, w_2$
$\overline{w}_1 = \overline{w}_3 \frac{\partial w_3}{\partial w_1} = \overline{w}_3$
$\overline{w}_2 = \overline{w}_3 \frac{\partial w_3}{\partial w_2} = \overline{w}_3$
$\overline{w}_3 = \overline{w}_4 \frac{\partial w_4}{\partial w_3} = \overline{w}_4 \cos(w_3)$
$\overline{w}_0 = \overline{w}_5 \frac{\partial w_5}{\partial w_0} = \overline{w}_5\, w_4$
$\overline{w}_4 = \overline{w}_5 \frac{\partial w_5}{\partial w_4} = \overline{w}_5\, w_0$
$\overline{w_5} = \overline{y}$

The reverse mode offers a significant advantage in terms of computational efficiency, particularly for functions with a large number of inputs. For example, in the case of $f : \mathbb{R}^n \to \mathbb{R}$, only one application of the reverse mode is sufficient

to compute the full gradient $\nabla f = (\frac{\partial y}{\partial x_1}, \ldots, \frac{\partial y}{\partial x_n})$, compared to the n steps required by the forward mode to achieve the same result.

3.2.2 Efficient Gradient Computation Method

Drawing inspiration from the principle behind the reverse mode of Automatic Differentiation, a new method for gradient computation has been developed. Before presenting this method, the notation used in its formulation is introduced: $\partial_k^u g \doteq \frac{\partial g}{\partial \mathfrak{u}_k}(\hat{x}_k, \mathfrak{u}_k)$, where g is a generic function and $\frac{\partial g}{\partial \mathfrak{u}_k}$ is its Jacobian (or gradient) with respect to $\mathfrak{u}_k$ computed at $(\hat{x}_k, \mathfrak{u}_k)$; $\partial_k^x g \doteq \frac{\partial g}{\partial x_k}(\hat{x}_k, \mathfrak{u}_k)$, where $\frac{\partial g}{\partial x_k}$ is the Jacobian (or gradient) of g with respect to x_k computed at $(\hat{x}_k, \mathfrak{u}_k)$.

It should be noted that the proposed method is general. It works for any differentiable objective function J, such that the gradients $\partial_k^u J$ and $\partial_k^x J$ depend only on the samples x_k and u_k at step k. Note that this condition is satisfied by the objective function in (2.5) and (2.8), in which case the gradients are given by $\partial_k^u J = 2u_k^\top R^\top + \sum_j \partial_k^u \rho_j$, $\partial_k^x J = -2\tilde{x}_k^\top Q^\top + \sum_j \partial_k^x \rho_j$ $(k \leq T)$, $\partial_k^x J = -2\tilde{x}_k^\top R^\top + \sum_j \partial_k^x \rho_j$ $(k = T+1)$.

A proposition is now presented, showing how the gradient of the objective function J can be computed iteratively. Note that the quantity Γ used below is a full-rank matrix of ones and zeros, introduced in Equation (2.6) for the parametrization of the input $\mathbf{u}$. For more detail see Section 2.3.

Proposition 1 (Efficient gradient computation)**.** *The gradient of J with respect to c is given by $\nabla_c J = \nabla_{\mathbf{u}} J\Gamma$, where $\nabla_{\mathbf{u}} J = [\nabla_1 J, \ldots, \nabla_T J]$ and $\nabla_k J$ is computed iteratively for $k = T : -1 : 1$ according to*

$$\begin{aligned} g_k &= \partial_{k+1}^x J + g_{k+1}\, \partial_{k+1}^x f \\ \nabla_k J &= \partial_k^u J + g_k\, \partial_k^u f \end{aligned} \tag{3.22}$$

assuming $g_{T+1} = 0$.

Proof. For simplicity of notation, in this proof, we set $\hat{x}_k \to x_k$ and $\mathfrak{u}_k \to u_k$.

Consider that the gradient of J with respect to (w.r.t.) c is given by

$$\nabla_c J = \nabla_{\mathbf{u}} J \frac{\partial \mathbf{u}}{\partial c} = \nabla_{\mathbf{u}} J\Gamma$$

where $\nabla_{\mathbf{u}} J$ is the gradient of J w.r.t. $\mathbf{u} \doteq (u_1, \ldots, u_T)$. By definition, this gradient is given by

$$\nabla_{\mathbf{u}} J = [\nabla_1 J, \ldots, \nabla_T J] \in \mathbb{R}^{1 \times T n_u}$$

where

$$\nabla_k J = \frac{\partial J}{\partial u_k} + \sum_{i=k+1}^{T} \frac{\partial J}{\partial x_i} \frac{\partial x_i}{\partial u_k} \in \mathbb{R}^{1 \times n_u}$$
$$\frac{\partial J}{\partial u_k} \in \mathbb{R}^{1 \times n_u}, \; \frac{\partial J}{\partial x_i} \in \mathbb{R}^{1 \times n_x}, \; \frac{\partial x_i}{\partial u_k} \in \mathbb{R}^{n_x \times n_u}.$$

Let us start with $k = T$:

$$\begin{aligned} \nabla_T J &= \frac{\partial J}{\partial u_T} + \frac{\partial J}{\partial x_{T+1}} \frac{\partial x_{T+1}}{\partial u_T} \\ &= \partial_T^u J + \partial_{T+1}^x J \partial_T^u f \\ &= \partial_T^u J + g_T \partial_T^u f \end{aligned}$$

where

$$\begin{aligned} g_T &= \partial_{T+1}^x J = \partial_{T+1}^x J + g_{T+1} \, \partial_{T+1}^x f \\ g_{T+1} &= 0. \end{aligned}$$

For $k = T - 1$, we have that

$$\begin{aligned} &\nabla_{T-1} J = \\ &= \frac{\partial J}{\partial u_{T-1}} + \frac{\partial J}{\partial x_T} \frac{\partial x_T}{\partial u_{T-1}} + \frac{\partial J}{\partial x_{T+1}} \frac{\partial x_{T+1}}{\partial x_T} \frac{\partial x_T}{\partial u_{T-1}} \\ &= \partial_{T-1}^u J + \partial_T^x J \, \partial_{T-1}^u f + \partial_{T+1}^x J \, \partial_T^x f \, \partial_{T-1}^u f \\ &= \partial_{T-1}^u J + (\partial_T^x J + \partial_{T+1}^x J \, \partial_T^x f) \, \partial_{T-1}^u f \\ &= \partial_{T-1}^u J + g_{T-1} \, \partial_{T-1}^u f \end{aligned}$$

where

$$\begin{aligned} g_{T-1} &= \partial_T^x J + \partial_{T+1}^x J \, \partial_T^x f \\ &= \partial_T^x J + g_T \, \partial_T^x f. \end{aligned}$$

For $k = T-2$, we have that

$$\begin{aligned}
&\nabla_{T-2} J = \\
&= \partial^u_{T-2} J + \partial^x_{T-1} J \, \partial^u_{T-2} f + \partial^x_T J \, \partial^x_{T-1} f \, \partial^u_{T-2} f \\
&+ \partial^x_{T+1} J \, \partial^x_T f \, \partial^x_{T-1} f \, \partial^u_{T-2} f = \partial^u_{T-2} J \\
&+ (\partial^x_{T-1} J + (\partial^x_T J + \partial^x_{T+1} J \, \partial^x_T f) \partial^x_{T-1} f) \, \partial^u_{T-2} f \\
&= \partial^u_{T-2} J + g_{T-2} \partial^u_{T-2} f
\end{aligned}$$

where

$$\begin{aligned}
g_{T-2} &= \partial^x_{T-1} J + (\partial^x_T J + \partial^x_{T+1} J \, \partial^x_T f) \partial^x_{T-1} f \\
&= \partial^x_{T-1} J + g_{T-1} \, \partial^x_{T-1} f.
\end{aligned}$$

For any k, it holds that

$$\begin{aligned}
g_k &= \partial^x_{k+1} J + g_{k+1} \, \partial^x_{k+1} f \\
\nabla_k J &= \partial^u_k J + g_k \, \partial^u_k f.
\end{aligned}$$

These equations allow us to compute $\nabla_{\mathbf{u}} J$. Finally, $\nabla_c J$ is obtained by multiplication of $\nabla_{\mathbf{u}} J$ by Γ and the claim follows. □

Based on this result, we propose an algorithm for iterative gradient computation, consisting of two main phases: (i) forward computation of the state sequence; (ii) backward gradient computation.

Algorithm 2 Iterative gradient computation

Input: initial state x_1; input sequence $\mathbf{u} \doteq (\mathrm{u}_1, \ldots, \mathrm{u}_T)$.
Output: $\nabla_c J$.

1: *Forward state sequence computation.* Starting from the initial condition x_1, iterate the state equation for $k = 1 : T$:

$$\hat{x}_{k+1} = f(\hat{x}_k, \mathrm{u}_k).$$

2: *Backward gradient computation.* Starting from the final condition $g_{T+1} = 0$, iterate equations (3.22) for $k = T : -1 : 1$ to obtain $\nabla_{\mathbf{u}} J$, and compute $\nabla_c J = \nabla_{\mathbf{u}} J \Gamma$.

Remark. *The computational complexity of Algorithm 4 is linear in T.*

Remark. *Algorithm 4 shares similarities with the reverse mode method, exhibiting also a comparable computational complexity. However, it is important*

to clarify that the aim of developing this novel algorithm is not based on making comparisons, either computationally or in terms of accuracy, with it or other state-of-the-art gradient computation methods described in Section 3.2.1.

3.3 Nonlinear Sparse Identification Method

Another key component of the dimensionality reduction approach developed in this thesis is the use of the sparse identification method presented in [15].

In general, the process of system identification focuses on the development of mathematical models that accurately reflect the behaviour of dynamic systems, using experimental data and a "weak" prior physical knowledge [120]. An example of such a dynamic system is represented in Figure 3.2. It is driven by a manipulated input signal $u(t)$, produces an output $y(t)$ and is subject to a non-manipulated disturbance $d(t)$.

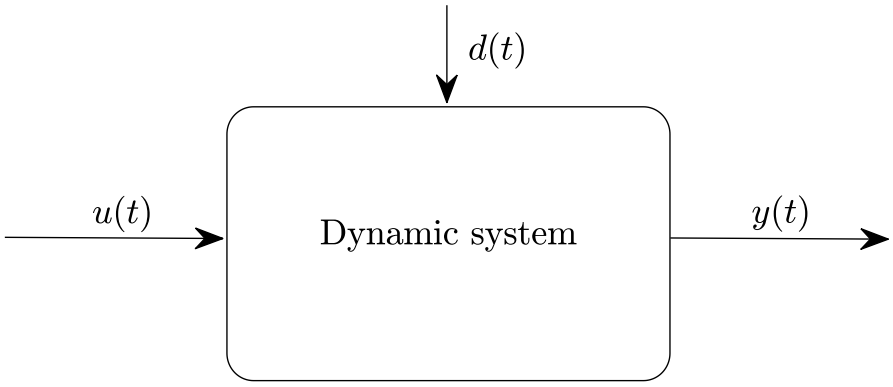

Fig. 3.2 Example of dynamic system.

In the context of sparse identification methods, sparse approximations are employed to construct these mathematical models. More in detail, these techniques involve approximating a function using only a few properly selected basis functions from a large set. This means representing the function as a linear combination of many basis functions, but with sparse coefficients, i.e., most coefficients are zero and only a few are non-zero. Sparse approxima-

tion/identification methods have been successfully introduced in the field of automatic control [121–123, 15], addressing several problems such as regularization, basis function selection, regressor selection, nonlinear control design, predictive control and "fast" online control implementation. The sparsity of a vector is usually measured by the ℓ_0 quasi-norm, which counts its non-zero elements. Sparse identification involves searching for a coefficient vector with a "small" ℓ_0 quasi-norm to form linear combinations of basis functions.

Consider a generic nonlinear function f^o defined by

$$y = f^o(w) \tag{3.23}$$

where $w \in \mathcal{W} \subset \mathbb{R}^n$ is the vector of independent variables, usually called regressor, $y \in \mathbb{R}$ and $f^o : \mathbb{R}^n \to \mathbb{R}$. It should be noted that the dimension n of the vector w is related to the type of system under consideration. Suppose that f^o is not known, but a set of corrupted data $D = \{\tilde{w}_k, \tilde{y}_k\}_{k=1}^L$ is available, described by

$$\tilde{y}_k = f^o(\tilde{w}_k) + d_k, \qquad k = 1, 2, \ldots, L, \tag{3.24}$$

where d_k accounts for noises, disturbances or errors. Assume that the noise sequence is unknown but bounded:

$$\|d\|_q \leq \mu, \tag{3.25}$$

where $d = (d_1, \ldots, d_L)$ and $\|\cdot\|_q$ is the vector ℓ_q norm (with $q = 2, \infty$).

Define the parametrized function:

$$\hat{f}(w) = \sum_{i=1}^{N} a_i \psi_i(w) = \psi(w) a \tag{3.26}$$

where $\psi(w) = [\psi_1(w), \psi_2(w), \ldots, \psi_N(w)]$, $\psi_i : \mathcal{W} \to \mathbb{R}$ are Lipschtiz continuous basis functions, and $a = (a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$ is a coefficients vector. The selection of the basis functions ψ_i is a crucial step in the identification process [124, 125]. In several practical scenarios, the basis functions are known a *priori* to belong to some 'large' set of functions (see, e.g., the example in [126]). In these cases, the sparse approximation algorithms described below can be employed to select, within this 'large' set, the functions that are important

for providing an accurate description of the system under investigation. Conversely, when the basis functions are not known a priori, their selection can be performed by considering the numerous options available in the literature (e.g. gaussian, sigmoidal, wavelet, polynomial, trigonometric). To obtain a sparse approximation from the dataset D, it is necessary to find a coefficient vector a that satisfies the following conditions: i) a is sparse; ii) the identification error $e(\hat{f}) \doteq \|f^o - \hat{f}\|_p$, where $\|\cdot\|_p$ is a function L_p norm, is "small".

Under the assumption (3.25), a coefficient vector a that verifies the above conditions can be found by solving the following optimization problem:

$$a^0 = \arg\min_{a \in \mathbb{R}^N} \|a\|_0 \tag{3.27a}$$

$$\text{subject to:} \quad \|\tilde{y} - \Psi a\|_q \leq \mu, \tag{3.27b}$$

where

$$\begin{aligned} \tilde{y} &\doteq (\tilde{y}_1, \ldots, \tilde{y}_L), \\ \Psi &\doteq \begin{bmatrix} \psi_1(\tilde{w}_1) & \ldots & \psi_N(\tilde{w}_1) \\ \vdots & \ddots & \vdots \\ \psi_1(\tilde{w}_L) & \ldots & \psi_N(\tilde{w}_L) \end{bmatrix} \\ &\doteq \begin{bmatrix} \psi_1(\tilde{w}) & \ldots & \psi_N(\tilde{w}) \end{bmatrix} \end{aligned}$$

$\psi_1(\tilde{w}) \doteq (\psi_1(\tilde{w}_1), \ldots, \psi_1(\tilde{w}_L)) \in \mathbb{R}^L$, and $\|a\|_0$ is the ℓ_0 quasi-norm of a, defined as the number of nonzero components of a. Note that minimizing this norm promotes sparsity, aiming for a solution with the fewest nonzero elements. Additionally, the constraint $\|\tilde{y} - \Psi a\|_q \leq \mu$ ensures that the identified coefficient vector a is consistent with both the measured data (3.24) and the prior assumption (3.25).

However, this optimization problem cannot be easily solved, since the ℓ_0 quasi-norm is a non-convex function and its minimization is an NP-hard problem. To overcome this issue, three main strategies are typically used: convex relaxation [127–129], greedy algorithms [130] and hybrid algorithms [131]. Convex relaxation involves minimizing a suitable convex function such as the ℓ_1 norm instead of the ℓ_0 quasi-norm. In greedy algorithms, sparse solutions are obtained iteratively. Hybrid algorithms combine ℓ_1 norm minimization and

iterative procedures to find a sparse solution. An interesting property of all these approaches is that, under certain conditions, they can achieve sparsity equivalent to the ℓ_0 quasi-norm, i.e., they identify solutions with the fewest non-zero coefficients [130, 128, 132]. Building on these advantages, the sparse identification method [15], used in this thesis, combines convex relaxation and greedy algorithms to obtain a combined ℓ_1-relaxed greedy algorithm, which is described in detail below.

3.3.1 ℓ_1-relaxed-greedy Identification Method

Without loss of generality, assume that the columns of Ψ, introduced in Eq. (3.27), are normalized: $\|\psi_i(\tilde{w})\|_2 = 1,\ i = 1, 2, \ldots, N$. Define the following quantity:

$$\xi(a) \doteq \frac{|\delta(a)|_1 + |\delta(a)|_{K_0}}{\underline{\sigma}^2(\Psi)}, \tag{3.29}$$

where $K_0 \doteq 2\|a\|_0$, $\underline{\sigma}(\Psi)$ is the minimum nonzero singular value of Ψ and

$$\begin{aligned} \delta(a) &\doteq \tilde{y} - \Psi a \\ |\upsilon|_K &\doteq \sqrt{\sum_{i \in I_K} (\upsilon^T \psi_i(\tilde{w}))^2}, \end{aligned} \tag{3.30}$$

being I_K the set of the K largest inner products $|\upsilon^T \psi_i(\tilde{w})|$, with υ a generic function. Moreover, let $\mathbf{card}(\cdot)$ denote the set cardinality.

Algorithm 3 describes the combined ℓ_1-relaxed-greedy algorithm, developed in [15]. Note that this approach relies completely on convex optimization and yields solutions with greater sparsity compared to conventional algorithms based on simple ℓ_1-norm minimization.

The resulting sparse approximation is given by:

$$f^*(w) = \sum_{i=1}^{N} a_i^* \psi_i(w). \tag{3.33}$$

where a_i^* are the coefficients of the vector a^* derived by Algorithm 3.

The rationale behind the algorithm can be explained as follows. In step 1, an optimization problem similar to (3.27) is solved, where the ℓ_0 quasi-norm

Algorithm 3 ℓ_1-relaxed-greedy algorithm

Input: $\tilde{y}$; Ψ.
Output: a^*.

1: Solve the optimization problem:

$$a^1 = \arg\min_{a\in\mathbb{R}^N} \|a\|_1 \tag{3.31a}$$

$$\text{subject to:} \quad \|\tilde{y} - \Psi a\| \leq \mu, \tag{3.31b}$$

where μ can be chosen as a positive number slightly larger than $\mu^{\min}$, i.e. the minimum value for which the problem is feasible. Provided that $\mu > \mu^{\min}$, the value of μ can be tuned to suitably manage the trade-off between accuracy and sparsity.

2: Let $r(a^1) \doteq \{i_1, \ldots, i_j : \xi(a^1) > |a^1_{i_1}| \leq \cdots \leq |a^1_{i_j}|\}$ and let $r_\lambda(a^1)$ denote the subset of $r(a^1)$ with indeces in λ. Compute the coefficient vector a^* as follows:

$$\begin{aligned}
&\text{for } k = 1 : \mathbf{card}(r_\lambda(a^1)) \\
&\quad c^k = \underset{a\in R^N}{\operatorname{argmin}} ||y - \Phi a||_q \\
&\quad \text{subject to } \; a_i = 0, \; \forall i \in r_\lambda(a^1) \\
&\qquad\qquad \lambda = \{k, ..., \operatorname{card}(r_\lambda(a^1))\} \\
&\quad \text{if } ||y - \Phi c^k||_q \leq \mu \\
&\qquad a^* = c^k \\
&\qquad \text{break} \\
&\quad \text{end} \\
&\text{end}
\end{aligned} \tag{3.32}$$

is replaced by the ℓ_1 norm. The ℓ_1 norm represents the convex envelope of the ℓ_0 quasi-norm, and its minimization produces a sparse vector a^1 [127–129]. However, there is no guarantee that all non-zero elements of a^1 are necessary to satisfy $\|\tilde{y} - \Psi a^1\|_q \leq \mu$. The step 2 addresses this limitation. In this step, only the elements of a^1 larger than $\xi(a^1)$ are kept, where the threshold $\xi(a^1)$ is introduced to discriminate between "important" and "less important" vector components [123]. The remaining elements are sorted by decreasing amplitude and progressively included to form the vector a^*. The algorithm stops when $\|\tilde{y} - \Psi c^k\|_q \leq \mu$. This two-step process refines the initial sparse solution a^1 by further reducing the number of non-zero elements, obtaining the final solution a^*.

3.4 Gradient computation and Sparse Nonlinear Identification technique for Dimensionality Reduction in NMPC

The novel dimensionality reduction method, developed in this thesis, is described here. It is applied to the single shooting NMPC with OCP (2.8), defined in Section 2.3, which is recalled below for the sake of clarity:

$$
\begin{aligned}
c^* &= \arg\min_{c} J(\Gamma c) \\
\text{subject to:} & \\
\hat{x}_1 &= f(x_t, u_t) \\
\hat{x}_{k+1} &= f(\hat{x}_k, S_k^u \Gamma c),\ k = 1:T \\
c &\in \mathcal{C}_c,
\end{aligned}
$$

where $x_t \in \mathbb{R}^{n_x}$ is the system state, $u_t \in U_c \subset \mathbb{R}^{n_u}$ is the command input, with U_c being a bounded set representing the admissible control space, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the dynamical model, T is the prediction horizon, $\Gamma \in \mathbb{R}^{Tn_u \times n_c}$ is a full-rank matrix with $n_c < Tn_u$, S_k^u is a selection matrix such that $u_k = S_k^u \Gamma c$, and $c \in \mathbb{R}^{n_c}$ is the parametrized command input. This sequence is subject to input saturations, denoted by $c \in \mathcal{C}_c \in \mathbb{R}^{n_c} \subset \mathbb{R}^{Tn_u}$, where $\mathcal{C}_c$ is a subset of U_c defined as $\mathcal{C}_c = \prod_j \mathcal{C}_{c_j}$, where $\prod_j$ denotes the Cartesian product, and $\mathcal{C}_{c_j} = [\underline{c}_j, \overline{c}_j]$ for $j = 1, \ldots, n_c$.

With the specific NMPC formulation established, we can now outline the proposed method for the dimensionality reduction. It is based on the strategies described in Setions 3.2 and 3.3, i.e., gradient computation and nonlinear identification. The way in which they come into play to mitigate the curse of dimensionality is illustrated below. To begin with, the steps that constitute this method are briefly defined. Afterwards, each step is described in detail.

The developed dimensionality reduction method consists in the following key steps:

1. *Offline data generation.* A dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{c}_l\}_{l=1}^{M}$ is generated, where:

- $\tilde{w}_l \doteq (\tilde{x}_l, \tilde{\mathfrak{r}}_l)$ is the regressor composed by a sample $\tilde{x}_l \in \mathbb{R}^{n_x}$ of the state and a sample $\tilde{\mathfrak{r}}_l = (\tilde{\mathfrak{r}}_{l1}, \ldots, \tilde{\mathfrak{r}}_{lT}) \in \mathbb{R}^{n_x \times T}$ of the reference sequence. These samples should be chosen to explore the state and reference domains of interest. For more details about this choice, see Section 3.4.1.
- $\tilde{c}_l$ is the command value corresponding to $\tilde{w}_l$, computed solving the optimization problem (2.8).

Note that the tilde symbol is used to indicate the collected data.

2. *Offline selection of the relevant decision variables.* The following selection matrices are defined: i) $\Gamma_r \in \mathbb{R}^{n_c \times n_{cr}}$ selects the "relevant" components of the decision vector c in (2.8); ii) $\Gamma_a \in \mathbb{R}^{n_c \times n_{ca}}$ selects the remaining components of c. These are matrices with elements 0 or 1, such that $c = \Gamma_r c_r + \Gamma_a c_a$, where $c_r \in \mathbb{R}^{n_{cr}}$ is the vector of the "relevant" components and $c_a \in \mathbb{R}^{n_{ca}}$ is the vector containing the other components, with $n_{cr}, n_{ca} \leq n_c$. In Section 3.4.2, we propose a systematic method for selecting the "relevant" components of c. This method relies on the objective function gradient, making use of the numerically efficient algorithm for gradient computation presented in Section 3.2.2.

3. *Offline approximation of the NMPC control law.* According to the formulation of Section 2.3, the optimal NMPC command is $u_{t+1} = S_1^u \Gamma c^*$, where c^* is the solution of the optimization problem (2.8). This solution can be seen as a static nonlinear function ϕ_o of the current state x_t and the reference sequence $\mathbf{r}_t = (r_{t+2}, \ldots, r_{t+T+1})$: $c^* = \phi_o(x_t, \mathbf{r}_t)$. An approximation ϕ_s of ϕ_o is derived from the dataset $\mathcal{D}$, using the Nonlinear Sparse Identification method described in Section 3.3.1.

4. *Online optimization on the reduced-dimensionality OCP.* At each time t:

   - the approximation ϕ_s is used to obtain the approximated solution $\hat{c} = \Gamma_r \hat{c}_r + \Gamma_a \hat{c}_a$;
   - the OCP is performed only on the relevant variables c_r, using $\hat{c}_r$ as a warm start, within the *reduced search domain* $\mathcal{C}_c^d$ (see Section 3.4.4 for more details);
   - the non-relevant variables c_a are kept fixed to the values $\hat{c}_a$.

Further details regarding these steps are provided below.

3.4.1 Offline data generation

A set of state data $\tilde{x}_l$ and reference signals $\tilde{\mathfrak{r}}_l = (\tilde{\mathfrak{r}}_{l1}, \ldots, \tilde{\mathfrak{r}}_{lT})$, where $l = 1 \ldots, M$, is generated to explore the state and reference domains using Latin Hypercube Sampling (LHS) [133].

Latin Hypercube Sampling. It is a statistical technique employed for the generation of nearly random samples of parameter values from a multidimensional distribution. The concept of LHS originates from the theory of Latin squares. A Latin square is a grid where each row and column contains a unique sample. LHS generalizes this notion to higher dimensions, ensuring that each sample point occupies a unique position within its corresponding axis-aligned hyperplane. In the context of N variables, LHS operates by dividing the range of each variable into M equiprobable intervals. Subsequently, M sample points are strategically placed to satisfy the Latin hypercube property. This characteristic necessitates an equal number of divisions, M, for all variables, offering a key advantage: the number of samples remains independent of the dataset size. Additionally, LHS allows a better coverage of the domain of interest with respect to random sampling, where new sample points are generated independently without considering previously chosen points.

