Université du Québec en Abitibi-Témiscamingue

CONCEPTION DES COMMANDES PRÉDICTIVES BASÉES SUR UN MODÈLE POUR LE SUIVI DE TRAJECTOIRE DES ROBOTS MOBILES À ROUES

Thèse présentée à l'Université du Québec en Abitibi-Témiscamingue comme exigence partielle du grade de Docteur en Philosophie en Ingénierie offert en extension en vertu d'un protocole d'entente avec l'Université du Québec à Chicoutimi

Par
Mahmoud El-Sayyah

07/2024

Université du Québec en Abitibi-Témiscamingue

DESIGN OF A MODEL-BASED PREDICTIVE CONTROLLERS FOR TRAJECTORY-TRACKING OF WHEELED MOBILE ROBOTS

Thesis presented to Université du Québec en Abitibi-Témiscamingue in partial fulfillment for the degree of Doctor of Philosophy (Ph.D.) in Engineering offered as an extension under an agreement with Université du Québec à Chicoutimi

By
Mahmoud El-Sayyah

07/2024

BOARD OF EXAMINERS

Prof. Mohamad Saad, Thesis Supervisor
École de génie, UQAT

Prof. Maarouf Saad, Co- supervisor
Département de génie électrique, ETS

Prof. Nahi Kandil, President of the Board of Examiners
École de génie, UQAT

Prof. Yassine Kali, Member of the jury
École de génie, UQAT

Prof. Azeddine Kaddouri, External Independent Examiner
Faculté d'ingénierie, UMoncton

DEDICATION

To my parents, whose unwavering support and encouragement have been my constant source of strength and inspiration.

To my family, for their love, understanding, and belief in my endeavors, which have driven me to achieve my goals.

To my wife Malaak, whose patience, love, and unwavering faith in me have been the cornerstone of my journey.

This modest work is dedicated to you all. Your presence and support have been invaluable, and I am deeply grateful.

ACKNOWLEGEMENT

Above all, I would like to thank Almighty ALLAH for giving me health, strength, and willingness to begin and complete this project.

First, this work would not have been possible without the invaluable help and guidance of Dr. Mohamad Saad and Dr. Maarouf Saad. I am deeply grateful to them for their exceptional supervision, patience, and valuable advice, as well as their availability throughout the preparation of this project.

I extend my sincere appreciation to the members of the jury, Dr. Nahi Kandil, Dr. Yassine Kali, and Dr. Azeddine Kaddouri. Thank you for your valuable time, insightful comments, and constructive feedback.

I would also like to acknowledge the financial support provided by the “Natural Sciences and Engineering Research Council of Canada (NSERC)”, “Fondation J.A. DeSève”, and “Fondation de l’UQAT”. Your generous contributions have enabled me to pursue and complete this research.

Lastly, I would like to thank the University for providing a stimulating environment and the resources necessary to complete this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AS	Active-Set
ARCPROP	A Robust Convergent Propagation
DARE	Discrete-time Algebraic Riccati Equation
IP	Interior-Point
iLQR	iterative Linear Quadratic Regulator
LQR	Linear Quadratic Regulator
LMPC	Laguerre-based Model Predictive Control
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
NMPCAS	Nonlinear Model Predictive Control using Active-Set
NMPCIP	Nonlinear Model Predictive Control using Interior-Point
NMPC-PROP	Nonlinear Model Predictive Control using Resilient Propagation
NOP	Nonlinear Optimization Problem
OMR	Omnidirectional Mobile Robot
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
QIH	Quasi Infinite Horizon
RPROP	Resilient Propagation
UGV	Unmanned Ground Vehicle
WMR	Wheeled Mobile Robot

RÉSUMÉ

Ce projet se concentre sur le contrôle autonome de suivi de trajectoire des robots mobiles à roues, avec une emphase particulière sur les robots omnidirectionnels capables de se déplacer instantanément dans n'importe quelle direction sans réorientation. L'objectif principal du projet est le développement de stratégies de contrôle non linéaires basées sur les principes de contrôle prédictif. Ces stratégies sont conçues pour assurer des performances de suivi fiables pour les systèmes de robots mobiles présentant des non-linéarités et des contraintes opérationnelles.

Le projet commence par développer la représentation cinématique du robot omnidirectionnel, qui sert de base à la conception ultérieure des commandes. Ensuite, deux approches de contrôle distinctes sont formulées pour la tâche de suivi de trajectoire des robots mobiles omnidirectionnels, en particulier dans des scénarios caractérisés par des non-linéarités et des contraintes opérationnelles. La première approche est une méthode prédictive optimale qui utilise des techniques de linéarisation non itératives pour gérer efficacement les non-linéarités du système. Elle intègre également les fonctions de Laguerre pour améliorer l'efficacité de calcul, réduisant ainsi le temps de calcul du contrôle. La deuxième approche exploite la nature non linéaire inhérente à la dynamique du robot et emploie des méthodes d'optimisation résilientes pour répondre à la complexité de calcul associée à cette méthode. L'analyse de stabilité est réalisée pour déterminer les conditions nécessaires pour assurer la stabilité nominale de système non linéaire

Les deux méthodes de contrôle sont rigoureusement vérifiées dans un environnement de simulation. De plus, les performances de ces méthodes sont comparées à des méthodes de référence de la littérature existante, démontrant leur efficacité et leurs capacités. Pour valider davantage l’aspect pratique et la pertinence des stratégies de contrôle proposées, des expériences en temps réel sont menées. Ces expériences confirment le développement théorique et montrent l’efficacité des méthodes pour des missions et des applications réelles.

Mots-clés: Robot mobile à roues, Robot mobile omnidirectionnel, Commande prédictive basée sur le modèle, Estimation stochastique, Fonctions de Laguerre, Propagation résiliente, Stabilité.

ABSTRACT

This project focuses on the autonomous tracking control of wheeled mobile robots, with a specific emphasis on omnidirectional robots capable of instantaneous movement in any direction without reorientation. The project's primary focus is the development of nonlinear control strategies based on predictive control principles. These strategies are designed to deliver reliable tracking performance for mobile robot systems that exhibit nonlinearities and operational constraints.

The project begins by developing the kinematic representation of the omnidirectional robot, which serves as the foundation for subsequent control design. Next, two distinct control approaches are formulated for the task of trajectory tracking for omnidirectional mobile robots, particularly in scenarios characterized by nonlinearities and operational constraints. The first method is an optimal predictive method utilizes non-iterative linearization techniques to effectively handle nonlinearities in the system. It also incorporates Laguerre functions to enhance computational efficiency, reducing the computational cost of control. The second approach leverages the inherent nonlinear nature of the robot's dynamics and employs resilient optimization methods to address the computational complexity associated with this approach. Stability analyses is conducted to determine the necessary conditions to achieve nominal stability for the nonlinear controller.

Both control methods are rigorously verified in a simulated environment. Additionally, the performance of these methods is compared against benchmark methods from existing literature, demonstrating their effectiveness and capabilities. To further validate the practicality and suitability of the proposed control strategies, real-time experiments are conducted. These experiments confirm the theoretical development and demonstrate the effectiveness of the methods for real-world missions and applications.

Keywords: Wheeled mobile robot, Omnidirectional mobile robot, Model predictive control, Stability, Stochastic estimation, Laguerre functions, Resilient propagation

1. INTRODUCTION

1.1 Overview

The past two decades have witnessed a huge evolution in the hardware and software technologies, which subsequently led to the presence of a wide range of automated machines in everyday activities, including mobile robots. With a quick look around, one can notice the increasing endorsement of mobile robots in different sectors of society, which nowadays are adopted for transportation, heavy-duty works, and repeatable tasks in warehouses and industries [1], for monitoring hazardous and contaminated environments [2], and for leading exploration missions in outer space [3], in underground mines [4], and even in the deep oceans [5]. Mobile robots have also entered the household environment as personal assistances [6] or for entertainment [7], they've joined rescue missions [8] and military operations [9] and became a powerful tool in educational systems [10].

The roles of mobile robots vary from one utilization to another, based on the needs of the field of application, and the type of solutions required. For instance, in the health sector, due to the recent COVID-19 pandemic, the closeness of a human has been considered dangerous for patients, hence, the urge to free some positions of the human operators increased [11]. Although the presence of the human staff remains mandatory, some basic operations, like patients monitoring, temperature measurements, and delivering products and documents can be performed with the assistance of well automated mobile robots. Another promising domain of application for mobile robots is in the mine industries. Underground mines are considered particularly unsafe environments, due to several sources of danger, including falling rocks, lack of light, confinement and the occurrence of toxic gases [12, 13]. In such an environment, mobile robots can be enrolled to perform a wide variety of tasks, including automatic inspections, exploration of unsafe areas, measurement of environmental conditions and participating in rescue missions [14]. By involving mobile robots in the mining operations, many dangers can be avoided when sending machines instead of humans in hazardous areas. These robots can also work alongside humans, which improve the efficiency and accuracy factors. Some mining

operations, such as transportation, have started the transition into automation [15]; however, a full automated mine is far from being complete.

Presently, mobile robots are increasingly being trusted to execute tasks in well-known constructed environments, on flat and compact floors, and they are widely commercialized [16]. Furthermore, with the advancements in sensor and camera technologies, mobile robots are now also used in unknown environments for exploration missions. However, their application in these unbuilt environments, such as agricultural fields and underground mines, is still constrained. In this kind of unknown terrains, roads conditions and environmental conditions are expected to be severe, and mobile robots require a higher degree of autonomy, adaptation and robustness, with good knowledge of their surroundings. One of the key operating conditions for mobile robots is trajectory tracking which aims to converge the robot to the desired trajectory which can be uploaded or generated using path-planning methods [17].

1.2 Mobile robots

Mobile robots are systems with moving platforms, remotely controlled, or free to move autonomously from one place to another in their predefined surrounding workspace without any human intervention. With a large variety of applications where mobile robots can be used, different types of robots have been developed, to cover this range of usability, and therefore, today's robots can fly, jump, run, walk, skid, swim and roll [18]. Based on their locomotion, mobile robots can be classified into three major categories: unmanned aerial vehicles, autonomous underwater vehicles and land-based robots [19].

Land-based robots, also known as Unmanned Ground Vehicles (UGVs), are the most common among the robotic society, which can be justified by their design's simplicity and the exposure to greater knowledge [20]. UGVs can be further divided, based on the locomotion's mechanism, into few sub-categories where the best-known ones are as follows:

- Wheeled mobile robots (WMRs)
- Legged (or walking) mobile robots
- Tracked mobile robots (slip/skid locomotion)
- Hybrid ground's robots

The majority of these robots are inspired from their biological analogous organisms, due to the success of these biological systems in moving smoothly through the harshest non-built environments. However, copying these systems can be extremely complicated for several reasons including the mechanical complexity, the high degree of balancing needed and the high cost.

The wheel's invention is considered one of the most important developments in human history, that allowed more efficient transportation and for longer distances. The wheel also had deep impact on the development of technologies including mobile robots. WMRs are commonly used in a wide range of applications, due to their advantages over other types of robots which includes the simpler and more cost-effective design, the ability to move faster and more efficiently on smooth surfaces and the higher carrying capacity [18].

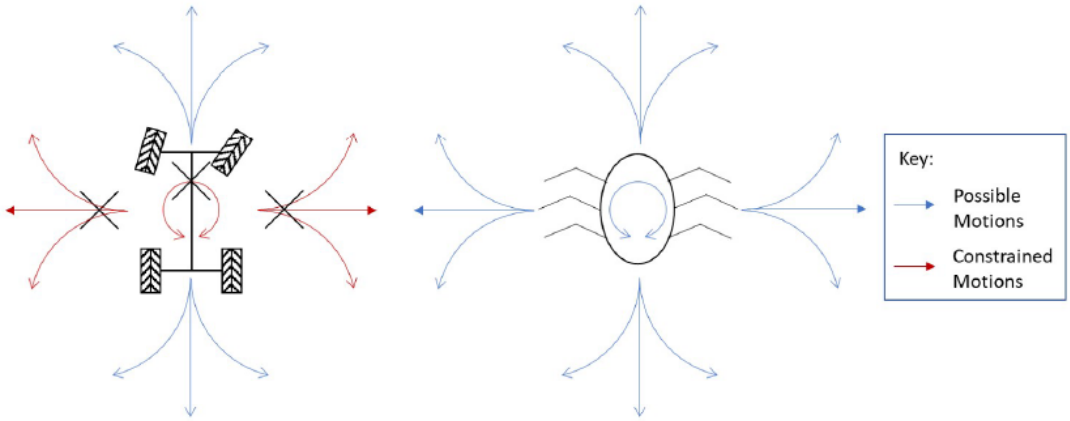


Figure 1
Non-holonomic (on the left) and holonomic (on the right) locomotion
Source: [19]

Depending on the type of wheels used, their number and their positions, we can distinguish many types of WMRs where most of them can be placed into two categories: Holonomic and non-holonomic mobile robots where the main difference between them lies in their ability to move and turn in different directions. Holonomic systems can move in any direction instantaneously, whereas the instant movement of non-holonomic systems is restricted as illustrated in Figure 1.

1.2.1 Non-Holonomic WMRs

This type of WMRs has more degrees of freedom than control inputs which means they don't have full control of all their degrees of freedom and therefore they must turn in order to change direction, and that limits their mobility. This type of robots uses conventional wheel types (Figure 2). There exist many types of non-holonomic WMRs, we mention two of them as follows:

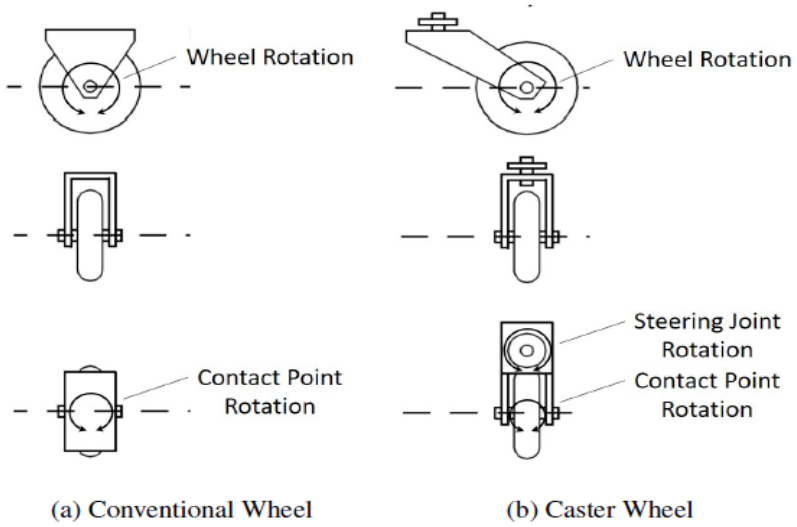


Figure 2
Conventional wheels
Source : [19]

- Differential wheeled robots

These are non-holonomic robots that have two separately driven wheels and at least one castor wheel for balance purpose. The change of direction is done by changing the rate of rotation of each driven wheel, and thus, there is no need of any steering motion (Figure 3).

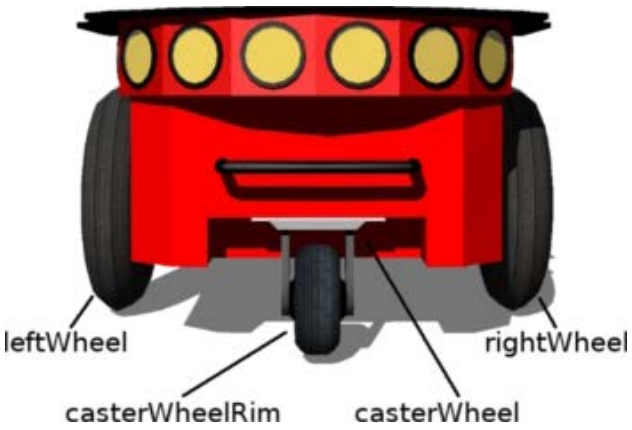


Figure 3
Differential wheeled robot: Pioneer 3-DX
Source: [21]

- Car-like robots

These robots have two fixed wheels placed on the same axis and two steerable wheels on another axis. The movement of this type of robots is determined by the speed of the fixed wheels and the orientation of the steering wheels (Figure 4).

Non-holonomic WMRs are often characterized by their simple mechanism along with the generally low price, however, their movement's constraint prevents them from being used in certain applications.

Figure 4
Car-like robot: MuSHR
Source: [22]

1.2.2 Holonomic WMRs

A robot is said to have holonomic movement when it's capable of moving in any direction without reorientation. This type of movement is achieved using special wheel types known as omnidirectional wheels which include Omni-wheels, Mecanum wheels and Ball wheels (Figure 5). The Omni-wheels have rollers attached tangentially around the girth of the wheel, which give the wheel three degrees of freedom: around the wheel axle, around the roller axle and around the contact point [19]. These wheels are mostly configured in three wheels format at 120 degrees from one another or using four wheels where adjacent wheels are perpendicular. Mecanum wheels have similar design to omni-wheels except that the rollers are placed at 45 degrees of the main wheel which allows the use of larger roller axis and hence increase the payload capacity.

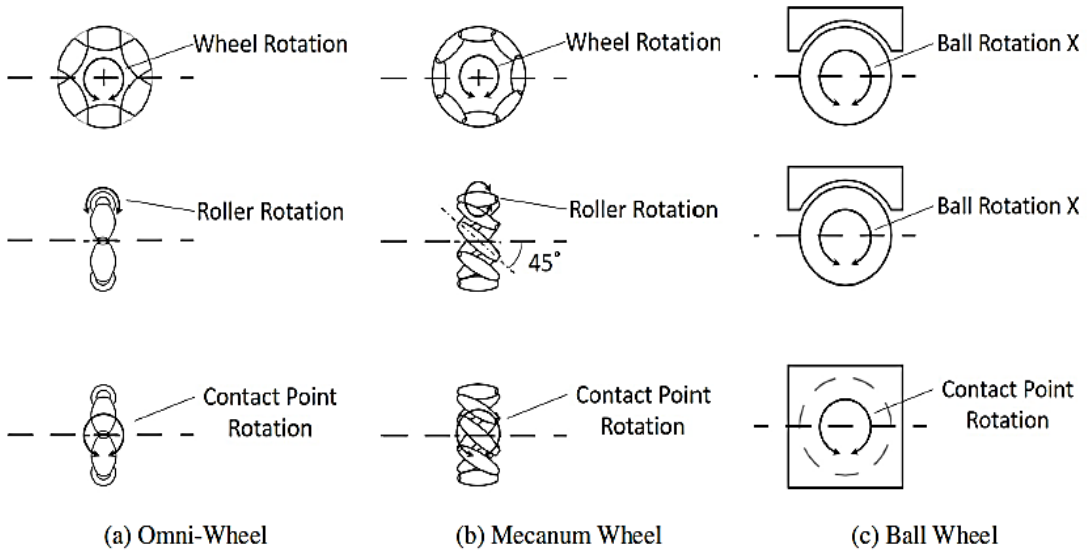


Figure 5
Omnidirectional wheels
Source: [19]

The Holonomic robots have the capacity to perform complex path following which gives it significant potential in different kinds of activities. On the other hand, the rollers mechanism adds mechanical complexity, and increases the skidding effects,

hence the need for advanced control methods that can achieve autonomy, stability, and robustness.

1.3 Control methods

As stated in the above sections, WMRs are intended for applications in different kinds of environments, which most likely to have severe operation conditions and to pose constraints on the robots' movement. Hence, a sufficient degree of autonomy is required to allow the WMRs to operate in such environments, which can be done by using advanced control methods. The nonlinearities in the WMRs systems and the complex operation conditions have motivated researchers to develop and apply a wide range of control architectures.

1.3.1 Trajectory tracking

Trajectory tracking is one of the most addressed problems in controller design which aims to bring the robot closer to the desired trajectory. Tracking accuracy and constraints handling are two main criteria in developing the control algorithms along with the ability to deal with nonlinear and multivariable characteristics of the systems and the applicability in real-time applications. In recent years, many control strategies have been proposed to solve the trajectory-tracking problem of the WMR [23, 24]. In [25], a PI controller tuned by an adaptive fuzzy logic was used as a high-level controller for an WMR. The fuzzy-PI, which corrects the kinematic errors, was paired with a linear quadratic regulator (LQR) as a low-level control of the velocities and accelerations. This combination showed significant improvement over a PI control alone; however, the use of an LQR for low-level control caused a deviation between the desired and the actual paths which greatly increased in the real-time application and led to unsatisfactory results. A control scheme taking into consideration the kinematic and dynamic uncertainties of the WMR was proposed in [26]. A sliding-mode-based observer was used to estimate these uncertainties, then a feedback linearization controller was used to handle such uncertainties. The controller showed a good trajectory-tracking performance. Nonetheless, since these methods cannot handle constraints directly, saturation was used to limit the control signals, which is not acceptable in practice. Another controller for the WMR was

presented in [27], which uses a linearizing adaptative algorithm for the kinematic control. This causes some singularities in the control signals. Such singularities are dealt with by switching to a sliding mode controller around them. Indeed, the resulting approach helps to reduce the control effort and eliminate the singularities; however, the risk of chattering appears which may harm the actuators. In [28], a bioinspired backstepping controller was proposed. It performs the tracking task while reducing the large velocity jumps that occur in the traditional backstepping control. A super-twisting sliding mode controller was also developed in [29] to overcome the chattering effect encountered in the traditional method. Both sliding mode and backstepping are efficient when controlling nonlinear systems. Nonetheless, they are limited to unconstrained problems as they cannot handle constraints explicitly. A combination of such methods with the barrier functions was proposed to deal with constrained problems [30, 31]. However, this often leads to an undesired design complication especially in the presence of complex nonlinear constraints. Model-free control schemes were used in [32, 33]. Using a visual serving strategy in [33] provides a unified algorithm for tracking and regulation. However, these model-free methods ignore useful information from the system model, and they are less adequate compared to systematic methods.

To overcome the above-mentioned problems, one can consider the model-based predictive control (MPC). It is one of the advanced techniques that now has a huge impact on the development of control systems and on research in feedback control areas and has achieved remarkable success in the practical field [34, 35]. This success of the MPC is attributed to many reasons. First, due to the finite control horizon, nonlinear systems dynamics, and process inputs, state and output constraints can be handled directly by the MPC algorithms. Moreover, the prediction aspect of this method over a future time horizon makes it possible to anticipate and remove the effect of disturbances, which leads to better tracking of the future trajectory. Finally, MPC principles and algorithms are relatively easy to understand and to extend to multi-input multi-output systems [36, 37]. The general idea of MPC is to solve an online open-loop optimization problem at each sampling time, and to find a trajectory

of future manipulated variables that optimize the future behavior of the system outputs within a limited time window. Traditionally, MPC was only applied to sufficiently slow systems due to the high computational cost required to perform the online optimization, but thanks to increased hardware efficiency, MPC is now applicable to systems with faster dynamics.