The generated samples $\tilde{x}_l$ and $\tilde{\mathfrak{r}}_l$ are combined into the regressor $\tilde{w}_l$. For each $\tilde{w}_l$, the corresponding optimal solution of the optimization problem (2.8) is computed: $\tilde{c}_l = \phi(\tilde{w}_l)$, giving a set of control data $\{\tilde{c}_l\}_{l=1}^{M}$. Combining the sets of the regressor and the control data, the resulting dataset is $\mathcal{D} = \{\tilde{w}_l, \tilde{c}_l\}_{l=1}^{M}$.

3.4.2 Offline selection of the relevant decision variables

The most relevant decision variables, i.e., those that exert the most influence on the OCP, can be selected by estimating the gradient of the objective function at all points of the dataset $\mathcal{D}$. In detail, the gradient of the cost function J with respect to the decision variable c is defined as $\nabla_c J \in \mathbb{R}^{n_c}$. To determine the most important components of the vector c, we define the mean-square

gradient $G \in \mathbb{R}^{n_c}$ as follows:

$$G = \frac{\sum_{i=1}^{M} (\nabla_c J_i)^2}{M}, \tag{3.35}$$

where M is the number of points in the dataset, $\nabla_c J_i$ denotes the gradient evaluated at the ith point and $(\cdot)^2$ is the component-wise square. Note that, the use of the mean-square gradient enhances the discrimination between the contributions of the most and least relevant decision variables for the OCP. Let ζ be an "importance" threshold, to be compared with each element of the vector G_j, with $j = 1 : n_c$. A component c_j of the decision vector c is considered significant if

$$G_j \geq \zeta. \tag{3.36}$$

The choice of the threshold ζ can based on practical considerations as well as on the decay of the gradient values, see for example Figure 5.5 in Chapter 5. At the end of this process, two vectors are obtained: the vector $c_r \in \mathbb{R}^{n_{cr}}$, with $n_{cr} < n_c$, which contains the relevant components, and the vector $c_a \in \mathbb{R}^{n_{ca}}$, with $n_{ca} < n_c$, of non-relevant components. In order to link these two vectors to the decision variables c, two selection matrices $\Gamma_r \in \mathbb{R}^{n_c \times n_{cr}}$ and $\Gamma_s \in \mathbb{R}^{n_c \times n_{ca}}$ are constructed such that:

$$c = \Gamma_r c_r + \Gamma_a c_a. \tag{3.37}$$

Note that these two matrices contain only zeros and ones, with the ones placed in the appropriate positions and corresponding to the components present on the vectors c_r and c_a. This implies that once the vector c_r and c_a are defined, the two matrices are consequently determined. To elucidate this concept, an example is provided below.

Example. Suppose to consider the same c of the example in Section 2.3, that is, $c = [c_1, c_2, c_3]$. After the computation of G and the appropriate choice of the parameter ζ, it is found that the only significant component is c_2. This implies that $c_r = [c_2]$ and $c_s = [c_1, c_3]^T$. Consequently:

$$c = \Gamma_r c_r + \Gamma_a c_a = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} c_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_3 \end{bmatrix}, \tag{3.38}$$

with $\Gamma_r \in \mathbb{R}^{3 \times 1}$ and $\Gamma_a \in \mathbb{R}^{3 \times 2}$.

The general procedure is formalized in Algorithm 4 below.

Algorithm 4 Computation of the most significant components

Input: regressor $\tilde{w}$; parameter ζ.
Output: c_r, c_s, Γ_r, Γ_s.

1: For $i = 1 : M$, compute the mean-squared gradient G using (3.35) for each component j of c.
2: Fix the threshold parameter ζ.
3: Set $c_r = [\,]$ and $c_s = [\,]$.
4: For each c_j, with $j = 1 : n_c$:
5: **if** $G_j > \zeta$ **then**
6: $c_r = [c_r; c_j]$
7: **else**
8: $c_s = [c_s; c_j]$
9: **end if**.
10: Construct the selective matrices Γ_r and Γ_s in order to satisfy (3.37).

To obtain the matrix G in equation (3.35), it is necessary to find the gradient of c for each element of the dataset. In this regard, the new efficient method for gradient computation described in Section 3.2.2, has been used.

3.4.3 Offline approximation of the NMPC control law

On the basis of the dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{M}$ the Nonlinear Sparse Identification method [15], described in Section 3.3.1, is used to compute the approximation ϕ_s of the NMPC control law. In the following, a detailed description is outlined.

According to the formulation of the OCP (2.8), the optimal NMPC solution c^* is a static nonlinear function ϕ_o of the current state x_t and the reference sequence $\mathbf{r}_t = (r_{t+2}, \ldots, r_{t+T+1})$. The NMPC solution c^* at time t is thus given by

$$c^* = \phi_o(w_t) \tag{3.39}$$

where $w_t \doteq (x_t, \mathbf{r}_t)$ is the *regressor*. For simplicity, here we consider $n_c = 1$. The generalization to the case of $n_c > 1$ is trivial and can be accomplished by applying the Nonlinear Sparse Identification method to each component of c^*. In general, due to the complexity of the OCP (2.8), writing the function ϕ_o in closed-form is not possible. To overcome this issue, an approximation of ϕ_o is derived, based on the off-line computation of its values $\tilde{c}_l$ at a given number

of points $\tilde{w}_l$, where $l = 1, \ldots, M$. From the resulting dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{c}_l\}_{l=1}^{M}$ a *sparse approximation* ϕ_s of the NMPC function ϕ_o is derived using the sparse approximation method described in Section 3.3. In particular, the NMPC control law ϕ_o is approximated using the parametrized function

$$\phi_s(\cdot) = \sum_{i=1}^{N} a_i^* \psi_i(\cdot) \tag{3.40}$$

where $\psi_i : \mathbb{R}^{(T+1)n_x} \to \mathbb{R}$ are Lipschitz continuous basis functions and $a_i^* \in \mathbb{R}$ are coefficients to identify. The basis functions are known *a priori* to belong to a large set of functions [126], and the coefficients a_i^* are used to select the most relevant functions for approximating ϕ_o accurately. In order to determine the "suitable" selection of these coefficients, the Algorithm 3 is used. For the sake of clarity, we recall the OCP (3.31) below, with minor adjustments made to its constraints (3.31b):

$$a^* = \arg\min_{a \in \mathbb{R}^N} \|a\|_1 \tag{3.41a}$$

$$\text{subject to:} \quad \|\tilde{c} - \Psi a\| \leq \mu, \tag{3.41b}$$

where

$$\tilde{c} \doteq (\tilde{c}_1, \ldots, \tilde{c}_M)$$

$$\Psi \doteq \begin{bmatrix} \psi_1(\tilde{w}_1) & \ldots & \psi_N(\tilde{w}_1) \\ \vdots & \ddots & \vdots \\ \psi_1(\tilde{w}_M) & \ldots & \psi_N(\tilde{w}_M) \end{bmatrix}.$$

As typical in sparse techniques, the term $\|\hat{a}\|_1$, i.e., the ℓ_1 norm of the parameter vector, is used to promote sparsity of this vector, and thus to reduce the complexity of the approximating function (3.40). The parameter $\mu \geq 0$ is a bound on the approximation error evaluated on the data. The μ parameter should be chosen larger than the minimum value for which (3.41) is feasible. Increasing μ leads to increased sparsity of the parameter vector, allowing us to manage the trade-off between approximation accuracy and complexity. To avoid using a regressor w_t with a too large number of components, instead of considering the entire reference sequence $\tilde{\mathfrak{r}}_l = (\tilde{\mathfrak{r}}_{l1}, \ldots, \tilde{\mathfrak{r}}_{lT})$, a subsequence

between 1 and T can be taken. Clearly, this operation introduces approximation errors, that can be dealt with by suitably choosing the parameter μ.

Remark. *To summarize, the sources of approximation, represented by the sequence d in Eq. (3.25), are as follows: (i) the approximation error given by ϕ_s ; (ii) the use of a subsequence of the reference $\mathbf{r}_t$. Therefore, in selecting μ, it is essential to consider both factors. It is also worth noting that, since we are identifying a control law that can be computed exactly, there are no additional disturbances or noise.*

3.4.4 Online optimization on the reduced-dimensionality OCP

At each time t, given the regressor w_t, the approximated solution

$$\hat{c} = \phi_s(w_t) = \Gamma_r \hat{c}_r + \Gamma_a \hat{c}_a \tag{3.43}$$

is computed online. Then, the optimization is executed only on the most relevant variables c_r, using $\hat{c}_r$ as a warm start, while leaving the remaining ones fixed at the values $\hat{c}_a$. This means performing the optimization process within a smaller search domain, called *reduced dimensionality search domain* $\mathcal{C}_c^d$. Let us define $\widehat{\mathcal{J}} = \{1, \ldots, n_{cr}\}$ as the set of indices $\widehat{j}$ that corresponds to the components $c_{\widehat{j}} \in c_r$. The reduced dimensionality search domain can be defined as follows:

$$\mathcal{C}_c^d = \prod_{\widehat{j}} \mathcal{C}_{c_j} = [\underline{c}_{\widehat{j}}, \overline{c}_{\widehat{j}}], \text{ for } \widehat{j} \in \widehat{\mathcal{J}}. \tag{3.44}$$

It should also be noted that the use of $\hat{c}_r$ provides a suitable starting point, which can reduce the time and computational resources required to reach an optimal solution.

The resulting OCP is the following:

$$c_r^* = \arg\min_{c_r} J(\Gamma c) \tag{3.45a}$$

subject to:

$$\hat{x}_1 = f(x_t, u_t) \tag{3.45b}$$

$$\hat{x}_{k+1} = f(\hat{x}_k, S_k^u \Gamma c),\ k = 1:T \tag{3.45c}$$

$$c_r \in \mathcal{C}_c^d \tag{3.45d}$$

$$c = \Gamma_r c_r + \Gamma_a \hat{c}_a. \tag{3.45e}$$

This approach streamlines the optimization process, focusing computational efforts on the most critical variables and thus enhancing the efficiency and effectiveness of the optimization algorithm. This allows a computational complexity reduction with respect to solving the OCP (2.8), as it will be shown in the following Section.

3.5 Computational Complexity Analysis

This section presents a theoretical analysis of the dimensionality reduction method, showing why it can be effective to reduce the computational complexity of nonlinear optimization algorithms.

Remark. *As a measure of the computational complexity, we consider the maximum number of operations required to execute a given algorithm, expressed using the Big O notation.*

As described in Section 3.4, our dimensionality reduction approach consists of the following main operations: (i) offline selection of the relevant decision variables (Subsection 3.4.2); (ii) offline approximation of the NMPC control law (Subsection 3.4.3); (iii) online optimization on the reduced dimensionality OCP (Subsection 3.4.4).

The goal of this section is to quantify the impact of the dimensionality reduction on the computational complexity of the OCP. In general, the computational complexity of most state-of-the-art algorithms used to solve nonlinear optimization problems, like Newton or Quasi-Newton methods, is $O(n^p)$, where

n is the number of decision variables and $p \geq 1$ (see, e.g., [71]). A proposition is now presented, showing how our dimensionality reduction method can significantly decrease the computational complexity necessary to solve the NMPC optimization problem.

Proposition 2 (Dimensionality reduction)**.** *Consider the following optimization problems:*

(i) OCP (2.8)*;*

(ii) OCP (3.45)*.*

Suppose a nonlinear optimization algorithm with computational complexity $O(n^p)$ *is used to solve both problems. The computational complexity of the algorithm is reduced from* $O(n_c^p)$ *in the case (i) to* $O(n_{cr}^p)$ *in the case (ii), resulting in an improvement factor of* $(\frac{n_c}{n_{cr}})^p$*.*

Proof. Consider the dimensionality reduction method introduced in Section 3.4. This method reduces the number of decision variables in the OCP from n_c to n_{cr}, resulting in the formulation presented in (3.45). Consequently, the number of operations needed by a nonlinear optimization algorithm to solve this OCP is $O(n_{cr})^p$. In contrast, solving the original OCP (2.8) required a number of operations proportional to $O(n_c)^p$. Therefore, with our dimensionality reduction method, we obtain an improvement in computation complexity of

$$A \propto \left(\frac{n_c}{n_{cr}}\right)^p,$$

where A represents an *efficiency gain.* □

Remark. *The reduction in computational complexity is directly related to the ratio of the total number of decision variables* n_c *to the relevant variables* n_{cr}*. This relationship is strongly influenced by the parameter* p*: the higher is* p*, the more pronounced the reduction in computational complexity becomes.*

Remark. *The result established in Proposition 2 is general: it holds for any nonlinear optimization algorithm characterized by a computational complexity* $O(n^p)$*.*

To illustrate the impact of the dimensionality reduction method on the computational complexity of a nonlinear algorithm, consider the following example.

Example. Assume to have an optimization problem with $n_c = 20$ decision variables. Suppose that, applying the method proposed in Section 3.4.2, we obtain $n_{cr} = 4$ relevant decision variables. Furthermore, consider a quasi-Newton optimization method, whose computational complexity is proportional to $O((\textit{number of decision variables})^2)$. By reducing the number of variables from n_c to n_{cr}, we achieve an efficiency improvement factor of $A \propto (20/4)^2 = 25$.

3.6 Chapter summary

This chapter presented a new dimensionality reduction method aimed at enhancing the computational efficiency of NMPC, enabling its use even in real-time applications with a fast sampling rate. The method is essentially based on the combination of two strategies: i) Gradient Computation, and ii) Sparse Nonlinear Identification. On the one hand, the gradient computation is used to identify the directions of greatest variability within the search domain of the optimization problem and consequently find the most relevant decision variables. On the other hand, the Nonlinear Sparse Identification method is used to approximate the NMPC control law employing the ℓ_1-relaxed-greedy algorithm. This allows both to obtain a good warm start of the optimization algorithm and to implement the OCP only on the most relevant decision variables, leaving the others fixed at the value found from the approximation of the control law. The effectiveness of the proposed method in reducing computational time for solving OCPs was theoretically demonstrated through an analysis of its computational complexity.

Chapter 4

Side-length Reduction for Fast NMPC

In the field of nonlinear optimization, efficiently exploring the extensive search domain to identify global or local optima represents a computationally demanding and time-consuming task, often posing a bottleneck in practical applications. To mitigate this problem, a promising solution could be the reduction of the side-length of the search domain. Indeed, by optimally shrinking the search space, the optimization process can efficiently exclude non-promising regions and focus computational resources only on areas that are more likely to contain high-quality solutions. Implementing such side-length reductions can speed up the rate of convergence of nonlinear optimization algorithms, allowing to achieve satisfactory solutions with a reduced number of iterations and objective function evaluations. This significantly decreases the overall computational cost, enhancing the usability of optimization techniques in real-world scenarios, such as real-time control systems.

In the context of real-time NMPC, the fast computation of control actions in response to continuously changing system states is crucial. When the search domain is concentrated on more relevant regions, the NMPC can generate control inputs more quickly without compromising quality and robustness. This is particularly important in systems where delays in response time can lead to suboptimal performance or even dangerous failures, such as in autonomous vehicles, industrial processes, or energy distribution networks.

Recalling the concept already discussed in Chapter 3, let us suppose that s^d represents the overall volume of the search domain, where s represents its side length and d is the number of decision variables of the problem. The main contribution of this chapter is to present a novel approach that aims to restrict, in an optimal way, the side-length s of the search domain, thereby reducing its overall volume and significantly decreasing the computational time. Specifically, the approach employs the Set Membership (SM) approximation method, proposed in [14], to derive from data tight bounds on the optimal NMPC control law, reducing accordingly the search range within which the optimization process operates. It should be noted that the proposed method is applied to the NMPC developed in Chapter 3, allowing for a simultaneous reduction in both dimensionality and side-length of the search domain, leading to a significant decrease in its total volume. However, the versatility of the side-length reduction method extends beyond its combination with the dimensionality reduction technique. Indeed, it can be effectively applied to traditional NMPC approaches as well. In this regard, Section 4.2.4 details the steps required for its implementation with the single-shooting NMPC described in Section 2.3.

The effectiveness of this method in reducing computational time in nonlinear optimization algorithms is theoretically demonstrated through a computational analysis of the rate of convergence.

It is worth noting that, in the context of NMPC, we have not found any other significant works addressing the concept of side-length reduction of the search domain. For this reason, throughout this chapter, there will be no references to the state-of-the-art literature.

Outline. The chapter is organized as follows. Section 4.1 provides an overview of the Nonlinear Set Membership approximation. Section 4.2 describes the novel method for optimally reducing the side-length of the search space. The computational complexity of this method is analyzed theoretically in Section 4.3.

4.1 Nonlinear Set Membership Approximation

System identification aims to create an accurate mathematical model of a dynamic system when the governing laws are complex or insufficiently known. The typical approach involves considering the dynamic system as belonging to a finite set of functions called basis functions, and then estimating the parameters of these functions using measured data. However, choosing the right parametric family of functions can be time-consuming, and its impact on identification errors remains an open problem. Additionally, parameter estimation is usually achieved through a prediction error method that requires minimizing an error function. This optimization problem is challenging due to issues like "curse of dimensionality", in the case of fixed basis functions, and non-convexity, when using tuneable basis functions like neural networks or wavelets. Furthermore, providing a measure of the identification error presents additional challenges due to the probabilistic uncertainty introduced by noise in the measurements.

To address these problems, in the last three decades, there has been a growing interest and research in formulating the identification problem in the Set Membership (SM) framework. The main reason is that the SM identification allows us to quantify the uncertainty of the identified model in a deterministic manner without making assumptions on the structure of the unknown system. Instead, two basic assumptions are made: i) regularity of the system given by bounds on its gradient, and ii) noise boundedness. Through this method, we obtain an optimal estimate with minimal guaranteed identification error and tight uncertainty bounds, which provides a bounded identification uncertainty description by considering all possible models equally probable. This nonlinear SM approach does not require iterative minimization or optimization procedures, making it well-suited for adaptive identification and control implementation.

4.1.1 Set Membership Formulation

Consider a generic nonlinear function f^o defined by

$$y = f^o(w) \tag{4.1}$$

where $w \in \mathcal{W} \subset \mathbb{R}^n$ is the vector of independent variables, usually called regressor, $y \in \mathbb{R}$ and $f^o : \mathbb{R}^n \to \mathbb{R}$. Note that the dimension n of the vector w is related to the specific system under consideration. Suppose that the function f^o is unknown but a set of noise corrupted data

$$D = \{\tilde{w}_k, \tilde{y}_k\}_{k=1}^{L} \tag{4.2}$$

generated by (4.1) is available. Then

$$\tilde{y}_k = f^o(\tilde{w}_k) + d_k, k = 1, \ldots, L \tag{4.3}$$

where the term d_k accounts for the fact that y and w are not exactly known, due to possible noise, disturbances or errors affecting the system.

The aim of system identification methods is to derive an estimate $\hat{f}$ of f^o from the available measurements D, using a suitable identification algorithm ϕ. This algorithm should be chosen in order to minimize the identification error

$$e(\hat{f}) = \|f^o - \hat{f}\|_p, \tag{4.4}$$

where $\|\cdot\|_p$ is the functional L_p norm, defined as

$$\|f\|_p \equiv \|f(\cdot)\|_p \doteq \begin{cases} [\int_{\mathcal{W}} |f(w)|^p dw]^{1/p}, & p \in [1, \infty) \\ \text{ess sup}_{w \in \mathcal{W}} |f(w)|, & p = \infty \end{cases} \tag{4.5}$$

being $\mathcal{W}$ a compact and connected set in $\mathbb{R}^n$.

The identification error (4.4) cannot be exactly computed since, from the available data, it is only know that $f^o \in \tilde{\mathcal{F}}$, where $\tilde{\mathcal{F}}$ represents the set of all functions that could have generated the data. Without making any assumptions on f^o, this set remains unbounded even in cases of exact measurements. Thus, it is not possible to derive any information about the identification error, unless certain assumptions are made about f^o and the noise term d. The conventional approach in literature involves assuming a finitely parametrized structure for f^o (linear, polynomial, neural network, etc.) and a statistical model for the noise, as mentioned in [124]. In contrast to this conventional approach, the set membership method is based on weaker assumptions, without requiring specific

parametric structures for f^o but rather focusing on its regularity. Specifically, it assumes only that the function f^o is Lipschitz continuous on $\mathcal{W}$.

Assumption 3. *The function f^o is Lipschitz continuous on $\mathcal{W}$, i.e.,*

$$f^o \in \mathcal{F}(\gamma) \tag{4.6}$$

for some $\gamma < \infty$, where

$$\mathcal{F}(\gamma) \doteq \{f : |f(w) - f(\hat{w})| \leq \gamma \|w - \hat{w}\|_2, \forall w, \hat{w} \in \mathcal{W}\}.$$

Additionally, it only supposes that the noise sequence $\{d_k\}_{k=1}^{L}$ is bounded.

Assumption 4. *The noise sequence $d = (d_1, ..., d_L)$ is unknown but bounded:*

$$\|d\|_q \leq \mu \tag{4.7}$$

where $\|\cdot\|_q$ is the vector l_q norm.

A general formulation is presented here to handle the most important cases (i.e. $q = 2, \infty$) in a unified framework. As commonly known, the ℓ_2 norm is associated with the energy of the signal being considered, while the ℓ_∞ norm relates to its amplitude. The selection of this norm can be determined based on prior knowledge of the energy or amplitude of the noise involved (if available), or through a trial and error procedure.

The problem at hand involves deriving an approximation $\hat{f}$ of f^o from the available data and evaluating tight estimate bounds on f^o. The approximation is required to be accurate on the whole domain $\mathcal{W}$, and it is evaluated using the approximation error (4.4). Since f^o is unknown, this error cannot be precisely computed. However, it is known that $f^o \in FFS^L$, where FFS^L represents the *Feasible Function Set*, that is, the set of all possible functions consistent with the available prior information and measured data. The formal definition of FFS^L will be provided in subsequent sections for two specific relevant cases. This motivates the following definition of identification error, often indicated as worst-case or guaranteed error.

Definition 16 (Identification Error)**.** *The identification error of* $\hat{f}$ *is*

$$EN(\hat{f}) = \sup_{f \in FFS^L} \|f - \hat{f}\|_p. \tag{4.8}$$

An optimal approximation is defined as a function f_{op} which minimizes the identification error.

Definition 17 (Optimal Approximation)**.** *An approximation* f_{op} *is* ***optimal*** *if*

$$EN(f_{op}) = \inf_{\hat{f}} EN(\hat{f}) \doteq \mathcal{R}_{\mathcal{I}}. \tag{4.9}$$

The quantity $\mathcal{R}_{\mathcal{I}}$, known as the *radius of information*, represents the minimum worst-case error that any estimate can guarantee based on the prior and experimental information available up to time L.

Finding optimal approximations may be hard or not convenient, leading to the search for suboptimal solutions. In the literature, interpolatory approximations are often considered (see, e.g., [134]).

Definition 18 (Interpolatory Approximation)**.** *An approximation* f_I *is said to be* ***interpolatory*** *if*

$$f_I \in FFS^T.$$

A key property of an interpolatory approximation is that it ensures a worst-case error degradation of at most 2 [134]. An approximation with this property is called almost-optimal.

Definition 19 (Almost-Optimal Approximation)**.** *An approximation* f_{ao} *is considered* ***almost-optimal*** *if*

$$EN(f_{ao}) \leq 2 \inf_{\hat{f}} EN(\hat{f}).$$

In the set membership methods, the aim is to solve the following problem.

Problem 1. From the dataset (4.2), find an approximation $\hat{f}$ of f^o

1. optimal or almost-optimal;

2. with tight interval estimates $\overline{f}$ and $\underline{f}$.

In the field of set membership and approximation theory, two main concepts of optimality are typically considered: local and global optimality [135, 14].

Remark. *In general, local optimality also implies global optimality, while the converse does not necessarily hold. Consequently, local optimality represents a more stringent and less conservative approach than global optimality.*

A detailed description of these concepts will be provided below.

4.1.2 Global Approach

Firstly, the formal definition of the Feasible Function Set is introduced.

Definition 20 (Global Feasible Function Set)**.** *The Feasible Function Set is*

$$FFS^L \doteq \{f \in \mathcal{F} : \|\tilde{y} - f(\tilde{w})\|_q \leq \mu\} \tag{4.10}$$

where $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_L)$ *and* $f(\tilde{w}) \doteq (f(\tilde{w}_1), \ldots, f(\tilde{w}_L))$.

The feasible function set FFS^L summarizes all available information on the mechanism generating the data up to time L. If the prior assumptions hold, then $f^o \in FFS^L$, which is an important property for evaluating the accuracy of any estimate.

In the context of set membership, the validation of prior assumptions is a fundamental step. This is typically defined as the consistency of the assumptions with the available data: if there exists at least one estimate that satisfies both the assumptions and the data (i.e., if FFS^L is not empty), then the prior assumptions are considered validated (see, e.g., [14, 136]).

Definition 21 (Validation of prior assumptions)**.** *Prior assumptions are validated if* $FFS^L \neq \emptyset$.

Necessary and sufficient conditions can be established to verify the validity of the underlying prior assumptions. Before presenting the theorem that defines

these conditions, it is necessary to introduce the following bounds:

$$\begin{aligned} \overline{f}(w) &\doteq \min_{k=1,\dots,L} (\overline{h}_k + \gamma \|w - \tilde{w}_k\|_2), \\ \underline{f}(w) &\doteq \max_{k=1,\dots,L} (\underline{h}_k - \gamma \|w - \tilde{w}_k\|_2) \end{aligned} \tag{4.11}$$

where $\overline{h}_k \doteq \tilde{y}_k + \mu$ and $\underline{h}_k \doteq \tilde{y}_k - \mu$.

Now, the theorem can be formally stated.

Theorem 3. *(i) A necessary condition for prior assumptions to be validated is:* $\overline{f}(\tilde{w}_k) \geq \underline{h}_k$, $\underline{f}(\tilde{w}_k) \leq \overline{h}_k, k = 1, \dots, L$.
(ii) A sufficient condition for prior assumptions to be validated is: $\overline{f}(\tilde{w}_k) > \underline{h}_k$, $\underline{f}(\tilde{w}_k) < \overline{h}_k, k = 1, \dots, L$.

Proof. See [14]. □

Note that validating prior assumptions by ensuring their consistency with the present data does not guarantee their validation for future data. For the remainder of the section, it is assumed that the sufficient condition holds. If this is not the case, the values of the constants in the assumptions regarding the function f^o and the noise d_k must be appropriately adjusted. The validation in Theorem 3 can be used to assess the values of these constants and ensure that sufficient conditions hold.

Let us now proceed to introduce the function f_c, which is defined as follows:

$$f_c(w) \doteq \frac{1}{2}[\underline{f}(w) + \overline{f}(w)] \tag{4.12}$$

where $\underline{f}(w)$ and $\overline{f}(w)$ are given in (4.11). The next results demonstrate that the estimate f_c is optimal according to Definition 17 for any L_p norm.

Theorem 4. *Assume that:*

(i) The noise affecting the measurements $\mathcal{D}$ is bounded according to (4.7).

(ii) The function f^o is Lipschitz continuous according to (4.6).

Then, for $q = 2, \infty$ and for any $p \in [1, \infty]$:

(i) The approximation f_c defined in (4.12) is optimal.

(ii) The worst-case approximation error of f_c is given by

$$E(f_c) = \frac{1}{2}\|\overline{f} - \underline{f}\|_p = \inf_{\hat{f}} EN(\hat{f}) = \mathcal{R}_{\mathcal{I}}. \tag{4.13}$$

Proof. See [14]. □

The interval estimates $\overline{f}$, $\underline{f}$ of the unknown function f^o are now given for the cases where the noise is bounded in ℓ_∞ norm (i.e. $q = \infty$ in (4.7)). According to the FFS^L definition, it follows that $f^o(w)$ is bounded as

$$\underline{f}(w) \leq f^o(w) \leq \overline{f}(w), \, \forall w \in W \tag{4.14}$$

where

$$\begin{aligned} \overline{f}(w) &= \sup_{f \in FFS^L} f(w), \\ \underline{f}(w) &= \inf_{f \in FFS^L} f(w). \end{aligned} \tag{4.15}$$

Theorem 5. *Assume that:*

(i) The noise affecting the measurements $\mathcal{D}$ is bounded according to (4.7).

(ii) The function f^o is Lipschitz continuous according to (4.6).

Then, for $q = \infty$ and for any $w \in W$, $f^o(w)$ is tightly bounded as

$$\underline{f}(w) \leq f^o(w) \leq \overline{f}(w). \tag{4.16}$$

Proof. See [14]. □

Figure 4.1 shows an example of the global set membership approach for a nonlinear function.

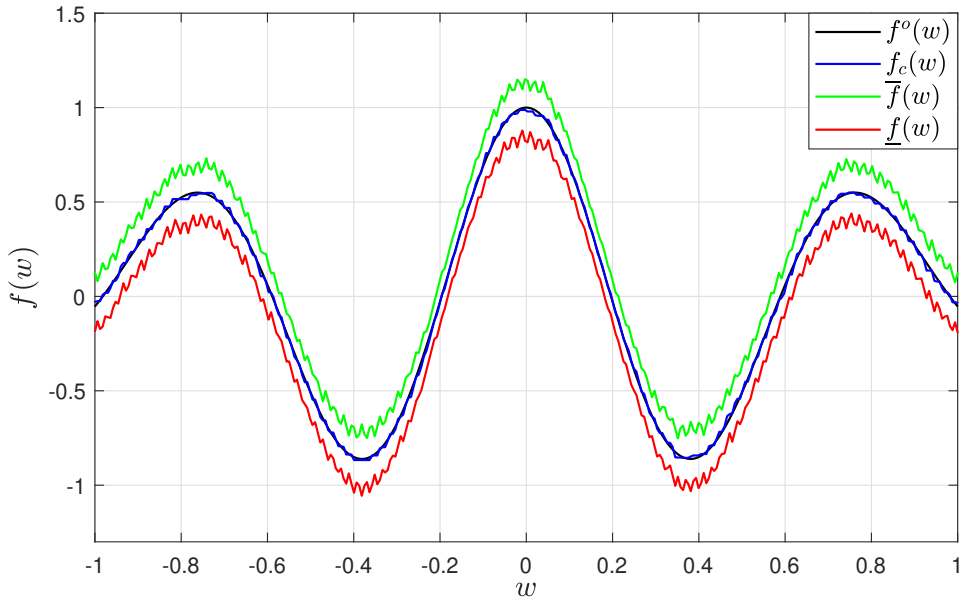

Fig. 4.1 Global Set Membership Approach.

4.1.3 Local Approach

Suppose that an initial approximation f^* of the function f^o has been obtained using any method. This approximation is of the form

$$f^*(w) = \sum_{i=1}^{N} a_i^* \psi_i(w) \tag{4.17}$$

where $\psi_i : \mathcal{W} \to \mathbb{R}$ are Lipschitz continuous basis functions and $a_i \in \mathbb{R}$ are parameters identified by means of some suitable algorithm. In this thesis, the Sparse Identification Algorithm described in Section 3.3.1 is used for generating the global approximation f^*.

Define the following *residue function*:

$$f_\Delta(w) \doteq f^o(w) - f^*(w). \tag{4.18}$$

From (4.6) and the Lipschitz continuity of ψ_i, it follows that f_Δ is Liptschiz continuous over the set $\mathcal{W}$:

$$f_\Delta \in \mathcal{F}(\gamma_\Delta) \tag{4.19}$$

for some $\gamma_\Delta < \infty$.

Under the above assumptions, particularly (4.6) and (4.19), we have that $f^o \in FFS^L$, where FFS^L is the Feasible Function Set defined as follows.

Definition 22 (Local Feasible Function Set)**.** *The Feasible Function Set is*

$$FFS^L \doteq \{f : f = f^* + f_\Delta, f_\Delta \in \mathcal{F}(\gamma_\Delta), \|\tilde{y} - f(\tilde{w})\|_q \leq \mu\}.$$

where $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_L)$ *and* $f(\tilde{w}) \doteq (f(\tilde{w}_1), \ldots, f(\tilde{w}_L))$.

The theorem presented herein demonstrates that the approximation f^*, defined in Eq. (4.17), has interpolatory characteristics, thereby rendering it nearly optimal. This theorem also provides an explicit bound on the worst-case approximation error. To proceed, let us define the following functions:

$$\begin{aligned} \overline{f}_\Delta(w) &\doteq \min_{k=1,\ldots,L} (\delta_k(a^*) + \mu + \gamma_\Delta \|w - \tilde{w}_k\|_2), \\ \underline{f}_\Delta(w) &\doteq \max_{k=1,\ldots,L} (\delta_k(a^*) - \mu - \gamma_\Delta \|w - \tilde{w}_k\|_2), \end{aligned} \tag{4.20}$$

$$\begin{aligned} \overline{f}(w) &= f^*(w) + \overline{f}_\Delta(w), \\ \underline{f}(w) &= f^*(w) + \underline{f}_\Delta(w), \end{aligned} \tag{4.21}$$

where $\delta_k(a^*) = \tilde{y}_k - f^*(\tilde{w}_k)$ and $\mu \geq 0$, for $k = 1, \ldots, L$.

Theorem 6. *Assume that:*

(i) The noise affecting the measurements $\mathcal{D}$ is bounded according to (4.7).

(ii) The function f^o is Lipschitz continous according to (4.6).

Then, for $q = 2, \infty$ and for any $p \in [1, \infty]$:

(i) The approximation f^* defined in (4.17) is interpolatory (and thus almost-optimal).

(ii) the worst-case approximation error of f^* is bounded as

$$EN(f^*) \leq \|\overline{f}_\Delta - \underline{f}_\Delta\|_p = 2 \inf_{\hat{f}} EN(\hat{f}). \tag{4.22}$$

Proof. See [15]. □

Now, define the function f_c as

$$f_c(w) \doteq f^*(w) + \frac{1}{2}[\overline{f}_\Delta(w) + \underline{f}_\Delta(w)] \tag{4.23}$$

where $\overline{f}_\Delta$ and $\underline{f}_\Delta$ are given in Eq. (4.20). The following results show that the function f_c is an optimal approximation of f^o.

Theorem 7. *Assume that:*

(i) The noise affecting the measurements $\mathcal{D}$ is bounded according to (4.7).

(ii) The function f^o is Lipschitz continuous according to (4.6).

Then, for $q = 2, \infty$ and for any $p \in [1, \infty]$:

(i) The approximation f_c defined in (4.23) is optimal.

(ii) the worst-case approximation error of f_c is bounded as

$$EN(f_c) = \frac{1}{2}\|\overline{f}_\Delta - \underline{f}_\Delta\|_p = \inf_{\hat{f}} EN(\hat{f}). \tag{4.24}$$

Proof. See [14]. □

In summary, we have demonstrated that the function $f^*(w)$ is an almost-optimal approximation, while the function $f_c(w) \doteq f^*(w) + [\overline{f}_\Delta(w) + \underline{f}_\Delta(w)]/2$ represents an optimal approximation of f^o. The correction term $[\overline{f}_\Delta(w) + \underline{f}_\Delta(w)]/2$ can be used to assess how close is f^* to the optimum.

Figure 4.2 shows an example of the local set membership approach for the same nonlinear function of Figure 4.1. It can be noted that the resulting uncertainty bounds $\overline{f}$ and $\underline{f}$ are tighter compared to those of the global approach. These bounds represent the *tightest guaranteed bounds*, as demonstrated in Theorem 8.

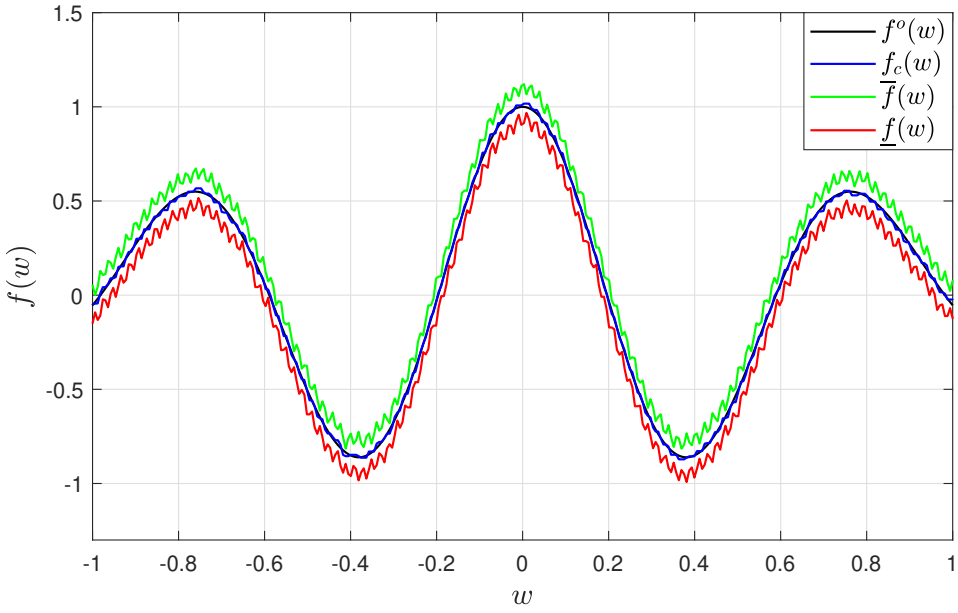

Fig. 4.2 Local Set Membership Approach.

4.2 Set Membership Method for Side-length Reduction in NMPC

This section outlines the novel approach for the side-length reduction of the search domain developed in the thesis. It uses the local Set Membership method described in Section 4.1.3 to derive from data tight bounds on the optimal NMPC control law, reducing accordingly the search domain within which the optimization process operates. Unlike many traditional estimation methods, the SM approach makes use of the so-called *interval bounds* to compute estimates of unknown functions, ensuring that the true value falls within the resulting uncertainty band (see Section 4.1 for more detail).

The side-length reduction method can be integrated with any NMPC technique. In this thesis, it was chosen to apply it in combination with the Dimensionality Reduction NMPC developed in Chapter 3, achieving a simultaneous reduction of both the dimensionality d and the side-length s of the volume of the search domain. The steps characterizing this method are presented below. Note that being in combination with dimensionality reduction, there are some points already described in Chapter 3.

1. *Offline data generation.* A dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{c}_l\}_{l=1}^{M}$ is generated, where:

   - $\tilde{w}_l \doteq (\tilde{x}_l, \tilde{\mathfrak{r}}_l)$ is the regressor composed by a sample $\tilde{x}_l \in \mathbb{R}^{n_x}$ of the state and a sample $\tilde{\mathfrak{r}}_l = (\tilde{\mathfrak{r}}_{l1}, \ldots, \tilde{\mathfrak{r}}_{lT}) \in \mathbb{R}^{n_x \times T}$ of the reference sequence. These samples should be chosen to explore the state and reference domains of interest.
   - $\tilde{c}_l$ is the command value corresponding to $\tilde{w}_l$, computed solving the optimization problem (2.8).

   For more detail, see Section 3.4.1.

2. *Offline dataset reduction through clustering.* Since in general the number of generated data M can be large, a clustering process is employed to reduce the size of $\mathcal{D}$. This pre-processing step groups similar data, effectively decreasing the size of the dataset while preserving its relevant information. The resulting reduced dataset is indicated with $\mathcal{D}_r$. The clustering technique used in this paper is described in Section 4.2.1.

3. *Offline selection of the relevant decision variables.* The following selection matrices are defined: i) $\Gamma_r \in \mathbb{R}^{n_c \times n_{cr}}$ selects the "relevant" components of the decision vector c in (2.8). ii) $\Gamma_a \in \mathbb{R}^{n_c \times n_{ca}}$ selects the remaining components of c. These are matrices with elements 0 or 1, such that $c = \Gamma_r c_r + \Gamma_a c_a$, where $c_r \in \mathbb{R}^{n_{cr}}$ is the vector of the "relevant" components and $c_a \in \mathbb{R}^{n_{ca}}$ is the vector containing the other components, with $n_{cr}, n_{ca} \leq n_c$. For further information, refer to Section 3.4.2.

4. *Offline approximation of the NMPC control law.* According to the formulation of Section 2.3, the optimal NMPC command is $u_{t+1} = S_1^u \Gamma c^*$, where c^* is the solution of the optimization problem (2.8). This solution can be seen as a static nonlinear function ϕ_o of the current state x_t and the reference sequence $\mathbf{r}_t = (r_{t+2}, \ldots, r_{t+T+1})$: $c^* = \phi_o(x_t, \mathbf{r}_t)$. An approximation ϕ_s of ϕ_o is derived from the dataset $\mathcal{D}$, using the Nonlinear Sparse Identification method described in Section 3.3.1. For additional information, see Section 3.4.3.

5. *Offline bounds definition of the most relevant decision variables.* The Set Membership method, described in Section 4.1.3, is applied to define guaranteed tight bounds $\overline{\phi}$ and $\underline{\phi}$ of the NMPC law ϕ_o, only for the

"relevant" decision variables. Since these bounds must be evaluated online, they are defined using the reduced dataset $\mathcal{D}_r$, obtained by means of the clustering procedure described above. See Section 4.2.2 for more details.

6. *Online optimization on the reduced dimensionality and side-length OCP.* At each time t:

   - the approximation ϕ_s is used to obtain the approximated solution $\hat{c} = \Gamma_r \hat{c}_r + \Gamma_a \hat{c}_a$;
   - the functions $\overline{\phi}$ and $\underline{\phi}$ are used to reduce the side-length of the search domain, obtaining the *reduced search domain* $\mathcal{C}_c^{ds}$ (see Section 4.2.3 for more details); u
   - the OCP is performed only on the relevant variables c_r, using $\hat{c}_r$ as a warm start, within the search domain $\mathcal{C}_c^{ds}$;
   - the non-relevant variables c_a are kept fixed to the values $\hat{c}_a$.

Further details regarding the steps characterizing the side-length reduction are provided below.

4.2.1 Offline data reduction through clustering

Due to the inherent relationship between the complexity of a Set Membership model and the size of its training dataset, a clustering process is employed to reduce the size of the dataset $\mathcal{D}$ from M to $K < M$, resulting in the reduced dataset $\mathcal{D}_r$. An overview of this approach and its application for dataset reduction is presented below.

Clustering

Clustering is one of the most important unsupervised learning problems, which involves grouping a set of objects in a way that objects within the same group (known as a *cluster*) are similar to each other and dissimilar to objects in other groups. In other words, clustering consist in finding a structure in a collection of unlabeled data. There exists a large number of clustering algorithms in the

literature. The choice of a specific clustering algorithm is influenced by the type of the available data as well as the particular purpose and application. Clustering methods can be broadly categorized into the following main groups [137]: i) Partitioning methods, ii) Hierarchical methods, iii) Density-based methods, iv) Grid-based methods, and v) Model-based methods. Among all these methods, the Partitioning methods are used in this thesis.

Partitioning algorithms involve specifying an initial number of groups and then iteratively reallocating objects among the groups until convergence. Typically, this algorithm determines all clusters at once. Many applications adopt one of two popular heuristic methods:

1. the K-means algorithm,
2. the K-medoids algorithm.

K-means algorithm. The K-means algorithm, a well-known partitioning method for clustering data [138, 139], iteratively partitions objects into a predefined number of clusters k, aiming to minimize the within-cluster variation. It works by initially placing k well-separated centroids (i.e., cluster centers) and then assigning each data point to the nearest centroid. Subsequently, the centroids are recalculated as the means of their respective clusters, and points are reassigned based on the new centroids. This iterative process continues until convergence, meaning no significant changes occur in the centroid locations. The goal of K-means is to minimize the sum of the squared distances between each data point and its centroid. Let Z_i represent the data point and C_j represent its cluster centroid, this approach uses the Euclidean distance as follows:

$$\|Z_{i(j)} - C_j\|^2. \tag{4.25}$$

An example of K-means clustering is shown in Figure 4.3. It should be noted that this approach is sensitive to outliers, since the mean can be strongly affected by extreme value. On the other hands, the computation of means is relatively straightforward, making this method efficient with a computational complexity that is proportional to $O(MK)$, where M is the number of data points and K is the number of clusters.

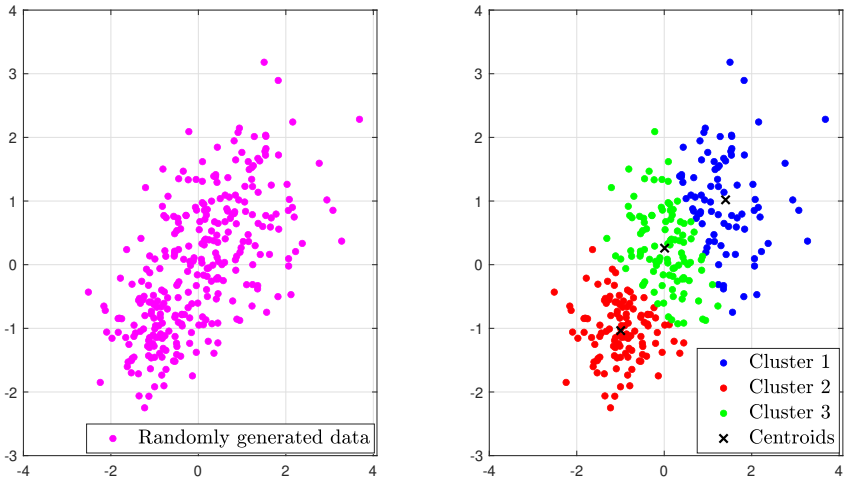

Fig. 4.3 Example of K-means clustering.

K-medoids algorithm. K-medoids clustering [140, 141] is a variant of K-means that minimizes the sum of absolute distances rather than the sum of squared distances. Moreover, unlike K-means which uses centroids, this approach employs the so-called medoids, which are points in the dataset whose sum of distances to other points in the cluster is minimal. Let Z_i represents the data point and M_j represents its cluster medoid. This approach uses the Manhattan distance as:

$$\|Z_{i(j)} - M_j\|_1. \tag{4.26}$$

Thanks to these features, this approach is more robust to noise and outliers than K-means. An example is shown in Figure 4.4. A popular method for K-medoids clustering is the so-called Partitioning Around Medoids (PAM), introduced by Kaufman and Rousseeuw in 1990 [142]. The basic idea is to select K representative points to form initial clusters and iteratively improve them. This process involves evaluating all possible combinations of representative and non-representative points to compute the quality of resulting clusters using a distortion function. An original representative point is replaced with a new point if it determines a greatest reduction in the distortion function, which means a better quality of the clusters. The best points for each cluster at each iteration form the new respective medoids. One of the disadvantages of this approach is its computational burden. Indeed, the time complexity of the PAM algorithm is proportional to $O(KM^2)$, which could pose challenges

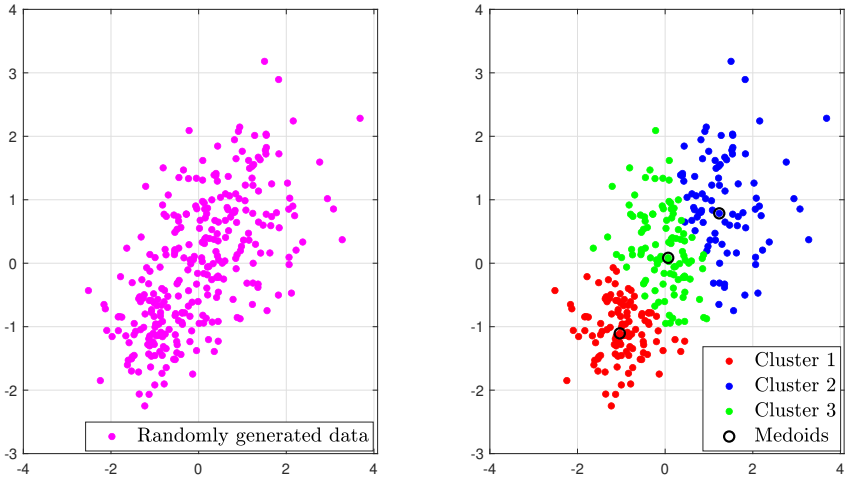

Fig. 4.4 Example of K-medoids clustering.

when dealing with large datasets. To improve the efficiency, algorithms like Clustering LARge Applications (CLARA) [143] have been proposed. This algorithm iteratively applies the PAM method on a smaller sample containing $M_{cl} << M$ objects, with the recommended value $M_{cl} = 40 + 2M$. Then, it assigns the remaining objects to their nearest medoid based on proximity.

In this thesis, to address the large size of collected data M, the CLARA algorithm is used. To effectively reduce the dimension, the number of clusters K should be significantly smaller than the original dataset size, i.e. $K \ll M$. The resulting dataset, which best captures the overall behavior of the system, is $\mathcal{D}_r = \{\tilde{w}_{lr}, \tilde{u}_{lr}\}_{lr=1}^{K}$, comprising K regressor $\tilde{w}_{lr}$ and commands $\tilde{u}_{lr}$.

4.2.2 Offline bounds definition of the most relevant decision variables

The optimal NMPC solution c^* is a static nonlinear function ϕ_o of the current state x_t and the reference sequence $\mathbf{r}_t = (r_{t+2}, \ldots, r_{t+T+1})$. The NMPC solution c^* at time t is thus given by

$$c^* = \phi_o(w_t) \tag{4.27}$$

where $w_t \doteq (x_t, \mathbf{r}_t)$ is the *regressor*.

Based on the reduced dataset $\mathcal{D}_r = \{\tilde{w}_{l_r}, \tilde{c}_{l_r}\}_{l_r=1}^{K}$ and the approximation ϕ_s (see Chapter 3), the Nonlinear Set Membership method (described in Section 4.1.3) is used to define *guaranteed tight bounds* $\overline{\phi}$ and $\underline{\phi}$ of the NMPC control law.

Let $\mathbb{W} \subset \mathbb{R}^{(T+1)n_x}$ be a region where the regressor w_t can evolve. Note that this region is bounded since the NMPC algorithm is assumed to guarantee a bounded tracking error and the reference is assumed bounded. As done in Section 3.4.3, we consider the mono-dimensional case $n_c = 1$. The generalization to the case of $n_c > 1$ is trivial and can be accomplished by applying the method proposed here to each component of c^*.

Consider the function Δ defined as

$$\Delta(w) \doteq \phi_o(w) - \phi_s(w) \tag{4.28}$$

where ϕ_o is the exact NMPC control law and ϕ_s is its approximation using the Nonlinear Sparse Identification method presented in Chapter 3. Δ is not known since the closed-form expression of ϕ_o is unknown. To derive tight bounds on ϕ_o, we assume that Δ is Lipschitz continuous. That is, $\Delta \in \mathcal{F}(\gamma_\Delta)$, for some $\gamma_\Delta < \infty$, where

$$\mathcal{F}(\gamma) \doteq \{g : |g(w) - g(\widehat{w})| \leq \gamma \|w - \widehat{w}\|, \forall w, \widehat{w} \in \mathbb{W}\}.$$

Note that the constant γ_Δ can be estimated using the validation procedure in [14].

Under this assumption, considering the constraint (3.41b), we can define the Feasible Function Set (FFS), i.e., the set of all functions consistent with the available assumptions and data.

Definition 23 (Feasible Function Set)**.** *The FFS is*

$$\text{FFS} \doteq \{\phi : \phi = \hat{\phi} + \Delta, \Delta \in \mathcal{F}(\gamma_\Delta), \|\tilde{c} - \phi_s(\tilde{w})\| \leq \mu\} \tag{4.29}$$

where $\tilde{c} = (\tilde{c}_1, \ldots, \tilde{c}_K)$ *and* $\phi_s(\tilde{w}) \doteq \Big(\phi_s(\tilde{w}_1), \ldots, \phi_s(\tilde{w}_K)\Big)$.

As already mentioned in the Remark of Section 3.4.3, μ accounts for the error introduced by the sparse approximation and the use of subsequences of the reference $\mathbf{r}_t$.