1.3.2 Linearization-based MPC

Since most systems are inherently nonlinear, linear MPC algorithm cannot be applied directly. Therefore, linearization mechanisms are needed to approximate the nonlinear behavior. In [38], the nonlinear system was modeled in the Weiner model structure, which divides the system into two parts, a linear time invariant system followed by a static nonlinear element. Then, linear MPC was used for the linear part and polynomial representation for the nonlinear part; however, for the Weiner model to properly describe the nonlinear aspects of the system, prior knowledge of these nonlinearities should be available, which is not the case for most systems, and a simple polynomial representation does not give an accurate description of these nonlinearities. Similarly, in [39], a NARMA-Volterra model was selected to represent the brain and used to predict neural activity. In addition, Laguerre functions were introduced to reduce the number of estimation parameters in the Volterra model, and then linear MPC was applied to solve the optimization problem. Nonetheless, Volterra models exhibit high level of complexity, which makes it impractical in modeling strong nonlinearities, and in order to reduce it, prior knowledge of the nonlinear aspect is required. Another popular way to deal with nonlinearities is to linearize the system at each time instant, along the desired trajectory using the Lyapunov method, and to use this linear approximation to compute the predicted future trajectory and then apply the well-known linear MPC [40]; however, when using an approximation at the current time instant to predict the whole future trajectory, the error of the linearization will accumulate, which leads to a poor prediction process. In [41], a duality-based control algorithm has been developed to control a two-wheeled differential robot. The approach uses the duality between optimal control and stochastic filtering to approximate the manipulated variables, and to linearize the nonlinear systems. The linearization and prediction processes were based on the duality without dependence on the future control signals. The algorithm

led to better approximation of the nonlinear plants compared to other linearization-based methods; however, the algorithm consists of two passes, forward for linearization and prediction, and backward for smoothing and control signals approximation, which double the computation time.

Although linear MPC does not usually require a high computational capacity, some multi-input multi outputs systems may have a large number of optimization variables, especially if a long control horizon is needed. This may increase the computational demand. One solution for this issue is to use Laguerre functions to parametrize the control variables. This allows the realization of a longer control horizon with fewer optimization parameters, which consequently reduces the computational time [42, 43]. In [44], Laguerre functions were used to parametrize linear MPC. The resulting algorithms were compared to other approaches and showed significant improvement regarding the computational cost and the number of optimization variables; however, only simulation results were given without real-time implementation.

In [45], the effect of the parametrization using Laguerre functions on the feasibility and performance of the MPC was analyzed, and it showed great improvement on the feasibility while maintaining a good performance; however, only dual mode MPC, which uses an infinite horizon for prediction, was considered. Recently in [46], an MPC controller parametrized by Laguerre functions was used to control a fast-switching electronic DC-DC converter allowing the use of a significantly short sampling time. Laguerre functions were first used with MPC in [47], which later has been expanded in [48] where a comprehensive study on the use of Laguerre functions with MPC is given, and which all the above-mentioned studies refer to; however, only linear systems that are supposed to remain constant during the entire prediction process are considered.

1.3.3 NMPC

MPC can explicitly handle nonlinearities, and since most systems are inherently nonlinear, many nonlinear model predictive control (NMPC) algorithms have been developed using iterative solutions to solve the optimization problem [49]. In [50], a

basic NMPC algorithm that uses the gradient descent method was applied to solve an WMR trajectory tracking with obstacle avoidance. The algorithm gave effective results in both simulation and real-time experiments; however, in the experiments, due to the high computational load, the movement direction and speed were fixed and only the orientation was controlled by NMPC. Using nonlinear systems directly in the NMPC algorithm often leads to undesired complexity and high computational demand. Therefore, studies have been conducted to overcome these problems [51-53].

Selection of NMPC parameters such as prediction and control horizons correctly could be a key factor to minimizing the computational burden [54]. In [55], the length of the prediction horizon was determined at each control step separately using a multilayer neural network based on the error magnitude. Moreover, hardware equipment was chosen effectively to optimize the speed of the performing actuators and increase the computational capability. This combination resulted in 40% faster computation compared to the conventional NMPC. Unfortunately, these adaptations are problem specific and cannot be applied to all systems.

The optimization algorithm is the most addressed component of NMPC to improve computational efficiency. To this end, numerous approaches were developed to deal with the Nonlinear Optimization Problem (NOP). In [56] and [57], the continuation method was combined with the Generalized Minimal Residual method (GMRES) to solve the NOP. The former method transforms the nonlinear problem to a linear one which is then solved by the latter one. This approach requires that the system and the objective function to have specific characteristics, hence it is not suitable for all problems. In addition, when dealing with complex systems and performance indices, programing this method is very challenging. Neural optimizations are commonly used when solving the NOP [58, 59]. In [58], a one-layer projection neural network was presented for the quadratic optimization. A faster convergence was achieved while maintaining computational efficiency. In [60] Neural-dynamic optimization was considered where the MPC was iteratively transformed to a quadratic programing problem, which is solved using primal-dual neural network. The proposed algorithm was applied to solve the trajectory tracking of a mobile robot and significantly

reduced the computation complexity. However, neural-based methods require a substantial amount of data for training which might be unavailable. Their generalization to different problems may also be difficult without retraining. Another type of optimization algorithm is the metaheuristic strategies. They are general-purpose techniques employed successfully in a vast range of NOP. The suitability of these methods to the NMPC was studied in [61] where three of them, the particle swarm optimization, the ant colony optimization, and the gravitational search algorithm, were compared. Results showed a good tracking performance with sufficiently low computational burden. However, these methods only provide an approximate solution, therefore, neither convergence nor optimality can be guaranteed.

Conventional exact methods remain the most reliable when it comes to guaranteeing convergence and optimality, which led to the development of a wide variety of these approaches, including Interior Point algorithm (IP) [62, 63], Active-Set algorithm (AS) [64, 65], sequential programing (SQ) and many others. Despite the great computational advantages achieved by these methods, their application to fast dynamic systems such as WMR is still restricted to specific cases, such as linear optimization and problems with limited number of optimization variables [66]. A gradient-based method was introduced in [67] to optimize the neural networks' parameters. This approach is called resilient propagation (RPROP), and it aims to achieve faster convergence by using only the sign of the partial derivative of the error function, and replacing its value with an adaptive step size which evolves during the optimization process. It is widely used to optimize neural networks, whereas its application outside of this domain remains limited [68, 69]. In [70], NMPC using the classical RPROP algorithm was presented for trajectory tracking of a quadrotor. Results showed important reduction in computational demand while maintaining tracking accuracy. However, the original RPROP does not ensure convergence as shown in [71, 72]. In [71], A Robust Convergent variant of RPROP (ARCPROP) was presented which ensures convergence. In addition to RPROP mechanism, the new variant considers the overall error function and backtracks

along all dimensions if the previous step did not achieve sufficient reduction. To the best of our knowledge, none of the existing approaches considered constrained NMPC using RPROP.

Overall, the control of WMRs remains an ongoing challenge within the research community. New methods are emerging daily, each with their advantages and disadvantages, to achieve full autonomy of WMR.

1.4 Motivation

Numerous control methodologies have been proposed with the aim of achieving effective autonomous control for WMR. The primary emphasis in many of these approaches has been on enhancing tracking performance, a critical component in progressing toward full autonomy. Ensuring stability and feasibility has also received attention to ensure good performance and real-time applicability.

However, a common limitation among these methods is the predominant focus on the primary control objective, often overlooking the constraints inherent in nearly all real-world systems. Moreover, some of these methods completely disregard the system model, which can contain valuable information essential for controller design in many cases. Additionally, a notable drawback in several proposed methods is their limited suitability for on-board implementation due to the high computational complexity involved.

Motivated by the promising advantages offered by MPC and NMPC in attaining autonomous control for mobile robots, this research aims to leverage these two approaches in the development of a novel controller for WMR.

1.5 Objectives

The overarching aim of this project is to develop a more practical tracking algorithm for WMR that adeptly handles nonlinearities, maintains high performance, and demands minimal computational resources, enabling real-time applicability. To achieve this objective, two distinct approaches are explored in this research: A linearization-based method and an approach that directly utilizes the nonlinear

characteristics of the system. Incorporating constraints on control and state variables is a key aspect to ensure that the proposed solutions lie within the feasible operating space of the robot. Comprehensive testing has been conducted using the new Robotino Festo Omnidirectional Mobile Robot (OMR) at UQAT's indoor laboratory to validate and assess these approaches.

The specific objectives of this research are as follows:

- Formulate the kinematic representations for the newly introduced OMR.
- Design model-based controllers adapted for the task of trajectory tracking, with a focus on accommodating both state and control constraints.
- Conduct a comparative analysis of the controllers' performance against other existing methods documented in the literature.
- Implement the designed controllers in practical, real-time applications to assess their effectiveness under real-world conditions.

1.6 Methodology

The objectives outlined above are realized through the following methodology:

- Develop kinematic representations of the newly introduced Robotino-Festo OMR. This involves the creation of MATLAB/Simulink® toolboxes to facilitate simulation and practical implementation.
- Establish the theoretical foundations for the novel controllers, including the formulation of associated optimization problems. These approaches are specifically designed to address the system's nonlinearities, constraints, and computational complexity.
- Conduct a stability analysis to determine the necessary conditions for achieving nominal stability of the nonlinear system.
- Perform a comparative evaluation of the proposed methods in a MATLAB/Simulink® environment against benchmark methods documented in the literature. The assessment encompasses tracking accuracy and computational efficiency.

- Validate the designed controllers through real-time trajectory tracking applications using the Robotino Festo mobile robot. These practical tests provide valuable insights into the controllers' real-world performance.

1.7 Project contributions

Despite the extensive efforts invested in investigating the tracking challenges faced by WMR, it remains a challenging subject in motion control. This work contributes to this domain in the following ways:

- Enhanced linearization-based tracking algorithm: Leveraging the duality principle linking stochastic filtering and optimal control, an advanced controller is introduced. This controller offers an improved approximation of the nonlinear system behavior by non-iteratively linearizing the system along the prediction horizon. This enhancement significantly boosts tracking performance. Furthermore, the integration of Laguerre functions helps offset the additional computational load. The novel contribution here is the fusion of stochastic estimation with the MPC setup, eliminating the iterative nature typically associated with these methods.
- NMPC with resilient propagation-based algorithm for optimization: This work investigates the utilization of an RPROP-based optimization algorithm within the framework of NMPC. The research explores properties related to convergence, the handling of constraints, and the real-time feasibility of this approach. Additionally, the stability properties of the controlled system are analysed and necessary conditions for nominal stability are presented.

1.8 Thesis outline

The outline of this thesis is as follows:

The first chapter provides an introduction to WMRs including a background on mobile robot technology and various control methods. It also explains the motivation behind this thesis and its objectives. The second chapter presents the kinematic properties of the OMR and their different representations. It also highlights the traditional setup for the two control methods MPC and NMPC.

In the third chapter, the duality between the stochastic filtering and optimal control is presented and incorporated into the MPC setup along with the Laguerre functions. Simulation and practical testing are conducted and reported to validate the new method.

The fourth chapter presents the second approach utilizing the resilient propagation method and the external penalty method to solve the constrained nonlinear optimization problem of the NMPC algorithm.

In the fifth chapter, a stability analyse for the NMPC controller is conducted and the necessary terminal components are constructed to achieve nominal stability. Finally, the conclusion and recommendations of the thesis are given.

1.9 Conclusion

This chapter highlights the expanding role of mobile robots across various domains, driven by technological progress. It discusses the different challenges that WMRs encounter, particularly in complex and unpredictable environments, and underscores the critical need for efficient trajectory tracking controllers. This chapter also outlines the motivation behind the research, defines the objectives, and describes the methodologies used. These elements collectively pave the way for the development of advanced model-based predictive control strategies tailored for WMRs.

2. PRELIMINARY SETUP

2.1 Introduction

This chapter presents an introduction to the various techniques applied in this project, including their essential configurations. We start by introducing the robot employed in this study, outlining its characteristics, constraints, and presenting its diverse kinematics representations. Following this, we introduce the Model Predictive Control (MPC) method, covering its application to both linear and nonlinear systems. Lastly, a concise overview of Laguerre functions and their relevance is provided.

2.2 Modeling the Omnidirectional Mobile Robot (OMR)

For this study, the Robotino-Festo three-wheeled OMR is considered. Robotino® (Figure 6) is a mobile robotic system created by Festo Didactic [73]. Its main purpose is to facilitate practical learning and skill development in various fields, including robotics, mechatronics, measurements, wireless control, signal processing, and programming, among others. Some of its notable features include autonomous movement in all directions, with the ability to identify and avoid obstacles.

Figure 6
Robotino-Festo OMR and its Omni-drive
Source: [73]

It is equipped with embedded sensors and actuators, enabling wireless communication with other devices. Additionally, Robotino® supports the integration of new components through a mounting tower, supports an open-source concept, and provides software interfaces compatible with various programming languages. The whole system is controlled by an embedded PC to COM Express specifications

with Intel i5, 2.4 GHz dual core, 8 GB RAM and 23 GB SSD. For the motors control, a 32-bit microcontroller is used. It generates the PWM signals for actuating the DC motors using a PID controller. The microcontroller is also used to correct the sensors data. A planetary gear unit with transition ratio 32:1 is used between the drive shafts and omni-wheels [74, 75]. The robot has a maximum translational and rotational speeds of 2m/s and 2rad/s respectively, and it accepts translational and rotational velocities as inputs, which are expected to be updated every 70ms.

Robotino® has three degrees of freedom and can achieve any translational and rotational movements regardless of its initial orientation. The three omni-wheels, placed at 120° from each other, allow the robot to turn on the spot and to move in any direction.

2.2.1 Kinematics modeling

To derive the kinematics representation of the OMR, we define a global frame (O, X, Y) and a locale (moving) frame (O_r, X_r, Y_r) connected to the robot (Figure 7). Let (x, y, θ) denote the position and orientation of the OMR in the global frame, and (x_r, y_r, θ_r) denote the position and orientation in the local frame. The local coordinates can be transposed into the global coordinates by:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = R_T . \begin{pmatrix} x_r \\ y_r \\ \theta_r \end{pmatrix} \tag{2.1}$$

where R_T is the transformation matrix which maps the locale frame into the global frame and is given as:

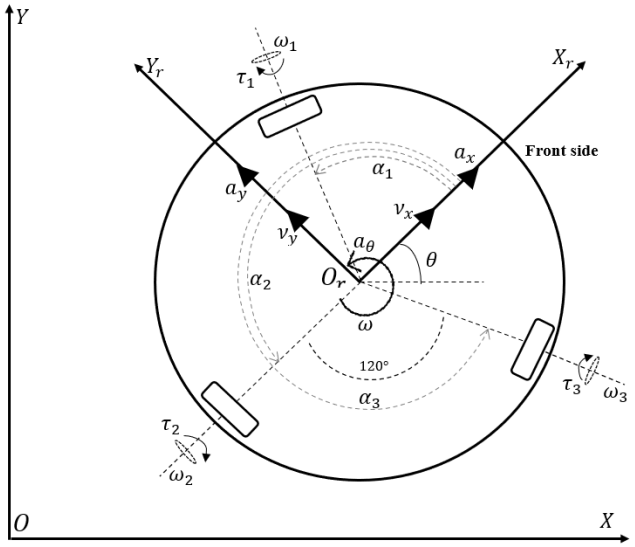


Figure 7
Locale and global frames

$$R_T = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

In the body frame, let (v_x, v_y) and (ω) be the translational and rotational velocities of the OMR respectively. Additionally, define (a_x, a_y) the translational accelerations and (a_θ) the rotational acceleration of the robot in the local frame. Then the state vector is chosen as:

$$q = \begin{pmatrix} x & y & \theta & v_x & v_y & \omega \end{pmatrix}^T \tag{2.3}$$

and the vector of manipulated variables as:

$$u = \begin{pmatrix} a_x & a_y & a_\theta \end{pmatrix}^T \tag{2.4}$$

By using the following relation:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = R_T . \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} \tag{2.5}$$

the kinematic equations of the OMR can be written as follows:

$$\begin{aligned} \dot{x} &= v_x \cos\theta - v_y \sin\theta \\ \dot{y} &= v_x \sin\theta + v_y \cos\theta \\ \dot{\theta} &= \omega \\ \dot{v}_x &= a_x \\ \dot{v}_y &= a_y \\ \dot{\omega} &= a_\theta \end{aligned} \tag{2.6}$$

which can be written in compact form as follows:

$$\dot{q} = f\left(q,u\right) = \begin{pmatrix} v_x \cos\theta - v_y \sin\theta \\ v_x \sin\theta + v_y \cos\theta \\ \omega \\ a_x \\ a_y \\ a_\theta \end{pmatrix} \tag{2.7}$$

Considering that the OMR requires translational and rotational velocities as inputs, the model from equation (2.7) is adequate for the development of various controllers and will be employed throughout the remaining chapters of this dissertation.

- Discretization

For many control methods, including MPC, a discrete model of the system is needed to design the controller. To obtain the discrete kinematic representation of the OMR, we apply the Euler method to equation (2.7) as follows:

$$q_{k+1} = q_k + T.f(q_k, u_k) \tag{2.8}$$

which yields the following discrete model:

$$q_{k+1} = f_k(q_k, u_k) \equiv f_k = \begin{pmatrix} x_k + \nu_{xk} T \cos\theta_k - \nu_{yk} T \sin\theta_k \\ y_k + \nu_{xk} T \sin\theta_k + \nu_{yk} T \cos\theta_k \\ \theta_k + \omega_k T \\ \nu_{xk} + a_x T \\ \nu_{yk} + a_y T \\ \omega_k + a_\theta T \end{pmatrix} \tag{2.9}$$

where T is the sampling time.

- Jacobian matrices

If a linearization-based method is to be used, computing the Jacobian matrices becomes mandatory, which can be done as follows:

$$A_k = \frac{\partial f_k}{\partial q_k} = \begin{pmatrix} 1 & 0 & -\nu_{xk} T \sin\theta_k - \nu_{yk} T \cos\theta_k & T\cos\theta_k & -T\sin\theta_k & 0 \\ 0 & 1 & \nu_{xk} T \cos\theta_k - \nu_{yk} T \sin\theta_k & T\sin\theta_k & T\cos\theta_k & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

and

$$B = \frac{\partial f_k}{\partial u_k} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{pmatrix} \tag{2.11}$$

which can be used to compute the linear approximation of the system.

- OMR geometry

In certain applications, it is necessary to include the OMR's wheels velocities and accelerations in the kinematic model. To accomplish this, the OMR's geometry must be taken into consideration. The transformation matrix that converts the local speeds of the robot into the wheel speeds can be expressed as follows:

$$S = \frac{1}{r}\begin{pmatrix} -\sin\alpha_1 & \cos\alpha_1 & L_r \\ -\sin\alpha_2 & \cos\alpha_2 & L_r \\ -\sin\alpha_3 & \cos\alpha_3 & L_r \end{pmatrix} \tag{2.12}$$

where α_i is the orientation of the axel of the *ith* wheel with respect to the local frame, r and L_r are the wheel's radius and the robot's radius respectively. Define the vector of the wheels' rotational velocities as:

$$\Omega_r = \begin{pmatrix} \omega_1 & \omega_2 & \omega_3 \end{pmatrix}^T \tag{2.13}$$

where ω_i is the rotational speeds of the *ith* wheel. Putting (2.4), (2.12) and (2.13) together, we obtain the relation between the robot accelerations and the wheels accelerations as:

$$u = S^{-1}\dot{\Omega}_r \tag{2.14}$$

Equation (2.7) can be written in matrix format as follows:

$$\dot{q} = f(q,u) = \begin{pmatrix} 0^{3\times3} & R_T \\ 0^{3\times3} & 0^{3\times3} \end{pmatrix} q + \begin{pmatrix} 0^{3\times3} \\ I^{3\times3} \end{pmatrix} u \tag{2.15}$$

where 0 and I represent the Zero matrix and the identity matrix respectively. Replacing (2.14) in (2.15), we obtain the following kinematic representation:

$$\dot{q} = f\left(q, \dot{\Omega}_r\right) = \begin{pmatrix} 0^{3\times3} & R_T \\ 0^{3\times3} & 0^{3\times3} \end{pmatrix} q + \begin{pmatrix} 0^{3\times3} \\ S^{-1} \end{pmatrix} \dot{\Omega}_r \tag{2.16}$$

2.2.2 Error Kinematics

When addressing a tracking problem, a frequently used method involves converting the problem into a regulation problem with the objective of minimizing the error to zero. To achieve this, a virtual reference robot is considered, and a model that describes the error kinematics between the actual robot and the reference robot is formulated. Define the global frame (O, X, Y), the moving frame (O_r, X_r, Y_r) associated with the real robot, and a second moving frame (O_d, X_d, Y_d) linked to the virtual robot that follows the desired trajectory (Figure 8).

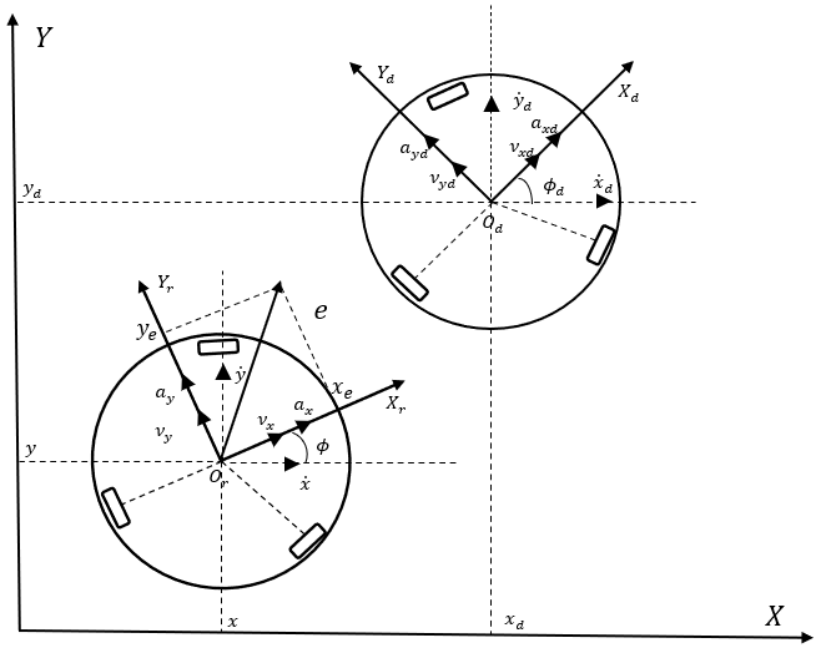


Figure 8
Real and virtual robots

The state vector of the reference robot defined in (O, X, Y) is given as:

$$q_d = \begin{pmatrix} x_d & y_d & \theta_d & \dot{x}_d & \dot{y}_d & \omega_d \end{pmatrix}^T \tag{2.17}$$

and the state vector of the real robot in (O, X, Y) is:

$$q = \begin{pmatrix} x & y & \theta & \dot{x} & \dot{y} & \omega \end{pmatrix}^T \tag{2.18}$$

Let q_e be the tracking error defined in the local frame and given as:

$$q_e = \begin{pmatrix} x_e & y_e & \theta_e & v_{xe} & v_{ye} & \omega_e \end{pmatrix}^T \tag{2.19}$$

Then we can write:

$$q_e = \begin{pmatrix} R_T^T & 0^{3\times3} \\ 0^{3\times3} & R_T^T \end{pmatrix} (q_d - q) \tag{2.20}$$

Here the inverse of R_T is equal to its transpose because it is an orthonormal matrix. Equation (2.20) can be further elaborated and lead to the following equations:

$$\begin{aligned} x_e &= (x_d - x)\cos\theta + (y_d - y)\sin\theta \\ y_e &= -(x_d - x)\sin\theta + (y_d - y)\cos\theta \\ \theta_e &= \theta_d - \theta \\ v_{xe} &= (\dot{x}_d - \dot{x})\cos\theta + (\dot{y}_d - \dot{y})\sin\theta \\ v_{ye} &= -(\dot{x}_d - \dot{x})\sin\theta + (\dot{y}_d - \dot{y})\cos\theta \\ \omega_e &= \omega_d - \omega \end{aligned} \tag{2.21}$$

To obtain the kinematic equations, we differentiate (2.21) with respect to time as shown below:

$$\begin{aligned} \dot{x}_e &= (\dot{x}_d - \dot{x})\cos\theta + (\dot{y}_d - \dot{y})\sin\theta - (x_d - x)\dot{\theta}\sin\theta + (y_d - y)\dot{\theta}\cos\theta \\ &= \omega\left(-(x_d - x)\sin\theta + (y_d - y)\cos\theta\right) - (\dot{x}\cos\theta + \dot{y}\sin\theta) + \dot{x}_d\cos\theta + \dot{y}_d\sin\theta \\ &= \omega y_e - v_x + \dot{x}_d\cos\theta + \dot{y}_d\sin\theta \\ &= \omega y_e + v_{xe} \end{aligned}$$

$$\begin{aligned}\dot{y}_e &= -\left(\dot{x}_d - \dot{x}\right)\sin\theta + \left(\dot{y}_d - \dot{y}\right)\cos\theta - \left(x_d - x\right)\dot{\theta}\cos\theta - \left(y_d - y\right)\dot{\theta}\sin\theta \\ &= -\omega\left(\left(x_d - x\right)\cos\theta + \left(y_d - y\right)\sin\theta\right) - \left(-\dot{x}\sin\theta + \dot{y}\cos\theta\right) - \dot{x}_d\sin\theta + \dot{y}_d\cos\theta \\ &= -\omega x_e - v_y - \dot{x}_d\sin\theta + \dot{y}_d\cos\theta \\ &= -\omega x_e + v_{ye}\end{aligned}$$

$$\dot{\theta}_e = \omega_d - \omega$$

$$\begin{aligned}\dot{v}_{xe} &= \left(\ddot{x}_d - \ddot{x}\right)\cos\theta + \left(\ddot{y}_d - \ddot{y}\right)\sin\theta - \left(\dot{x}_d - \dot{x}\right)\dot{\theta}\sin\theta + \left(\dot{y}_d - \dot{y}\right)\dot{\theta}\cos\theta \\ &= \omega\left(-\left(\dot{x}_d - \dot{x}\right)\sin\theta + \left(\dot{y}_d - \dot{y}\right)\cos\theta\right) - \left(\ddot{x}\cos\theta + \ddot{y}\sin\theta\right) + \ddot{x}_d\cos\theta + \ddot{y}_d\sin\theta \\ &= \omega v_{ye} - a_x + \ddot{x}_d\cos\theta + \ddot{y}_d\sin\theta \\ &= \omega v_{ye} - a_x + \ddot{x}_d\cos\left(\theta_d - \theta_e\right) + \ddot{y}_d\sin\left(\theta_d - \theta_e\right) \\ &= \omega v_{ye} - a_x + \ddot{x}_d\left(\cos\theta_d\cos\theta_e + \sin\theta_d\sin\theta_e\right) + \ddot{y}_d\left(-\cos\theta_d\sin\theta_e + \sin\theta_d\cos\theta_e\right) \\ &= \omega v_{ye} - a_x + \left(\ddot{x}_d\cos\theta_d + \ddot{y}_d\sin\theta_d\right)\cos\theta_e - \left(-\ddot{x}_d\sin\theta_d + \ddot{y}_d\cos\theta_d\right)\sin\theta_e \\ &= \omega v_{ye} - a_x + a_{xd}\cos\theta_e - a_{yd}\sin\theta_e\end{aligned}$$

$$\begin{aligned}\dot{v}_{ye} &= -\left(\ddot{x}_d - \ddot{x}\right)\sin\theta + \left(\ddot{y}_d - \ddot{y}\right)\cos\theta - \left(\dot{x}_d - \dot{x}\right)\dot{\theta}\cos\theta - \left(\dot{y}_d - \dot{y}\right)\dot{\theta}\sin\theta \\ &= -\omega\left(\left(\dot{x}_d - \dot{x}\right)\cos\theta + \left(\dot{y}_d - \dot{y}\right)\sin\theta\right) - \left(-\ddot{x}\sin\theta + \ddot{y}\cos\theta\right) - \ddot{x}_d\sin\theta + \ddot{y}_d\cos\theta \\ &= -\omega v_{xe} - a_y - \ddot{x}_d\sin\theta + \ddot{y}_d\cos\theta \\ &= -\omega v_{xe} - a_y - \ddot{x}_d\sin\left(\theta_d - \theta_e\right) + \ddot{y}_d\cos\left(\theta_d - \theta_e\right) \\ &= -\omega v_{xe} - a_y - \ddot{x}_d\left(\sin\theta_d\cos\theta_e - \cos\theta_d\sin\theta_e\right) + \ddot{y}_d\left(\cos\theta_d\cos\theta_e + \sin\theta_d\sin\theta_e\right) \\ &= -\omega v_{xe} - a_y + \left(-\ddot{x}_d\sin\theta_d + \ddot{y}_d\cos\theta_d\right)\cos\theta_e + \left(\ddot{x}_d\cos\theta_d + \ddot{y}_d\sin\theta_d\right)\sin\theta_e \\ &= -\omega v_{xe} - a_y + a_{yd}\cos\theta_e + a_{xd}\sin\theta_e\end{aligned}$$

$$\dot{\omega}_e = a_{\theta d} - a_\theta$$

Finally, we obtain the tracking error kinematics as follows:

$$\dot{q}_e = \begin{pmatrix} \omega y_e + v_{xe} \\ -\omega x_e + v_{ye} \\ \omega_d - \omega \\ \omega v_{ye} - a_x + a_{xd}\cos\theta_e - a_{yd}\sin\theta_e \\ -\omega v_{xe} - a_y + a_{yd}\cos\theta_e + a_{xd}\sin\theta_e \\ a_{\theta d} - a_\theta \end{pmatrix} \tag{2.22}$$

- Discretization

To obtain the discrete model of the error kinematics, we apply the Euler method as in (2.8) which leads to the following representation.

$$q_{e,k+1} = \begin{pmatrix} x_{ek} + T\left(\omega_k y_{ek} + v_{xek}\right) \\ y_{ek} + T\left(-\omega_k x_{ek} + v_{yek}\right) \\ \theta_{ek} + T\omega_{ek} \\ v_{xek} + T\left(\omega_k v_{yek} - a_{xk} + a_{xdk}\cos\theta_{ek} - a_{ydk}\sin\theta_{ek}\right) \\ v_{yek} + T\left(-\omega_k v_{xek} - a_{yk} + a_{ydk}\cos\theta_{ek} + a_{xdk}\sin\theta_{ek}\right) \\ \omega_{ek} + T\left(a_{\theta dk} - a_{\theta k}\right) \end{pmatrix} \tag{2.23}$$

where T is the sampling time. Define:

$$u_{ek} = \begin{pmatrix} u_{e1k} \\ u_{e2k} \\ u_{e3k} \end{pmatrix} = \begin{pmatrix} a_{xdk}\cos\theta_{ek} - a_{xk} \\ a_{ydk}\cos\theta_{ek} - a_{yk} \\ a_{\theta dk} - a_{\theta k} \end{pmatrix} \tag{2.24}$$

By substituting (2.24) in (2.23), we obtain:

$$q_{e,k+1} = f_e(q_{ek}, u_{ek}) = \begin{pmatrix} x_{ek} + T\left(\omega_k y_{ek} + v_{xek}\right) \\ y_{ek} + T\left(-\omega_k x_{ek} + v_{yek}\right) \\ \theta_{ek} + T\omega_{ek} \\ v_{xek} + T\left(\omega_k v_{yek} - a_{ydk}\sin\theta_{ek}\right) + Tu_{e1k} \\ v_{yek} + T\left(-\omega_k v_{xek} + a_{xdk}\sin\theta_{ek}\right) + Tu_{e2k} \\ \omega_{ek} + Tu_{e3k} \end{pmatrix} \tag{2.25}$$

- Linearization around the reference trajectory

When needed, a linearization around the reference trajectory can be done by computing the Jacobian matrices. In such a case, q_k would be equal to q_{dk}, which leads to $q_{ek} = 0^{6\times1}$. The Jacobian matrices can be found as follows:

$$A_{ek} = \left.\frac{\partial f_e(q_{ek}, u_{ek})}{\partial q_{ek}}\right|_{q_k = q_{dk}} = \begin{pmatrix} 1 & T\omega_{dk} & 0 & T & 0 & 0 \\ -T\omega_{dk} & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & -Ta_{ydk} & 1 & T\omega_{dk} & 0 \\ 0 & 0 & Ta_{xdk} & -T\omega_{dk} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.26}$$

$$B_{ek} = \frac{\partial f_e(q_{ek}, u_{ek})}{\partial u_{ek}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{pmatrix} \tag{2.27}$$

These matrices can be used to approximate the nonlinear behavior around any reference point.

2.3 Model predictive control

Model Predictive Control (MPC) emerged in the late seventies and has undergone significant development since its inception. The term "Model Predictive Control" encompasses a wide spectrum of control methods that explicitly rely on a process model to generate the control signals. The fundamental ideas in predictive control methods involve explicit use of a model to predict the process output at future time instances. This includes minimizing an objective function taking into account the predicted behavior to compute the control signals, and applying the first control signal from the calculated sequence at each step to control the system [76]. The different MPC algorithms primarily distinguish themselves based on the model used to describe the system, and the objective functions to be minimized.

MPC offers several advantages over other control methods, making it a compelling choice for various applications. These advantages include:

- Intuitive Concepts: MPC concepts are easy to understand making them accessible to individuals with limited knowledge of control systems.
- Constraint Handling: MPC can easily incorporate constraints into the control design.
- Multivariable Control: MPC handles multivariable systems effectively, making it applicable to processes of large scale.
- Future References: MPC is valuable when future references are known, such as in robotics, as it can optimize control based on anticipated changes.

While MPC offers numerous advantages, it's not without drawbacks. One of them is the requirement for an appropriate model of the controlled process to be available. The design algorithm relies on prior knowledge of the process model and operates independently of the real system. However, discrepancies between the actual, real-world process and the model used in the control algorithm may yield suboptimal performance. Another drawback of MPC is its computational complexity, especially in the nonlinear case and in the presence of constraints. While the computational power available today can handle the demands of MPC algorithms, it's essential to

recognize that many industrial process-control computers may not have abundant computing resources. That why the balance between control algorithm execution and other essential tasks is a crucial aspect of practical MPC implementation in industrial settings.

2.3.1 Linear model predictive control

In the context of linear time-invariant systems, MPC has reached a high level of development and acceptance. This maturity is attributed to the inherent linearity of these systems which simplifies the prediction process, allowing for a straightforward formulation. Even when the control objective involves nonlinear terms, MPC remains capable of handling it without resorting to complex iterative optimization methods. In this section, the setup for the traditional linear MPC of the OMR is presented. The main elements of the optimal predictive controller are the cost function that describes the control objectives, and the system model used for prediction.

- Prediction process

After linearization around an operating point, the nonlinear discrete state space representation (2.9) becomes:

$$\begin{aligned} q_{k+1} &= Aq_k + Bu_k \\ z_k &= Cq_k \end{aligned} \tag{2.28}$$

where A and B are given in equations (2.10) and (2.11) respectively, $z_k\left(6\times 1\right)$ is the output vector which contains the position, orientation, and velocities of the OMR, and $C\left(6\times 6\right)$ is the output matrix.

At an ith random iteration, the future inputs variables are:

$$u_{k_i}, u_{k_i+1}, \ldots, u_{k_i+N_p-1} \tag{2.29}$$

with N_p is the prediction horizon representing the length of the optimization window. For a given q_{k_i}, the future state variables are predicted using the state space representation as follows:

$$\begin{aligned} q_{k_i+1} &= Aq_{k_i} + Bu_{k_i} \\ q_{k_i+2} &= A^2 q_{k_i} + ABu_{k_i} + Bu_{k_i+1} \\ &\text{M} \\ q_{k_i+N_p|k_i} &= A^{N_p} q_{k_i|k_i} + A^{N_p-1}Bu_{k_i} + A^{N_p-2}Bu_{k_i+1} + \ldots + Bu_{k_i+N_p-1} \end{aligned} \tag{2.30}$$

Using the predicted state variable, we can find the predicted output as:

$$\begin{aligned} z_{k_i+1|k_i} &= CAq_{k_i} + CBu_{k_i} \\ z_{k_i+2|k_i} &= CA^2 q_{k_i} + CABu_{k_i} + CBu_{k_i+1} \\ &\text{M} \\ z_{k_i+N_p|k_i} &= CA^{N_p} q_{k_i} + CA^{N_p-1}Bu_{k_i} + CA^{N_p-2}Bu_{k_i+1} + \ldots + CBu_{k_i+N_p-1} \end{aligned} \tag{2.31}$$

By defining the future control variables vector *U* and the predicted outputs vectors *Z* as follows:

$$U = \begin{pmatrix} u_{k_i}^T & u_{k_i+1}^T & \ldots & u_{k_i+N_p-1}^T \end{pmatrix}^T \tag{2.32}$$

$$Z = \begin{pmatrix} z_{k_i+1|k_i}^T & z_{k_i+2|k_i}^T & \ldots & z_{k_i+N_p|k_i}^T \end{pmatrix}^T \tag{2.33}$$

Equation (2.31) can be written in matrix form as follows:

$$Z = Fq_{k_i} + \Phi U \tag{2.34}$$

where

$$F = \begin{pmatrix} CA \\ CA^2 \\ CA^3 \\ . \\ . \\ . \\ CA^{N_p} \end{pmatrix} \quad \text{and} \quad \Phi = \begin{pmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ . & & & & \\ . & \text{M} & \text{M} & \dots & \text{M} \\ . & & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CB \end{pmatrix} \tag{2.35}$$

- Cost function

The cost function, often referred to as the objective function, is a mathematical function that depends on both the control inputs and the output variables. This function encapsulates the control objectives and aims to quantify the performance of the control system. In the scope of this study, the cost function is selected to be a quadratic function. It serves the purpose of penalizing the output error and the magnitude of the control inputs. The quadratic cost function, used in this context, is defined as follows:

$$J_k = \sum_{k=1}^{N_p} \left(q_{dk} - z_k\right)^T Q \left(q_{dk} - z_k\right) + \sum_{k=0}^{N_p} u_k^T R u_k \tag{2.36}$$

where q_{dk} are the desired set points, and $Q(6\times 6)$ and $R(3\times 3)$ are known symmetric positive definite matrices also called penalization matrices. This cost function, through its quadratic form, makes it possible to evaluate the performance of the controller by assessing the output error and control effort. It plays a fundamental role in the MPC framework by guiding the controller's decisions to optimize control performance based on the specified control objectives. Using (2.32) and (2.33), the cost function (2.36) can be written in compact form as:

$$J_k = \left(Q_d - Z\right)^T \bar{Q} \left(Q_d - Z\right) + U^T \bar{R} U \tag{2.37}$$

where $Q_d(N_p \times 1)$ is a vector containing the future desired trajectory supposed to be known, and $\bar{Q}(6N_p \times 6N_P)$ and $\bar{R}(3N_p \times 3N_p)$ are block diagonal matrices of Q and R, respectively. Substituting (2.34) in (2.37) we obtain:

$$J_k = \left(Q_d - Fq_{k_i}\right)^T \bar{Q}\left(Q_d - Fq_{k_i}\right) - 2\Phi^T U^T \bar{Q}\left(Q_d - Fq_{k_i}\right) + U^T\left(\Phi^T \bar{Q}\Phi + \bar{R}\right)U \qquad (2.38)$$

and differentiating (2.38) with respect to *U*:

$$\frac{\partial J_k}{\partial U} = -2\Phi^T \bar{Q}(Q_d - Fq_{k_i})^T + 2(\Phi^T \bar{Q}\Phi + \bar{R})U \qquad (2.39)$$

The necessary condition to minimize J_k is:

$$\frac{\partial J_k}{\partial U} = 0 \qquad (2.40)$$

which leads the optimal controls as follows:

$$U = \left(\Phi^T \bar{Q}\Phi + \bar{R}\right)^{-1} \Phi^T \bar{Q}\left(Q_d - Fq_{k_i}\right) \qquad (2.41)$$

from which we apply the first three (3) elements to the system and proceed to the next iteration.

2.3.2 Nonlinear model predictive control

While linear MPC is a powerful and versatile control strategy, it encounters limitations when applied to real-world systems characterized by inherent nonlinearity. Attempting to linearize such systems often leads to inadequate control performance. To address this challenge, a wide array of Nonlinear Model Predictive Control (NMPC) algorithms has been developed. NMPC has emerged as a valuable approach, capable of effectively managing inherently nonlinear systems, and accommodating nonlinear constraints. In this section, we provide the preliminary

setup of NMPC algorithm for OMR. In this context, we utilize the nonlinear kinematic model of the OMR from (2.9):

$$q_{k+1} = f(q_k, u_k) \tag{2.42}$$

Define the tracking error $e_k = q_{dk} - q_k$. The objective function used in this section is slightly modified compared to the one used in the previous section. It now includes the terminal cost which can be used later for stability analyses, and it's given as follows:

$$J_k = \phi(e_{N_p}) + \sum_{k=0}^{N_p-1} \frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) \tag{2.43}$$

where $\phi(e_N) = \frac{1}{2} e_N^T Q_0 e_N$ is the terminal cost, and $Q_0\,(6\times 6)$ is the penalization matrix of the terminal cost. The unconstrained NMPC control problem of the robot motion is written as:

$$\begin{aligned} \min \quad & J_k = \phi(e_{N_p}) + \sum_{k=0}^{N_p-1} \frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) \\ \text{s.t.} \quad & q_{k+1} = f(q_k, u_k) \end{aligned} \tag{2.44}$$

Solving (2.44) determines the control inputs ensuring that the state errors are decreasing along the control horizon. To incorporate the system's equations into the objective function Lagrange multipliers vectors $\lambda_k\ (6\times 1)$ with $(k = 1, \mathrm{K}, N)$ are introduced in (2.44) and the NMPC control problem is transformed to:

$$\min\ J = \phi(e_{N_p}) + \sum_{k=0}^{N_p-1} \left(\frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) + \lambda_{k+1}^T (f_k - q_{k+1}) \right) \tag{2.45}$$

where $f_k \equiv f(q_k, u_k)$. Define the Hamiltonian function of problem (2.45) as:

$$H_k = \frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) + \lambda_{k+1}^T f_k \tag{2.46}$$

Substituting (2.46) into the cost function (2.45) yields:

$$J = \phi(e_{N_p}) + H_0 - \lambda_{N_p}^T q_{N_p} + \sum_{k=1}^{N_p-1}\left(H_k - \lambda_k^T q_k\right) \tag{2.47}$$

To minimize *J*, we differentiate (2.47) as:

$$\begin{aligned} dJ = &\left(\frac{\partial\phi(e_{N_p})}{\partial q_{N_p}} - \lambda_{N_p}^T\right) dq_{N_p} + \frac{\partial H_0}{\partial q_0} dq_0 \\ &+ \frac{\partial H_0}{\partial u_0} du_0 + \sum_{k=1}^{N_p-1}\left(\frac{\partial H_k}{\partial q_k} - \lambda_k^T\right) dq_k + \frac{\partial H_k}{\partial u_k} du_k \end{aligned} \tag{2.48}$$

In equation (2.45), the Lagrange multipliers vectors are multiplying zeros, therefore they can be chosen arbitrarily. To simplify (2.48), we chose:

$$\begin{aligned} \lambda_{N_p}^T &= \frac{\partial\phi(e_{N_p})}{\partial q_{N_p}} = e_{N_p}^T Q_0 \frac{\partial e_{N_p}}{\partial q_{N_p}} \\ \lambda_k^T &= \frac{\partial H_k}{\partial q_k} = e_k^T Q_k \frac{\partial e_k}{\partial q_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial q_k} \end{aligned} \tag{2.49}$$

With these choices, (2.48) becomes:

$$dJ_k = \sum_{k=0}^{N_p-1} \frac{\partial H_k}{\partial u_k} du_k \tag{2.50}$$

with

$$\frac{\partial H_k}{\partial u_k} = u_k^T R + \lambda_{k+1}^T \frac{\partial f_k}{\partial u_k} \tag{2.51}$$

Here we relied on the fact that q_0, the state vector at the current sample instant of the controller, remains constant during the optimization process, which means that $dq_0 = 0$. Equation (2.50) illustrates that the minimization of the Hamiltonian automatically results in the minimization of the objective function. To solve equations (2.42), (2.49) and (2.51), an iterative approach is necessary. However, this iterative nature of the solution process can result in a significant computational burden. Therefore, the selection of an appropriate iterative approach becomes critical to minimize computational complexity, rendering NMPC more practical and feasible for real-world applications.

2.4 Conclusion

This chapter lays the foundational framework for the thesis. It explores the kinematic modeling for OMRs, a crucial step for controlling their movement. Various kinematic representations of OMRs were discussed, including their discretization and linear approximations. The development of tracking error kinematics is also covered. Furthermore, the chapter introduces the conventional frameworks for both linear and nonlinear MPC approaches. These foundational models and frameworks set the stage for the subsequent design and implementation of advanced MPC strategies.

3. ENHANCED MPC FOR OMNIDIRECTIONAL ROBOT MOTION TRACKING USING LAGUERRE FUNCTIONS AND NON-ITERATIVE LINEARIZATION

3.1 Introduction

This chapter proposes an enhanced MPC algorithm, based on Laguerre functions (LMPC), for trajectory tracking of OMR. To ensure a good tracking performance and reduce the linearization's errors, this controller deals with nonlinearities by noniteratively linearizing the system along the predicted trajectory, which, to the best of our knowledge, has never been done before. The duality between optimal control and stochastic filtering is used to compute the linearization points, which allows the linearization of the system without dependence on the to be computed control variables. Contrary to existing approaches, the duality is used only for the prediction, then it is combined with an optimizer to compute the optimal solution. This will enhance the prediction process, but also increase the computation cost. To compensate for this increase, Laguerre functions will be used to parametrize the control variables, which will reduce the number of optimization variables and consequently the computational burden. This makes it suitable for real-time implementation allowing the robot to make fast decisions and swiftly adapt to sudden changes in complex environments. The existing Laguerre parametrization method is further developed to consider the change of the linear system at each prediction instant. The performance of the proposed algorithm is evaluated by simulation and by experiment on the Robotino-Festo OMR and a comparative study of accuracy and computational efficiency is carried out with the traditional MPC and NMPC. In the following, first the kinematic model of the OMR is derived in Section 2.2, next the classical MPC setup is presented in Section 2.3, then in Section 2.4 the duality principal is described. The Laguerre functions are introduced in the MPC setup in Section 2.5, and finally a comparative study with MPC and NMPC is given in Section 2.6. The content of this chapter has been published in the IEEE Access journal [77].