Based on the above definition, the functions

$$\sup_{\phi \in \mathrm{FFS}} \phi(w), \ \inf_{\phi \in \mathrm{FFS}} \phi(w)$$

are the tightest upper and lower bounds of ϕ_o that can be defined on the basis of the available assumptions and data. The theorem presented in the following shows that these bounds can be computed in closed-form and are given by

$$\begin{aligned} \overline{\phi}(w) &\doteq \phi_s(w) + \overline{\Delta}(w) \\ \underline{\phi}(w) &\doteq \phi_s(w) + \underline{\Delta}(w) \end{aligned} \tag{4.30}$$

where

$$\begin{aligned} \overline{\Delta}(w) &\doteq \min[\overline{c}, \min_{l_r=1,\dots,K} (\tilde{c}_{l_r} - \phi_s(\tilde{w}_{l_r}) + \mu + \gamma_\Delta \|(w - \tilde{w}_{l_r})\|)] \\ \underline{\Delta}(w) &\doteq \max[\underline{c}, \max_{l_r=1,\dots,K} (\tilde{c}_{l_r} - \phi_s(\tilde{w}_{l_r}) - \mu - \gamma_\Delta \|(w - \tilde{w}_{l_r})\|)]. \end{aligned} \tag{4.31}$$

The functions $\overline{\phi}$ and $\underline{\phi}$ in (4.30) are *tightest guaranteed bounds* of ϕ_o: they are the tightest upper and lower bounds that can be derived from the available prior information on the function and the data. Moreover, the true value of the function ϕ_o is guaranteed to lie within these bounds. These two properties are formally proven in the following Theorem.

Theorem 8 (Tightest guaranteed bounds)**.** *The functions $\overline{\phi}$ and $\underline{\phi}$ are the tightest guaranteed bounds of ϕ_o, i.e.,*

$$\underline{\phi}(w) \leq \phi_o(w) \leq \overline{\phi}(w) \tag{4.32}$$

and

$$\overline{\phi}(w) = \sup_{\phi \in \mathrm{FFS}} \phi(w) \tag{4.33a}$$

$$\underline{\phi}(w) = \inf_{\phi \in \mathrm{FFS}} \phi(w). \tag{4.33b}$$

Proof. According to (4.30), we have $\overline{\phi}(w) \doteq \phi_s(w) + \overline{\Delta}(w)$. Under the assumption that $\Delta \in \mathcal{F}(\gamma_\Delta)$, implying Lipschitz continuity of Δ with Lipschitz constant γ_Δ, the following inequality holds:

$$\Delta(w) \leq \Delta(\tilde{w}_l) + \gamma_\Delta \|w - \tilde{w}_l\|, \forall w \in \mathbb{W}$$

for $l = 1, \ldots, K$, which yields

$$\Delta(w) \leq \min_{l=1,\ldots,K} (\Delta(\tilde{w}_l) + \gamma_\Delta \|w - \tilde{w}_l\|) \doteq \overline{\Delta}(w), \forall w \in \mathbb{W}.$$

That is, the function $\overline{\Delta}(\cdot)$ is an upper bound of the function $\Delta(\cdot)$. For given $w, \widehat{w} \in \mathbb{W}$, let

$$\begin{aligned} i &\doteq \arg \min_{l=1,\ldots,K} (\Delta(\tilde{w}_l) + \gamma_\Delta \|w - \tilde{w}_l\|), \\ \widehat{i} &\doteq \arg \min_{l=1,\ldots,K} (\Delta(\tilde{w}_l) + \gamma_\Delta \|\widehat{w} - \tilde{w}_l\|). \end{aligned}$$

Then,

$$\begin{aligned} \overline{\Delta}(w) &= (\Delta(\tilde{w}_i) + \gamma_\Delta \|w - \tilde{w}_i\|) \\ &\leq (\Delta(\tilde{w}_{\widehat{i}}) + \gamma_\Delta \|w - \tilde{w}_{\widehat{i}}\|), \\ \overline{\Delta}(\widehat{w}) &= (\Delta(\tilde{w}_{\widehat{i}}) + \gamma_\Delta \|\widehat{w} - \tilde{w}_{\widehat{i}}\|). \end{aligned}$$

This yields

$$\begin{aligned} \overline{\Delta}(w) - \overline{\Delta}(\widehat{w}) &\leq \gamma_\Delta \|w - \tilde{w}_{\widehat{i}}\| - \gamma_\Delta \|\widehat{w} - \tilde{w}_{\widehat{i}}\| \\ &\leq \gamma_\Delta \|w - \widehat{w}\|, \end{aligned}$$

which shows that $\overline{\Delta}(w)$ is Lipschitz continuous with Lipschitz constant γ_Δ, implying that $\overline{\Delta} \in \mathcal{F}(\gamma_\Delta)$. Introduce now a FFS for Δ

$$\mathrm{FFS}_\Delta \doteq \{\Delta : \Delta \in \mathcal{F}(\gamma_\Delta), \|\Delta(\tilde{w})\| \leq \mu\}.$$

Being $\overline{\Delta}(\cdot)$ an upper bound of $\Delta(\cdot)$, it follows that

$$\sup_{\Delta \in FFS_\Delta} \Delta(w) = \overline{\Delta}(w), \forall w \in \mathbb{W}.$$

This establishes that $\overline{\Delta}(\cdot)$ is the tightest upper bound of the unknown function $\Delta(\cdot)$. From (4.30) and Definition 23, it follows that

$$\sup_{\phi \in \text{FFS}} \phi(w) = \overline{\phi}(w), \forall w \in \mathbb{W}.$$

This demonstrates that $\overline{\phi}(\cdot)$ is the tightest upper bound of the unknown function $\phi_o(\cdot)$. Analogously, it can be shown that $\underline{\phi}(\cdot)$ is the tightest lower bound of $\phi_o(\cdot)$. □

4.2.3 Online optimization on the reduced dimensionality and side-length OCP.

At each time t, given the regressor $w_t = (x_t, \mathbf{r}_t)$, the approximated solution

$$\hat{c} = \phi_s(w_t) = \Gamma_r \hat{c}_r + \Gamma_a \hat{c}_a \tag{4.34}$$

is computed online. The functions $\overline{\phi}(w_t)$ and $\underline{\phi}(w_t)$ are used to reduce the side-length of the search domain, resulting in the *reduced dimensionality and side-length search domain*

$$\mathcal{C}_c^{ds} = \prod_j \mathcal{C}_{c_j}^{ds} = [\underline{\phi}_j(w_t), \overline{\phi}_j(w_t)], \text{ for } j = 1, \ldots, n_{cr}. \tag{4.35}$$

Then, the optimization is performed: (i) only on the most relevant variables c_r, using $\hat{c}_r$ as a warm start, while leaving the remaining ones fixed at the values $\hat{c}_a$; (ii) within the search domain $\mathcal{C}_c^{ds}$. The resulting OCP is the following:

$$c_r^* = \arg\min_{c_r} J(\Gamma c) \tag{4.36a}$$

subject to:

$$\hat{x}_1 = f(x_t, u_t) \tag{4.36b}$$

$$\hat{x}_{k+1} = f(\hat{x}_k, S_k^u \Gamma c),\ k = 1 : T \tag{4.36c}$$

$$c_r \in \mathcal{C}_c^{ds} \tag{4.36d}$$

$$c = \Gamma_r c_r + \Gamma_a \hat{c}_a. \tag{4.36e}$$

This approach leads to a simultaneous reduction of the number of variables involved in the optimization and the number of iterations and cost function evaluations to find the optimal solution. This allows a computational complexity reduction with respect to solving the OCP (2.8). In Section 4.3, we delve into the correlation between the side-length of the search domain and the number of iterations required by the optimization algorithm, providing a comprehensive analysis of the rate of convergence.

It should be noted that the proposed Side-length reduction method works effectively not only when combined with the dimensionality reduction technique, but also when applied directly to traditional NMPC approaches. The following subsection outlines the steps needed for its implementation with the OCP (2.8). The performance of the resulting Side-length reduction NMPC will be shown in the chapter 5.

4.2.4 Side-length reduction method applied to single shooting NMPC

Here, the side-length reduction method is applied directly to the single shooting NMPC with OCP (2.8), introduced in Section 2.3, without employing any dimensionality reduction technique. The main steps that characterize this method are: (i) Offline data generation; (ii) Offline dataset reduction through clustering; (iii) Offline approximation of the NMPC control law; (iv) Offline bounds definition of all the decision variables; (iv) Online optimization on the reduced side-length OCP.

In this case, no distinction is made between relevant and non-relevant decision variables. This implies that the selection matrices are

$$\Gamma_r = I_{n_c}, \quad \Gamma_a = 0. \tag{4.37}$$

Therefore, during the offline phase, first the Nonlinear Sparse Identification method, described in Chapter 3, is used to derive an approximation ϕ_s of the NMPC control law ϕ_o. Then, the set membership method is applied to define guaranteed tight bounds $\overline{\phi}$ and $\underline{\phi}$ of the NMPC law ϕ_o, for all the decision variables c. During the online procedure, at each time t, the approximated

control law ϕ_s is applied to the regressor $w_t = (x_t, \mathbf{r}_t)$ for computing the approximated solution

$$\hat{c} = \phi_s(w_t) = \Gamma_r \hat{c}_r + \Gamma_a \hat{c}_a = \hat{c}_r. \tag{4.38}$$

The functions $\overline{\phi}(w_t)$ and $\underline{\phi}(w_t)$ are used to reduce the search domain in which the decision variables can be found, obtaining the *reduced side-length search domain*

$$\mathcal{C}_c^s = \prod_j \mathcal{C}_{c_j}^s = [\underline{\phi}_j(w_t), \overline{\phi}_j(w_t)], \text{ for } j = 1, \ldots, n_c. \tag{4.39}$$

Then, the optimization is performed: (i) on all the decision variables c, using $\hat{c}$ a warm start; (ii) within the search domain $\mathcal{C}_c^s$. The resulting OCP is defined as:

$$c^* = \arg\min_c J(\Gamma c) \tag{4.40a}$$

subject to:

$$\hat{x}_1 = f(x_t, u_t) \tag{4.40b}$$

$$\hat{x}_{k+1} = f(\hat{x}_k, S_k^u \Gamma c),\ k = 1:T \tag{4.40c}$$

$$c \in \mathcal{C}_c^s. \tag{4.40d}$$

This method leads to a reduction of the number of iterations and cost function evaluations, without requiring the use of the gradient procedure for finding the most relevant decision variables.

Remark. *Both dimensionality reduction and side-length reduction methods have a worse performance in terms of computational complexity when used individually compared to their combined use. However, they can be useful for obtaining simple methods with shorter computational times than NMPC with OCP (2.8).*

4.3 Computational Complexity Analysis

The proposed side-length reduction approach consists of the following main operations. Offline operations: set membership bounds definition (Section 4.2.2). Online operation: solution of the optimization problem (4.36d) within

the reduced search domain $\mathcal{C}_c^r$ (Section 4.2.3). In this section, we first provide some general considerations about the advantages of reducing the side-length of an OCP search domain that hold for all nonlinear optimization algorithms. Then, we conduct a theoretical analysis considering the Sequential Quadratic Programming, that is one of the most common nonlinear optimization techniques.

4.3.1 General considerations

Typically, nonlinear optimization algorithms comprise two mechanisms for the search process of the optimal solution: exploration and exploitation. Exploration involves the inspection of the search domain to identify the possible optimal solutions, while exploitation focuses on finding the optimal solution around the candidate solutions obtained during the exploration phase. The computational burden of the exploration phase increases as the volume/size of the search domain grows, requiring more iterations and evaluations of the cost function to find an optimal solution. Consequently, a side-length reduction method can lead to a significant reduction in the computational complexity required for this phase. Indeed, by reducing the volume/size of the search space, the algorithm can focus only on the search region where the optimal solution is present (as guaranteed by Theorem 8), thereby avoiding time-consuming iterations and function evaluations. For a sufficiently small volume, the algorithm no longer needs to explore the search space and can move to the optimal solution with improved convergence properties.

An example of how the side-length reduction shrinks the search domain is shown in Figure 4.5.

4.3.2 Theoretical Analysis of the Rate of Convergence of SQP methods

For quantifying the impact of the side-length reduction on the computational complexity of the OCP, a theoretical analysis is now conducted, considering the Sequential Quadratic Programming (SQP) approach. As already discussed in Chapter 2, SQP is one of the most successful methods for numerically solving

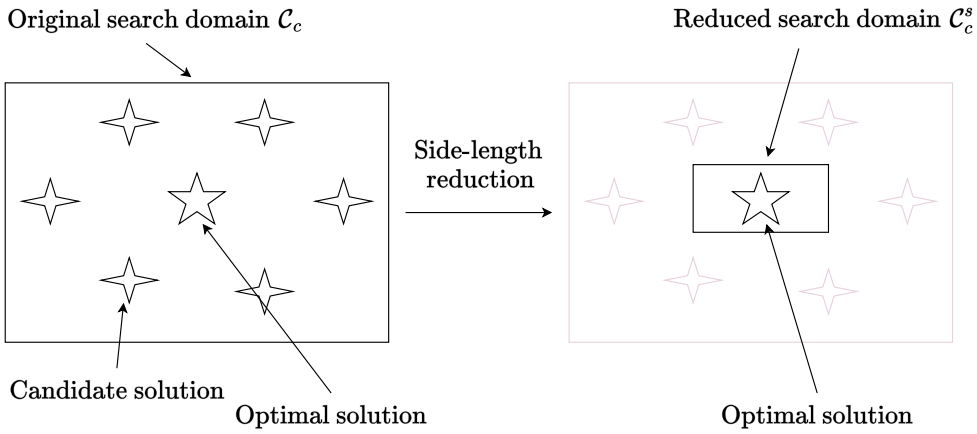

Fig. 4.5 Graphical illustration of a side-length reduction of the search domain.

constrained nonlinear optimization problems. As well-known, the fundamental idea is to solve, at each iteration, a Quadratic Programming (QP) subproblem which is obtained by linearizing the constraints and approximating the objective function quadratically.

A brief and simplified description of this method and one of its first-order approximation is re-proposed below (for more detail please refer to Section 2.5.2). Consider a general nonlinear optimization problem (NLP):

$$\min_{c \in \mathcal{C}_c} J(c), \text{ subject to: } b(c) = 0,\ g(c) \leq 0 \tag{4.41}$$

where $c \in \mathcal{C}_c$ is the decision vector, $\mathcal{C}_c \subset \mathbb{R}^{n_c}$ is an hyper-rectangle, J is the objective function, while $b : \mathbb{R}^{n_c} \to \mathbb{R}^{n_b}$ and $g : \mathbb{R}^{n_c} \to \mathbb{R}^{n_g}$ describe respectively the equality and inequality constraints.

The formulation of a QP subproblem is based on the following Lagrangian function:

$$L(c, \lambda, \mu) \doteq J(c) + \lambda^T b(c) + \mu^T g(c) \tag{4.42}$$

where $\lambda \in \mathbb{R}^{n_b}$ and $\mu \in \mathbb{R}^{n_g}$ are the Lagrange multipliers.

Starting with an initial guess c_0, a full step SQP iteration is performed at each iteration i:

$$c_{i+1} = c_i + \alpha_i p_i \tag{4.43}$$

where $\alpha_i \in (0,1]$ is the step length determined by an appropriate line search procedure and $p_i \in \mathcal{C}_c$ is the search direction computed as the solution of the following QP subproblem:

$$\begin{aligned} &\min_{p \in \mathcal{C}_c} \quad \nabla J(c_i)^T p + \tfrac{1}{2} p^T H_i p \\ &\text{subject to:} \\ &\qquad b(c_i) + \nabla b(c_i)^T p = 0 \\ &\qquad g(c_i) + \nabla g(c_i)^T p \leq 0. \end{aligned} \tag{4.44}$$

Therein, $\nabla b(c_i)$ and $\nabla g(c_i)$ are the constraint Jacobians, the matrix H_i denotes either the exact Hessian of the Lagrangian $\nabla^2 L(c_i, \lambda_i, \mu_i)$ or a positive definite quasi-Newton approximation B_i of it, and $\nabla J(c_i)$ is the cost function gradient.

SQP methods that use exact Hessian rely on the computation of second-order derivatives, which can be computationally expensive. Indeed, the number of operations to compute the Hessian is proportional to $O(n_c^2)$, while its inversion scales as $O(n_c^3)$.

These operations can be prohibitively expensive for real-time applications. As a result, many popular variants of SQP methods based on Hessian approximation techniques have been developed. An important class of such SQP algorithms uses exact constraint Jacobians but approximates the Hessian matrix with a first-order derivative based updating formula. These approaches are usually referred to as Quasi-Newton methods and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) is the most widely used formula. With this method, the Hessian approximation B_i is updated according to

$$B_{i+1} = B_i + \frac{q_i q_i^T}{q_i^T s_i} - \frac{B_i s_i s_i^T B_i^T}{s_i^T B_i s_i} \tag{4.45}$$

where $s_i = c_{i+1} - c_i$, $q_i = \nabla L(c_{i+1}, \lambda_i) - \nabla L(c_i, \lambda_i)$, and λ_i is a current estimate of the Lagrange multiplier λ. An interesting property of Equation (4.45) is that Hessian positivity can be maintained, satisfying the curvature condition $q_i^T s_i > 0$ at each update.

Since the BFGS-SQP formulation does not require matrix inversion, the number of operations necessary for updating the solution c_i in (4.43) is $O(n_c^2)$. This solution is iteratively updated using (4.43), until, under certain conditions,

it converges to the optimal solution c^*. For BFGS-SQP methods, the speed of this convergence in a neighborhood of c^*, called *rate of convergence*, can be Q-linear or Q-superlinear, where Q means "Quotient". In the case of Q-linear convergence, the error between successive iterations decreases with a constant factor for all i. In the case of Q-superlinear convergence, this error tends to 0 as $i \to \infty$. This means that, while linear convergence guarantees a proportional decrease in error, superlinear convergence implies a faster reduction in error, that accelerates with each iteration tending to 0 when i becomes large.

Definition 24 (Q-superlinear convergence rate)**.** *Consider a sequence* $\{c_i\} \in \mathbb{R}^{n_c}$ *that converges asymptotically to a solution* c^* *of problem* (4.41)*. The converge of* $\{c_i\}$ *is said* ***Q-superlinear*** *if:*

$$\lim_{i\to\infty} \frac{\|c_{i+1} - c^*\|}{\|c_i - c^*\|} = 0.$$

It is worth mentioning another type of rate of convergence, known as root convergence (R-convergence). This type of convergence considers the overall rate of error reduction, while quotient convergence requires error reduction at each iteration of the algorithm. Thus, Q-convergence is a stronger form of convergence than R-convergence.

Suppose now that the BFGS-SQP method is used to solve the OCP (4.36). This optimization problem features a side-length reduction of the search domain, obtained through the application of the Set Membership approach described in Subsection 4.2.2. A theorem is presented below, giving conditions under which the side-length reduction allows a Q-superlinear convergence. Define

$$\breve{\Delta}_j(w_t) \doteq \frac{\overline{\Delta}_j(w_t) + \underline{\Delta}_j(w_t)}{2},\ j = 1, \ldots, n_{cr} \tag{4.46}$$

where $\overline{\Delta}_j$ and $\underline{\Delta}_j$ are given in (4.31), and $w_t \doteq (x_t, \mathbf{r}_t)$ is the current regressor at time t. Define also the set

$$\mathcal{D}_\Delta = \{c \in \mathbb{R}^{n_{cr}} : |c_j - c_j^*| \leq \breve{\Delta}_j(w_t),\ j = 1, \ldots, n_{cr}\}. \tag{4.47}$$

With an abuse of mathematical notation, let us henceforth define $J(c) \doteq J(\Gamma c)$.

Theorem 9 (Q-superlinear convergence of BFGS-SQP)**.**
Assume that:

(i) The function J is twice differentiable in $\mathcal{D}_\Delta$.

(ii) $\nabla^2 J(c^*)$ is symmetric, positive definite and $\|\nabla^2 J(c^*)^{-1}\| \leq \beta$, with $\beta > 0$.

(iii) $\nabla^2 J(c)$ is Lipschitz continuous on $\mathcal{D}_\Delta$, with Lipschitz constant $\gamma > 0$:

$$\|\nabla^2 J(c) - \nabla^2 J(c^*)\| \leq \gamma \|c - c^*\|, \; \forall c \in \mathcal{D}_\Delta.$$

(iv) The step length α_i in (4.43) is chosen equal to $1 \; \forall i$.

(v) The following inequality holds:

$$\|\breve{\Delta}(w_t)\| \leq \frac{1}{9\gamma\beta}. \tag{4.48}$$

Then, the sequence $\{c_i\}$ generated by the BFGS-SQP method for solving OCP (4.36) is well defined, remains in $\mathcal{D}_\Delta$ and converges Q-superlinearly to c^*.

Proof. According to Theorem 8.2.2 of [144], if there exist positive constants ϵ and δ, and an initial Hessian approximation B_0 such that:

$$\|c_0 - c^*\| \leq \epsilon \tag{4.49}$$

$$\|B_0 - \nabla^2 J(c^*)\| \leq \delta \tag{4.50}$$

under the conditions that:

$$6\beta\delta \leq 1 \tag{4.51}$$

$$3\gamma\epsilon \leq 2\delta \tag{4.52}$$

then the statement holds true.

The contribution of this proof consists in relating ϵ, δ and B_0 with $\breve{\Delta}$. Suppose to be at the time t. According to the Theorem 8, both the initial solution $c_o = \phi_s(w_t)$ and the optimal solution c^* lie within the guaranteed bounds defined in (4.30). This implies that

$$|c_{0_j} - c_j^*| \leq \breve{\Delta}_j(w_t), \; j = 1, \ldots, n_{cr}. \tag{4.53}$$

From now on, we will omit w_t for simplicity of notation. Using (4.53) in (4.49), we obtain

$$\|c_0 - c^*\| = \sqrt{\sum_{j=1}^{n_{cr}} (c_{0_j} - c_j^*)^2} \leq \sqrt{\sum_{j=1}^{n_{cr}} \breve{\Delta}_j^2} = \|\breve{\Delta}\|. \tag{4.54}$$

Moreover, under the assumption *(iii)*, if we choose $B_0 = \nabla^2 J(c_0)$, we can establish the following relationship between B_0 and $\|\breve{\Delta}\|$:

$$\|\nabla^2 J(c_0) - \nabla^2 J(c^*)\| = \|B_0 - \nabla^2 J(c^*)\| \leq \gamma \|\breve{\Delta}\|. \tag{4.55}$$

Let us also choose $\epsilon = \|\breve{\Delta}\|$ and $\delta = \frac{3}{2}\gamma\|\breve{\Delta}\|$. Consequently, condition (4.52) is always satisfied. Furthermore, if (4.48) holds true, we have

$$\delta = \frac{3}{2}\gamma\|\breve{\Delta}\| \leq \frac{1}{6\beta}. \tag{4.56}$$

This ensures the validity of condition (4.51) and concludes the proof. □

Remark. *The innovative aspect of this work, compared to previous ones, lies in the ability of the SM to define and guarantee the maximum distance* (4.48) *between the optimal solution and the initial guess of the optimization problem. Knowing this distance, it is possible to understand if the derived bounds are such as to allow superlinear convergence. Note that these bounds can be reduced to very small values by increasing the number of data used to identify the SM model. Furthermore, by choosing $\alpha_i = 1\ \forall i$, the line search procedure is no longer required, resulting in a further computational time saving.*

Some comments are now provided, about the technical assumptions *(i)*, *(ii)* and *(iii)* in Theorem 9. Assumption *(i)* is necessary to guarantee that the cost function is smooth, i.e., it has a well-defined gradient and Hessian. Assumption *(ii)* ensures that c^* is an optimal solution with a bounded inverse of the Hessian. Finally, assumption *(iii)* guarantees that the curvature of the cost function does not change too abruptly near c^*. These assumptions are generally reasonable and appropriate. In our case, where we use a standard cost function, these assumptions are verified.

4.4 Chapter summary

The key contribution of this chapter lies in presenting a novel approach for the optimal side-length reduction of the search domain. This approach uses the Set Membership approximation method to derive tight guarantee bounds on the optimal NMPC control law directly from available data. This effectively reduces the search domain that the optimization process needs to explore for finding the optimal solution. It should be noted that the proposed method is applied to the Dimensionality Reduction NMPC developed in Chapter 3, achieving a simultaneous reduction in both dimensionality and side-length of the search domain. However, the applicability of this approach extends beyond this specific framework. It can be directly applied to traditional NMPC formulations as well, demonstrably improving computational times. The effectiveness of this method in reducing computational burden for nonlinear optimization algorithms is rigorously demonstrated through a theoretical analysis. This analysis focuses on the convergence rate of a specific algorithm, the Sequential Quadratic Programming algorithm, when the side-length reduction method is implemented. In particular when the side-length, determined by the SM bounds, is reduced below to a certain threshold, the convergence rate of the algorithm shows Q-superlinear behavior towards the optimal solution, which is proven to lie within this volume. This is one of the most innovative aspects of this work.

Chapter 5

Simulation results in Autonomous Driving scenarios

In this chapter, the Dimensionality and Side-length reduction methods are used for trajectory planning and control of autonomous vehicles in real-world scenarios. Their performance is compared against standard NMPCs, i.e. NMPCs that integrate both single shooting and multiple shooting approaches without dimensionality and side-length reduction strategies. This comparison aims to demonstrate the effectiveness of the proposed methods in reducing the computational complexity of nonlinear optimization algorithms not only theoretically (as can be seen in Sections 3.5 and 4.3) but also through practical examples.

The autonomous vehicle scenarios chosen for the evaluation are:

1. Lane Keeping on Urban Road: This scenario evaluates the ability of a control system to maintain a designated lane on a road with some curves.

2. Obstacle Avoidance in Rural Road: This scenario assesses the capability of a control system to safely navigate around obstacles encountered on rural roads.

3. Parallel Parking: This scenario evaluates the performance of a control system in executing complex maneuvers to complete the parking avoiding collisions with other cars.

These three scenarios are selected to showcase the versatility of the developed methods in handling different tasks, from simple lane keeping to more complex maneuvers involving obstacle avoidance and parking.