3.2 Model of the omnidirectional mobile robot

For this study, the three-wheels OMR is considered., and the model to be used in the discrete model derived in (2.9) written in the following form:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{x,k+1} \\ v_{y,k+1} \\ \omega_{k+1} \end{pmatrix} = \begin{pmatrix} x_k + v_{xk}T\cos\theta_k - v_{yk}T\sin\theta_k \\ y_k + v_{xk}T\sin\theta_k + v_{yk}T\cos\theta_k \\ \theta_k + \omega_k T \\ v_{xk} \\ v_{yk} \\ \omega_k \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{pmatrix} \begin{pmatrix} a_{xk} \\ a_{yk} \\ a_{\theta k} \end{pmatrix} \quad (3.1)$$

which in compact form becomes:

$$\begin{aligned} q_{k+1} &= f(q_k) + Bu_k \\ z_k &= Cq_k \end{aligned} \quad (3.2)$$

with T is the simulation step, $f(q_k)$ is a nonlinear function of the state, B is the input matrix, z_k is the output vector which contains the position, orientation, and velocities of the OMR, and C is the output matrix. The proposed controller is based on linearization, thus the transition matrix to the linearized system, i.e. the Jacobian matrix, is needed and can be computed using (2.10).

3.3 Constraints

The key advantage of MPC is the capability to handle inequality constraints explicitly, and the OMR exhibits numerous physical and operational constraints that need to be satisfied by the control algorithm. First, there are limits on the acceleration of the OMR, which in this case represent the control variables and can be expressed as follows:

$$u_{\min} \le u_k \le u_{\max} \quad (3.3)$$

where $u_{\min}$ and $u_{\max}$ are vectors of the same size as u_k that contain the lower and upper acceleration limits, respectively. Furthermore, when tracking a reference trajectory, the robot velocities must not exceed the velocity constraints. In this study, v_x and v_y have the same maximum value denoted $v_{\max}$ and the maximum rotational speed is denoted $\omega_{\max}$, then the speeds constraints can be written as:

$$\begin{bmatrix} -v_{\max} \\ -v_{\max} \\ -\omega_{\max} \end{bmatrix} \leq \begin{bmatrix} v_{xk} \\ v_{yk} \\ \omega_k \end{bmatrix} \leq \begin{bmatrix} v_{\max} \\ v_{\max} \\ \omega_{\max} \end{bmatrix} \tag{3.4}$$

Finally, using (3.2), the operational constraints of the OMR can be written together as follows:

$$\begin{aligned} -u_k &\leq -u_{\min} \\ -u_k &\leq \frac{1}{T}\left(\begin{bmatrix} v_{\max} \\ v_{\max} \\ \omega_{\max} \end{bmatrix} + \begin{bmatrix} v_{xk} \\ v_{yk} \\ \omega_k \end{bmatrix}\right) \\ u_k &\leq \frac{1}{T}\left(\begin{bmatrix} v_{\max} \\ v_{\max} \\ \omega_{\max} \end{bmatrix} - \begin{bmatrix} v_{xk} \\ v_{yk} \\ \omega_k \end{bmatrix}\right) \\ u_k &\leq u_{\max} \end{aligned} \tag{3.5}$$

3.4 Duality principal and non-iterative linearization

Since nonlinear optimal control problems are hard to solve and computationally demanding, linearization is often used. Choosing the linearization points is the main problem of linearization approaches. Using the optimal trajectory as linearization points would be the best solution but since they depend on the control to be computed and the control depends on them, iterative methods are required such as NMPC and iterative linear quadratic regulator (iLQR) [78]. To avoid applying iterative algorithms, we are going to use the duality between stochastic filtering and optimal

control to achieve non-iterative linearization by approximating the future optimal trajectory. This duality principle was first presented by Kalman [79], where it showed that the optimal control and the Kalman filtering are dual to each other, and therefore the solution for the estimation problem can be used to solve the optimal problem and vice versa.

To use such duality, we first consider the stochastic dynamics for the control problem as:

$$\begin{aligned} q_{k+1} &= A_k q_k + B u_k + w_k \\ z_k &= C q_k + \sigma_k \end{aligned} \tag{3.6}$$

where w_k and σ_k are fictitious Gaussian noise with covariances V_k and W_k, respectively. The cost function to be minimized at each simulation step is considered quadratic similar to (2.36) of the form:

$$J_k = \sum_{k=1}^{N_p} \left(q_{dk} - z_k\right)^T Q \left(q_{dk} - z_k\right) + \sum_{k=0}^{N_p} u_k^T R u_k \tag{3.7}$$

The dual estimation problem is defined as estimation of q_{k+1} knowing q_k and the whole observation sequence which is considered to be the reference trajectory q_{dk} [41], therefore the stochastic dynamics for the dual estimation problem is considered as:

$$\begin{aligned} \hat{q}_{k+1} &= f(\hat{q}_k) + w_k \\ q_{dk} &= C\ \hat{q}_k + \sigma_k \end{aligned} \tag{3.8}$$

where the “hat” represents estimated values.

The duality between the optimal control problem (3.6) and (3.7), and the estimation dynamic (3.8) is established by choosing $V_k = BR^{-1}B^{-1}$ and $W_k = Q^{-1}$ [80]. The computation of the optimal linearization points can be done using the following Kalman filter equations:

$$
\begin{aligned}
K_k &= P_k C^T (C P_k C^T + Q^{-1})^{-1} \\
P_{k+1} &= A_k (I - K_k C) P_k A_k^T + B R^{-1} B^T \\
\hat{q}_{k+1} &= \hat{q}_k + K_k (q_{dk} - C\hat{q}_k)
\end{aligned}
\tag{3.9}
$$

with K_k is the Kalman gain matrix, P_k is the estimation error covariance matrix [41, 80].

3.5 MPC with Laguerre functions

Although linearizing the system at each prediction step will give more accurate approximation, it is more demanding computationally. Therefore, we introduce the Laguerre functions in the problem formulation to solve the open-loop optimization which will help reducing the computational burden.

3.5.1 Introduction to Laguerre functions

To reduce the computational complexity of the standard MPC, we approximate the future control trajectory by combining a set of orthonormal functions (Laguerre Functions) linearly with few coefficients, which helps to cover the entire control horizon without the need for massive optimization parameters [42]. The Laguerre orthonormal sequence is described by the following z-transforms:

$$
\begin{aligned}
\Gamma_1(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}} \\
\Gamma_2(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}} \frac{z^{-1}-a}{1-az^{-1}} \\
&\vdots \\
\Gamma_N(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}} \left(\frac{z^{-1}-a}{1-az^{-1}} \right)^{N-1}
\end{aligned}
\tag{3.10}
$$

where *a* is the scaling factor of the Laguerre sequence, and $0 \le a < 1$ for the stability of the sequence [47]. Let $l_i(k)$ be the inverse z-transform of $\Gamma_i(z,a)$, then the set of discrete-time Laguerre functions can be written in vector form as:

$$L(k) = \begin{bmatrix} l_1(k) & l_2(k) & \cdots & l_N(k) \end{bmatrix}^T \tag{3.11}$$

Taking advantage of the sequence realization

$$\begin{aligned} \Gamma_1(z) &= \left(\sqrt{1-a^2}\right)/\left(1-az^{-1}\right) \\ \Gamma_k(z) &= \Gamma_{k-1}(z)\frac{z^{-1}-a}{1-az^{-1}} \quad k = 2,3,\ldots N \end{aligned} \tag{3.12}$$

We can describe the sequence by the following state space representation:

$$L(k+1) = A_l L(k) \tag{3.13}$$

with A_l ($N \times N$) and the initial condition $L(0)$ given by:

$$A_l = \begin{bmatrix} a & 0 & 0 & \cdots & 0 \\ \beta & a & 0 & \cdots & 0 \\ -a\beta & \beta & a & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ (-1)^{N-2}a^{N-2}\beta & \cdots & \cdots & \beta & a \end{bmatrix} \tag{3.14}$$

$$L(0) = \sqrt{\beta}\begin{bmatrix} 1 & -a & a^2 & -a^3 & \ldots & (-1)^{N-1}a^{N-1} \end{bmatrix}^T \tag{3.15}$$

where $\beta = 1-a^2$. The orthonormality of Laguerre functions can be expressed in the time domain by:

$$\begin{aligned} \sum_{k=0}^{\infty} l_i(k)l_j(k) &= 0 \ \ for \ \ i \neq j \\ \sum_{k=0}^{\infty} l_i(k)l_j(k) &= 1 \ \ for \ \ i = j \end{aligned} \tag{3.16}$$

Finally, this set of Laguerre functions can be used to capture the response $H(k)$ of an arbitrary system by:

$$H(k) = c_1 l_1(k) + c_2 l_2(k) + \mathrm{K} + c_N l_N(k) \tag{3.17}$$

where $c_1, c_2, \ldots, c_N$ are the coefficients to be determined using the system data, and N is the number of terms used to capture the response [48].

3.5.2 Laguerre-based MPC

Since the OMR has three motors, i.e., three control inputs, let the matrix B be partitioned into:

$$B = \begin{bmatrix} B_1 & B_2 & B_3 \end{bmatrix} \tag{3.18}$$

and define:

$$A_m = \prod_{i=m}^{1} A_{k,i} \tag{3.19}$$

where $A_{k,i}$ is the transition matrix computed at the ith future instant, and m is the current prediction instant. Here, the linearization at every prediction sample is considered, and $A_{k,i}$ is computed using the duality principle. Using (2.34), each control variable can be approximated at an arbitrary future instant with:

$$u_i(k+m) = L_i(m)^T \eta_i = \sum_{j=1}^{N_i} c_j^i(k) l_j^i(m) \tag{3.20}$$

where k is the initial time of the moving horizon, m is the future instant where $k \le m \le k + N_p - 1, i = 1,2,3$ implies the ith control variable, c_j^i are the coefficients, which are functions of the initial time of the moving horizon, N_i is the number of parameters used to capture the ith control variable, η_i is a vector containing the coefficients, $\eta_i = \begin{bmatrix} c_1^i & \ldots & c_N^i \end{bmatrix}^T$, and N_p is length of the prediction horizon. By

using the system (2.7) and (2.37), the prediction of the future state variables can be written as:

$$\begin{aligned} q(k+m) &= A_m q(k) \\ &+\left[B_1 L_1(m-1)^T \quad B_2 L_2(m-1)^T \quad B_3 L_3(m-1)^T \right]\eta \\ &+\sum_{j=0}^{m-2} A_{m-j-1}\left[B_1 L_1(j)^T \quad B_2 L_2(j)^T \quad B_3 L_3(j)^T \right]\eta \\ &= A_m q(k) + \phi(m)^T \eta \end{aligned} \tag{3.21}$$

with $\eta = \left[\eta_1^T \quad \eta_2^T \quad \eta_3^T \right]^T$ and

$$\begin{aligned} \phi(m)^T &= \left[B_1 L_1(m-1)^T \quad B_2 L_2(m-1)^T \quad B_3 L_3(m-1)^T \right] \\ &+\sum_{j=0}^{m-2} A_{m-j-1}\left[B_1 L_1(j)^T \quad B_2 L_2(j)^T \quad B_3 L_3(j)^T \right] \end{aligned} \tag{3.22}$$

Here we have taken into consideration that the matrix A_k is not constant during the prediction process, it is changing at each future instant. With a sufficiently large prediction horizon, the orthonormal property (2.33) becomes:

$$\begin{aligned} &\sum_{k=0}^{N_p} l_i(k) l_j(k) = 0 \ \ for \ \ i \neq j \\ &\sum_{k=0}^{N_p} l_i(k) l_j(k) = 1 \ \ for \ \ i = j \end{aligned} \tag{3.23}$$

And using (2.37) and (2.40), the sum of the future control inputs can be computed by:

$$\sum_{m=0}^{N_p} u\left(k+m\right)^T R_k u(k+m) = \eta^T R_L \eta \tag{3.24}$$

where R_L is a block-diagonal matrix where each block contains one of the elements of R on its diagonal. Define $Q_L = C^T Q C,$ putting (2.38) and (2.41) in the cost function (2.15) will lead to the following form:

$$\begin{aligned} J &= \eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q_L \phi(m)^T + R_L \right) \eta \\ &+ 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q_L A_m \right) q(k) - 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) C^T Q q_d(k+m) \right) \\ &+ \sum_{m=1}^{N_p} \left(q_d(k+m) - CA_m q(k) \right)^T Q \left(q_d(k+m) - CA_m q(k) \right) \\ &= \eta^T \Omega \eta + 2\eta^T (\Psi q(k) - \xi) \\ &+ \sum_{m=1}^{N_p} \left(q_d(k+m) - CA_m q(k) \right)^T Q \left(q_d(k+m) - CA_m q(k) \right) \end{aligned} \tag{3.25}$$

where

$$\begin{aligned} \Omega &= \sum_{m=1}^{N_p} \phi(m) Q_L \phi(m)^T + R_l \\ \Psi &= \sum_{m=1}^{N_p} \phi(m) Q_L A_m \\ \xi &= \sum_{m=1}^{N_p} \phi(m) C^T Q q_d(k+m) \end{aligned} \tag{3.26}$$

By setting the partial derivative (relative to η) of the cost function (2.42) to zero, the optimal solution can be found as:

$$\eta = \Omega^{-1} \left(\xi - \Psi q(k) \right) \tag{3.27}$$

and the first control action can be computed using:

$$u_k = \begin{bmatrix} L_1(0)^T & 0_2{}^T & 0_3{}^T \\ 0_1{}^T & L_2(0)^T & 0_3{}^T \\ 0_1{}^T & 0_2{}^T & L_3(0)^T \end{bmatrix} \eta \tag{3.28}$$

Here $L_i(0)$ is the initial condition for the Laguerre functions of the ith input, and 0_i is a zero vector with the same dimension as $L_i(0)$.

The scaling factor a and the number of terms needed to approximate u_k are closely related. If we set $a = 0$ and the number of terms $N = N_c$ the control horizon, we obtain the traditional MPC approach, and by choosing $0 < a < 1$, we can achieve similar performance with N far less than N_c and reduce the computational cost [48].

3.5.3 Constrained solution using Laguerre functions

The Laguerre functions can also be introduced in the constraints' description, which gives more flexibility for the designer to force the constraints at any specified future instant. The constraints on the control variables at an arbitrary future time m are:

$$U_{\min} \le u(k+m) \le U_{\max} \tag{3.29}$$

with $m = 0,1,\ldots,N_p - 1$, and $U_{\min}, U_{\max}$ are the control bounds from (2.23). This can be written in terms of η as:

$$U_{\min} \le \begin{bmatrix} L_1(m)^T & 0_2{}^T & 0_3{}^T \\ 0_1{}^T & L_2(m)^T & 0_3{}^T \\ 0_1{}^T & 0_2{}^T & L_3(m)^T \end{bmatrix} \eta \le U_{\max} \tag{3.30}$$

The constrained optimal solution is obtained by solving a dual-quadratic problem using the Hildreth method [46, 48]. First the active set of the inequalities constraints

is selected in matrix M_{act}, then the Lagrange multipliers λ_{act} are found using the Hildreth' algorithm, and finally the optimal constrained solution is computed by:

$$\eta = \Omega^{-1}\left(\xi - \Psi q(k) - M_{act}^{T}\lambda_{act}\right) \tag{3.31}$$

3.5.4 The LMPC algorithm

Algorithm 1 is the resulting algorithm named LMPC. It shows the two steps, the linearization, and the control computing. Unlike existing algorithms, which use the duality to linearize the system and approximate the control inputs, the LMPC uses the duality only for linearization. The control inputs are computed by introducing the Laguerre functions and performing online optimization.

Algorithm 1: LMPC

Initialization

$\hat{q}_k = q_k; P_k = 0$

Prediction

for $\mathrm{m} = \mathrm{k}+1,\ldots,\mathrm{k}+\mathrm{N_p}$

Linearization

$\overline{\mathrm{q}} = \mathrm{f}\left(\hat{\mathrm{q}}_{\mathrm{m-1}}\right)$

$$\mathrm{A_{m-1}} = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{q_{m-1}}}\right|_{\mathrm{q_{m-1}}=\hat{\mathrm{q}}_{\mathrm{m-1}}}$$

$$\mathrm{k_m} = \mathrm{P_{m-1}C^T}\left(\mathrm{CP_{m-1}C^T} + \mathrm{Q_k^{-1}}\right)^{-1}$$

$$\mathrm{P_m} = \mathrm{A_{m-1}}\left(\mathrm{I} - \mathrm{K_m C}\right)\mathrm{P_{m-1}A_{m-1}^T} + \mathrm{BR_k^{-1}B^T}$$

$$\hat{\mathrm{q}} = \overline{\mathrm{q}}_{\mathrm{m}} + \mathrm{k_m}\left(\mathrm{q_{dm}} - \mathrm{C}\overline{\mathrm{q}}_{\mathrm{m}}\right)$$

Compute Convolution Sums

$\Omega;\ \Psi;\ \xi;$ *(Eq.2.43)*

end for

Set The Constraints

$\mathrm{M_{act}}$ *and* λ_{act} *using Hildreth Algorithm*

Compute Optimal Coefficients Vector

$$\eta_{\mathrm{k}} = \Omega^{-1}\left(\xi - \Psi \mathrm{q_k} - \mathrm{M_{act}^T}\lambda_{\mathrm{act}}\right)$$

Compute First Optimal control action

$$\mathrm{u_k} = \mathrm{L_{zero}}\eta_{\mathrm{k}}$$

Contrary to existing methods, the parameterization using Laguerre functions takes into consideration the linearization at each future instant. In the case of constrained control, the LMPC uses the Hildreth algorithm to identify the active constraints and compute the Lagrange multipliers. All the state variables are considered available for measurement. Therefore, the LMPC is a deterministic state feedback controller.

3.6 Comparative studies and analyzes

In this section, the performance of the proposed LMPC algorithm will be analyzed. To show the outstanding performance, the traditional linear MPC and NMPC approaches are introduced for comparisons.

3.6.1 Simulations results

The simulations are carried out using the MATLAB/Simulink software. The aim is to drive the OMR to track a given trajectory by minimizing the cost function (2.15). All the strategies compared will minimize the same cost function. The following approaches will be compared:

1) LMPC: this algorithm linearizes the system at each future prediction step using the duality principle, and then solves the optimality using Laguerre functions. The number of terms and the scaling factor will be the same for all input variables.

2) MPC: this method solves the optimization problem using a linearized model and standard optimization algorithm. Implementation is based on [48].

3) NMPC: this algorithm solves the open-loop optimization using the nonlinear model. Implementation is based on the optimized algorithm in [37], where the active-set method is used for the minimization. The convergence threshold is set to 10^{-8} and the maximum number of iterations is 10^{3}.

To ensure a fair comparison, some unifying conditions need to be set:

1) The prediction horizon N_p is the same for the three strategies and it is set to $N_p = 20$, which ensure the convergence and practical feasibility.

2) For both MPC and NMPC, control horizon N_c is the same; however, the term N_c does not appear in the LMPC algorithm, since it has been replaced by the number of parameters N and scaling factor a. In [47], it has been shown that for a small N, N_c and a are related by $a \approx e^{-5/N_c}$.

3) The weighting matrices for the strategies compared are set to:

$$R_k = diag(0.01, 0.01, 0.01)$$

$$Q_k = diag(25, 25, 25, 0.1, 0.1, 0.1)$$

These tuning parameters were chosen through trial and error; therefore, their optimality cannot be guaranteed. Nonetheless, they have demonstrated good performance in this study.

To thoroughly evaluate the tracking performance, we use a desired trajectory given by reference speed components $v_{xd} = 0.5\text{ms}^{-1}$ and $v_{yd} = 0.5\text{ms}^{-1}$, reference positions $x_d = v_{xd}t$; $y_d = v_{yd}t$ m, orientation $\theta_d = 0$ rad, and angular velocity $\omega_d = 0$ rads^{-1}. The trajectory is supposed to be known along the prediction horizon, and all the state variables are considered available for measurement.

We consider a set of $H = 5$ simulations, and a random uniformly distributed initial state:

$$q_0^h = Rand([-1,1],[-1,1],[-\pi/6, \pi/6], 0, 0, 0)^T \qquad (3.32)$$

For each simulation h, and iteration k, the cost achieved for each approach is denoted $J_r(h,k), \quad r=1,\ldots,3.$ The initial reference minimum of the cost function is chosen to be 5. On each iteration, the method with the best cost, lower than 5, is taken as reference $J_{best}(h,k)$ to be compared to the other methods [41], i.e, $J_{best}=\min_{1<r<3}(J_r(h,k),5).$ The average cost ratio (ACR) over all simulations is computed by:

$$ACR_r=(1/H)\times\sum_{h=1}^{H}\frac{J_r(h,k)}{J_{best}(h,k)}, \quad r=1,\ldots,3. \tag{3.33}$$

Furthermore, to compare the tracking performance of different strategies, an additional index is used, which quantifies the tracking quadratic error, and it is given by:

$$M_i=1/H\sum_{h=1}^{H}\sqrt{\frac{\sum_{j=1}^{T_{sim}/T}\left(y_{ref}(j)-y_{sys}(j)\right)^2}{T_{sim}/T}}; \quad i\in\{x,y,\theta\} \tag{3.34}$$

with $\Delta t=0.07$ s. In this experiment, the control horizon for MPC and NMPC is set to $N_c=5$, and for the parameters of LMPC are chosen $a=0.5,$ and $N=3.$

Figure 9 shows the ACR of each method for the first 10 iterations in the logarithmic scale. It can be clearly seen that after the first iteration, the initial cost realized by the LMPC is significantly lower than the ones realized by MPC and NMPC, and after 10 iterations, LMPC yields the best performance. Both the LMPC and NMPC converge after the third iteration with LMPC achieving a lower final cost, while MPC took much longer to converge.