To compare the performance of the proposed methods against standard NMPCs, both software-in-the-loop (SIL) and hardware-in-the-loop (HIL) simulations are conducted. In a SIL simulation, the entire closed loop control system is designed in Simulink, a graphical programming environment for modeling and simulating dynamic systems. This implies that the NMPC algorithm is implemented and executed using a `Matlab Function`. On the other hand, in a HIL simulation, the NMPC algorithm is implemented and executed on a hardware platform rather than in a simulated environment. In this thesis, an *Nvidia Jetson Nano* board is used.

Outline. The chapter is organized as follows. Section 5.1 provides an overview of the autonomous driving scenarios chosen for the performance evaluation and discusses the rationale behind their selection. Sections 5.2, 5.3 and 5.4 focus on performance evaluations, providing a detailed description of the results achieved by both developed methods for each scenario and comparing them with standard NMPCs. Additionally, a comparison using HIL simulation is performed for the first scenario.

5.1 Autonomous driving scenarios

This section describes the realistic autonomous vehicle scenarios considered in the thesis, providing also the motivations behind the choice of these specific maneuvers.

5.1.1 Lane Keeping on Urban Road

In recent years, the increasing number of cars has led to a rise in traffic accidents. As a result, driving safety has become a major concern in the social transportation and automotive sectors. Extensive investigation and analysis have revealed that unintentional lane departures are a leading factor in road accidents involving passenger cars (see, e.g.,[145] and [146]). Advanced driver

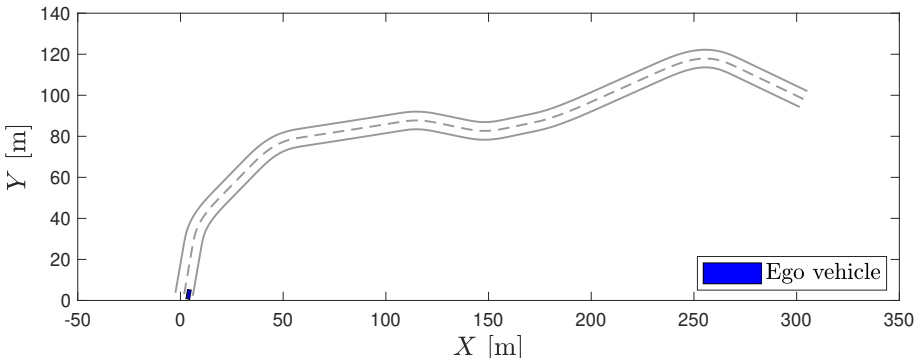

Fig. 5.1 Example of a road scenario for the lane-keeping control system.

assistance systems and autonomous driving functions, such as lane keeping, offer great potential for mitigating or even preventing many of these accidents [147]. Due to its significant impact on traffic safety, extensive research has been recently conducted on the control of the lateral vehicle dynamics (see, e.g., [148]). In this context, the developed NMPC approaches are used for implementing the lane keeping control system, and tested on various real-world urban roads. An example of one of these scenarios is shown in Figure 5.1.

5.1.2 Obstacle Avoidance in Rural Road

In the last decades, a remarkable amount of research and progress has been made towards creating intelligent technologies for autonomous vehicles [149, 150]. Significant developments include adaptive cruise control, lane-keeping system (see Section 5.1.1), and decision-making algorithms. These technologies are designed with a fundamental objective: enabling autonomous vehicles to track given reference trajectories, efficiently avoiding obstacles and ensuring safety. Indeed, as vehicles become increasingly autonomous, the need for effective obstacle avoidance mechanisms becomes crucial, with these systems playing an essential role in enhancing the following aspects:

- Safety: The primary importance of obstacle avoidance lies in ensuring the safety of all passengers, pedestrians, and other road users.

- Efficiency: Obstacle avoidance systems enable autonomous vehicles to choose the most efficient path by avoiding obstacles rather than stopping or slowing down when they come across one of them.
- Reliability: A robust obstacle avoidance system contributes significantly to the reliability of an autonomous vehicle by ensuring its ability to handle a wide range of scenarios and conditions.
- Legal and Ethical Considerations: Autonomous vehicles are expected to comply with traffic laws and ethical considerations.
- Public Acceptance: Effective obstacle avoidance systems can help build public trust in autonomous vehicles by demonstrating their ability to navigate safely and efficiently.

The NMPC is an attractive approach for autonomous driving due to its capability to manage input and state constraints [151]. However, generating a real-time path in an environment with obstacles continues to be a complex task. When an obstacle lies directly ahead, the controller must strategically select a turning direction to navigate around it. This becomes particularly challenging when the obstacle and the vehicle share a similar path, resulting in a non-convex optimization problem for the optimal trajectory.

In this context, we consider the scenario depicted in Figure 5.2. It presents a rural road divided into two lanes, each designated for a specific direction of travel. Additionally, at the exit of a curve, there is an obstacle due to roadworks and a truck traveling in the opposite direction. The aim of the vehicle is to avoid the construction site and return to its lane before the arrival of the truck.

5.1.3 Parallel Parking

While many advanced driver assistance systems (ADAS) are used in everyday transportation, implementing a fully autonomous application remains a challenge due to safety and legal concerns. Autonomous parking may become the first application to achieve full autonomy in the near future due to well-known environments and relatively low risks. Additionally, the increasing number of vehicles has created a significant challenge in metropolitan areas, where

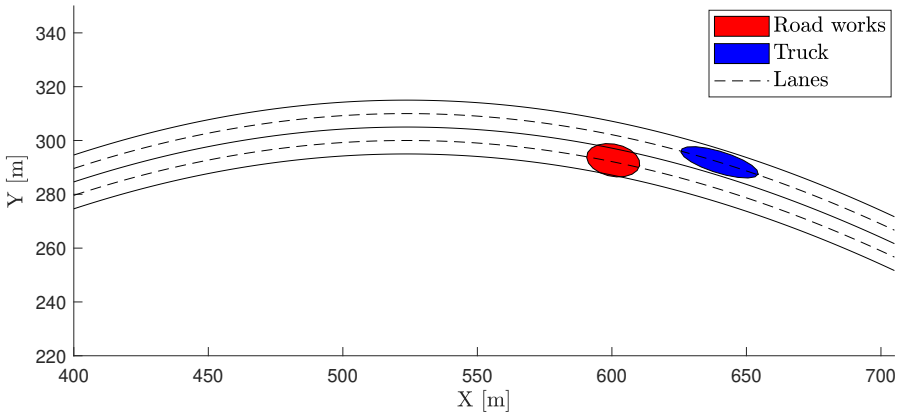

Fig. 5.2 Obstacle avoidance scenario.

finding a suitable location to park is becoming increasingly difficult. This situation is exacerbated by the shrinking size of the available spaces, making manual parking more challenging and contributing to traffic congestion. The autonomous parking technology would simplify as much as possible the actions required by the driver to park, reducing time (to perform all the maneuvers) and spaces [152]. In this regard, many algorithms for trajectory planning and control have been developed in recent years (see, e.g., [153–155]).

In this thesis, we consider the Parallel Parking scenario shown in Figure 5.3. Starting from an initial pose, the ego vehicle must first pull alongside the front vehicle (Target 1) and then perform the necessary maneuvers for entering the parking lot (Target 2) without colliding with other vehicles and obstacles.

5.1.4 Real Vehicle Model

To simulate the real vehicle, the Matlab `Vehicle Body 3DOF Dual Track` block [156] is used, which implements a rigid two-axle vehicle body model to calculate longitudinal, lateral, and yaw motion. The block accounts for body mass, aerodynamic drag, and weight distribution between the axles due to acceleration and steering. It therefore provides a good high-fidelity model for

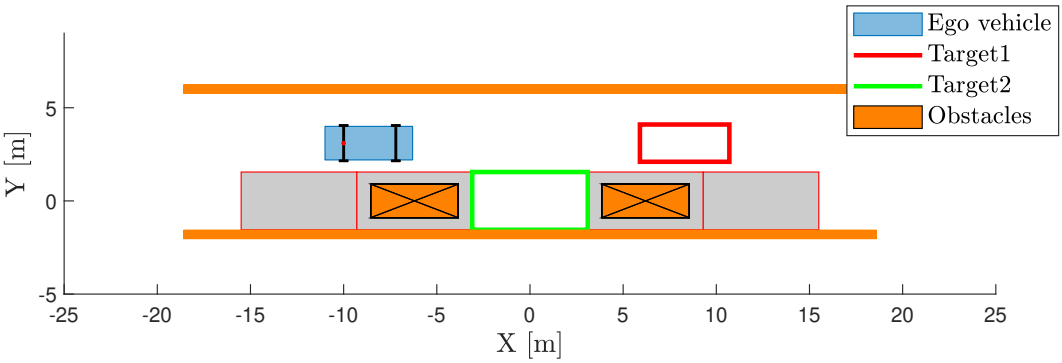

Fig. 5.3 Parallel parking scenario.

testing the developed NMPC methods and comparing them with standard NMPCs.

5.2 Case Study I: Lane Keeping on Urban Road

For this case study, four different configurations of the NMPC have been considered, evaluating their performance in terms of both computational time and accuracy. These configurations are:

1. *Standard single shooting NMPC* (*s-NMPC*): This refers to the NMPC approach with the optimization control problem formulation presented in (2.8). It does not apply any dimensionality or side-length reduction techniques.

2. *Standard multiple shooting NMPC* (*m-NMPC*): This denotes the NMPC with multiple shooting configuration, described in Section 2.2.1. Also in this case, no volume reduction techniques are used.

3. *Dimensionality reduction NMPC* (*DR-NMPC*): This configuration employes the dimensionality reduction approach developed in Chapter 3.

4. *Dimensionality and side-length reduction NMPC* (*DSR-NMPC*): This configuration uses both dimensionality and side-length reduction techniques, as detailed in Chapter 4.

The s-NMPC, DR-NMPC and DSR-NMPC configurations have been tested using both software-in-the-loop (SIL) and hardware-in-the-loop (HIL) simulations for lane-keeping maneuvers in real-world road scenarios (described in Section 5.1.1). An example is shown in Figure 5.1.

Firstly, the models representing the ego vehicle are introduced. Subsequently, the different methods are described and compared through both SIL and HIL simulations. Finally, the performance of the DSR-NMPC is evaluated in comparison to the m-NMPC. The detailed steps are outlined below.

5.2.1 Vehicle models

The models used to describe the ego vehicle are first introduced. When implementing the NMPC approach, it is important to consider two types of models: i) a "high-fidelity" plant model that replicates the real vehicle, and ii) a prediction model used by the NMPC optimization algorithm to predict future system behavior. Note that, in general, the prediction model has a lower complexity compared to the high-fidelity plant model. The Matlab `Vehicle Body 3DOF Dual Track` block (described in Section 5.1.4) is used for simulating real vehicle dynamics. In terms of the NMPC prediction model, a standard model called Dynamic Single-Track (DST) Model is used, which provides a representation of both lateral and longitudinal vehicle dynamics. This model represents a single-track vehicle, also known as a bicycle model. It simplifies a four-wheeled vehicle by assuming symmetry between the left and right sides. This allows us to model the behavior of the vehicle using only two wheels: a front wheel and a rear wheel. A graphical representation of this model is shown in Figure 5.4. Although simpler than other models, it includes the key aspects required for designing and conducting preliminary tests on vehicle

control systems. The state equations characterizing the DST are as follows:

$$\begin{aligned}
\dot{X} &= v_x \cos\psi - v_y \sin\psi \\
\dot{Y} &= v_x \sin\psi + v_y \cos\psi \\
\dot{\psi} &= \omega \\
\dot{v}_x &= v_y \dot{\psi} + a_x \\
\dot{v}_y &= -v_x \dot{\psi} + \frac{2}{m}\left(F_{yf} + F_{yr}\right) \\
\dot{\omega} &= \frac{2}{I_z}\left(l_f F_{yf} - l_r F_{yr}\right)
\end{aligned} \tag{5.1}$$

where X and Y are the coordinates of the vehicle in an inertial frame, ψ is the yaw angle, ω denotes the yaw rate, and v_x, v_y are the longitudinal and lateral speeds, respectively. The main physical parameters of the model and their values are as follows: mass $m = 1575\,\text{kg}$; moment of inertia $I_z = 4000\,\text{kg*m}^2$; distances of Center of Gravity (CoG) to front wheels $l_f = 1.2\,\text{m}$ and rear wheels $l_r = 1.6\,\text{m}$. Moreover, F_{yf} and F_{yr} are the lateral forces between the wheels and the vehicle:

$$F_{yf} = -c_f \beta_f, \qquad F_{yr} = -c_r \beta_r \tag{5.2}$$

where $c_f = 2.7 \cdot 10^4\,\text{N/rad}$ and $c_r = 2 \cdot 10^4\,\text{N/rad}$ are the front/rear cornering stiffnesses. The tire slip angles β_f and β_r are defined as:

$$\beta_f = \text{atan}\left(\frac{v_y + l_f \dot{\psi}}{v_x}\right) - \delta_f,\ \beta_r = \text{atan}\left(\frac{v_y - l_r \dot{\psi}}{v_x}\right). \tag{5.3}$$

The longitudinal acceleration a_x and the steering angle δ_f are the control variables. The output of the system is (X, Y). Note that the prediction model is a simplified version of the plant, not accounting for aerodynamic forces and weight distribution between the axles due to acceleration and steering.

5.2.2 Closed-loop system

A closed-loop system has been implemented in Simulink (see Figures 5.7 and 5.9), comprising of the plant controlled through feedback by a NMPC block. The input of the plant is the command $u = (a_x, \delta_f)$ generated by the NMPC block on the basis of the plant state and the reference trajectory. The NMPC block

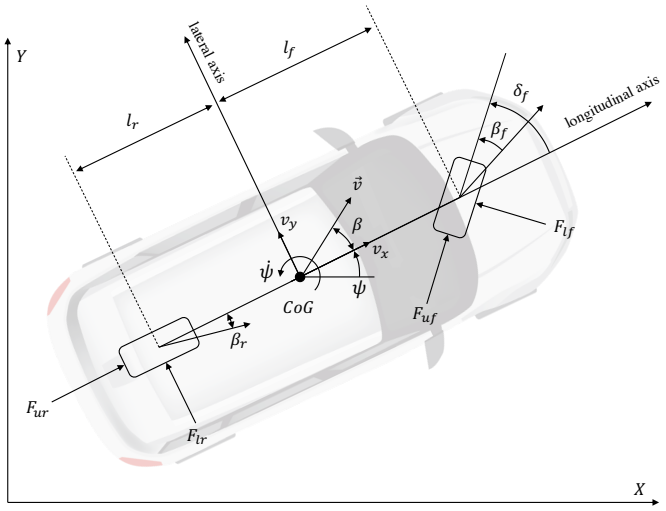

Fig. 5.4 Dynamic Single-Track Model.

can be either a standard single shooting NMPC, a dimensionality reduction NMPC or a dimensionality and side-length reduction NMPC. The description of these three different types of NMCP algorithms follows below. It should be noted that for this comparison, the multiple shooting NMPC configuration is not taken into account.

5.2.3 Standard single shooting NMPC algorithm

A standard single shooting NMPC algorithm was designed to provide both lateral and longitudinal control of the vehicle dynamics. The algorithm is formulated based on the optimization problem (2.8) and uses equations (5.1) as the internal prediction model. It takes the plant state and reference trajectory as inputs and generates the command $u_{t+1} = (a_x, \delta_f)$ to track the reference trajectory, enabling lane-keeping while maintaining a desired speed. The NMPC parameters were selected through a trial-and-error process and are listed in Table 5.6. Note that, in order to obtain the OCP (2.8), it is necessary to parametrize the command $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_T)$ through the selection matrix Γ (as described in Section 2.3). According to the value of $T = 30$ in Table 5.6 and considering that $n_u = 2$, the command $\mathbf{u}$ is a vector of 60 variables. In

order to reduce the complexity of the optimization problem, it was decided to parametrize $\mathbf{u}$ using a new command input sequence c of 20 decision variables, i.e. $n_c = 20$. In particular, the strategy involves the use of 2 c every 6 $\mathbf{u}$. For illustrative purpose, considering the first set $\mathbf{u}_{(1:6)}$, the corresponding sub-selection matrix is:

$$\mathbf{u}_{(1:6)} = \Gamma_{(1:6)}\, c_{(1:2)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{5.4}$$

This process is repeated for each group of 6 $\mathbf{u}$, obtaining the total selection matrix Γ. In other words, the prediction interval has been divided into 10 segments. Defining n_s as the number of segments, this means that $n_s = 10$. Subsequently, for each segment, the pair of control commands (a_x, δ_f) is considered. This results in a total of 20 command samples: 10 for longitudinal acceleration a_x and 10 for steering angle δ_f.

Table 5.1 NMPC design parameters.

Parameter	**Value**
T_s	0.1 s
T	30
Q	$\mathrm{diag}(5, 5)$
R	$\mathrm{diag}(0.5, 1)$
Upper bounds	$[3\,\mathrm{m/s^2}, \pi/8, 3\,\mathrm{m/s^2}, \pi/8$
Lower bounds	$[-3\,\mathrm{m/s^2}, -\pi/8, -3\,\mathrm{m/s^2}, -\pi/8]$

5.2.4 Dimensionality Reduction NMPC

The practical implementation of the procedure outlined in Chapter 3 is now discussed, focusing on the specific case of lane-keeping maneuvers.

Offline data generation

A Monte Carlo campaign of 100 simulations of the closed-loop system with the standard NMPC algorithm was carried out, considering urban roads (such the one in Figure 5.1) with varying curvatures and assuming a speed of 40 km/h. Note that: i) to obtain always a different road for each simulation, the curvatures were modified in a pseudo-random way using the Latin Hypercube Sampling (LHS) technique (see [133]); ii) to find pseudo-optimal minima, the optimization problem was solved using the Matlab function `fmincon` with the Sequential Quadratic Programming (SQP) algorithm, initialized at various points within the search domain in order to enhance its exploration phase. This simulation campaign resulted in a dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{M}$ containing approximately $M = 3e4$ samples. It is noteworthy that while the current dataset was constructed through closed-loop simulations, it is also possible to generate it in an "open-loop" fashion. This alternative approach would involve evaluating the output of the NMPC law for different values of the regressor.

Offline selection of the relevant decision variables

In the standard NMPC, the use of 10 segments implies that each command sample $\tilde{u}_l$ in the dataset is a vector of 20 elements. To reduce the complexity of the algorithm, it is possible to identify the decision variables that significantly influence the optimization process by computing their mean-square gradient, as described in Section 3.4.2. Practically, starting with the dataset obtained in the previous step, the vectors c_r and c_a are computed along with their corresponding selection matrices Γ_r and Γ_a using the Algorithm 4. The process of choosing the relevant components involves the selection of a proper threshold, denoted as ζ. In this case, $\zeta = 0.1$ is considered, indicating that only components with a mean-square gradient greater than 10% of the total mean-square gradient will be considered relevant. The choice of the threshold depends on practical considerations and the specific scenario. Figure 5.5 presents the obtained results. For the sake of simplicity, we consider the segments instead of the individual decision variables c. It can be observed that only the first three segments have a mean-square gradient greater than the threshold, implying that only the first 6 decision variables c can be considered significant. As a result, the vector c_r

can be defined as:

$$c_r = \begin{pmatrix} c_1, & c_2, & c_3, & c_4, & c_5, & c_6 \end{pmatrix} \in \mathbf{R}^{(6\times 1)}. \tag{5.5}$$

Furthermore, the vector c_a and the selective matrices Γ_r and Γ_a can be trivially computed. They are reported below:

$$\begin{aligned} c_a &= \begin{pmatrix} c_7, & c_8, & \dots & c_{20} \end{pmatrix} \in \mathbf{R}^{(14\times 1)}, \\ \Gamma_r &= \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} = \begin{bmatrix} I_6 \\ \mathbf{0}_{(14,6)} \end{bmatrix} \in \mathbb{R}^{20\times 6}, \\ \Gamma_a &= \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{(6,14)} \\ I_{14} \end{bmatrix} \in \mathbb{R}^{20\times 14}. \end{aligned} \tag{5.6}$$

Offline approximation of the NMPC control law

Based on the dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{3e4}$, the approximated control law ϕ_s was computed using the Nonlinear Sparse Identification method described in Section 3.3.1. It should be noted that, to reduce the number of components in the regressor $\{\tilde{w}_l\}_{l=1}^{M} = \{\tilde{x}_l, \tilde{\mathfrak{r}}_l\}_{l=1}^{M}$ with $\tilde{x}_l \in \mathbb{R}^{n_x}$ and $\tilde{\mathfrak{r}}_l = (\tilde{\mathfrak{r}}_{l1}, \dots, \tilde{\mathfrak{r}}_{lT}) \in \mathbb{R}^{n_x \times T}$, it was decided to select only a subsequence of 3 samples of the reference $\tilde{\mathfrak{r}}_l$ between 1 and T. In particular, the first, middle, and last samples were used. This source of approximation, in combination with the use of the sparse function ϕ_s, introduces errors that can be taken into account in the parameter μ (see Section 3.4.3). For this case study, $\mu = 0.01$ was chosen.

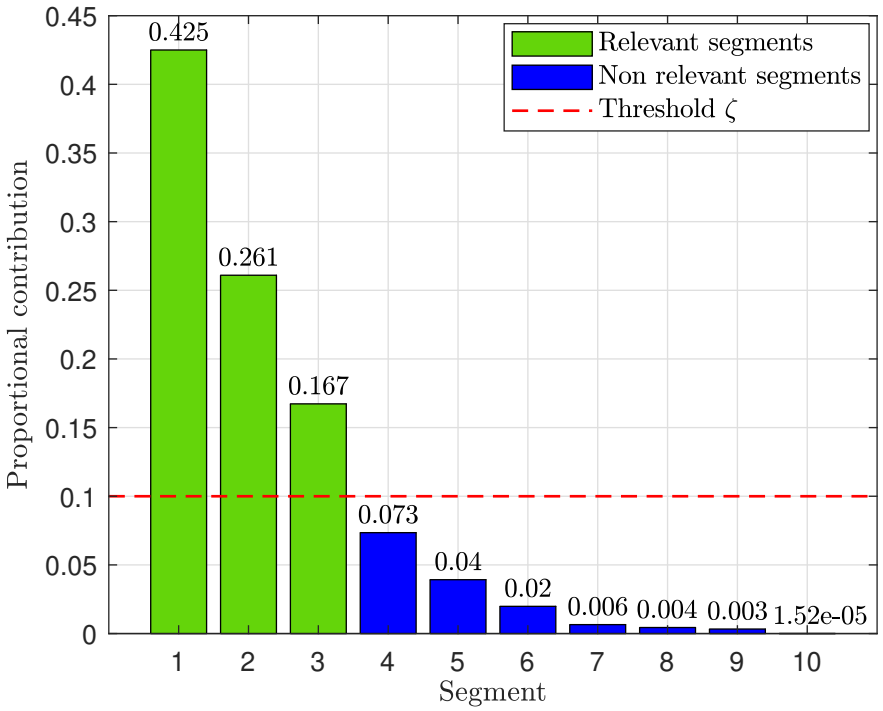

Fig. 5.5 Proportional contribution of each segment with respect to the total mean square gradient.

Online optimization on the reduced-dimensionality OCP

At each time t, given the regressor w_t, the approximated control law ϕ_s is applied online to compute the approximated solution

$$\hat{c} = \phi_s(w_t)\Gamma_r\hat{c}_r + \Gamma_a\hat{c}_a, \tag{5.7}$$

with Γ_r and Γ_a defined in Eq. (5.6). Then, the OCP (3.45) is solved using only the most relevant decision variables c_r, considering $\hat{c}_r$ as warm start. The other variables c_a are kept constant and set to the values $\hat{c}_a$. This approach leads to a reduction of the computational complexity as already discussed theoretically in Chapter 3 and as will be shown numerically in Section 5.2.6.

5.2.5 Dimensionality and Side-length Reduction NMPC

Below are described only the additional steps required to achieve a simultaneous reduction of both dimensionality and side-length of the search domain.

Offline data reduction through clustering

Since the complexity of an SM method increases with the size of the dataset, a clustering procedure was carried out to reduce the dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{3e4}$. Specifically, the K-medoids clustering method with the CLARA algorithm was used. After multiple attempts, a reduced set $\mathcal{D}_r = \{\tilde{w}_{lr}, \tilde{u}_{lr}\}_{lr=1}^{1e3}$ was found as an "optimal" compromise between data quantity (and thus complexity of the SM method) and exploration of the control law domain.

Offline bounds definitions of the most relevant decision variables

Based on the reduced dataset $\mathcal{D}_r = \{\tilde{w}_{lr}, \tilde{u}_{lr}\}_{lr=1}^{1e3}$ and the approximation ϕ_s, the corresponding tight bounds $\overline{\phi}$ and $\underline{\phi}$ of the most significant variables c_r were computed using the SM approach detailed in Section 4.1.3. Figure 5.6 shows the approximation and the relative bounds of c_{r_2}, i.e. the first steering control command. As can be seen, the bounds were reduced by approximately tenfold compared to the original ones.

Online optimization on the reduced dimensionality and side-length OCP

At each instant t, given the regressor w_t, the approximated solution

$$\hat{c} = \phi_s(w_t) = \Gamma_r \hat{c}_r + \Gamma_a \hat{c}_a \tag{5.8}$$

is computed online. The functions $\overline{\phi}(w_t)$ and $\underline{\phi}(w_t)$ are used to obtain the following reduced search domain:

$$\mathcal{C}_c^{ds} = \prod_j \mathcal{C}_{c_j}^{ds} = [\underline{\phi}_j(w_t), \overline{\phi}_j(w_t)], \text{ for } j = 1, \ldots, n_{cr}. \tag{5.9}$$

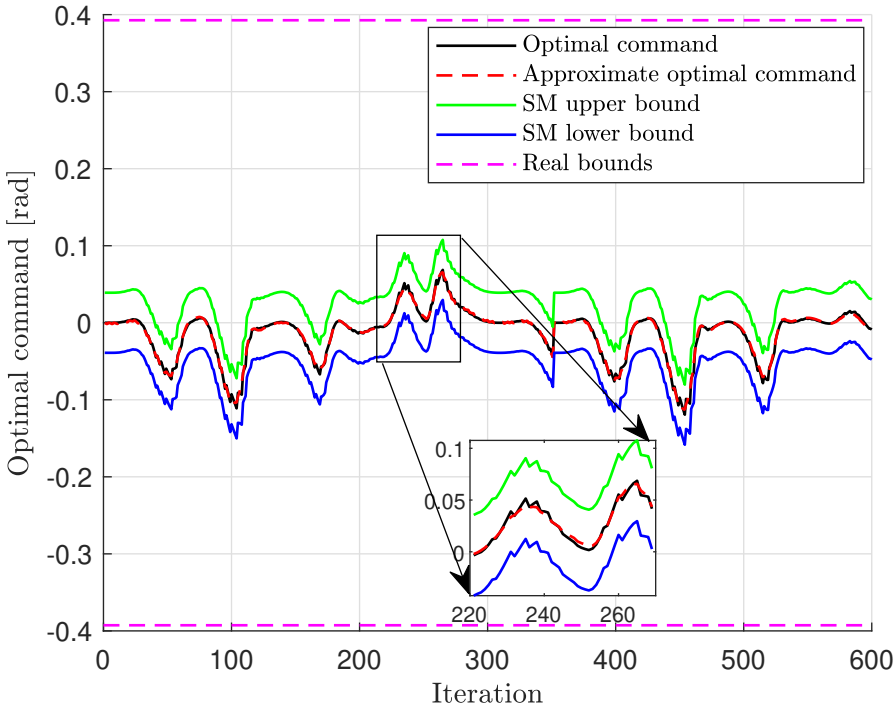

Fig. 5.6 Set Membership approximation of c_{r_2}.

Then, the OCP (4.36) is solved by optimizing only the most relevant decision variables c_r, using $\hat{c}_r$ as warm start and $\mathcal{C}_c^{ds}$ as search domain, while leaving the remaining variables fixed at the values $\hat{c}_a$. It is important to highlight that the use of the SM approach further decrease the overall volume of the search domain, compared to just employing dimensionality reduction. This, in turn, results in an additional reduction of the computational complexity, improving the rate of convergence. This is numerically shown in Section 5.2.6.

5.2.6 Performance Comparison using SIL

The comparison of the three configurations described above is performed firstly using SIL simulations.

Software-in-the-loop. Software-In-the-Loop (SIL) is a simulation methodology that is extensively used in the field of systems engineering, particularly within the realms of automotive and aerospace development. This technique

involves the execution of embedded software algorithms within a simulated environment that emulates the behavior of the physical system. The main goal of SIL is to validate the accuracy and robustness of software algorithms before deploying them onto actual hardware. By integrating the software with a mathematical model of the system and running it on a high-fidelity simulation platform, SIL provides a controlled setting that can replicate real-world conditions without the associated risks and costs. It plays an essential role in the model-based design (MBD) process, allowing for early detection and correction of software defects. It facilitates iterative development and testing, enabling engineers to enhance their designs with a high degree of accuracy. By employing SIL, developers can ensure that the software works correctly in response to a wide range of inputs and scenarios, thereby improving the reliability of the final product.

The two numerical nonlinear optimization algorithms used for the comparison are:

- Sequential Quadratic Programming (SQP), see Section 2.5.2 for more detail.
- Particle Swarm Optimization (PSO), see the following paragraph for a brief description of this algorithm.

Particle Swarm Optimization Inspired by the collective movements of birds and fish flocks, Particle Swarm Optimization is a population-based optimization technique introduced by Kennedy and Eberhart in 1995 for continuous search spaces [20]. Unlike single-point search methods such as gradient descent, PSO leverages a swarm of particles to explore the search space in pursuit of the optimal solution. These particles, initialized randomly within the search space, iteratively update their positions based on their individual best experiences and the best solution of the swarm. This collaborative approach promotes the exploration of diverse regions, enhancing the probability of finding the optimal solution. Furthermore, one of the strengths of PSO lies in its gradient-free nature; unlike gradient-based methods that require readily available and well-behaved gradients, PSO can tackle problems with non-differentiable or computationally expensive objective functions.

The selection of these two algorithms aims to highlight the versatility of the proposed methods, demonstrating their effectiveness regardless of the specific optimization algorithm employed. Indeed, it is noteworthy that SQP is a local optimization algorithm, while PSO is a global one.

In order to evaluate and compare the performance of the three NMPC configurations, another Monte Carlo campaign of 100 simulations was carried out, using urban roads different from those considered during the offline data collection. In each trial, three simulations of the closed-loop system were run: one with the standard NMPC (St-NMPC) controller, another with the DR-NMPC algorithm, and the third one with the DSR-NMPC algorithm. Note that the plant was always the same. The simulations were run on a Dell Precision 5820 (Processor: Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz). The optimization problem was solved using the Matlab function `fmincon`. The corresponding Simulink scheme used for the SIL simulation is shown in Figure 5.7. The performance of the three NMPC algorithms was compared using the following indexes:

1. Mean number of iterations (Num. of Iteration);
2. Mean number of evaluated cost functions (Num. of Cost Fun.) for finding the minimum;
3. Mean total computational time for solving an OCP (Comp. Time);
4. Mean Root-Mean-Square (RMS) Error for the cross track error (RMS CT. Err.).

It should be noted that the term "Mean" represents the average number of iterations, evaluated cost functions, computational times, and RMS errors throughout the Monte Carlo simulations. For example, the mean number of evaluated cost functions (M_{ecf}) means:

$$M_{ecf} = \frac{\sum_{t=1}^{n_t} ecf_t}{n_t} \tag{5.10}$$

where ecf_t is the number of evaluated functions for the t-th time index and n_t is the number of iterations required to complete the simulation.

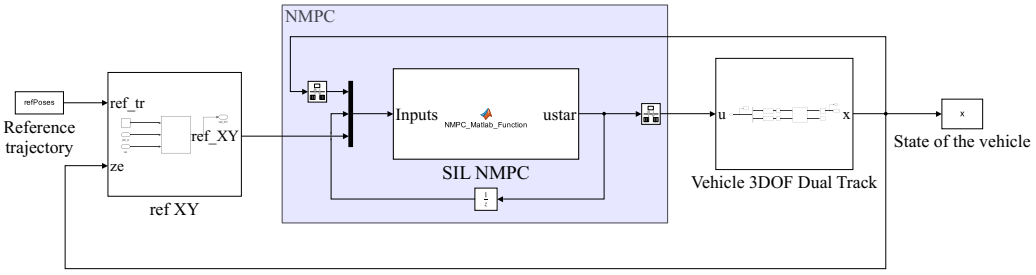

Fig. 5.7 Closed-loop system in Simulink for SIL.

In Tables 5.2 and 5.3, the resulting performance indexes for the three configurations using both the SQP and PSO algorithms are presented. Below are the comments for each of the two developed methods.

Dimensionality Reduction NMPC

By using dimensionality reduction method, the optimization problem is transformed from one with 20 variables to one involving only 6 relevant variables c_r. This results in a decrease in both the number of iterations and, more importantly, the number of cost function evaluations, as theoretically demonstrated in Section 3.5. Consequently, there is a significant reduction in the computational time required to solve the OCP. Specifically, for both the SQP and the PSO, we observe the following benefits compared to the s-NMPC: i) more than a halving of the number of iterations; ii) a reduction of over five times in the number of cost function evaluations; iii) an improvement in computational time by nearly an order of magnitude. Importantly, in the case of the SQP, the dimensionality reduction does not degrade the quality of the solution found, since the cross-track error remains practically unchanged. Regarding the PSO, it is worth noting that the standard NMPC could not be implemented in real-time as it requires a computational time greater than the sampling time ($T_s = 0.1\,\mathrm{s}$). This issue is solved by introducing dimensionality reduction. Furthermore, the solution quality is enhanced, since the warm start of the PSO algorithm exploits a model identified from data generated using a SQP method.

Table 5.2 Comparison between the three NMPC methods using SQP algorithm with SIL simulations.

	SQP		
	s-NMPC	DR-NMPC	DSR-NMPC
Num. of Iteration	31.7	12.96	2.82
Num. of Cost Fun.	788.33	143.1	35.53
Comp. Time [ms]	29.6	4	1.2
RMS CT. Err. [m]	0.098	0.0995	0.0991

Table 5.3 Comparison between the three NMPC methods using PSO algorithm with SIL simulations.

	PSO		
	s-NMPC	DR-NMPC	DSR-NMPC
Num. of Iteration	110.5	68.73	20.97
Num. of Cost Fun.	2210	412.4	125.8
Comp. Time [ms]	165.4	33.5	10.6
RMS CT. Err. [m]	0.2264	0.1377	0.0998

Dimensionality and Side-length Reduction NMPC

The integration of the side-length reduction method further enhances the performance of the two algorithms. Specifically, as described theoretically in Section 4.3, it improves the convergence rate for Sequential Quadratic Programming and reduces the exploration phase for Particle Swarm Optimization, thereby decreasing the number of iterations required to find a solution. Consequently, this leads to fewer cost function evaluations and reduced computational times. For SQP, times close to milliseconds are achieved, while for PSO, times are in the order of tens of milliseconds. It should be noted that the ability to focus the optimization in the region of the search domain containing the minimum allows for a better solution compared to dimensionality reduction alone. In fact, a higher accuracy is achieved, resulting in a reduction of the cross-track error.

5.2.7 Performance Comparison using HIL

The three NMPC methods are now compared using hardware-in-the-loop (HIL) simulations.

Hardware-In-the-loop. Hardware-In-the-Loop (HIL) is an advanced simulation technique that integrates hardware components within the simulation loop. This approach is crucial in the development and validation of complex embedded systems (control algorithms and physical plant systems), particularly in the automotive and aerospace industries. In this thesis, the HIL simulation is used to verify the behavior of the developed control algorithms in real-time, while executing on the target processor. In particular, the HIL is leveraged to simulate the interaction between the embedded software and the hardware, without the need for a physical prototype. This is achieved by running the control code directly on an electronic board and interfacing it with a simulated model of the system, which is executed on a host computer. The simulated environment replicates the dynamics of the actual system, allowing for a thorough analysis of the control strategy under various scenarios. One of the primary advantages of HIL is the significant reduction in development time and cost. By circumventing the necessity for early-stage physical prototypes, engineers can iterate and refine their designs rapidly. Moreover, HIL facilitates the identification of issues related to timing and integration, which are critical in real-time systems. Overall, HIL provides a more accurate representation of the system behavior by incorporating the actual processor into the simulation, thus offering a balance between simulation fidelity and resource consumption.

The Nvidia Jetson Nano board has been chosen to implement the three NMPC methods.

Nvidia Jetson Nano. Jetson Nano is a small, powerful computer for embedded applications (see Figure 5.8 [157]). It is engineered around a 64-bit quad-core Arm Cortex-A57 Central Processing Unit (CPU) that operates at a frequency of 1.43GHz. This CPU is complemented by an NVIDIA Maxwell Graphics Processing Unit (GPU) equipped with 128 CUDA cores. This GPU configuration is capable of achieving a computational speed of 472 GFLOPs when operating in FP16 mode. The module is further enhanced with 4GB of

64-bit Low Power Double Data Rate 4 (LPDDR4) Random Access Memory (RAM) and 16GB of embedded MultiMediaCard (eMMC) storage. It operates on the Linux for Tegra operating system. The physical dimensions of the module are 70mm by 45mm, and it features a 260-pin Small Outline Dual In-line Memory Module (SODIMM) connector. This connector facilitates the breakout of various interfaces, including video, audio, Universal Serial Bus (USB), and networking. The presence of this connector enables the module to be interfaced with a compatible carrier board.

Fig. 5.8 NVIDIA Jetson Nano board.

For the comparison between the three NMPC methods, only the Sequential Quadratic Programming (SQP) algorithm was considered. Indeed, in the case of Particle Swarm Optimization (PSO), even when used with Software-in-the-Loop (SIL), it fails to achieve real-time NMPC (see s-NMPC in Table 5.3). Additionally, the inherent performance limitations introduced by the algorithm implementation on a physical board would further worsen these shortcomings, rendering PSO unsuitable for a meaningful performance comparison within the scope of this analysis. Conversely, SQP is a well-established optimization technique frequently employed in real-time NMPC applications due to its efficiency and convergence properties. Therefore, SQP represents the most appropriate choice for this comparative study.

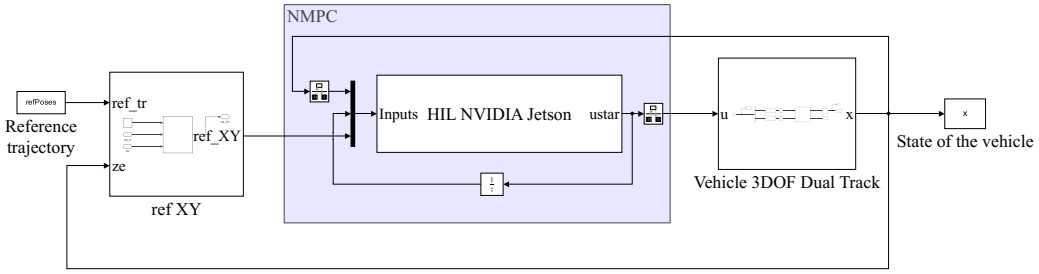

Fig. 5.9 Closed-loop system in Simulink for HIL.

The simulation environment is shown in Figure 5.9. All the blocks are designed in Simulink, with the exception of the controller implemented on the Jetson Nano board. The performance of the three NMPC algorithms was compared using the same indexes as before (see Section 5.2.6). Please note that all performances are expected to be very similar to those obtained with SIL, except for the Computational Time, which should significantly increase with the use of the board.

The obtained results are reported in Table 5.4. As can be seen, the HIL validation becomes crucial when considering real-time implementation of optimization algorithms. Indeed, while the standard NMPC in SIL simulations achieved computation times well below the sampling time (see Table 5.2), the same method proved to be too computationally expensive for real-time hardware implementation at a sampling rate of $T_s = 0.1$ seconds (see the NMPC parameters in Table 5.6). This highlights the limitations of SIL simulations for real-time control applications. Regarding the two developed methods, they demonstrate their effectiveness for real-time control by achieving sampling rates significantly lower than the required threshold (i.e., 0.1 s). In particular, the DSR-NMPC method achieves a mean computational time of approximately 7 ms, a remarkable result that enables its use in fast real-time applications. The other performance metrics remain consistent with those presented in Table 5.2. This is due to the fact that the board only affects the computation time, leaving the other performance indexes unaffected.

Table 5.4 Comparison between the three NMPC methods using SQP algorithm with HIL simulations.

	SQP		
	s-NMPC	DR-NMPC	DSR-NMPC
Num. of Iteration	31.67	12.97	2.79
Num. of Cost Fun.	787.34	143.2	34.9
Comp. Time [ms]	151.8	24.3	7.4
RMS CT. Err. [m]	0.098	0.0993	0.0985

5.2.8 Performance comparison with NMPC in Multiple Shooting configuration

Direct multiple shooting methods belong to the group of simultaneous approaches for the numerical solution of dynamic optimization problems. They are characterized by dividing the time horizon into multiple segments and introducing additional intermediate state variables into the optimization problem. This approach allows for the explicit incorporation of knowledge about the system dynamics into the problem by an adequate choice of initial guesses for these additional variables. For more detail please refer to Chapter 2.

In MATLAB, CasADi provides a powerful tool for implementing NMPC with the multiple shooting approach [70]. CasADi, an open-source software tool, has emerged as a powerful solution for nonlinear optimization and algorithmic differentiation (AD). It is designed to facilitate the efficient implementation of various techniques for numerical optimal control, making it suitable for both offline and online applications. The core of CasADi is a symbolic framework that includes both forward and reverse modes of AD on expression graphs. This unique feature enables the construction of gradients, large-and-sparse Jacobians, and Hessians, thereby enhancing the efficiency of numerical computations. CasADi is interfaced with a variety of Nonlinear Programming solvers. Among these, the most widely used is `IPOPT`. This open-source solver employs a primal-dual interior point method, which is highly effective for solving large-scale optimization problems.

For the comparison, we take into account the following NMPC configurations: standard multiple shooting NMPC (implemented using CasADi and `IPOPT`) and

Table 5.5 Comparison between m-NMPC with `IPOPT` and DSR-NMPC with SQP.

	m-NMPC	DSR-NMPC
Comp. Time [ms]	31.6	9.2
RMS CT. Err. [m]	0.098	0.099

DSR-NMPC (implemented using `fmincon` and SQP). The evaluated metrics are the mean total computational time for solving the OCP (Comp. Time) and the mean root-mean-square error for the cross track error (RMS Ct. Err.).

Remark. *It is important to note that in this scenario, we consider SIL simulations using the Interpreted MATLAB Function in Simulink. This is due to the fact that CasADi cannot be implemented using a Matlab Function. However, the use of Interpreted MATLAB Functions results in a significant computational overhead, often increasing simulation time by an order of magnitude compared to standard MATLAB functions.*

The obtained results are presented in Table 5.5. As can be seen, the combination of both dimensionality and side-length reduction methods leads to a more computationally efficient single-shooting NMPC approach, implemented with `fmincon`, compared to a multiple shooting NMPC, implemented with CasADi and the solver `IPOPT`. It should be noted that the CasADi implementation, being symbolic and using algorithm differentiation along with `IPOPT`, generally outperforms the use of `fmincon` in the standard NMPC case. This further highlights the effectiveness of the developed approaches, even against powerful tools like CasADi. Moreover, for the cross track error, both methods achieve comparable results.

5.3 Case Study II: Obstacle Avoidance on Rural Roads

For this case study, we consider the following NMPC configurations:

1. *Standard single shooting NMPC (s-NMPC)*: The same configuration considered in the previous case study.

2. *Dimensionality and side-length reduction NMPC (DSR-NMPC)*: This configuration uses both dimensionality and side-length reduction techniques, as detailed in Chapter 4.

3. *Side-length reduction NMPC (SR-NMPC)*: This configuration employs the side-length reduction approach described in Section 4.2.4 without using the dimensionality reduction.

These configurations have been tested using software-in-the loop (SIL) for an autonomous obstacle avoidance in real-world road scenarios (described in Section 5.1.2). An example is shown in Figure 5.2. To ensure that the ego vehicle safely maneuvers around the construction site and returns to its designated lane before the truck arrives, safety ellipses are strategically designed around obstacles:

$$\frac{(X - X_{v_1})^2}{a_1^2} + \frac{(Y - Y_{v_1})^2}{b_1^2} \geq 1, \tag{5.11a}$$

$$\frac{(X - X_{v_2})^2}{a_2^2} + \frac{(Y - Y_{v_2})^2}{b_2^2} \geq 1. \tag{5.11b}$$

The equation (5.11a) describes the roadworks obstacle centered at X_{v_1} and Y_{v_1}, with major and minor axes represented by a_1 and b_1, respectively. On the other hand, equation (5.11b) refers to the obstacle posed by the truck. In order to create a critical scenario for the vehicle, forcing it to return promptly to its lane, the truck is considered to be in motion at a speed of 20 km/h. To account for this during optimization, a prediction of its position is performed, instant by instant, by using a rough estimate of the truck velocity. This implies that the center characterized by X_{v_2} and Y_{v_2} is not fixed but changes at every time instant. Note that equations (5.11a)-(5.11b) will be included as constraints within the NMPC optimization problems.

5.3.1 Vehicle models

As in the previous case, the Matlab `Vehicle Body 3DOF Dual Track` block is used for simulating the real vehicle, while the DST model is used for the NMPC internal prediction model. The model state is $x = (X, Y, \psi, v_x, v_y, \omega)$, while

the longitudinal acceleration and the steering angle are the control variables: $u = (a_x, \delta_f)$. The model output is (X, Y).

5.3.2 Closed-loop system

The same closed-loop system of Section 5.2.2 has been implemented in Simulink, with the plant being controlled via feedback by an NMPC block. The input of the plant is the command $u = (a_x, \delta_f)$ generated by the NMPC block using information from the plant state and reference trajectory. The NMPC block can be either a standard single shooting NMPC, a side-length reduction NMPC or a dimensionality and side-length reduction NMPC. These three NMPC algorithms are described below.

5.3.3 Standard single shooting NMPC algorithm

A standard single shooting NMPC algorithm was developed to handle the lateral and longitudinal control of vehicle dynamics. The algorithm relies on solving the optimization problem (2.8), using equations (5.1) as the internal prediction model. Its inputs include the plant state and the reference trajectory. The output is the command $u_{t+1} = (a_x, \delta_f)$, used to track the reference trajectory, allowing the vehicle to accomplish the lane keeping task, and to avoid potential obstacles along the path, while maintaining a desired speed. This command was parametrized considering a new command input sequence c of 8 decision variables, i.e. $n_c = 8$. This means that the prediction interval was divided into 4 segments, that is, $n_s = 4$. As a result, in the prediction interval $[t, t+T]$, there are a total of 8 command samples: 4 for the longitudinal acceleration a_x and 4 for the steering angle δ_f. The values of the NMPC parameters were chosen through a trial-and-error procedure and are listed in Table 5.6.

5.3.4 Dimensionality and side-length Reduction NMPC

The steps required for the implementation of the approach described in Chapter 4 are now discussed.

Table 5.6 NMPC design parameters.

Parameter	**Value**
T_s	0.1 s
T	30
Q	$\text{diag}(1,1)$
R	$\text{diag}(0.01,1)$
Upper bounds	$[3\,\text{m/s}^2, \pi/4, 3\,\text{m/s}^2, \pi/4]$
Lower bounds	$[-3\,\text{m/s}^2, -\pi/4, -3\,\text{m/s}^2, -\pi/4]$

Offline data generation

A Monte Carlo campaign of 500 simulations of the closed-loop system with the standard NMPC algorithm was carried out, considering rural roads (such as the one in Figure 5.12) with slightly different curvatures (obtained through the LHS technique) and assuming a speed of 60 km/h.

This simulation campaign provided a dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{M}$ of about $M = 2e5$ samples. Note that here the dataset was constructed by means of closed-loop simulations but it is also possible to generate it in "open-loop", just by evaluating the output of the NMPC law for different values of the regressor.

Offline dataset reduction through clustering

To reduce the size of the dataset obtained in the previous step a clustering procedure was used. The K-medoids clustering method with the CLARA algorithm was employed. A reduced set $\mathcal{D}_r = \{\tilde{w}_{lr}, \tilde{u}_{lr}\}_{lr=1}^{1e4}$ was constructed, giving an optimal compromise between amount of data, memory usage and exploration of the control law domain.

Offline selection of the relevant decision variables

In the standard NMPC, using 4 segments implies that each command sample $\tilde{u}_l$ in the dataset is a vector of 8 elements. To simplify the algorithm, it is possible to identify the relevant variables through the method described in Chapter 3. Starting with the dataset obtained in the previous step, we compute the vectors c_r and c_a, along with their respective selection matrices Γ_r and Γ_a, using Algorithm 4. As in the previous case study, a threshold $\zeta = 0.1$ has

been chosen. Figure 5.10 presents the results. It can be seen that only the first segment has a mean-square gradient greater than the threshold, meaning that only the first 2 decision variables c are significant. Consequently, the vector c_r can be defined as:

$$c_r = \begin{pmatrix} c_1, & c_2 \end{pmatrix} \in \mathbf{R}^{(2\times 1)}. \tag{5.12}$$

Additionally, the vector c_s and the selective matrices Γ_r and Γ_s are as follows:

$$\begin{aligned} c_a &= \begin{pmatrix} c_3, & c_4, & c_5, & c_6, & c_7, & c_8 \end{pmatrix} \in \mathbf{R}^{(6\times 1)}, \\ \Gamma_r &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} I_2 \\ \mathbf{0}_{(6,2)} \end{bmatrix} \in \mathbb{R}^{8\times 2}, \\ \Gamma_a &= \begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{(2,6)} \\ I_6 \end{bmatrix} \in \mathbb{R}^{8\times 6}. \end{aligned} \tag{5.13}$$

Offline approximation of the NMPC control law

Based on the dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{2e5}$, the approximated control law ϕ_s was computed using the Nonlinear Sparse Identification method described in Section 3.3.1. As in the previous case study, it was decided to select only a subsequence of 3 samples of the reference $\tilde{\mathfrak{r}}_l$ between 1 and T. In order to take into account this source of approximation as well as the error introduced by the sparse function ϕ_s, $\mu = 0.05$ was chosen.

Offline bounds definition of the most relevant decision variables

Using the reduced dataset $\mathcal{D}_r = \{\tilde{w}_{lr}, \tilde{u}_{lr}\}_{lr=1}^{1e4}$ and the approximation ϕ_s, the corresponding tight bounds $\overline{\phi}$ and $\underline{\phi}$ of the most relevant variables c_r were computed employing the SM approach detailed in Section 4.1.3. The approxi-

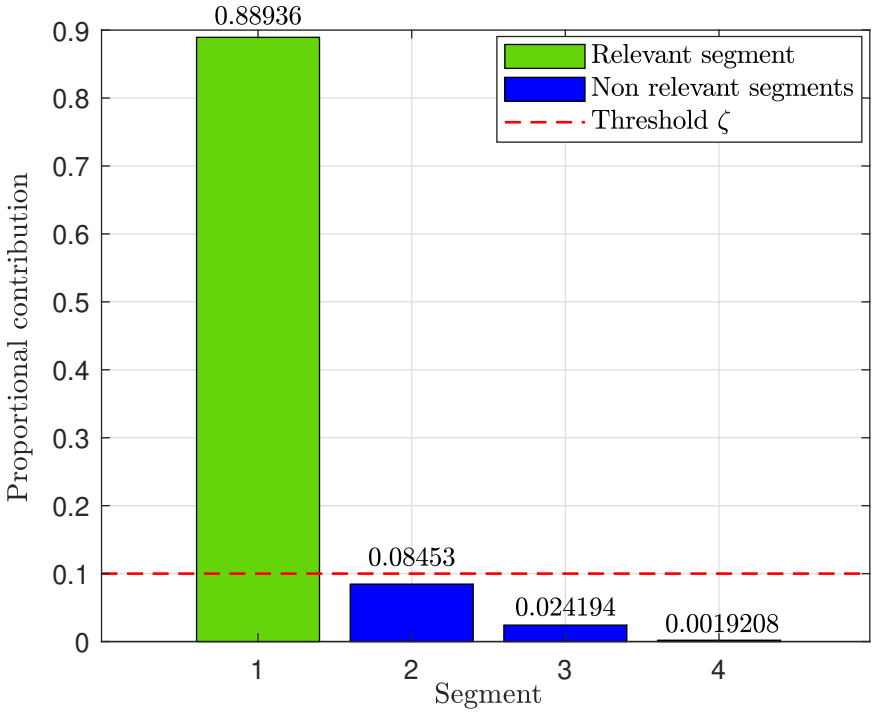

Fig. 5.10 Proportional contribution of each segment with respect to the total mean square gradient.

mation ϕ_s and the bounds for one of the steering angle commands are shown in Figure 5.11. It can be observed that the interval between the bounds is reduced more than 10 times compared to the original one.