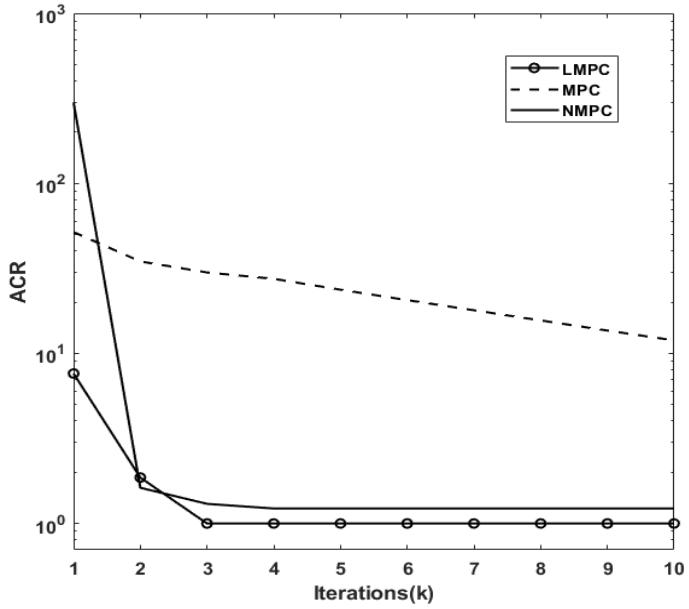


Figure 9
Average cost ratio (ACR) per iteration
Source: [77]

The numeric values of Table 1 represent the number of variables involved in the optimization process of each strategy, along with the average time for the first ten iterations. For MPC and NMPC, the number optimization variables are computed by multiplying the number of control inputs by the length of the control horizon N_c, whereas for LMPC, it is equal to the number of parameters used to parametrize the manipulated variables. Only 9 variables are involved in the optimization process of the LMPC, while 15 parameters are involved in both MPC and NMPC. This gives the LMPC a great computational advantage (100 times faster) over the NMPC while maintaining good performance. Even with linearization performed at each prediction instant, LMPC still managed to keep up with the MPC with only 1ms difference.

Table 1
Number of control parameters and time per iteration

	LMPC	MPC	NMPC
Number of optimization variables	9	15	15
Time per iteration (s)	0.002	0.001	0.266

Table 2 shows the mean-quadratic tracking errors for the three methods. It can be appreciated that LMPC significantly reduces the tracking error compared to MPC with almost the same computational demand, and it is not that far behind NMPC. In fact, in the orientation tracking, LMPC showed better performance than the other two methods. NMPC showed slightly lower M_x and M_y compared to LMPC, at the cost of higher computation time. This is due to the NMPC iterative aspect. As a result, LMPC can be considered a computationally effective method to obtain optimal performance in practice.

Table 2
Mean quadratic tracking errors for different strategies

	LMPC	MPC	NMPC
M_x	0.0083	0.0663	0.0058
M_y	0.0113	0.0631	0.0012
M_ϕ	0.0035	0.0065	0.0217

3.6.2 Experimental results

In this section, the three algorithms are tested on the Robotino-Festo omnidirectional robot. The whole system is controlled by an embedded PC to COM Express specifications with Intel i5, 2.4 GHz dual core, 8 GB RAM and 23 GB SSD. For the motor control, a 32-bit microcontroller is used. It generates the PWM signals for actuating the DC motors using a PID controller. The microcontroller is also used to correct the sensors data. A planetary gear unit with transition ratio 32:1 is used between the drive shafts and omni-wheels. The robot has a maximum translational and rotational speeds of 2m/s and 2rad/s respectively. The algorithms are implemented using the Robotino MATLAB-Simulink toolbox. The robot accepts translational and rotational velocities as inputs, which are expected to be updated every 70ms. Since the compared approaches have accelerations as manipulated variables, integrators are included in the algorithms. The aim is to drive Robotino to follow the eight-shaped trajectory described by the following equations:

$$
\begin{aligned}
x_d &= -\sin((2\pi / T_p)t) \\
y_d &= 0.5\sin(2(2\pi / T_p)t) \\
v_{xd} &= \dot{x}_d \quad ; \quad v_{yd} = \dot{y}_d
\end{aligned} \tag{3.35}
$$

where $T_p = 25\,\text{s}$ is the trajectory period, $\theta_d = 0$ and $\omega_d = 0$. The control horizon is chosen 10 for MPC and 5 for NMPC, as for the LMPC, $N = 3$ and $a = 0.8$. The weighting matrices are chosen as:

$$
\begin{aligned}
R &= diag(15,15,15) \\
Q &= diag(80,80,80,0.1,0.1,0.1)
\end{aligned}
$$

and initial position is given as:

$$
q_0 = [-0.5,\ 0,\ \pi/6,\ 0,0,0].
$$

Figure 10 shows the performance of each method when tracking the eight-shaped trajectory. In Figure 10(a) the advantage of the LMPC performance over MPC is well illustrated.

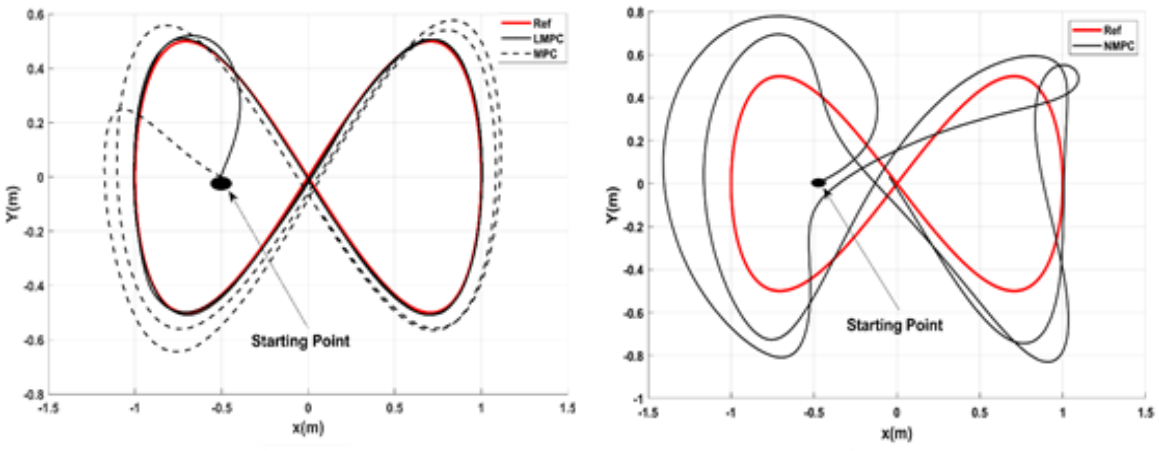


Figure 10
Tracking the eight shaped trajectory in real-time: (a) using LMPC and MPC, and (b) using NMPC.
Source: [77]

The NMPC needs longer than the update time of the microcontroller (70ms) to compute the next inputs values, which makes it unsuitable for practice, as shown in Figure 10(b).

Figure 11 shows the state variables of the system over time when using the LMPC: x and y positions, orientation θ, translational velocities v_x and v_y, and angular velocity ω. The computed input accelerations before integration (black lines) and the real robot acceleration (red lines) estimated by differentiating and filtering the odometry data are shown in Figure 12.

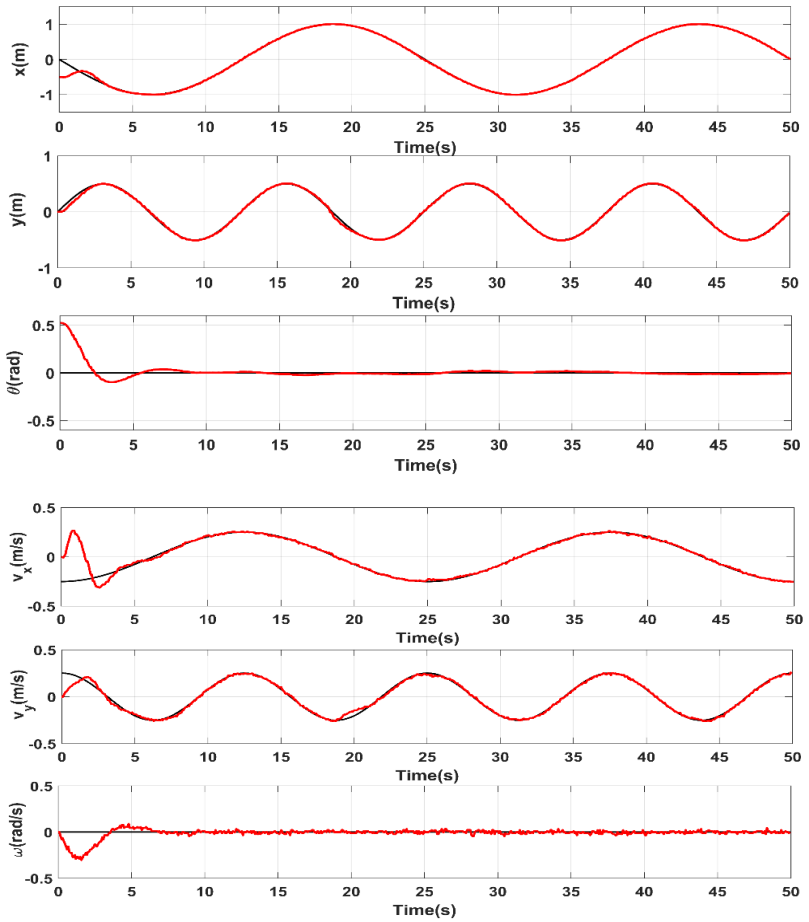


Figure 11
Reference States (black solid line) and real positions (red solid line) using LMPC
Source: [77]

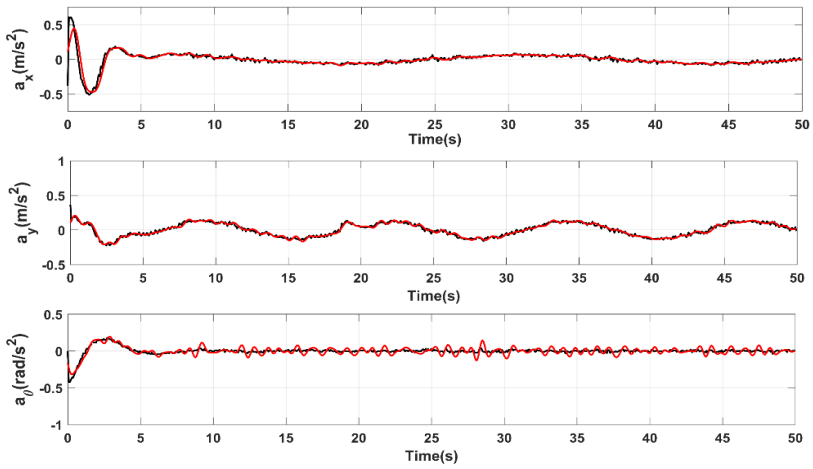


Figure 12
Applied control inputs (black solid line) and estimated accelerations (red solid line) using LMPC
Source: [77]

3.7 Conclusion

In this chapter, an enhanced MPC algorithm based on Laguerre functions, LMPC, for trajectory tracking of an OMR has been presented. Non-iterative linearization was performed using the duality between optimal control and stochastic filtering to approximate the nonlinear system, and the Laguerre functions were used to describe the control variables and reduce the number of optimization variables. The method presented provides a way to ameliorate the prediction process, prevent the accumulation of the linearization's error and improve the tracking performance. The computational time was also reduced significantly allowing the algorithm to make fast accurate decisions. The performance of the proposed algorithm was validated on the trajectory tracking problem of the OMR and compared to the traditional linear MPC algorithm and the NMPC. Experiments show that LMPC can achieve high tracking accuracy, outperforming both MPC and NMPC. Feasibility and suitability for real-time applications were also demonstrated by experiment on Robotino Festo mobile robot.

4. NMPC FOR TRAJECTORY TRACKING OF OMNIDIRECTIONAL ROBOT USING RESILIENT PROPAGATION

4.1 Introduction

While linearization-based methods have demonstrated good performance, they inherently rely on approximating the true nonlinear behavior of systems. Consequently, they exhibit limitations, particularly in terms of stability analyses and the provision of convergence proofs. These limitations arise from the necessity to linearize the system at each time step. Furthermore, linearization-based methods often disregard valuable system information that could be beneficial for control system design and understanding.

To overcome these limitations, NMPC offers the advantage of directly utilizing nonlinear system models in the prediction and optimization processes. This approach enables the possibility of conducting stability analyses and utilizing the full extent of system information, ultimately leading to improved tracking performance.

However, employing nonlinear system models comes with its challenges, particularly in terms of computational complexity. The resulting nonlinear optimization problem is often demanding in terms of computational resources. To address this issue, extensive research has been carried out in this field. Much of this research has focused on enhancing the optimization algorithms used within NMPC to reduce computational costs. Several nonlinear optimization algorithms have been explored to streamline the computational burden, including active-set methods, interior-point techniques, trust-region methods, and algorithms based on neural networks. While these algorithms offer precision, they can still be time-intensive, limiting their application to slower systems or specific cases where computational time is not a critical constraint.

Another promising algorithm from the literature is the resilient propagation algorithm (RPROP) first introduced in [67]. It’s a gradient descent-based algorithm widely used in the optimization of neural networks. Differently to existing gradient-based approaches, the RPROP uses only the sign of the gradient to decide the search’s

direction and to update the step length which is independent of the gradient value. Compared to existing methods, the RPROP shows great computational advantages without compromising the accuracy [72, 81]. Despite its great advantages, the application of RPROP remains very limited in the robotic field. The advantages of RPROP are the following:

1. RPROP is a fast method with high accuracy, which does not rely on the system's properties.
2. RPROP is a first-order method, which makes it easy to comprehend and implement without mathematical or numerical complexities.
3. RPROP depends only on the gradients' signs, and that makes it suitable for applications where the derivatives are estimated or noisy.

In this chapter, and motivated by the potential of RPROP, we introduce an improved NMPC approach that utilizes A Robust Convergent Resilient Propagation (ARCPROP) algorithm to address and solve the trajectory tracking challenge of an OMR. The method is applied to both unconstrained and constrained scenarios. In the latter case, we utilize the external penalty method to effectively manage the constraints. The superiority of our proposed method is demonstrated through a comparative study against the well-known Interior Point (IP) and Active Set (AS) methods. The content of this chapter has been submitted for publication in the IEEE Access journal [82].

4.2 Constrained NMPC setup

4.2.1 Prediction model

For the prediction process of the nonlinear predictive controller, the discrete kinematic model of the OMR form equation (2.9) is used which is given as follows:

$$q_{k+1} = f(q_k, u_k) \tag{4.1}$$

with

$$f(q_k,u_k)\equiv f_k=\begin{pmatrix} x_k+\nu_{xk}T\cos\theta_k-\nu_{yk}T\sin\theta_k \\ y_k+\nu_{xk}T\sin\theta_k+\nu_{yk}T\cos\theta_k \\ \theta_k+\omega_kT \\ \nu_{xk}+a_xT \\ \nu_{yk}+a_yT \\ \omega_k+a_\theta T \end{pmatrix} \tag{4.2}$$

with $q_k=\begin{pmatrix} x_k & y_k & \theta_k & v_{xk} & v_{yk} & \omega_k \end{pmatrix}$ is the state vector, and $(a_x \quad a_y \quad a_\theta)$ is the inputs vector.

4.2.2 Optimization problem

The objective function used in this chapter is similar to (2.43) and given as follows:

$$J_k=\phi(e_{N_p})+\sum_{k=0}^{N_p-1}\frac{1}{2}(e_k^TQe_k+u_k^TRu_k) \tag{4.3}$$

where $\phi(e_{N_p})=\frac{1}{2}e_{N_p}^TQ_0e_{N_p}$ is the terminal cost. Q, Q_0 (both 6×6) and R (3×3) are diagonal weight matrices and $e_k=q_{dk}-q_k$ is the tracking error. In real-time applications, there are always limitations and restrictions that need to be taken into consideration when designing the control algorithm. One of the main advantages of NMPC is the capability to handle control constraints explicitly. When constraints are imposed, the NMPC control problem becomes as follows:

$$\begin{aligned} \min \quad & J=\frac{1}{2}e_{N_p}^TQ_0e_{N_p}+\sum_{k=0}^{N_p-1}\frac{1}{2}(e_k^TQe_k+u_k^TRu_k) \\ \text{s.t.} \quad & q_{k+1}=f(q_k,u_k) \\ & h(q_k,u_k)\le 0 \end{aligned} \tag{4.4}$$

where $h(q_k, u_k)$ are the inequality constraints set as:

$$\begin{cases} -v_{\max} \le v_{xk}; v_{yk} \le v_{\max} \\ -\omega_{\max} \le \omega_k \le \omega_{\max} \\ -a_{t\max} \le a_{xk}; a_{yk} \le a_{t\max} \\ -a_{n\max} \le a_{\theta k} \le a_{n\max} \end{cases} \tag{4.5}$$

Here $v_{\max}$ and $\omega_{\max}$ are the maximum allowable translational and rotational velocities respectively, and $a_{t\max}$ and $a_{n\max}$ are the maximum translational and rotational accelerations respectively. These inequality constraints can be handled using different approaches such as the auxiliary variable method or the external penalty method. In [56], the auxiliary variable method was used to transform the inequality constraints to equality ones by adding a dummy variable which was later considered an additional variable in the optimization problem. However, using a dummy variable for each inequality constraint will increase the dimension of the optimization problem and consequently increase the computational burden. The external penalty method aims to transform the constrained problem to an unconstrained one by adding a penalty term to the objective function which penalizes the violation of constraints. It is a straightforward method proven to be effective [83], and it will be adopted in this work. The additional penalty term is chosen as:

$$\mathrm{M}_i(q_k, u_k) = \begin{cases} 0 & , h_i(q_k, u_k) \le 0 \\ \mu_i \times h_i(q_k, u_k)^2 & , h_i(q_k, u_k) > 0 \end{cases} \tag{4.6}$$

where $\mathrm{M}_i(q_k, u_k)$ and μ_i are the penalty cost and the weight factor of the $i\mathrm{th}$ inequality constraint $h_i(q_k, u_k)$. In this work, $h_i(q_k, u_k)$ is:

$$\begin{cases} h_1(q_k,u_k) = v_{xk}^2 - v_{\max}^2 \\ h_2(q_k,u_k) = v_{yk}^2 - v_{\max}^2 \\ h_3(q_k,u_k) = \omega_k^2 - \omega_{\max}^2 \\ h_4(q_k,u_k) = a_{xk}^2 - a_{t\max}^2 \\ h_5(q_k,u_k) = a_{yk}^2 - a_{t\max}^2 \\ h_6(q_k,u_k) = a_{\theta k}^2 - a_{n\max}^2 \end{cases} \tag{4.7}$$

The obtained unconstrained optimization problem can be written as follows:

$$\begin{aligned} \min \quad & J_c = \frac{1}{2} e_{N_p}^T Q_0 e_{N_p} \\ & + \sum_{k=0}^{N_p-1} \left[\frac{1}{2} (e_k^T Q e_k + u_k^T R u_k) + \sum_{i=1}^{6} \mathrm{M}_i(q_k,u_k) \right] \\ \text{s.t.} \quad & q_{k+1} = f(q_k,u_k) \end{aligned} \tag{4.8}$$

which is subject to the dynamic of the plant. We introduce the Lagrange multipliers λ_k (6×1) with $(k=1,\ldots,N_p)$, which are used to incorporate the system's equations in the cost function as follows:

$$\begin{aligned} \min \quad & J_c = \frac{1}{2} e_{N_p}^T Q_0 e_{N_p} \\ & + \sum_{k=0}^{N_p-1} \left[\frac{1}{2} (e_k^T Q e_k + u_k^T R u_k) + \lambda_{k+1}^T (f_k - q_{k+1}) + \sum_{i=1}^{6} \mathrm{M}_i(q_k,u_k) \right] \end{aligned} \tag{4.9}$$

Since $f_k - q_{k+1} = 0$, the Lagrange multipliers can be chosen arbitrarily to simplify the development. Define the Hamiltonian function of problem (4.9) as:

$$H_k = \frac{1}{2} (e_k^T Q e_k + u_k^T R u_k) + \lambda_{k+1}^T f_k + \sum_{i=1}^{6} \mathrm{M}_i(q_k,u_k) \tag{4.10}$$

and substituting (4.10) in the cost function (4.9), we obtain:

$$
\begin{aligned}
J_{ck} &= \phi(e_{N_p}) + \sum_{k=0}^{N_p-1} \frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) + \lambda_{k+1}^T f_k - \lambda_{k+1}^T q_{k+1} \\
&= \phi(e_{N_p}) + H_0 - \lambda_k^T q_{N_p} + \sum_{k=1}^{N_p-1} H_k - \lambda_k^T q_k
\end{aligned}
\tag{4.11}
$$

The differential of (4.11) is computed as:

$$
\begin{aligned}
dJ_{ck} = &\left(\frac{\partial \phi(e_{N_p})}{\partial q_{N_p}} - \lambda_{N_p}^T \right) dq_{N_p} + \frac{\partial H_0}{\partial q_0} dq_0 + \frac{\partial H_0}{\partial u_0} du_0 \\
&+ \sum_{k=1}^{N_p-1} \left(\frac{\partial H_k}{\partial q_k} - \lambda_k^T \right) dq_k + \frac{\partial H_k}{\partial u_k} du_k
\end{aligned}
\tag{4.12}
$$

To simplify the expression of (4.12), we choose the Lagrange multipliers as follows:

$$
\begin{aligned}
\lambda_{N_p}^T &= \frac{\partial \phi(e_{N_p})}{\partial q_{N_p}} = e_{N_p}^T Q_0 \frac{\partial e_{N_p}}{\partial q_{N_p}} \\
\lambda_k^T &= \frac{\partial H_k}{\partial q_k} = e_k^T Q \frac{\partial e_k}{\partial q_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial q_k} + \sum_{i=1}^{6} \frac{\partial \mathrm{M}_i(q_k, u_k)}{\partial q_k}
\end{aligned}
\tag{4.13}
$$

which leads to:

$$
dJ_k = \sum_{k=0}^{N_p-1} \frac{\partial H_k}{\partial u_k} du_k \tag{4.14}
$$

Here we took into consideration that the initial state remains constant during the optimization process, which means that $dq_0 = 0$. The differential of the cost function is proportional to the partial derivative of the Hamiltonian relative to *u*, which means that minimizing H will lead to minimization the cost function. We can compute:

$$\lambda_{N_p}^T = \frac{\partial \phi(e_{N_p})}{\partial q_{N_P}} = (q_{N_p} - q_{dN_p})^T Q_0$$
$$\lambda_k^T = \frac{\partial H_k}{\partial q_k} = (q_k - q_{dN_p})^T Q + \lambda_{k+1}^T A_k + \sum_{i=1}^{6} \frac{\partial \mathrm{M}_i(q_k, u_k)}{\partial q_k} \qquad (4.15)$$
$$\frac{\partial H_k}{\partial u_k} = u_k^T R + \lambda_{k+1}^T \frac{\partial f_k}{\partial u_k} + \sum_{i=1}^{6} \frac{\partial \mathrm{M}_i(q_k, u_k)}{\partial q_k}$$

where A_k is the transition matrix from(2.10).

4.3 Optimization algorithm

The second component of a predictive controller is the optimization process. In this project, given that NMPC entails online optimization at each sampling instance, the selected optimization algorithm must satisfy two primary criteria: computational efficiency and convergence.

4.3.1 RPROP

Gradient-based methods remain the most widely used in optimization problems, especially back-propagation algorithms, which minimize the objective function and update the optimization's variables using steepest descent as follows:

$$u_{k+1} = u_k - \eta \nabla H(u_k) \qquad (4.16)$$

where *H* is the function to be minimized, u_k is a vector containing the optimization variables, and η is called the learning rates. Choosing the leaning rates that ensure convergence and avoid oscillation is well known to be a difficult task.