Online optimization on the reduced dimensionality and side-length OCP

The DSR-NMPC consists in applying online the OCP (4.36), considering the same procedure described in Section 5.2.5.

5.3.5 Side-length reduction NMPC

This approach is a subcase of the method developed in Chapter 4 (see Section 4.2.4). In this case, the OCP encompasses all the decision variables without

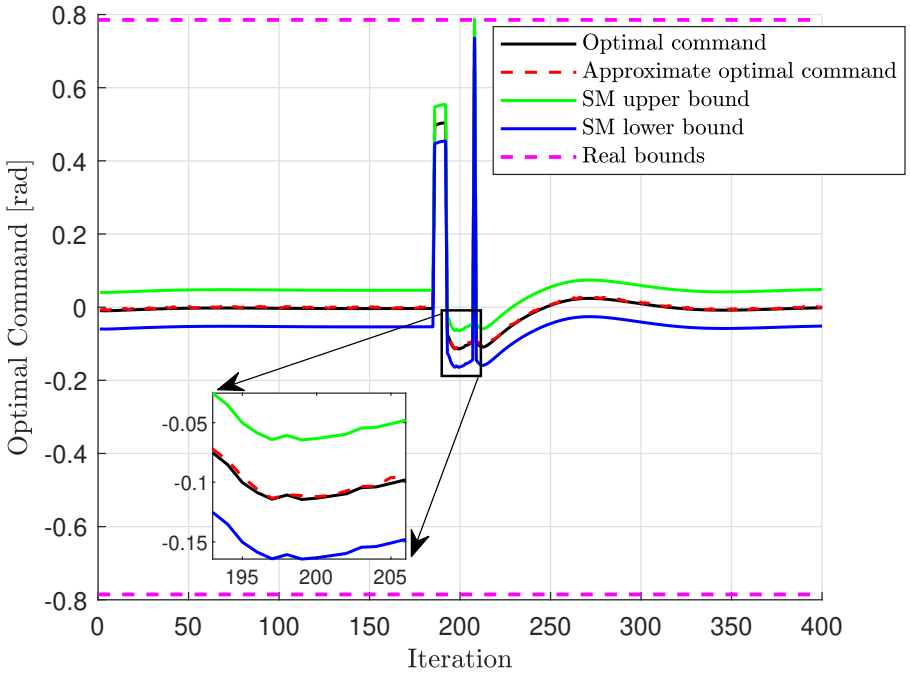

Fig. 5.11 Set Membership approximation of δ_f.

distinction between relevant and non-relevant. This means that:

$$c_r \equiv c,\ c_a = 0,\ \Gamma_r = I_8,\ \Gamma_a = \mathbf{0}_{(8,1)}. \tag{5.14}$$

During the offline phase, the approximation ϕ_s is computed based on the dataset $\mathcal{D}$, while the SM approach is used to compute the corresponding bounds $\overline{\phi}$ and $\underline{\phi}$ for all the variables c using the reduced dataset $\mathcal{D}_r$. Then, during the online procedure, the approximated control law ϕ_s is applied to compute the approximated decision variables $\hat{c}$, while the function $\overline{\phi}$ and $\underline{\phi}$ are used to reduce the search domain in which the decision variables can be found, obtaining

$$\mathcal{C}_c^s = \prod_{j=1}^{8} \mathcal{C}_{c_j}^s = \prod_{j=1}^{8} [\underline{\phi}_j(w_t), \overline{\phi}_j(w_t)]. \tag{5.15}$$

The OCP is performed for all the decision variables, considering $\hat{c}$ as warm start and $\mathcal{C}_c^s$ as search domain, obtaining the Eq. (4.40).

It should be noted that in both DSR-NMPC and SC-NMPC, equations (5.1) are used as the internal prediction model. The design parameters are the same as those of the standard NMPC, see Table 5.6.

5.3.6 Comparison between DSR-NMPC, SR-NMPC and s-NMPC.

A Monte Carlo campaign of 100 trials was carried out, considering rural roads with different curvatures compared to those used during the offline data generation phase. The reference velocity was 60 km/h.

The simulations were run on a Dell Precision 5820 (Processor: Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz). The optimization problems were solved using the Matlab function `fmincon` with the SQP algorithm. The performance of the three NMPC algorithms was compared using the following indexes:

1. Mean number of evaluated cost functions (Num. of Cost Fun.) for finding the minimum;
2. Mean NMPC total computational time (NMPC Time);
3. Mean optimization computational time (Opt. Time);
4. Mean Root-Mean-Square (RMS) Error both for the Lateral Error (Lat. E.) and the Orientation Error (Orient. E.);
5. Mean minimum distance from the obstacles (Min. Dist.).

Note that "NMPC total computational time" refers to the elapsed time between when the NMPC receives inputs and when it produces the commands. This encompasses not only the optimization part but also the entire pre-processing phase. Specifically, in the case of DSR-NMPC and SR-NMPC, it includes the use of the approximation functions to obtain the warm start and the bounds. On the other hand, "Optimization computational time" only considers the time required by the solver (`fmincon`) to find the optimal command. Table 5.7 shows the mean value of the performance indexes for the three NMPC algorithms. The term "Mean" represents the average number of evaluated cost functions, computational times, RMS errors and minimum distances throughout

Table 5.7 Comparison between the three NMPC methods using SQP algorithm with SIL simulations.

	SQP		
	St-NMPC	SR-NMPC	DSR-NMPC
Num. of Cost Fun.	194.65	16.64	4.52
NMPC Time [ms]	92.6	14.6	9.1
Opt. Time [ms]	91.8	10.2	4.9
RMS Lat. E. [m]	0.2105	0.2179	0.2256
RMS Or. E. [rad]	0.0135	0.0137	0.0142
Min. Dist. [m]	2.814	2.806	2.792

the Monte Carlo simulations. To compute the RMS error, only the reference tracking phase has been taken into consideration, neglecting the overtaking part where the vehicle must move away from the reference. Indeed, to make the scenario challenging, the same reference, i.e., the center of the lane, has always been considered, leaving the NMPC to autonomously avoid the obstacle. Note that this is more challenging than just imposing a reference change to obtain the overtaking. Clearly, the RMS, being referred to the center of the lane aligned with the travel direction of the vehicle, becomes very large during the obstacle avoidance phase. Finally, the minimum distance from the obstacle highlights that, throughout the entire overtaking maneuver, a safe distance was always maintained from all obstacles. An example of an obstacle avoidance performed by the DSR-NMPC algorithm is shown in Figure 5.12.

From these results, we can observe that: i) The DSR-NMPC reduces the number of function evaluations by a factor of approximately 50 compared to the standard NMPC and 4 with respect to the SR-NMPC. ii) The DSR-NMPC improves the computational time of 5 ms compared to the SR-NMPC and about 10 times with respect to the Standard NMPC. The discrepancy between this result and the one obtained for the cost function evaluations is due to the fact that the DSR-NMPC and SR-NMPC algorithms require to evaluate the approximated control laws before performing optimization, which is not present in the standard NMPC. This operation implies additional computation time. iii) Regarding the times required to solve the OCP with `fmincon`, the DSR-NMPC shows more pronounced improvements with respect to both the Standard NMPC (a factor of 20) and the SR-NMPC (a factor of 2). iv) Finally,

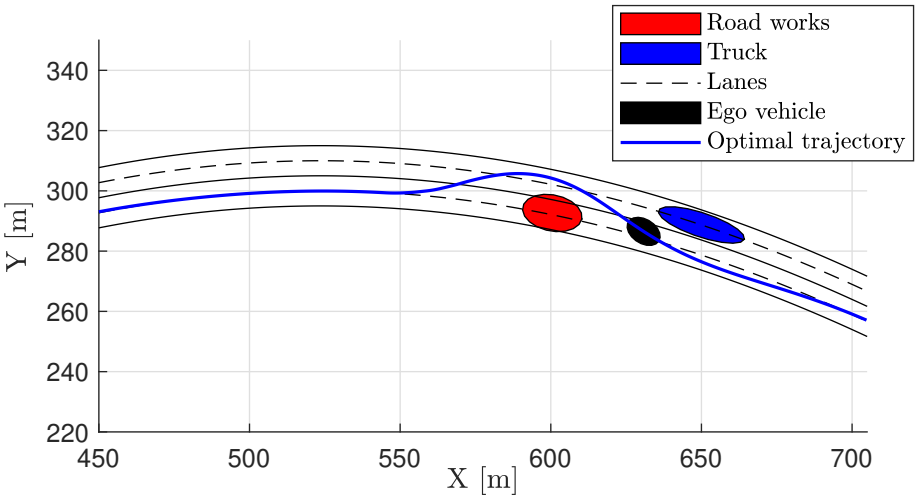

Fig. 5.12 Example of obstacle avoidance performed by DSR-NMPC.

the RMS errors and the minimum distances are quite similar for all the NMPC algorithms.

5.4 Case Study III: Parallel Parking

For this case study, two configurations have been considered:

1. *Standard single shooting NMPC (s-NMPC)*: The same configuration considered in the previous case studies.
2. *Side-length reduction NMPC (SR-NMPC)*: In this configuration only the side-length reduction is employed.

Remark. *It should be noted that in this case, the combination of dimensionality and side-length reduction is not used because, as described below, only two nodes are considered.*

These two configurations have been tested using SIL simulations for a real road scenario regarding a Parallel Parking maneuver (described in Section 5.1.3). An example is shown in Figure 5.3.

5.4.1 Vehicle models

As in the previous two case studies, to simulate the real vehicle, the Matlab `Vehicle Body 3DOF Dual Track` block is used. Regarding the NMPC prediction model, the classical kinematic bicycle equations are considered. Since the vehicle travels at low speed, these equations provide a sufficiently accurate description of the vehicle motion. The kinematic bicycle model is the following:

$$\begin{aligned} \dot{X} &= v_x \cos\psi \\ \dot{Y} &= v_y \sin\psi \\ \dot{\psi} &= \frac{v_x}{w_b}\tan(\delta_f) \end{aligned} \tag{5.16}$$

where X and Y denote the position of the vehicle, ψ its yaw angle, and the parameter $w_b = 2.8$ m represents the wheelbase of the vehicle. The longitudinal speed v_x and the steering angle δ_f are the control variables. The output of the system is (X, Y, ψ). Concerning the state constraint, safety ellipses were designed around the parking vehicles in order to avoid possible collisions:

$$\frac{(X - X_{v_I})^2}{a_I^2} + \frac{(Y - Y_{v_I})^2}{b_I^2} = 1, \text{ for } I = 1, 2, \tag{5.17}$$

where the major axes a_I are defined as $(l_{f_I} + l_{r_I})/2$, the minor axes b_I as $w_{b_I}/2$, and (X_{v_I}, Y_{v_I}) represent the centers of the two vehicles.

5.4.2 Closed-loop system

The same closed-loop system of the previous case studies has been used in Simulink, where the plant is controlled through feedback by an NMPC block. The input of the plant is the command $u = (v_x, \delta_f)$ generated by the NMPC block using information from the plant state and reference trajectory. The NMPC block can implement either a standard single shooting NMPC or a side-length reduction NMPC, which will be described in the following.

5.4.3 Standard single shooting NMPC algorithm

A standard single shooting NMPC algorithm was developed for the control of the lateral and longitudinal vehicle dynamics. The algorithm relies on solving the optimization problem (2.8), using equations (5.1) as the internal prediction model. Its inputs include the plant state and the reference trajectory, while its output $u_{t+1} = (v_x, \delta_f)$ are used to perform the parking maneuver without collisions with the other vehicles. The NMPC command was parametrized considering a new command input sequence c of 4 decision variables, i.e., $n_c = 4$. This means that the prediction interval was divided into 2 segments, that is, $n_s = 2$. As a result, in the prediction interval $[t, t+T]$, there are a total of 4 command samples: 2 for the speed v_x and 2 for the steering angle δ_f. The values of the NMPC parameters are listed in Table 5.8.

Table 5.8 NMPC design parameters.

Parameter	**Value**
T_s	0.1 s
T	100
Q	$\text{diag}(0.25, 0.25, 0.5)$
R	$\text{diag}(0.5, 0.5)$
P	$\text{diag}(2, 10, 20)$
Upper bounds	$[2\,\text{m/s}, \pi/4, 2\ \text{m/s}, \pi/4]$
Lower bounds	$[-2\,\text{m/s}, -\pi/4, -2\ \text{m/s}, -\pi/4]$

5.4.4 Side-length Reduction NMPC

The steps required for the implementation of this approach are now discussed. Note that all the procedures characterizing the dimensionality reduction are not taken into account in this case study.

Offline data generation

A starting set of initial state conditions x_{0p}, $p = 1, \ldots, 1000$ of the vehicle was obtained through the Latin Hypercube Sampling (LHS) technique (see [133]). Starting from these initial conditions, a Monte Carlo (MC) campaign

was carried out using the standard single shooting NMPC. At the end of this campaign, a dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{M}$, with $M = 3e5$, was obtained.

Offline data reduction through clustering

A clustering procedure was carried out to reduce the size of the dataset generated in the previous step using CLARA algorithm. After several trials, a reduced dataset $\mathcal{D}_r = \{\tilde{w}_{lr}, \tilde{u}_{lr}\}$ of $1e4$ data was found.

Offline approximation of the NMPC control law

Based on the dataset $\mathcal{D} = \{\tilde{w}_l, \tilde{u}_l\}_{l=1}^{3e5}$, the approximated control law ϕ_s was computed using the Nonlinear Sparse Identification method described in Section 3.3.1. For this case study, $\mu = 0.1$ was chosen.

Offline bounds definition of all the decision variables

Based on the reduced dataset $\mathcal{D}_r = \{\tilde{w}_{lr}, \tilde{u}_{lr}\}_{lr=1}^{1e4}$, the corresponding bounds $\overline{\phi}$ and $\underline{\phi}$ were computed by means of the SM approach described in Section 4.1.3. Figure 5.13 shows the approximation ϕ_s and the tight bounds of the velocity control command. As it can be seen, the bounds were reduced by about 10 times with respect to the original ones.

Online optimization on the reduced side-length OCP

The SR-NMPC consists in applying online the OCP (4.40), considering the same procedure described in the case study II (see Section 5.3.5).

5.4.5 Comparison between SR-NMPC and Standard single shooting NMPC

In order to test the effectiveness of the SR-NMPC technique and the robustness of the obtained model, different initial state conditions of the ego vehicle, from those considered previously, were taken into account. Then, an MC campaign of 100 simulations was carried out.

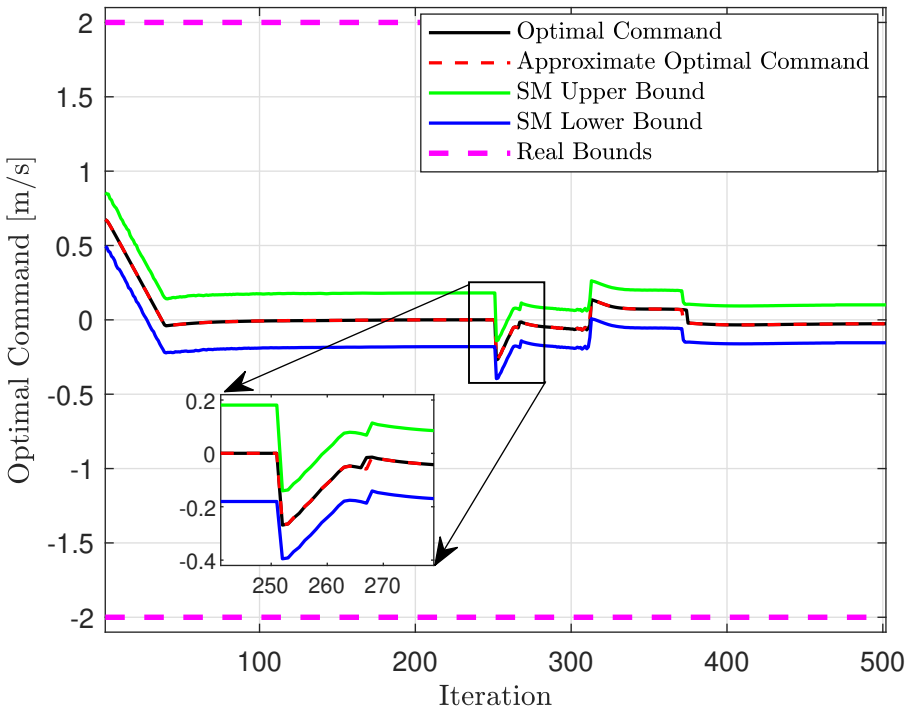

Fig. 5.13 Set Membership approximation of v_x.

The standard single shooting NMPC and the developed SR-NMPC were simulated on a Dell Precision 5820 (Processor: Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz). The optimization problem was solved using the Matlab function `fmincon` with the SQP algorithm.

The metrics used for comparing the performance of the two algorithms are:

1. Number of evaluated cost functions (Num. of Cost Fun.) for finding the minimum;
2. Computational time (Comp. Time);
3. Accuracy in reaching the final target considering the Position Tracking Error (Pos. Track. Err.) and the Orientation Tracking Error (Orient. Track. Err.).

In Table 5.9, the mean and maximum values of the above performance indexes are shown for the two NMPC algorithms. Note that the term Mean Value refers to the average number of evaluated cost functions, computational times, and

Table 5.9 Comparison between Standard NMPC and SR-NMPC using SQP algorithm with SIL simulations.

	SQP			
	St.-NMPC		SR-NMPC	
	Mean Value	Maximum Value	Mean Value	Maximum Value
Num. of Cost Fun.	94.4	106.1	5.46	12.16
Comp. Time [ms]	31.3	36.5	5.7	5.9
Pos. Track. Err. [m]	0.1237	0.7	0.1106	0.1388
Orient. Track. Err. [rad]	0.0241	0.29	0.0213	0.029

tracking errors throughout the simulation (see the example in Eq. (5.10)). The term Maximum Value refers to the highest value during the 100 simulations.

Regarding the number of evaluated cost functions, with the SR-NMPC a reduction of about 17 times, on average, is obtained. Note that, since in this nonlinear optimization problem there are 4 variables to be optimized, at least 5 evaluations of the cost function are required to obtain a numerical estimate of the gradient. Thus, in the case of SR-NMPC, results very close to this minimum number are obtained. For the computational time, the use of SR-NMPC leads to an improvement in the performance of about 6 times, on average. The same considerations also apply to the maximum values of both metrics. With regard to the tracking error, since the values of the position (X, Y) are generally larger than those of the orientation ψ, it is split into: Position Tracking Error and Orientation Tracking Error. For the computation of the Position Tracking Error, the Euclidean distance between Target 2 and the final pose of the vehicle is considered (see Figure 5.3). Regarding the Mean Value, the obtained results are quite similar. Instead for the Maximum Value, there is a considerable difference. Indeed, in the case of standard NMPC, the very high value reported in Table 5.9 is due to the fact that out of 100 simulations, the parking maneuver fails four times. The SR-NMPC, instead, always succeeds in completing it. Thus, besides being more efficient from a computational point of view, this

approach is also more robust. An example of a complete maneuver performed by the SR-NMPC is shown in Figure 5.14.

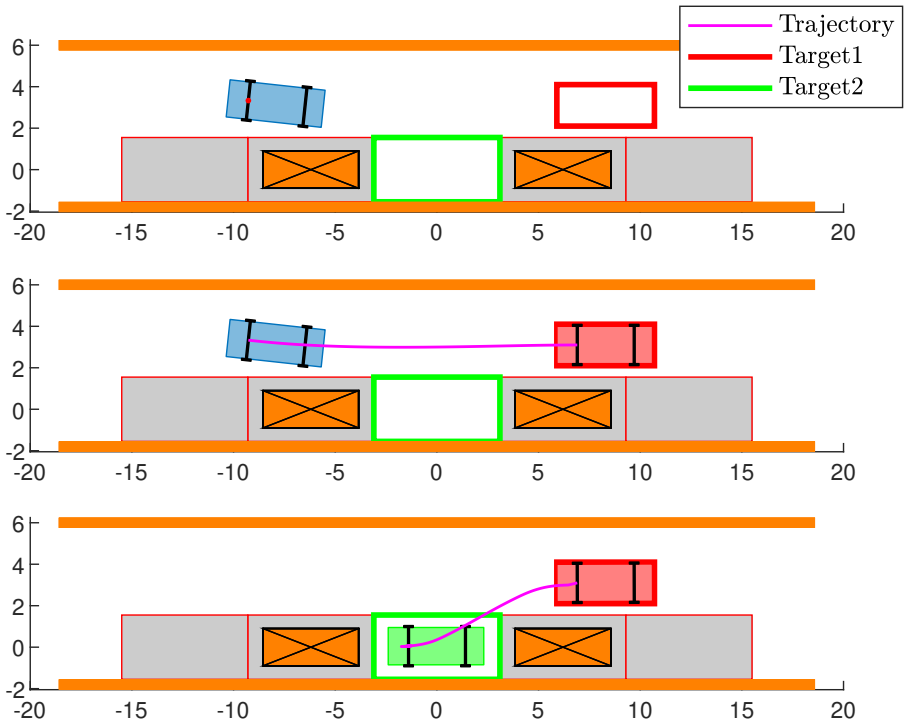

Fig. 5.14 Example of Autonomous Parallel Parking performed by SR-NMPC.

5.5 Chapter summary

In this chapter, the two methods developed in the thesis were tested and compared with standard NMPCs, which integrate both single shooting and multiple shooting approaches. These comparisons were made through both software-in-the-loop and hardware-in-the-loop simulations. The obtained results have demonstrated the benefits of using dimensionality and side-length reduction techniques to decrease the computational complexity of nonlinear optimization algorithms, in particular Sequential Quadratic Programming and Particle Swarm Optimization. Importantly, these benefits were achieved without compromising the quality of the solution found. This represents a significant advancement as it enhances the optimization efficiency while maintaining solution accuracy, potentially enabling the implementation of these methods

in real-time applications where computational resources and time are critical factors. Furthermore, the successful application of these techniques underlines their potential for a wide range of optimization problems beyond those explored in this thesis.

Chapter 6

Conclusions and future research

This thesis addressed the challenge of reducing the computational burden associated with Nonlinear Model Predictive Control (NMPC) for real-time applications, particularly within the domain of autonomous driving.

It began by establishing a comprehensive foundation in NMPC covering its formulation, theoretical underpinnings, and optimization methods. This provided a crucial understanding of the theoretical and numerical complexities that limit the use of standard NMPC algorithms in real-time scenarios. Notably, key elements that strongly affect the computation time of nonlinear optimization algorithms are: i) the dimensionality d of the search domain (i.e., the number of decision variables); ii) the side-length s of the search domain (i.e., the region where the decision variables are defined); and iii) the starting point of the algorithm.

To overcome these limitations, the current study proposed two novel strategies, called *Dimensionality Reduction* and *Side-length Reduction*, specifically designed to reduce the computational burden of the underlying optimal control problem by decreasing the volume of the search domain. This volume can be represented as s^d, where s represents its side length and d its dimensionality.

The Dimensionality Reduction method operates on the parameter d by employing a combination between gradient computation and Sparse Nonlinear Identification. Gradient computation identifies the directions of greatest variability of the optimization problem within the search domain to find the relevant decision variables. The Nonlinear Sparse Identification method approximates

the NMPC control law using the ℓ_1-relaxed-greedy algorithm, providing a good warm start for the optimization algorithm and enabling the implementation of the Optimal Control Problem (OCP) only on the relevant decision variables.

The Side-length Reduction addresses the problem of the search domain size s by using the Set Membership (SM) method to derive tight bounds on the most relevant decision variables. These bounds are then employed online to restrict in an optimal and adaptive way the search domain of the optimization problem. This strategy allows to focus the optimization only in the region of the search domain containing the minimum. Moreover, it enhances robustness compared to previous methods that used SM for open-loop control law approximation.

The second contribution of this thesis involved presenting a theoretical analysis of the inherent computational complexity of the two proposed methods. This study showed that reducing the dimensionality d can significantly decrease the number of mathematical operations required for solving the numerical optimization problem. This, in turns, leads to a reduction in computational time and resources, making the proposed methods more efficient and scalable. Moreover, the analysis also revealed that shrinking the side length s of the search domain can improve the convergence rate of nonlinear algorithms. A faster convergence rate means that these algorithms can reach their optimal solution in fewer iterations, thus enhancing their efficiency.

The third contribution of this thesis was to test the proposed methods on three realistic autonomous driving scenarios, demonstrating their efficiency and versatility in handling a range of tasks. The performance of the proposed methods was compared with a standard NMPC, using both Software-In-the-Loop (SIL) and Hardware-In-the-Loop (HIL) simulations. Two different nonlinear numerical optimization algorithms, namely Sequential Quadratic Programming (SQP) and Particle Swarm Optimization (PSO), were used in the SIL simulation, highlighting the effectiveness of the proposed methods regardless of the specific optimization algorithm employed. For the HIL simulation, only the SQP algorithm was considered. The benefits of using the proposed approaches were evident in all the considered scenarios and with both algorithms, achieving significantly better results in terms of computation time, without compromising the quality of the solution found. In particular, the developed methods enabled the design of nonlinear optimization algorithms that can compute the optimal

control command within 1-2 milliseconds in SIL and around 7 milliseconds on hardware. This represents a significant advancement, potentially allowing the implementation of these methods in real-time applications where computational resources and time are critical factors. This demonstrates that the aim of the thesis has been achieved.