To overcome this problem, RPROP uses an adaptative individual step-size for each optimization variable, which helps minimize oscillations and maximize the length of the step-size. Each iteration of the algorithm is composed of two phases: first, the step-sizes are updated to accelerate convergence and improve performance. Each optimization variable u_i has its own step-size Δ_i. If the sign of the gradients remains unchanged for two consecutive iterations, the step-size is increased. Whereas if it

flips sign, the step-size is shortened. These rules are illustrated in the following equations:

$$\Delta_i^k = \begin{cases} \min\left(\eta^+ \Delta_i^{(k-1)}, \Delta_{\max}\right) & \text{if } \dfrac{\partial H(u_{k-1})}{\partial u_{i,k}} \dfrac{\partial H(u_k)}{\partial u_{i,k}} > 0 \\ \max\left(\eta^- \Delta_i^{(k-1)}, \Delta_{\min}\right) & \text{if } \dfrac{\partial H(u_{k-1})}{\partial u_{i,k}} \dfrac{\partial H(u_k)}{\partial u_{i,k}} < 0 \\ \Delta_i^{(k-1)} & \text{otherwise} \end{cases} \tag{4.17}$$

where $0 < \eta^- < 1 < \eta^+$ are the update rates, and $\Delta_{\min}$, $\Delta_{\max}$ are the boundary limits of the step-sizes.

Thus, whenever the partial derivative flips sign, which implies that the last update was large, the RPROP assumes that a local minimum has been jumped over, so it does not update the correspondent optimization variable; instead, it decreases the step-size to enforce returning to the local minimum's region. If the derivative retains its sign, then step-size is increased to accelerate convergence in shallow regions.

After adjusting the step-sizes, the second part of the RPROP algorithm is to update the value of the optimization variables, which is done following the rules below:

$$\begin{cases} \text{if } \dfrac{\partial H(u_{k-1})}{\partial u_{i,k}} \dfrac{\partial H(u_k)}{\partial u_{i,k}} \geq 0 \quad \text{then } \Delta u_{i,k} = -\text{sign}\left(\dfrac{\partial H(u_k)}{\partial u_{i,k}}\right) \Delta_i^k \\ \text{if } \dfrac{\partial H(u_{k-1})}{\partial u_{i,k}} \dfrac{\partial H(u_k)}{\partial u_{i,k}} < 0 \quad \text{then } \Delta u_{i,k} = -\Delta u_{i,k-1};\ \dfrac{\partial H(u_k)}{\partial u_{i,k}} = 0 \end{cases} \tag{4.18}$$

and

$$u_{i,k+1} = u_{i,k} + \Delta u_{i,k} \tag{4.19}$$

This approach proved to be very powerful with high accuracy and less dependence on the system characteristics, especially in cases where the derivatives are estimated and noisy.

4.3.2 ARCPROP

Algorithm 2: ARCPROP

while $iter \le I_{\max}$

if $\max\left(\frac{\partial H_k}{\partial u_k}\right) \le \delta$ **then return** u_k

update the step sizes

if $H_k > H_{k-1} - \delta \min(\Delta_{i,k-1})$ **then**

  if $\max(\Delta_{i,k-1} \le \Delta_{\min})$ **then return** u_{k-1}

  For each $u_{i,k}$ **do**

    backtracking

    $u_{i,k} = u_{i,k-1}$; $\frac{\partial H_k}{\partial u_i} = \frac{\partial H_{k-1}}{\partial u_i}$

    $\Delta_{i,k} = \max\left(\eta^{-}\Delta_{i,k-1}, \Delta_{\min}\right)$

else

  for each $u_{i,k}$ **do**

$$\Delta_{i,k} = \begin{cases} \min\left(\eta^{+}\Delta_{i,k-1}, \Delta_{\max}\right) & \text{if } \frac{\partial H_{k-1}}{\partial u_i}\frac{\partial H_k}{\partial u_i} > 0 \\ \max\left(\eta^{-}\Delta_{i,k-1}, \Delta_{\min}\right) & \text{if } \frac{\partial H_{k-1}}{\partial u_i}\frac{\partial H_k}{\partial u_i} < 0 \\ \Delta_{i,k-1} & \text{otherwise} \end{cases}$$

update the optimization variables

for each $u_{i,k}$ **do**

$u_{i,k+1} = u_{i,k} - \text{sign}\left(\frac{\partial H_k}{\partial u_i}\right).\Delta_{i,k}$

Despite its advantages, it has been proven that the traditional RPROP algorithm does not guarantee convergence as shown in [71], [72]. In [71], it is shown that adjusting the step size for each optimization variable individually does not necessarily result in a decrease in the overall objective function. Inadvertently, it may cause an increase. To address this limitation, a new robust convergent variant ARCPROP was proposed. This new algorithm adds another layer of verification which considers the impact of the updated step sizes on the overall objective function. This approach is illustrated in Algorithm 2.

In this algorithm, the original RPROP approach is combined with an additional step size updating phase that forces each individual update to decrease the objective function by a factor of $\delta \min(\Delta_{i,k-1})$ where δ is the threshold of the optimization. This will ensure the convergence of the algorithm to a local minimum in a finite time. For further details and proof of convergence, refer to [71].

4.4 Simulation Results

In this section, the performance of the proposed method is evaluated. Comparison with other benchmark approaches is done to demonstrate the capabilities of the ARCPROP optimization algorithm to perform online optimization, handle constraints and reduce the computational burden.

4.4.1 Simulation setup

Three methods will be compared:

1) NMPC-PROP: This approach solves the NMPC problem using ARCPROP algorithm.
2) NMPCIP: This method uses the Interior Point (IP) algorithm to solve the optimization problem of NMPC. Its implementation is based on [37].
3) NMPCAS: This algorithm uses the AS methods for the optimization problem. It is implemented using MATLAB “fmincon” function.

The goal is to drive the OMR to follow a given trajectory by minimizing an objective function. Constrained and unconstrained cases will be considered. All methods

minimize the same objective function (4.3). The NMPC parameters are: the length of the prediction horizon $N_P = 6,$ the length of the control horizon $N_c = 3$, the penalization matrices $R = diag(0.01, 0.01, 0.01)$ and $Q = Q_0 = diag(75, 75, 75, 5, 5, 5)$. These penalization matrices are chosen by trial and error so we cannot guarantee their optimality, nonetheless, they showed satisfactory performance in this study. We note that the term control horizon N_c does not appear in the objective functions. It is the number of variables optimized at each iteration which is less or equal to N_p.

For the optimization algorithms, the parameters are chosen as follows: the convergence's threshold $\delta = 10^{-6}$ and maximum number of iterations $I_{\max} = 20$. In addition, for the ARCPROP, we have: $\eta^+ = 1.6,\ \eta^- = 0.5,\ \Delta_{\max} = 15$ and $\Delta_{\min} = 10^{-10}$. The robot is set to track an eight-shaped trajectory described by the following equations:

$$\begin{aligned} x_d &= -2\sin((2\pi / P)t) \\ y_d &= 1\sin(2(2\pi / P)t) \\ v_{xd} &= \dot{x}_d \quad ; \quad v_{yd} = \dot{y}_d \end{aligned} \tag{4.20}$$

where $P = 24$ is the trajectory period. The desired orientation and rotational speed are selected to be zero and the initial position is set to $q_0 = [-0.35,\ -0.2,\ 0,\ 0, 0, 0]$. In the constrained case, the maximum speeds are set to $v_{\max} = 2$ m/s and $\omega_{\max} = 3$ rad/s, and the maximum accelerations to $a_{t\max} = 1$ m/s^2 and $a_{n\max} = 4$ rad/s^2, which reflect the limitations on the real robot. The tracking performance of the compared strategies is evaluated using a performance index which qualifies the quadratic error as follows:

$$M_i = \sqrt{\frac{\sum_{j=1}^{T_{sim}/T} \left(y_{ref}(j) - y_{sys}(j)\right)^2}{T_{sim} / T}}; \quad i \in \{x, y, \theta\} \tag{4.21}$$

where T_{sim} is the simulation time, y_{ref} is the reference value and y_{sys} is the measured value of the position variables.

4.4.2 Tracking analysis and computational resources

The simulation results of the tracking problem are illustrated in the figures and tables below.

Figure 13 shows the performance of the three methods when tracking the eight-shaped trajectory without constraints. It is evident that NMPC-PROP achieved faster convergence compared to the other two methods.

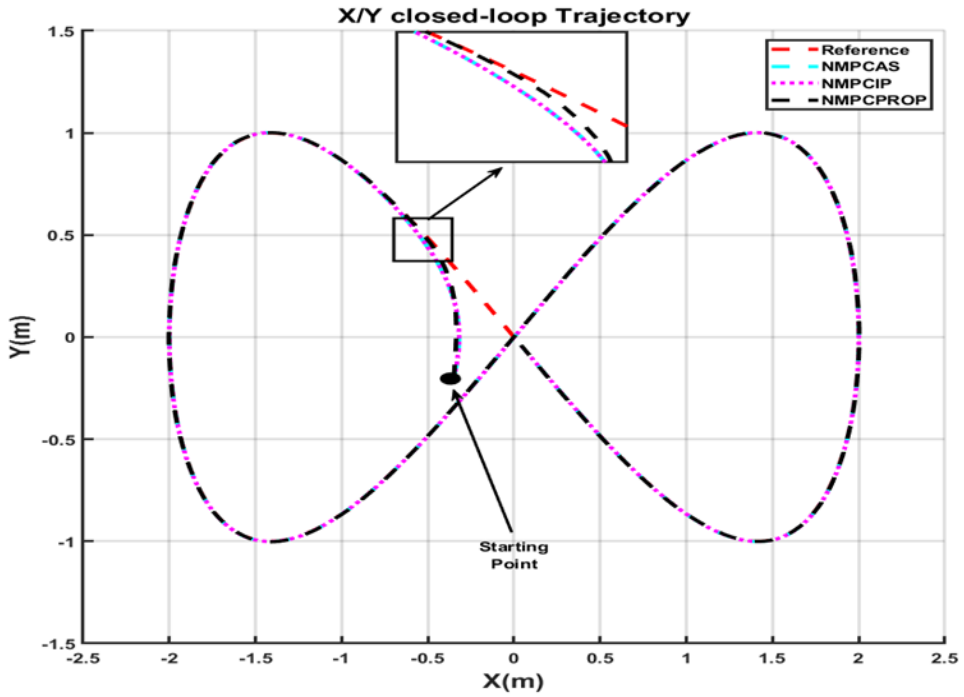


Figure 13
Tracking the eight shaped trajectory: Unconstrained case
Source:[82]

The optimized control signals are shown in Figure 14. It can be appreciated that the initial control effort resulted using NMPC-PROP is notably lower compared to the initial efforts of NMPCIP and NMPCAS.

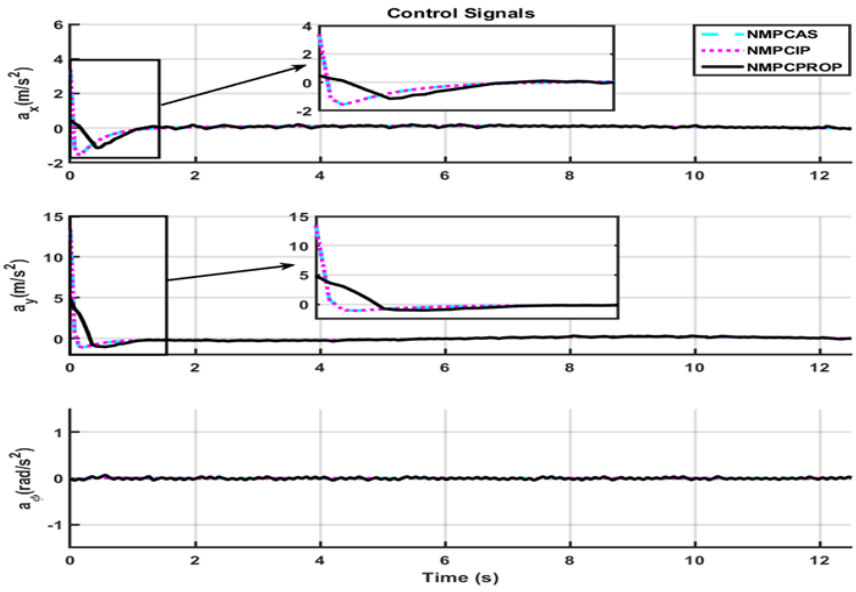


Figure 14
Optimized control signals: Unconstrained case
Source: [82]

Table 3 conveys the average and maximum time necessary to complete one iteration by each of the compared strategies along with the tracking performance measure from (4.21) for the unconstrained case. It is notable that NMPC-PROP has a greater advantage regarding computational cost. It is 12 to 17 times faster than NMPCIP and 7 to 8 times faster than NMPCAS algorithm. This renders it more suitable for real-time applications. Some of NMPCIP and NMPCAS iterations take more than 70ms to be completed which may affect their performance when applied to the real robot that requires updates every 70ms. Despite this substantial reduction in computational burden, NMPC-PROP is attaining a tracking performance remarkably similar to NMPCIP and NMPCAS. The tracking error of NMPC-PROP is only slightly higher by an order of 10^{-3} compared to the other two methods. This discrepancy is justified by the attained computational efficiency.

Table 3
Maximum time per iteration, average time per iteration, time ratios and mean quadratic tracking errors for different strategies: Unconstrained case

	NMPC-PROP	NMPCIP	NMPCAS
Max time/iteration (s)	0.0278	0.3352	0.1992
Max time ratio	1	12	7
Average time/iteration (s)	0.0014	0.0244	0.0121
Average time ratio	1	17	8
M_x **(m)**	0.0365	0.0349	0.0349
M_y **(m)**	0.0314	0.0251	0.0251
M_θ **(rad)**	0.0013	0.0006	0.0006

Figure 15 shows the tracking performance of the compared methods when following the eight-shaped trajectory in the presence of constraints.

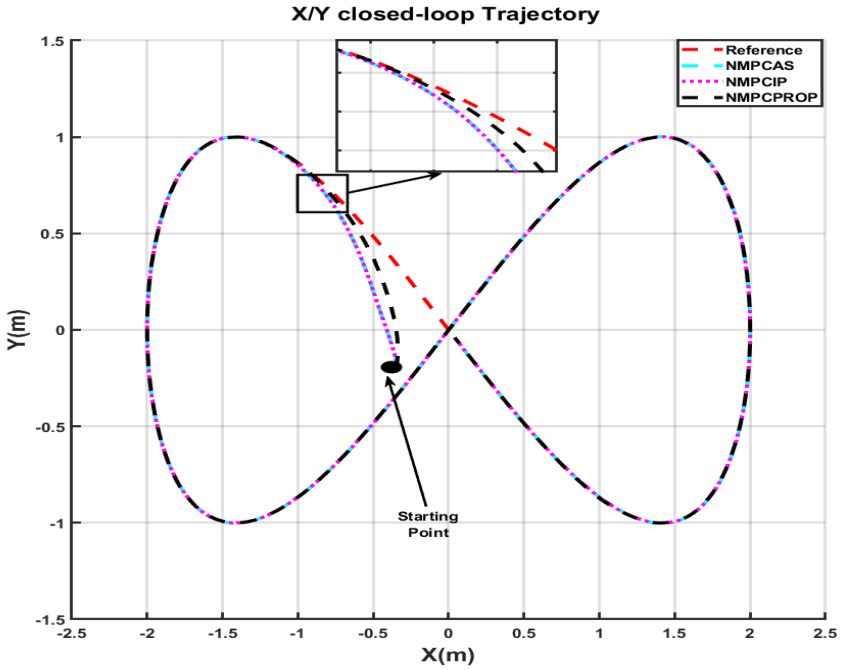


Figure 15
Tracking the eight-shaped trajectory: Constrained case.
Source: [82]

We can clearly see that NMPC-PROP maintains faster convergence than the other two methods in the constrained case.

Table 4 shows that even in the presence of constraints, NMPC-PROP kept a low computational cost which was 6 to 11 times faster than NMPCIP and 2 to 4 times faster than NMPCAS. The maximum time per iteration for NMPC-PROP remained under 70ms which kept it suitable for real-time applications. NMPC-PROP has also achieved better tracking performance than the other two methods for the x and y positions.

Figure 16 visualises the tracking errors for the constrained case, where the faster convergence of NMPC-PROP can be seen. The capability of NMPC-PROP to handle constraints is illustrated in Figure 17 which compares the resulted optimal control variables in the constrained and unconstrained case. It manages to bring the control amplitudes under the maximum allowed values after they were violated in the unconstrained case.

Table 4
Maximum time per iteration, average time per iteration, time ratios and mean quadratic tracking errors for different strategies: Constrained case

	NMPC-PROP	**NMPCIP**	**NMPCAS**
Max time/iteration (s)	0.0492	0.5452	0.2023
Max time ratio	1	11	4
Average time/iteration (s)	0.0047	0.0311	0.0122
Average time ratio	1	6	2
M_x (m)	0.0361	0.0373	0.0373
M_y (m)	0.0609	0.0636	0.0636
M_θ (rad)	0.0018	0.0013	0.0013

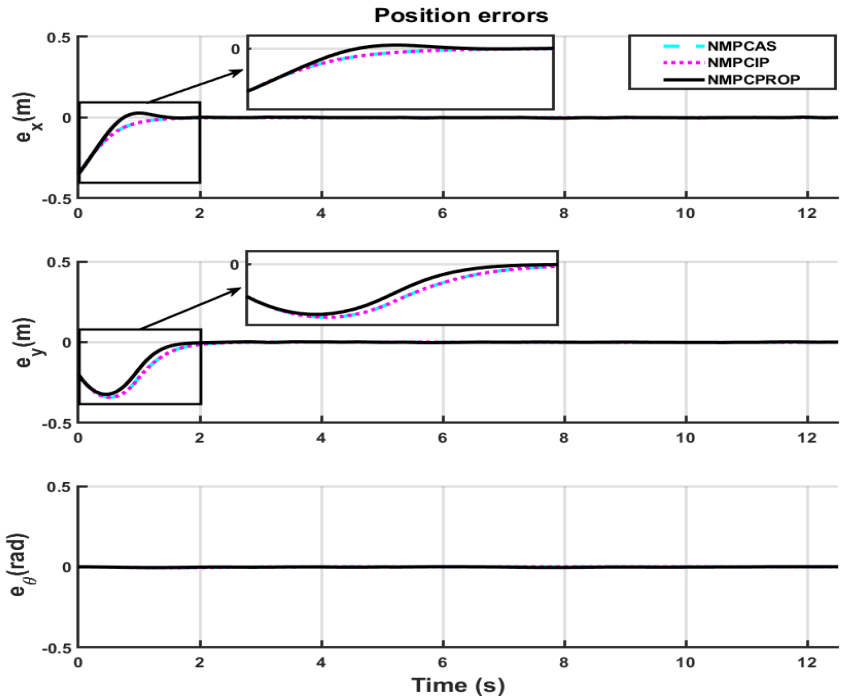


Figure 16
Position tracking errors: Constrained case

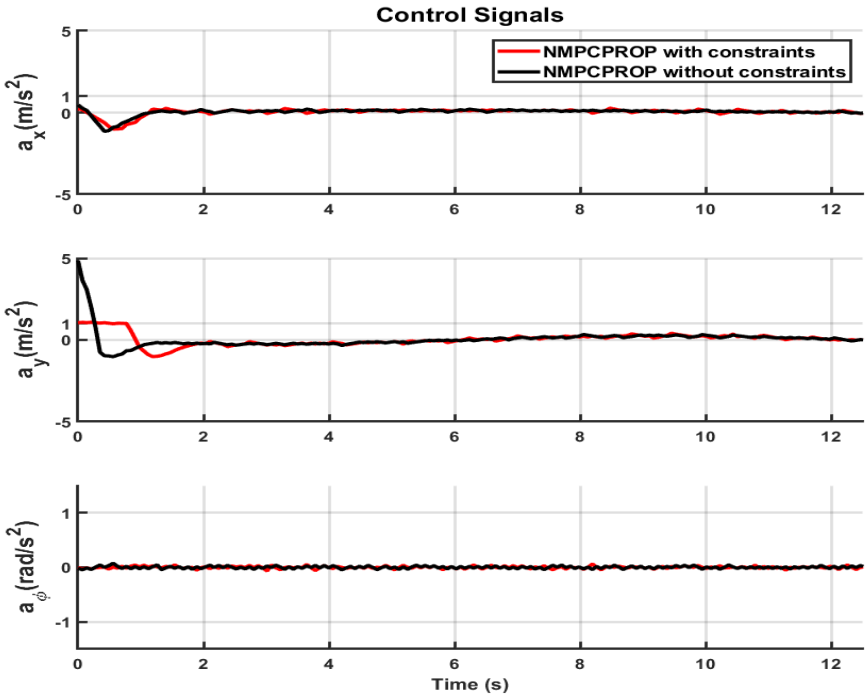


Figure 17
Comparison of control signals in constrained and unconstrained cases

4.5 Experimental results

This section provides the experimental results of testing the three methods on the Robotino-Festo robot (Figure 18).

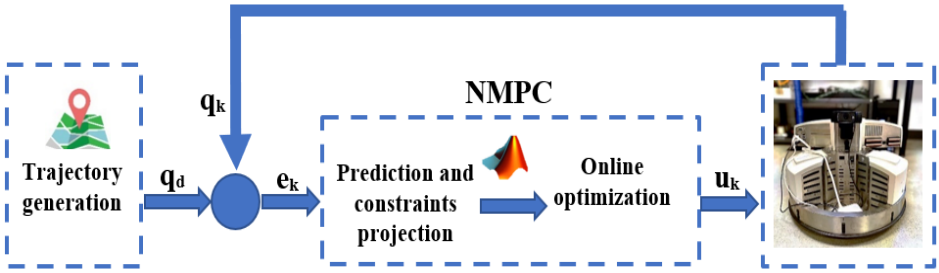


Figure 18
Overview of the trajectory tracking experiment.
Source: [82]

A MATLAB-Simulink toolbox, that was developed to communicate with Robotino, is used to implement the algorithms. The robot expects the translational and rotational speeds as inputs; therefore, the controls generated by the control algorithms were integrated prior to their application to the actual robot. The aim is to drive the robot to follow the eight-shaped trajectory given in (4.20) where $P = 50$. The initial position is chosen to be $q_0 = [-0.15,\ 0.1,\ 0,\ 0,0,0]$, and the penalization matrices are set as follows:

$$R = diag(7,7,7)$$
$$Q = Q_0 = diag(750,750,750,10,10,10) \tag{4.22}$$

Figure 19 illustrates the real-time tracking performance of the compared strategies. It is evident that NMPC-PROP exhibits quicker convergence in comparison to the other methods, which take more time to converge. This delay in convergence can be attributed to the extended computational time required during the transitory phase.