In conclusion, this thesis successfully fulfilled its objectives, proposing novel methods and demonstrating their effectiveness in reducing the computational complexity associated with NMPC.

Future research

This thesis has laid the groundwork for several promising avenues of future research. The following proposals are suggested for further investigations.

Use of Autonomous Driving Simulators. After performing a Hardware-In-the-Loop (HIL) simulation, the subsequent step may involve testing the developed algorithms on simulators specifically designed for the validation of autonomous driving systems. One of the most well-known options is CARLA (Car Learning to Act), an open-source simulator tailored for autonomous driving research. Built on the Unreal Engine, CARLA provides a high-fidelity visual context. It uses the ASAM OpenDRIVE standard to define roads and urban settings. This combination allows for the creation of a wide variety of realistic and complex driving environments, as can be seen in Figure 6.1. Furthermore, CARLA incorporates a scalable client-server architecture, enabling the simulator to handle large-scale and complex simulations. It also includes a Traffic Manager, a system that controls all vehicles in the simulation except the one used for learning. In addition to these features, CARLA offers various sensors such as cameras, lidar, radar, and GPS, that vehicles can use to gather information about their surroundings.

Stability and Robust Stability Analysis. The study of stability in NMPC remains an active area of research with promising avenues for further exploration. One particularly interesting direction lies in leveraging finite-gain stability as an

Fig. 6.1 Example of a Carla environment.

alternative to the well-established Lyapunov function approach. In this regard, the use of finite-gain stability will be another focus of my future research.

Cross-Disciplinary Applications. The algorithms developed in this thesis have shown promising results in the automotive sector, demonstrating their potential in controlling complex systems and improving performance. However, their application is not limited to this field. The versatility and robustness of these algorithms make them suitable for a wide range of applications. In this context, a future area of exploration lies in the aerospace sector, whose progress towards increasingly autonomous vehicles requires sophisticated control strategies. The developed algorithms could be adapted to handle the complex dynamics of the spacecraft and the challenging conditions of space exploration. This includes guiding satellites on Earth orbits, where they must deal with gravitational forces and atmospheric drag, and on interplanetary orbits, where they must navigate vast distances and gravitational fields of the solar system. The control of rovers is another attractive application of these algorithms. Rovers, such as those used in Mars exploration missions, operate in extremely harsh environments and need to perform complex tasks autonomously, such as navigating rough terrains, avoiding obstacles, and carrying out scientific experiments. The algorithms developed in this thesis could be adapted to address these challenging operations.

Use of Quantum Computing Paradigms. Research in NMPC has seen a surge in recent years, driven by two key factors: the development of ever-more powerful processors and the continuous refinement of numerical algorithms. However, a technological ceiling is approaching, beyond which significant performance improvements for NMPC through traditional methods may become exceedingly challenging. Moreover, the emergence of large-scale applications, such as those related to power grids, further highlights the limitations imposed by the computational constraints of NMPC. Consequently, exploring alternative computational approaches becomes an urgent necessity. One of the most promising solutions is Quantum Computing (QC), which was first introduced in the early 1980s by physicist Paul Benioff [158] and independently by Richard Feynman [159]. QC is a revolutionary technology that uses the principles of quantum mechanics, such as *superposition* and *entanglement*, to perform calculations exponentially faster than classical computers [160]. Unlike classical computers that rely on bits, which can be either 0 or 1, quantum computers leverage *qubits*, which can exist in a superposition of both states simultaneously. This unique characteristic allows quantum computers to explore a vast number of possibilities concurrently, leading to a significant increase in processing power. There are various paradigms and hardware for building QCs, with *gate-based quantum computing* [161] and *adiabatic quantum computation* (AQC) [162] leading the way. In the gate model of QC, also known as circuit-based or digital quantum computing, the computation is performed by applying a sequence of unitary gates to a set of qubits, whose states can be measured at the end of the computation. On the other hand, adiabatic quantum computation (AQC) is a method that relies on the adiabatic theorem of quantum mechanics. This theorem states that a quantum-mechanical system remains in its instantaneous ground state if a given Hamiltonian that governs it varies slowly enough. The idea behind AQC is to encode the solution of a problem in the ground state of a Hamiltonian and to change this Hamiltonian slowly from an initial one, whose ground state is easy to prepare, to a final one, whose ground state encodes the solution. This method is particularly suited for solving optimization problems. An approach closely related to AQC is quantum annealing (QA), first proposed in 1988 by Apolloni et al. [163] and then formulated in its present form by Kadowaki and Nishimori [164]. QA can be viewed as a relaxation of the AQC model, where the strict conditions of adiabaticity are not met, resulting in

a heuristic variational quantum algorithm. Consequently, this approach can be used to find the ground state of Ising models, a known NP-hard problem. It is well-documented in the literature how to transform canonical NP-hard and NP-complete combinatorial optimization problems into forms suitable for quantum annealers. These problems can be formulated either in Ising form using a $\{-1, 1\}$ basis and spin variables, or as a quadratic unconstrained binary optimization (QUBO) problem using the $\{0, 1\}$ basis and binary variables. The two forms are equivalent, and problems in one form can be readily represented in the other via a simple change of basis. QA is used to solve real-world problems in optimization, scheduling, and machine learning, among others. While it is still debated whether QA can provide a quantum speed-up over classical approaches [165], technological advances fuel hope for future competitiveness. The solution to a problem in QA goes through several stages, depending on the type of qubits used and the adiabatic protocol implemented. D-Wave Systems annealers, which use superconducting qubits, are the most popularly used QA devices [166, 167]. The exploration of quantum computing represents a significant opportunity to overcome the computational limitations of traditional NMPC and open up new avenues in the control of complex systems. Research in this field is progressing rapidly, and the future prospects are highly promising. Quantum NMPC has the potential to transform the way we control complex systems, offering innovative and high-performance solutions to problems that are currently intractable.

References

[1] S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[2] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[3] Jay H Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9:415–424, 2011.

[4] David Mayne. Nonlinear model predictive control:challenges and opportunities. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, pages 23–44. Birkhäuser Basel, 2000.

[5] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[6] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010.

[7] S. Joe Qin and Thomas A. Badgwell. An overview of nonlinear model predictive control applications. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, pages 369–392. Birkhäuser Basel, 2000.

[8] Mark Campbell, Magnus Egerstedt, Jonathan P. How, and Richard M. Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4649–4672, 2010.

[9] Daniel J. Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[10] Travis J. Crayton and Benjamin Mason Meier. Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy. *Journal of Transport & Health*, 6:245–252, 2017.

[11] Ryosuke Okuda, Yuki Kajiwara, and Kazuaki Terashima. A survey of technical trend of ADAS and autonomous driving. In *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*, pages 1–4, Hsinchu, Taiwan, 2014.

[12] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[13] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[14] Mario Milanese and Carlo Novara. Set membership identification of nonlinear systems. *Automatica*, 40(6):957–975, 2004.

[15] Carlo Novara. Sparse set membership identification of nonlinear functions and application to fault detection. *International Journal of Adaptive Control and Signal Processing*, 30(2):206–223, 2016.

[16] Mattia Boggio, Carlo Novara, and Michele Taragna. Trajectory planning and control for autonomous vehicles: a "fast" data-aided NMPC approach. *European Journal of Control*, page 100857, 2023.

[17] Mattia Boggio, Carlo Novara, and Michele Taragna. Nonlinear model predictive control: an optimal search domain reduction. *IFAC-PapersOnLine*, 56(2):6253–6258, 2023. 22nd IFAC World Congress.

[18] Mattia Boggio, Carlo Novara, and Michele Taragna. Set membership identification for NMPC complexity reduction. *IFAC-PapersOnLine*, 58(15):217–222, 2024. 20th IFAC Symposium on System Identification SYSID 2024.

[19] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[20] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[21] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[22] Moritz Diehl, H.Georg Bock, Johannes P. Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[23] Hans Georg Bock, Moritz Diehl, Ekaterina Kostina, and Johannes P Schlöder. Constrained optimal feedback control of systems governed by large differential algebraic equations. In *Real-Time PDE-constrained optimization*, pages 3–24. SIAM, 2007.

[24] Lorenz T. Biegler. Efficient solution of dynamic optimization and NMPC problems. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, pages 219–243, Basel, 2000. Birkhäuser Basel.

[25] Wei C. Li and Lorenz T. Biegler. A multistep, newton-type control strategy for constrained, nonlinear processes. In *1989 American Control Conference*, pages 1526–1527, 1989.

[26] Toshiyuki Ohtsuka. A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4):563–574, 2004.

[27] Dimitris Kouzoupis, Gianluca Frison, Andrea Zanelli, and Moritz Diehl. Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, 46:863–882, 2018.

[28] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*, pages 391–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[29] T. Parisini and R. Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

[30] Tor A. Johansen. Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, 40(2):293–300, 2004.

[31] T.A. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Transactions on Automatic Control*, 48(5):810–815, 2003.

[32] Munoz de la Pena, A. Bemporad, and C. Filippi. Robust explicit MPC based on approximate multi-parametric convex programming. In *2004 43rd IEEE Conference on Decision and Control (CDC)*, volume 3, pages 2491–2496 Vol.3, 2004.

[33] Alexandra Grancharova and Tor A. Johansen. Computation, approximation and stability of explicit feedback min–max nonlinear model predictive control. *Automatica*, 45(5):1134–1143, 2009.

[34] M. Canale, M. Milanese, and C. Novara. Semi-active suspension control using "fast" model-predictive techniques. *IEEE Transactions on Control Systems Technology*, 14(6):1034–1046, 2006.

[35] M. Canale, L. Fagiano, and M. Milanese. Set membership approximation theory for fast implementation of model predictive control laws. *Automatica*, 45(1):45–54, 2009.

[36] Guanru Pan and Timm Faulwasser. NMPC in active subspaces: Dimensionality reduction with recursive feasibility guarantees. *Automatica*, 147:110708, 2023.

[37] E. F. Camacho and C. Bordons. *Model Predictive Controllers*, pages 13–30. Springer London, London, 2007.

[38] David Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[39] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29(5):1251–1274, 1993.

[40] Manfred Morari and Jay H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682, 1999.

[41] C.C. Chen and L. Shaw. On receding horizon feedback control. *Automatica*, 18(3):349–352, 1982.

[42] D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 464–465 vol.1, 1988.

[43] Wei Chong Li and Lorenz T Biegler. Newton-type controllers for constrained nonlinear processes with uncertainty. *Industrial & engineering chemistry research*, 29(8):1647–1657, 1990.

[44] Frank Allgöwer, Rolf Findeisen, and Zoltan Kalman Nagy. Nonlinear model predictive control: From theory to application. *Journal of The Chinese Institute of Chemical Engineers*, 35:299–315, 2004.

[45] Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control*, pages 45–69. Springer International Publishing, Cham, 2017.

[46] Hong Chen, H Kremling, and Frank Allgöwer. Nonlinear predictive control of a benchmark CSTR. *Proceedings of the 3rd European Control Conference, Rome-Italy.*, pages 3247–3252, 01 1995.

[47] Lino O. Santos, Paulo A.F.N.A. Afonso, José A.A.M. Castro, Nuno M.C. Oliveira, and Lorenz T. Biegler. On-line implementation of nonlinear MPC: an experimental case study. *Control Engineering Practice*, 9(8):847–857, 2001. Advanced Control of Chemical Processes.

[48] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

[49] Sébastien Gros, Rien Quirynen, and Moritz Diehl. Aircraft control based on fast non-linear MPC & multiple-shooting. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1142–1147, 2012.

[50] Janick V. Frasch, Andrew Gray, Mario Zanon, Hans Joachim Ferreau, Sebastian Sager, Francesco Borrelli, and Moritz Diehl. An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles. In *2013 European Control Conference (ECC)*, pages 4136–4141, 2013.

[51] Thivaharan Albin, Dennis Ritter, Dirk Abel, Norman Liberda, Rien Quirynen, and Moritz Diehl. Nonlinear MPC for a two-stage turbocharged gasoline engine airpath. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 849–856, 2015.

[52] Bárbara Barros Carlos, Tommaso Sartor, Andrea Zanelli, Gianluca Frison, Wolfram Burgard, Moritz Diehl, and Giuseppe Oriolo. An efficient real-time NMPC for quadrotor position control under communication time-delay. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 982–989, 2020.

[53] D. Bertsekas. *Dynamic Programming and Optimal Control: Volume I*. Athena Scientific optimization and computation series. Athena Scientific, 2012.

[54] Thomas J. Böhme and Benjamin Frank. *Indirect Methods for Optimal Control*, pages 215–231. Springer International Publishing, Cham, 2017.

[55] Thomas J. Böhme and Benjamin Frank. *Direct Methods for Optimal Control*, pages 233–273. Springer International Publishing, Cham, 2017.

[56] Constantin Carathéodory. *Calculus of variations and partial differential equations of the first order*. Teubner, Berlin, 1935.

[57] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[58] Amnon Rapoport. Dynamic programming models for multistage decision-making tasks. *Journal of Mathematical Psychology*, 4(1):48–71, 1967.

[59] P.L. Lions. *Generalized Solutions of Hamilton-Jacobi Equations*. London Pitman, 1982.

[60] R. Weinstock. *Calculus of Variations: With Applications to Physics and Engineering*. Mineola, New York, Dover Publications, 1974.

[61] L.S. Pontryagin, V.G. Boltyanski, R.V. Gamkrelidze, and E.F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Interscience Publishes, New York, 1962.

[62] Benjamin Passenberg. *Theory and Algorithms for Indirect Methods in Optimal Control of Hybrid Systems*. PhD thesis, Technische Universität München, 2012.

[63] Michele Pagone, Mattia Boggio, Carlo Novara, and Simone Vidano. A pontryagin-based NMPC approach for autonomous rendez-vous proximity operations. In *2021 IEEE Aerospace Conference (50100)*, pages 1–9, 2021.

[64] D P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[65] GA Hicks and WH Ray. Approximation methods for optimal control synthesis. *The Canadian Journal of Chemical Engineering*, 49(4):522–528, 1971.

[66] RWH Sargent and GR Sullivan. The development of an efficient optimal control package. In *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*, pages 158–168. Springer, 1978.

[67] Dieter Kraft. On converting optimal control problems into nonlinear programming problems. In *Computational mathematical programming*, pages 261–280. Springer, 1985.

[68] D. M. Himmelblau T. H. Tsang and T. F. Edgar. Optimal control via collocation and non-linear programming. *International Journal of Control*, 21(5):763–768, 1975.

[69] Lorenz T Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers & chemical engineering*, 8(3-4):243–247, 1984.

[70] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[71] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[72] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492, Berkeley and Los Angeles, 1951. University of California Press.

[73] William Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, Chicago, IL, USA, 1939.

[74] L.T. Biegler. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2010.

[75] Tjalling J. Ypma. Historical development of the newton–raphson method. *SIAM Review*, 37(4):531–551, 1995.

[76] I. Newton and D.T. Whiteside. *The Mathematical Papers of Isaac Newton: Volume 7, 1691-1695*. The Mathematical Papers of Isaac Newton. Cambridge University Press, 2008.

[77] Thomas Simpson. *Essays on Several Curious and Useful Subjects, in Speculative and Mix'd Mathematicks*. H. Woodfall, jun., 1740.

[78] Stephen J. Wright. *Primal-Dual Interior-Points Methods*. SIAM, Philadelphia, Pa, USA, 1997.

[79] Paul Boggs, Paul Domich, and Janet Rogers. An interior point method for general large-scale quadratic programming problems. *Annals of Operations Research*, 62:419–437, 12 1996.

[80] Seung-Jean Kim, K. Koh, M. Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, 2007.

[81] Andreas Wächter and Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 03 2006.

[82] Roger Fletcher. *The Sequential Quadratic Programming Method*, pages 165–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[83] A. Forsgren. On warm starts for interior methods. In F. Ceragioli, A. Dontchev, H. Futura, K. Marti, and L. Pandolfi, editors, *System Modeling and Optimization*, pages 51–66, Boston, MA, 2006. Springer US.

[84] J. E. Dennis, Jr. and Jorge J. Moré. Quasi-newton methods, motivation and theory. *SIAM Review*, 19(1):46–89, 1977.

[85] WC Davidon. Variable metric method for minimization. Technical report, Argonne National Lab., Lemont, Ill., 1959.

[86] William C. Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17, 1991.

[87] R. Fletcher and M. J. D. Powell. A Rapidly Convergent Descent Method for Minimization. *The Computer Journal*, 6(2):163–168, 08 1963.

[88] C. G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 03 1970.

[89] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 01 1970.

[90] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

[91] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

[92] Michael JD Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*, pages 144–157. Springer, 2006.

[93] H. G. Bock. *Recent Advances in Parameteridentification Techniques for O.D.E.*, pages 95–121. Birkhäuser Boston, Boston, MA, 1983.

[94] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[95] Moritz Diehl, Rolf Findeisen, and Frank Allgöwer. A stabilizing real-time implementation of nonlinear model predictive control. In *Real-Time PDE-Constrained Optimization*, pages 25–52. SIAM, 2007.

[96] Victor M. Zavala and Lorenz T. Biegler. The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica*, 45(1):86–93, 2009.

[97] Sébastien Gros, Mario Zanon, Rien Quirynen, Alberto Bemporad, and Moritz Diehl. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*, 93(1):62–80, 2020.

[98] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[99] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.

[100] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

[101] Melanie Nicole Zeilinger, Colin Neil Jones, and Manfred Morari. Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization. *IEEE Transactions on Automatic Control*, 56(7):1524–1534, 2011.

[102] Osvaldo J. Rojas, Graham C. Goodwin, María M. Serón, and Arie Feuer. An SVD based strategy for receding horizon control of input constrained linear systems. *International Journal of Robust and Nonlinear Control*, 14(13-14):1207–1226, 2004.

[103] Chong-Jin Ong and Zheming Wang. Reducing variables in model predictive control of linear system with disturbances using singular value decomposition. *Systems & Control Letters*, 71:62–68, 2014.

[104] Ayorinde Bamimore. Laguerre function-based quasi-infinite horizon nonlinear model predictive control. *International Journal of Dynamics and Control*, 11, 01 2023.

[105] Liuping Wang. Discrete model predictive controller design using laguerre functions. 14:12, 2004.

[106] Maciej Ławryńczuk. Nonlinear model predictive control for processes with complex dynamics: A parameterisation approach using laguerre functions. *International Journal of Applied Mathematics and Computer Science*, 30(1):35–46, 2020.

[107] Xiuhui Hou, Fang Deng, Hualin Yang, Dunjing Yu, Hanlin Zhang, and Boyang Li. Robust nonlinear model predictive control for ship dynamic positioning using laguerre function. *IEEE Access*, 10:127563–127574, 2022.

[108] B. Wahlberg. System identification using laguerre models. *IEEE Transactions on Automatic Control*, 36(5):551–562, 1991.

[109] R. Cagienard, P. Grieder, E.C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563–570, 2007.

[110] Rohan C. Shekhar and Chris Manzie. Optimal move blocking strategies for model predictive control. *Automatica*, 61:27–34, 2015.

[111] Yutao Chen, Nicoló Scarabottolo, Mattia Bruschetta, and Alessandro Beghi. Efficient move blocking strategy for multiple shooting-based nonlinear model predictive control. *IET Control Theory & Applications*, 14(2):343–351, 2020.

[112] Christoph Rösmann Artemi Makarow and Torsten Bertram. Suboptimal nonlinear model predictive control with input move-blocking. *International Journal of Control*, 0(0):1–10, 2022.

[113] Trent M. Russi. *Uncertainty quantification with experimental data and complex system models*. PhD thesis, 2010.

[114] Paul G. Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: Applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[115] Richard L. Burden and J. Douglas. Faires. *Numerical analysis / Richard L. Burden, J. Douglas Faires*. Thomson Brooks/Cole, Belmont, Calif, 8th ed. edition, 2005.

[116] J. Grabmeier, E. Kaltofen, and V. Weispfenning. *Computer Algebra Handbook: Foundations, Applications, Systems*. Springer Berlin Heidelberg, 2003.

[117] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, USA, second edition, 2008.

[118] John C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations, Second Edition*. Society for Industrial and Applied Mathematics, 2004.

[119] Charles C. Margossian. A review of automatic differentiation and its efficient implementation. *WIREs Data Mining and Knowledge Discovery*, 9(4):e1305, 2019.

[120] Lennart Ljung. *System identification: theory for the user*. Prentice-Hall, Inc., USA, 1986.

[121] Yilun Chen, Yuantao Gu, and Alfred O. Hero. Sparse LMS for system identification. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3125–3128, 2009.

[122] Necmiye Ozay, Mario Sznaier, Constantino M. Lagoa, and Octavia I. Camps. A sparsification approach to set membership identification of switched affine systems. *IEEE Transactions on Automatic Control*, 57(3):634–648, 2012.

[123] Carlo Novara. Sparse identification of nonlinear functions and parametric set membership optimality analysis. In *Proceedings of the 2011 American Control Conference*, pages 663–668, 2011.

[124] Jonas Sjöberg, Qinghua Zhang, Lennart Ljung, Albert Benveniste, Bernard Delyon, Pierre-Yves Glorennec, Håkan Hjalmarsson, and Anatoli Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

[125] C. Novara, T. Vincent, K. Hsu, M. Milanese, and K. Poolla. Parametric identification of structured nonlinear systems. *Automatica*, 47(4):711–721, 2011.

[126] Carlo Novara. Sparse identification of nonlinear functions and parametric set membership optimality analysis. In *Proceedings of the 2011 American Control Conference*, pages 663–668, 2011.

[127] J.J. Fuchs. Recovery of exact sparse representations in the presence of bounded noise. *IEEE Transactions on Information Theory*, 51(10):3601–3608, 2005.

[128] J.A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.

[129] D.L. Donoho, M. Elad, and V.N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006.

[130] J.A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[131] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier analysis and applications*, 14:877–905, 2008.

[132] R. Gribonval, R.M. Figueras i Ventura, and P. Vandergheynst. A simple test to check the optimality of a sparse signal approximation. *Signal Processing*, 86(3):496–510, 2006.

[133] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[134] M. Milanese, J. Norton, H. Piet-Lahanier, and É. Walter. *Bounding Approaches to System Identification*. Plenum Press, New York, 1996.

[135] J.F. Traub, G.W. Wasilkowski, and H. Woźniakowski. *Information-based Complexity*. Computer science and scientific computing. Academic Press, 1988.

[136] J. Chen and G. Gu. *Control-Oriented System Identification: An H∞ Approach*. Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control. Wiley, 2000.

[137] Lior Rokach and Oded Maimon. Clustering methods. *Data mining and knowledge discovery handbook*, pages 321–352, 2005.

[138] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[139] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[140] Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416, 01 1987.

[141] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, Ltd, 1990.

[142] Leonard Kaufman and Peter J. Rousseeuw. *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley & Sons, Ltd, 1990.

[143] Leonard Kaufman and Peter J. Rousseeuw. *Clustering Large Applications (Program CLARA)*, chapter 3, pages 126–163. John Wiley & Sons, Ltd, 1990.

[144] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[145] Kristofer D. Kusano and Hampton C. Gabler. Comprehensive target populations for current active safety systems using national crash databases. *Traffic Injury Prevention*, 15(7):753–761, 2014.

[146] Yang Ding, Weichao Zhuang, Liangmo Wang, Jingxing Liu, Levent Guvenc, and Zhen Li. Safe and optimal lane-change path planning for automated driving. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 235(4):1070–1083, 2021.

[147] Simon Sternlund, Johan Strandroth, Matteo Rizzi, Anders Lie, and Claes Tingvall. The effectiveness of lane departure warning systems-a reduction in real-world passenger car injury crashes. *Traffic Injury Prevention*, 18(2):225—229, February 2017.

[148] Noor Amer, Hairi Zamzuri, Khisbullah Hudha, and Zulkiffli Kadir. Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges. *Journal of Intelligent & Robotic Systems*, 86:1–30, 05 2017.

[149] Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi, and Tracey Oliver. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. *Journal of modern transportation*, 24:284–303, 2016.

[150] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(Volume 1, 2018):187–210, 2018.

[151] Muhammad Awais Abbas, Ruth Milman, and J. Mikael Eklund. Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles. *Canadian Journal of Electrical and Computer Engineering*, 40(1):12–22, 2017.

[152] Holger Banzhaf, Dennis Nienhüser, Steffen Knoop, and J. Marius Zöllner. The future of parking: A survey on automated valet parking with an outlook on high density parking. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1827–1834, 2017.

[153] Ming Feng Hsieh and Umit Ozguner. A parking algorithm for an autonomous vehicle. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1155–1160, 2008.

[154] Bai Li, Kexin Wang, and Zhijiang Shao. Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3263–3274, 2016.

[155] Xiaojing Zhang, Alexander Liniger, Atsushi Sakai, and Francesco Borrelli. Autonomous parking using optimization-based collision avoidance. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4327–4332, 2018.

[156] MATLAB. Vehicle body 3DOF. https://it.mathworks.com/help/vdynblks/ref/vehiclebody3dof.html, 2018.

[157] NVIDIA. Jetson nano. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/, 2019.

[158] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22:563–591, 05 1980.

[159] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[160] Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117, feb 1998.

[161] Sangil Kwon, Akiyoshi Tomonaga, Gopika Lakshmi Bhai, Simon J. Devitt, and Jaw-Shen Tsai. Gate-based superconducting quantum computing. *Journal of Applied Physics*, 129(4):041102, 01 2021.

[162] Tameem Albash and Daniel A. Lidar. Adiabatic quantum computation. *Rev. Mod. Phys.*, 90:015002, Jan 2018.

[163] B. Apolloni, Nicolò Cesa-Bianchi, and Diego de Falco. A numerical implementation of quantum annealing. *International Conference on Stochastic Processes, Physics and Geometry*, pages 97–111, 07 1988.

[164] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Phys. Rev. E*, 58:5355–5363, Nov 1998.

[165] Philipp Hauke, Helmut G Katzgraber, Wolfgang Lechner, Hidetoshi Nishimori, and William D Oliver. Perspectives of quantum annealing: methods and implementations. *Reports on Progress in Physics*, 83(5):054401, may 2020.

[166] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, may 2011.

[167] D-Wave Systems. The practical quantum computing company. https://www.dwavesys.com/, 2011.