Position tracking and velocity tracking are given in Figure 20 and Figure 21 respectively. While the position tracking performance of three methods was similar, the superiority of the NMPC-PROP is clear in speed tracking thanks to its computational advantage over the other two methods.

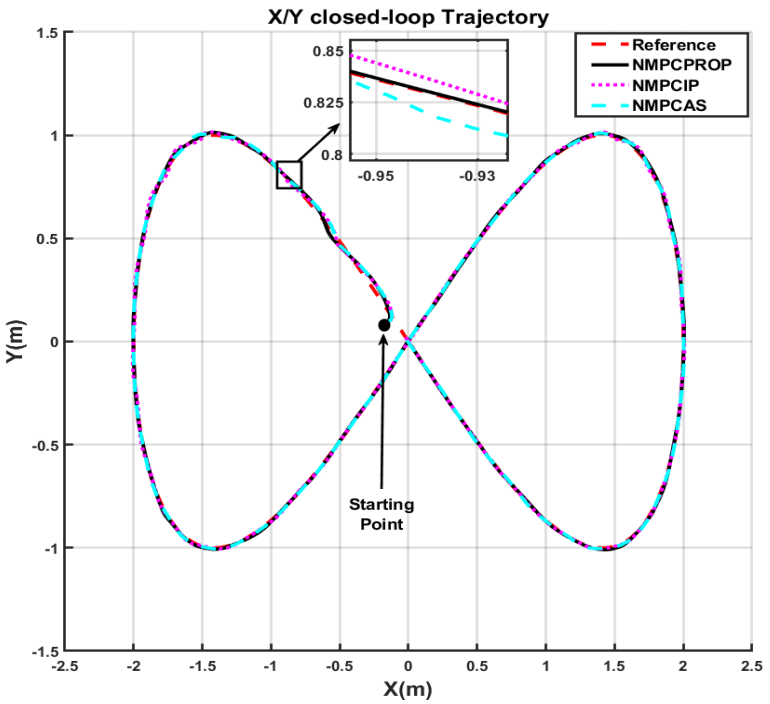


Figure 19
Real-time tracking of the eight shaped trajectory

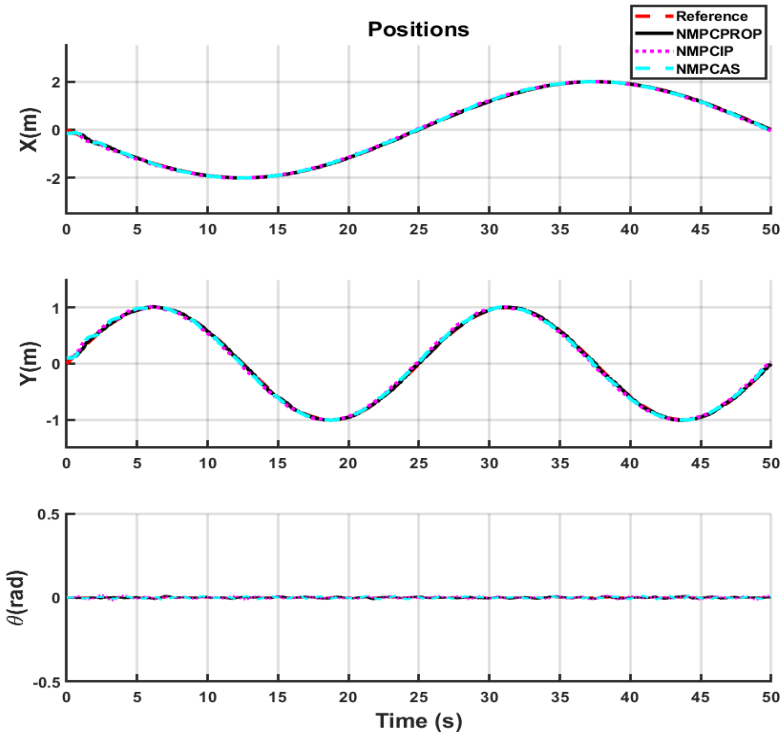


Figure 20
Real-time position tracking.

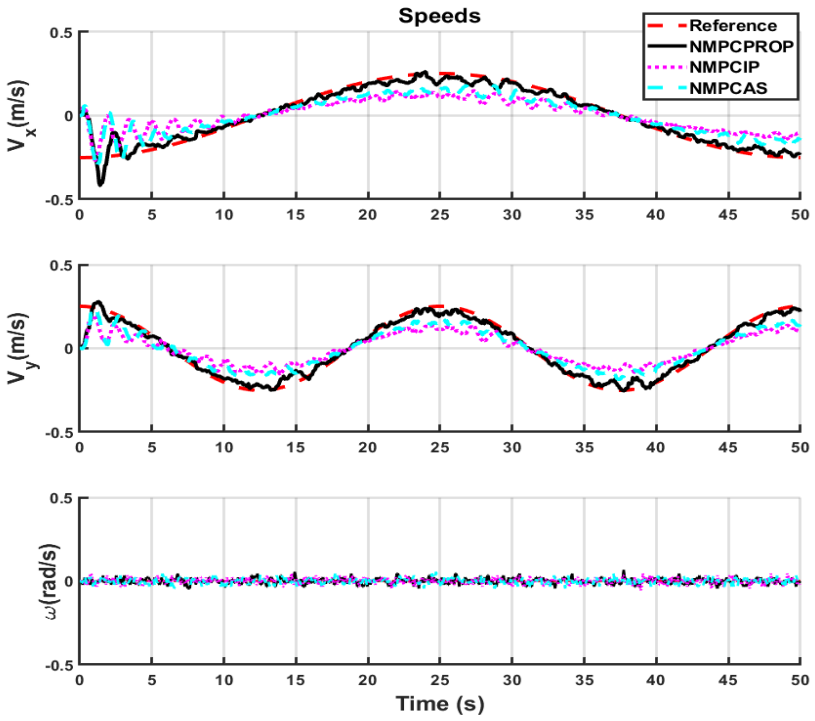


Figure 21
Real-time velocities tracking.

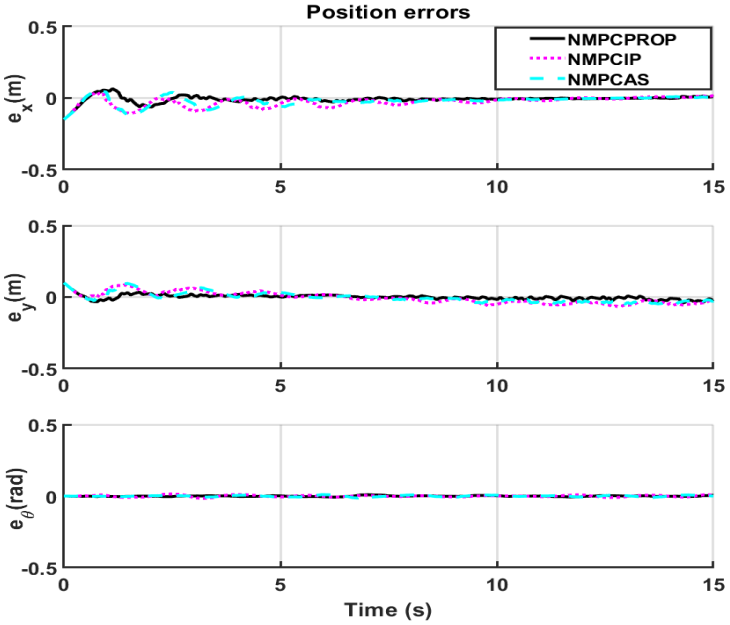


Figure 22
Real-time position tracking errors.

The position tracking errors are displayed in Figure 22 where the faster convergence of NMPC-PROP can be seen. The numeric values of the tracking errors are given in Table 5. It is shown that NMPC-PROP resulted in about a two-time reduction in tracking errors compared NMPCIP and NMPCAS.

Table 5
Quadratic tracking errors for different strategies: Real-time experiment

	NMPC-PROP	NMPCIP	NMPCAS
M_x (m)	0.0169	0.0372	0.0307
M_y (m)	0.0140	0.0347	0.0275
M_θ (rad)	0.0034	0.0049	0.0052

4.6 Conclusion

In this chapter, a NMPC controller for the trajectory tracking problem of OMR was presented. The proposed controller used ARCPROP algorithm to solve the optimization problem leading to rapid convergence, precise tracking, and low computational burden. The capability of this algorithm to solve constraint problems during online optimization was successfully demonstrated. To validate the performance of the proposed controller, comparison studies were conducted against benchmark methods, namely Interior Point and Active Set. Simulation and experimental results proved that NMPC-PROP outperformed both NMPCIP and NMPCAS in terms of computational efficiency, convergence speed and in real-time tracking performance. The results of this study highlighted the superiority of NMPC-PROP in practical applications.

5. STABITITY ANALYSES

5.1 Introduction

When designing a control algorithm, it is crucial to conduct a stability analysis to provide insights on the long-term behavior of the system, and to ensure that the control strategy does not lead to any divergence or instability. A key advantage of NMPC is the use of the nonlinear model in the controller's design, which allows the stability analyses to be conducted.

Over the past three decades, wide investigation has been made by the academic community to establish nominal stability of NMPC schemes [84-87]. Motivated by the established stability of infinite-time optimization, such as LQR, an NMPC scheme with an infinite prediction horizon was initially introduced, ensuring nominal stability. Initially applicable to unconstrained problems, the method was subsequently extended to constrained scenarios. However, the trade-off for this stability lies in elevated computational complexity due to the infinite horizon, posing challenges for real-time implementation [87, 88].

The finite-horizon criterion of NMPC is not inherently designed to ensure asymptotic properties such as stability. Therefore, achieving closed-loop stability necessitates fine tuning of design parameters, including the prediction horizon, control horizon, and weighting matrices [87]. Another way to ensure nominal stability of NMPC is to force the state to be zero at the end of the prediction process by adding a terminal equality constraint to the optimization problem [89]. However, in the nonlinear case, satisfaction of the terminal equality constraint may require an infinite number of iterations which is computationally expensive. To ease the terminal equality constraint, an alternative approach involves introducing a terminal region. This region is defined so that, at the end of the prediction, the system's state lies within it. Subsequently, a dual-mode control strategy was implemented. Outside the terminal region, the control operates in a receding horizon mode. However, upon entering the terminal region, the control mode transitions to a local feedback controller, guiding the states toward the origin [90]. To eliminate the need for transitioning to an

alternative controller within the terminal region, a terminal cost term was introduced into the objective function in [87]. This terminal cost bounds an infinite horizon cost controlled by a virtual (non-implemented) local controller. This approach is commonly referred to as Quasi-Infinite Horizon (QIH) NMPC. By appropriately choosing the terminal cost and terminal region, the nominal stability of NMPC can be ensured.

5.2 NMPC setup using the error dynamics

To analyze the stability characteristics of the NMPC schemes, a common approach involves converting the problem into a stabilization problem, where the origin serves as an equilibrium point. This transformation is achieved by incorporating the error kinematics into the NMPC algorithm. From (2.25), the error model is given as follows:

$$q_{e,k+1} = f_e(q_{ek}, u_{ek}) = \begin{pmatrix} x_{ek} + T\left(\omega_k y_{ek} + v_{xek}\right) \\ y_{ek} + T\left(-\omega_k x_{ek} + v_{yek}\right) \\ \theta_{ek} + T\omega_{ek} \\ v_{xek} + T\left(\omega_k v_{yek} - a_{ydk}\sin\theta_{ek}\right) + Tu_{1k} \\ v_{yek} + T\left(-\omega_k v_{xek} + a_{xdk}\sin\theta_{ek}\right) + Tu_{2k} \\ \omega_{ek} + Tu_{3k} \end{pmatrix} \tag{5.1}$$

where

$$u_{ek} = \begin{pmatrix} u_{1k} \\ u_{2k} \\ u_{3k} \end{pmatrix} = \begin{pmatrix} a_{xdk}\cos\theta_{ek} - a_{xk} \\ a_{ydk}\cos\theta_{ek} - a_{yk} \\ a_{\theta dk} - a_{\theta k} \end{pmatrix} \tag{5.2}$$

Using (5.1) and (5.2), the cost function to be minimized is chosen as:

$$J\left(q_{ek}, u_{ek}\right) = \frac{1}{2} q_{eN_p}^T Q_0 q_{eN_p} + \sum_{k=0}^{N_p-1} \frac{1}{2}\left(q_{ek}^T Q q_{ek} + u_{ek}^T R u_{ek}\right) \tag{5.3}$$

Based on the discrete QIH method [84], the finite-horizon optimal control problem to be solved online is formulated as follows:

$$\begin{aligned} &\arg\min_{u_{ek}} J\left(q_{ek}, u_{ek}\right) \\ &\text{subject to:} \\ &q_{e,k+1} = f_e\left(q_{ek}, u_{ek}\right) \\ &q_{ek} \in X_e \\ &u_{ke} \in U_e \\ &q_{eN_p} \in \Omega \end{aligned} \tag{5.4}$$

where X_e and U_e are the state and control constraints which can be derived from (4.7), and Ω is the terminal region in the neighborhood of the origin. The terminal cost is chosen to bound the infinite horizon cost function as follows:

$$\frac{1}{2} q_{eN_p}^T Q_0 q_{eN_p} \geq \sum_{k=N_p}^{\infty} \frac{1}{2} (q_{ek}^T Q q_{ek} + u_{lk}^T R u_{lk}) \tag{5.5}$$

where u_{lk} is a fictitious linear controller in the terminal region. The local linear controller will solely be employed to define the characteristics of the terminal cost and terminal region and will never be implemented.

5.3 Feasibility of NMPC

Feasibility in the context of NMPC refers to the ability of the algorithm to find a solution that satisfies both the system dynamics and any imposed constraints within a given prediction horizon. Specifically, feasibility is concerned with whether it is possible to find a sequence of control inputs and state trajectories that respect the system dynamics and adhere to all the specified constraints. Assuming the existing of an optimal solution for the finite optimal problem at a given time instant k_1, we can write the sequence of optimal control inputs and the sequence of optimal state trajectory as follows:

$$
\begin{aligned}
U_{k_1}^{*} &= \left(u_e^{*}(k_1), u_e^{*}(k_1+1), \ldots, u_e^{*}(k_1+N_p-1)\right)^T \\
Q_{k_1}^{*} &= \left(q_e^{*}(k_1), q_e^{*}(k_1+1), \ldots, q_e^{*}(k_1+N_p)\right)^T
\end{aligned}
\tag{5.6}
$$

When the next instant arrives, $k_2 = k_1 + 1$, and assuming a nominal case without disturbances, the control inputs and state sequences can be expressed as follows:

$$
\begin{aligned}
U_{k2} &= \left(u_e(k_2), u_e(k_2+1), \ldots, u_e(k_2+N_p-1)\right)^T \\
&= \left(u_e^{*}(k_1+1), u_e^{*}(k_1+2), \ldots, u_e^{*}(k_1+N_p)\right)^T \\
Q_{k2} &= \left(q_e(k_2), q_e(k_2+1), \ldots, q_e(k_2+N_p)\right)^T \\
&= \left(q_e^{*}(k_1+1), q_e^{*}(k_1+2), \ldots, q_e^{*}(k_1+N_p+1)\right)^T
\end{aligned}
\tag{5.7}
$$

From the definition of the terminal region and terminal cost, we know that $u_e^{*}(k_1+N_p) = u_{le}(k_1)$ and $q_e^{*}(k_1+N_p+1)$ will remain invariant in the terminal region, therefore, (5.7) is a feasible (not necessarily optimal) solution for the optimization problem at time k_2. In a recursive manner, we can deduce that if a solution exists for the finite optimal problem at the initial time instant, then the problem remains feasible for all subsequent instants.

5.4 Stability of NMPC

In this section, the stability proof of the NMPC with terminal components selected using the Quasi-Infinite Horizon (QIH) method is outlined. Define a Lyapunov candidate function $V\left(q_{ek}, u_{ek}\right)$ as the value function of the objective function (5.5):

$$
V\left(q_{ek}, u_{ek}\right) = J^{*} = \frac{1}{2} q_{eN_p}^{*T} Q_0 q_{eN_p}^{*} + \sum_{k=0}^{N_p-1} \frac{1}{2} \left(q_{ek}^{*T} Q q_{ek}^{*} + u_{ek}^{*T} R u_{ek}^{*}\right) \tag{5.8}
$$

We can easily see that $V(0,0)=0$ and $V>0$ if $q_{ek}\neq 0$ and $u_{ek}\neq 0$. Moreover, V is positive definite with holding inequality: $V\left(q_{ek},u_{ek}\right)\geq\frac{1}{2}q_{ek}^{*T}Qq_{ek}^{*}$.

It is demonstrated in [84] that when choosing the terminal terms according to the QIH principle, the following inequality holds:

$$J\left(q_{ek},u_{ek}\right)\leq\frac{1}{2}q_{ek}^{T}Q_{0}q_{ek}\qquad\text{for all } q_{ek}\in\Omega \tag{5.9}$$

which leads to:

$$\frac{1}{2}q_{ek}^{*T}Qq_{ek}^{*}\leq V\left(q_{ek},u_{ek}\right)_{k}\leq V\left(q_{ek},u_{ek}\right)\leq\frac{1}{2}q_{ek}^{T}Q_{0}q_{ek}\qquad\text{for all } q_{ek}\in\Omega \tag{5.10}$$

From (5.6) and (5.7) the cost $J\left(q_{e,k+1},u_{e,k+1}\right)$ (not necessary optimal) can be computed as:

$$\begin{aligned}J\left(q_{e,k+1},u_{e,k+1}\right)=V\left(q_{e,k+1},u_{e,k+1}\right)&-\frac{1}{2}q_{ek}^{*T}Qq_{ek}^{*}-\frac{1}{2}u_{ek}^{*T}Ru_{ek}^{*}\\&-\frac{1}{2}q_{eN_p}^{*T}Q_{0}q_{eN_p}^{*}+\frac{1}{2}q_{e,N_p+1}^{T}Q_{0}q_{e,N_p+1}\\&+\frac{1}{2}q_{eN_p}^{*T}Qq_{eN_p}^{*}+\frac{1}{2}u_{eN_p}^{T}Ru_{eN_p}\end{aligned} \tag{5.11}$$

Since Ω is an invariant set of the nominal system, we can write:

$$\frac{1}{2}q_{e,N_p+1}^{T}Q_{0}q_{e,N_p+1}\leq\frac{1}{2}q_{eN_p}^{*T}Q_{0}q_{eN_p}^{*}-\frac{1}{2}q_{eN_p}^{*T}Qq_{eN_p}^{*}-\frac{1}{2}u_{eN_p}^{T}Ru_{eN_p} \tag{5.12}$$

Combining inequality (5.12) with (5.11) we obtain:

$$J\left(q_{e,k+1},u_{e,k+1}\right)\leq V\left(q_{ek},u_{ek}\right)-\frac{1}{2}q_{ek}^{*T}Qq_{ek}^{*}-\frac{1}{2}u_{ek}^{*T}Ru_{ek}^{*} \tag{5.13}$$

Given the optimal solution at the instant $k+1$, we can write:

$$V\left(q_{e,k+1},u_{e,k+1}\right)\leq J\left(q_{e,k+1},u_{e,k+1}\right) \tag{5.14}$$

Combining inequalities (5.13) and (5.14) we get:

$$V\left(q_{e,k+1},u_{e,k+1}\right)-V\left(q_{ek},u_{ek}\right)\leq -\left(\frac{1}{2}q_{ek}^{*T}Qq_{ek}^{*}+\frac{1}{2}u_{ek}^{*T}Ru_{ek}^{*}\right) \tag{5.15}$$

Given that Q and R are both positive, we conclude that the value function V is strictly decreasing. From equations (5.8) to (5.15), we conclude the properties of V as follows: $V(0,0)=0$ and $V>0$ when $q_{ek}\neq 0$, V is bounded and strictly decreasing along a trajectory started form an admissible initial set. Thus, V is a Lyapunov function for the system (5.1) controlled by NMPC with terminal components chosen using the QIH method in the absence of disturbances ensuring nominal stability.

5.5 Characterizing the terminal components using QIH method

To ensure the nominal stability of NMPC, it is essential to determine a local linear controller for the terminal region. Given the optimality aspect of the problem, an LQR controller is selected. The linearized equations of (5.1) are of the form:

$$q_{e,k+1}=A_{ek}q_{ek}+B_{ek}u_{ek} \tag{5.16}$$

where A_{ek} and B_{ek} are given from (2.26) and (2.27) respectively as follows:

$$A_{ek}=\left.\frac{\partial f_e(q_{ek},u_{ek})}{\partial q_{ek}}\right|_{q_k=q_{dk}}=\begin{pmatrix}1 & T\omega_{dk} & 0 & T & 0 & 0\\ -T\omega_{dk} & 1 & 0 & 0 & T & 0\\ 0 & 0 & 1 & 0 & 0 & T\\ 0 & 0 & -Ta_{ydk} & 1 & T\omega_{dk} & 0\\ 0 & 0 & Ta_{xdk} & -T\omega_{dk} & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}$$

$$B_{ek} = \frac{\partial f_e(q_{ek}, u_{ek})}{\partial u_{ek}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{pmatrix}$$

The cost function to be minimized by the LQR is as follows:

$$J_{LQ} = \sum_{k=N_p+1}^{\infty} \frac{1}{2}(q_{ek}^T Q q_{ek} + u_{lk}^T R u_{lk}) \tag{5.17}$$

The associated discrete-time Algebraic Riccati Equation (DARE) is [91, 92]:

$$P_{LQ} = A_{ek}^T P_{LQ} A_{ek} - \left(A_{ek}^T P_{LQ} B_{ek}\right)\left(R + B_{ek}^T P_{LQ} B_{ek}\right)^{-1}\left(B_{ek}^T P_{LQ} A_{ek}\right) + Q \tag{5.18}$$

and the control gain matrix is:

$$L_{LQ} = \left(R + B_{ek}^T P_{LQ} B_{ek}\right)^{-1}\left(B_{ek}^T P_{LQ} A_{ek}\right) \tag{5.19}$$

Then the control input u_{lk} is computed as:

$$u_{lq} = -L_{LQ} q_{ek} \tag{5.20}$$

The terminal region Ω is then determined by finding the largest constant $\alpha \in (0, \infty)$ such that:

$$\Omega \equiv \left\{q_{ek} : q_{ek} \in \mathbb{R}^6,\ \ q_{ek}^T P_{LQ} q_{ek} \le \alpha,\ \ u_{lk} \in U_e\right\} \tag{5.21}$$

According to the QIH method [84, 87], the solution of the DARE (5.18) P_{LQ} and the region Ω can serve as terminal penalty matrix and terminal region to achieve the nominal stability of the NMPC controller.

5.6 Experimental results

In this section, we proceed to assess the performance of the proposed NMPC controller through real-time experiments. Unlike the previous tests, we extend our evaluation to include orientation tracking. This expansion allows us to check the method's efficacy across various application scenarios, providing a more comprehensive understanding of its capabilities.

Three NMPC approaches will be compared, each employing different online optimizers: ARCPROP, IP, and AS. The prediction horizon is set to $N_p = 7,$ and the control horizon is set to $N_c = 3.$ Regarding the optimization algorithms, the convergence threshold is $\delta = 10^{-6},$ and the maximum number of iterations is $I_{\max} = 40.$ The experiments are conducted using the Robotino Festo OMR as shown in Figure 18. The penalization matrices for NMPC-PROP are $R = diag(4,4,4),$ and $Q = diag(100,100,100,5,5,8)$, while for NMPCIP and NMPCAS, they are $R_n = diag(1,1,1),$ and $Q_n = diag(750,750,750,5,5,8)$. These matrices are selected through a trial-and-error process to achieve a balance between rapid convergence and overall performance; thus, their optimality cannot be ensured. Nevertheless, they exhibit good performance in this experiment.

5.6.1 Eight-shaped trajectory with time-variant orientation

In the first scenario, the robot will track an eight-shaped path given in as follows:

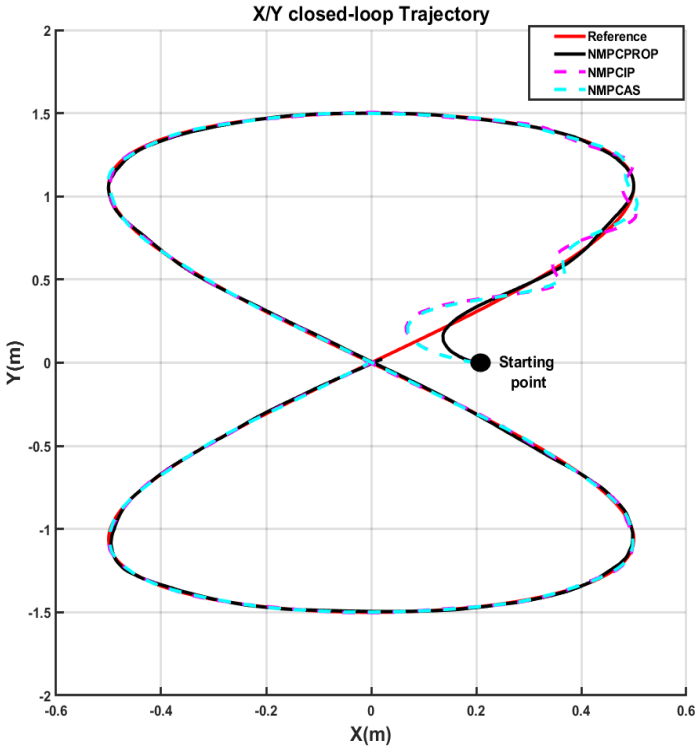
$$\begin{aligned} x_d &= 0.5\sin((4\pi / P)t) \\ y_d &= 1.5\sin((2\pi / P)t) \\ \theta_d &= \operatorname{atan}\left(\dot{y}_d / \dot{x}_d\right) + k\pi, \quad k = 0,1 \\ v_{xd} &= \sqrt{\dot{x}_d^2 + \dot{y}_d^2} \quad ; \quad v_{yd} = 0 \end{aligned} \tag{5.22}$$

where $P = 50$ is the trajectory period. The initial states are set to:

$$q_0 = [0.2,\ 0,\ \pi / 2,\ 0,0,0]$$

Figure 23 illustrates the performance of the three methods when tracking the eight-shaped trajectory. NMPC-PROP demonstrated faster convergence compared to NMPCIP and NMPCAS. Furthermore, Figure 24 and Figure 25 depict the positions and speeds tracking, respectively. The superior performance of NMPC-PROP is evident, particularly in the velocities, where it exhibits more stable tracking compared to the other two methods.

The computed control inputs are shown in Figure 26. It is notable that the initial control effort of NMPC-PROP is significantly lower than that of the other two



methods.

Figure 23
Tracking the eight-shaped trajectory with time variant orientation

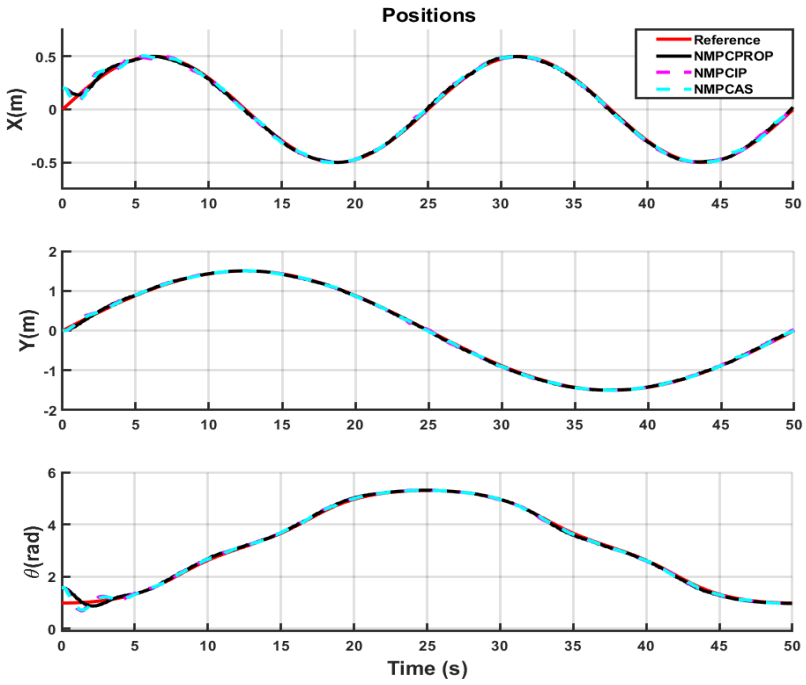


Figure 24
Position Tracking of the eight-shaped trajectory with time-variant orientation

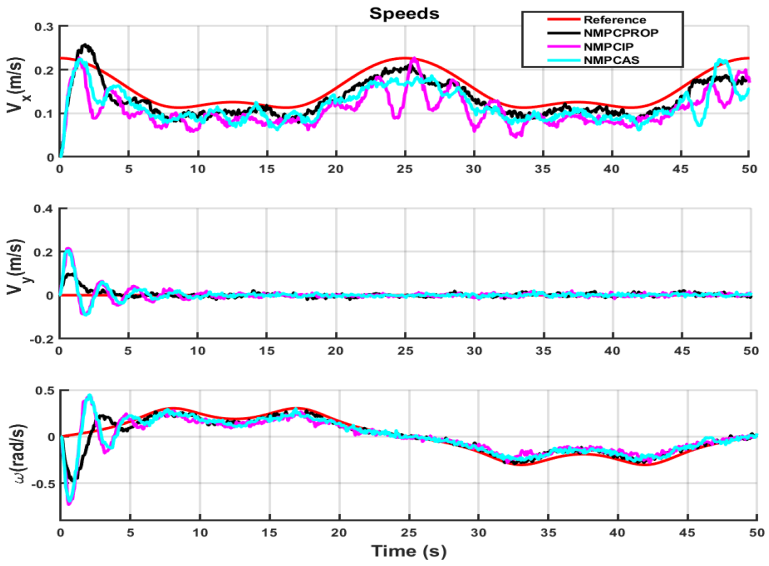


Figure 25
Velocities Tracking of the eight-shaped trajectory with time-variant orientation

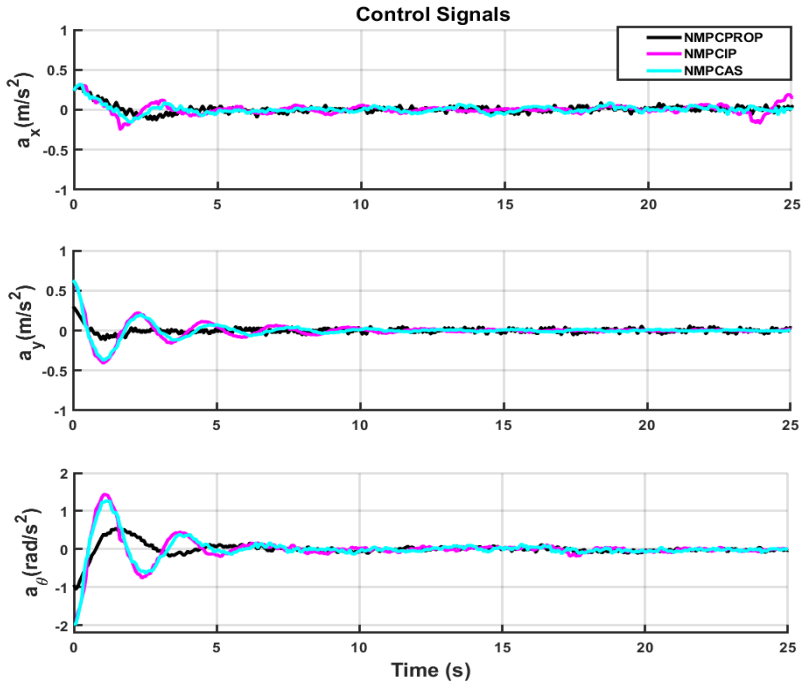


Figure 26
Control inputs computed while tracking the eight-shaped trajectory with time-variant orientation

The root mean square errors calculated from equation (4.21) are utilized for further comparison of the data, as presented in Table 6. It is evident that NMPC-PROP not only attained faster convergence but also maintained comparable position errors with the other two methods. These tracking errors are given in Figure 27.

Table 6
Root mean square errors for different strategies: Real-time experiment

	NMPC-PROP	NMPCIP	NMPCAS
M_x (m)	0.0207	0.0230	0.0212
M_y (m)	0.0173	0.0197	0.0156
M_θ (rad)	0.0802	0.0766	0.0716

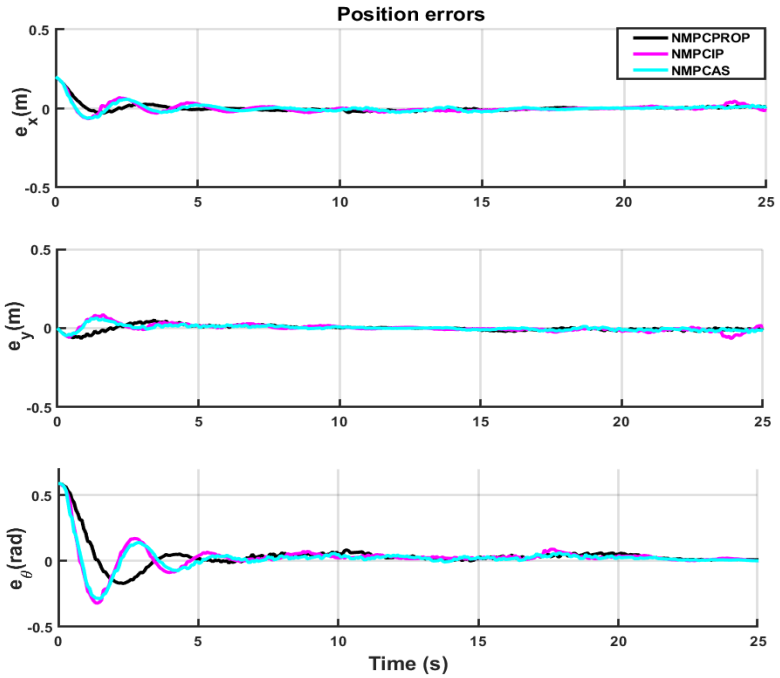


Figure 27
Tracking errors when tracking the eight-shaped trajectory with time-variant orientation

5.6.2 Square trajectory

In this experiment, the robot is instructed to follow a square path defined by the points $\{(0,\ 0),\ (2,\ 0),\ (2,\ 2),\ (0,\ 2)\}$, with a reference speed $\mathrm{v}_{xd} = 0.4\ m/s$, moving forward. This enables us to evaluate the abilities of the compared methods in handling abrupt changes in the trajectory. The initial states are set to:

$$q_0 = [0,\ 0,\ 0,\ 0,0,0]$$

Figure 28 illustrates the tracking performance in the 2-D plane. Following the first change in the desired path, NMPC-PROP adeptly resumed alignment with the path. In contrast, NMPCIP and NMPCAS exhibited oscillations around the desired trajectory without convergence.

Position tracking over time is depicted in Figure 29, while the tracking error is shown in Figure 30. It is noteworthy that NMPC-PROP exhibits lower overshooting after each trajectory change compared to the other two methods, facilitating smoother convergence.

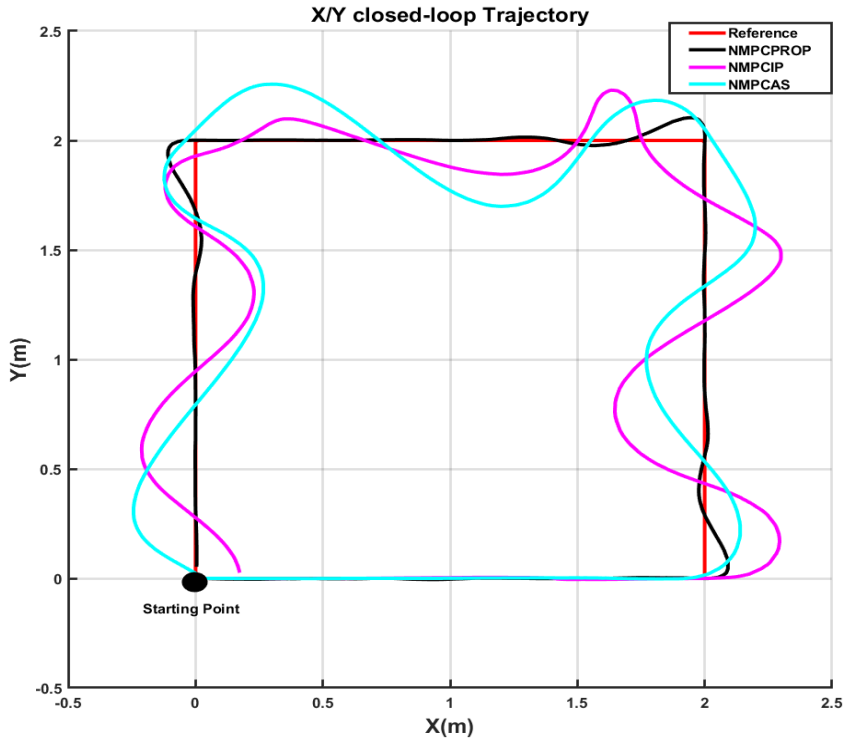


Figure 28
Tracking the square trajectory

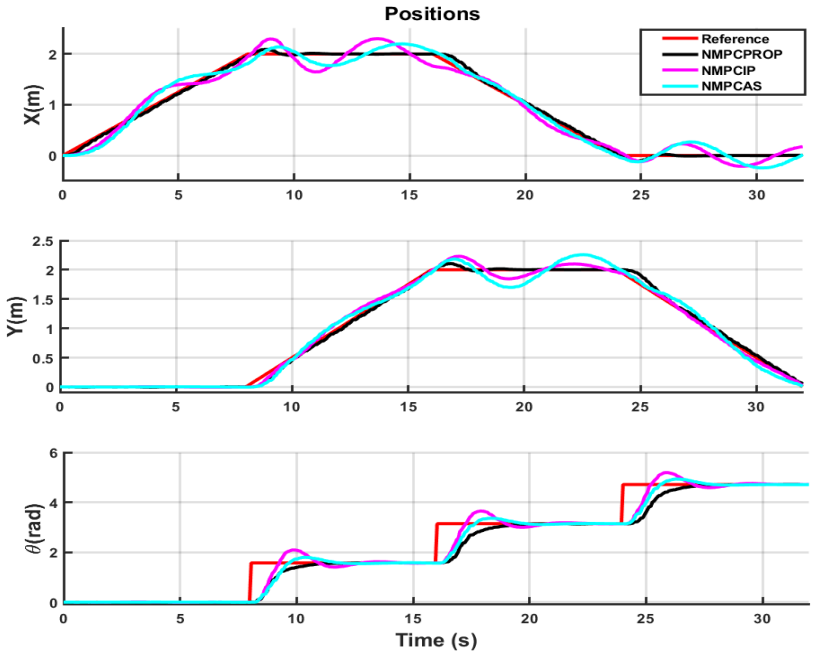


Figure 29
Tracking the square trajectory

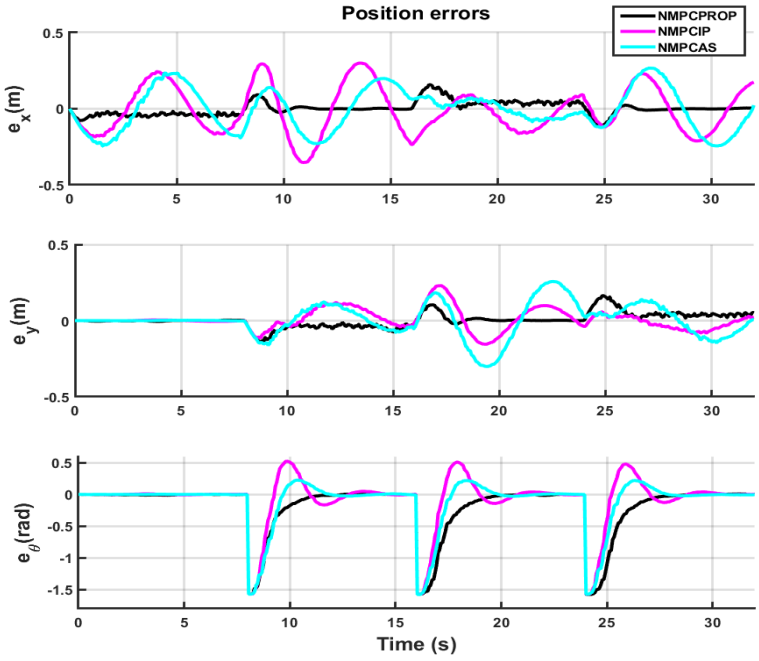


Figure 30
Tracking the square trajectory

5.7 Conclusion

This chapter provides a thorough analysis of the stability of the NMPC-PROP strategy developed in the previous chapter. Using error dynamics and the Quasi Infinite Horizon (QIH) method, the chapter establishes conditions for ensuring nominal stability of the control system through the application of Lyapunov theory. Experimental results, including complex trajectory tracking scenarios, validate the theoretical stability claims and demonstrate the practical applicability of the NMPC strategies.

6. CONCLUSION

This thesis represents a step forward in the field of control systems for OMRs. The research was driven by the need to address the challenge of trajectory tracking in these robots, especially within environments that demand high precision and adaptability. The study developed and validated advanced control algorithms designed to enhance the performance and feasibility of OMRs in real-world applications.

The first phase of the research involved developing a detailed kinematic model of the OMR. This model was crucial for predicting the robot's behavior and validating the effectiveness of the control strategies. The constructed model captured the essential kinematics of the OMR, incorporating its nonlinearities and the operational constraints, providing a reliable foundation for developing control algorithms. This detailed representation ensured that the control strategies could reliably manage the complexities of the OMR's kinematics.

Building on the model, the research proceeded to develop a theoretical framework for the proposed controllers. This framework took into account the inherited nonlinearities on OMR systems. State and control constraints were also integrated into the framework, ensuring that the developed strategies could manage the robot's physical limitations and operational boundaries effectively.

The research introduced two enhanced control algorithms, each designed to offer unique advantages in the trajectory tracking for OMRs:

- Laguerre-Based MPC approach: An innovative linearization-based MPC approach was introduced and implemented for the OMR. Traditional linearization-based methods suffer accumulation of linearization errors over the prediction horizon. To address this issue, without using typical iterative solutions, this method incorporated a non-iterative linearization process, leveraging the duality between optimal control and stochastic filtering. The duality was employed to estimate the optimal linearization points that are used

to linearize the system at each prediction instant, which enhances the prediction process. To manage the added computational load from the estimation and linearization steps required, Laguerre functions were introduced to characterize the control inputs and reduce the number of optimization variables which consequently reduces the time required for online optimization. The newly designed controller demonstrated superior performance compared to traditional MPC and NMPC and proved suitable for real-time implementation.

- Resilient nonlinear predictive controller: This controller was designed to operate directly on the nonlinear system model without resorting to any form of linearization. NMPC often requires iterative methods to solve the nonlinear optimization problem, most of which result high computational complexity particularly when performed online. To cope this computational complexity, our approach integrates a convergent variant of the Resilient Backpropagation (RPROP) algorithm for efficient online optimization. RPROP optimizes by adapting individual step-sizes and utilizing only the sign of the gradient, making it less dependent on system characteristics and thus significantly reducing computational demands. To handle state and control constraints, we employed the external penalties method, incorporating these constraints into the NMPC framework. Stability analyses were conducted, using Quasi Infinite Horizon (QIH) method to construct the terminal cost and terminal region ensuring the feasibility and nominal stability of the controlled nonlinear system. Simulation and experimental results demonstrated superiority of the proposed algorithm over other benchmark methods from the literature regarding both computational cost and tracking performance.

This research represents a significant advancement in the development of controllers for Omnidirectional Mobile Robots, offering the potential for enhanced real-world performance and future improvements.

Recommendation for future work

The present work acknowledges a reliance on trial-and-error parameter tuning in the design and implementation of the control algorithms. Both MPC parameters (horizons lengths and penalization matrices), and the optimization algorithms parameters (initial step size and increment ratio) are chosen by iteratively adjusting parameters through experimentation. Despite being a common practice, the trial-and-error process can be time-consuming and lack precision. Recognizing the potential for improvement, future research directions may explore the implementation of automated tuning algorithms. Automated algorithms provide more efficiency, consistency, and a more systematic optimization of parameters.

ANNEXE A – IMPLEMENTATION SETUP

The research presented in this thesis utilizes the Robotino Festo 3, an advanced omnidirectional mobile robot known for its versatility in both educational and industrial applications. This section provides a detailed overview of the implementation methodologies, hardware specifications, and software integration used to control the Robotino during the experiments.

- Hardware specifications

The Robotino-Festo is an omnidirectional mobile robot equipped with robust hardware to support advanced control and navigation tasks. It operates on an embedded PC that adheres to COM Express specifications and is powered by an Intel i5 dual-core processor clocked at 2.4 GHz. This setup includes 8 GB of RAM and a 23 GB SSD, ensuring adequate storage and processing capacity for real-time applications.

For motor control, Robotino utilizes a 32-bit microcontroller that generates Pulse Width Modulation (PWM) signals to actuate the DC motors. This microcontroller is integral not only in driving the motors through a PID controller but also in correcting sensor data, thereby enhancing the robot's responsiveness and precision. The system features a planetary gear unit with a 32:1 gear ratio, connecting the drive shafts to the omni-wheels. Robotino-Festo is capable of reaching a maximum translational speed of 2 meters per second and a rotational speed of 2 radians per second.

- Interaction methods with Robotino

Robotino-Festo offers two primary methods for interaction and control:

1- On board computer: The onboard computer allows direct interaction with the robot's embedded system, enabling users to deploy and run algorithm locally. This method leverages the processing power and internal resources of Robotino to execute control algorithms and process sensor data.

2- REST API Method: In this work, we utilized the REST API method, which provides a flexible and efficient way to control and monitor the robot. Robotino-Festo has its own integrated router that hosts the robot's odometry and sensor data on a specific IP address. This IP address is also used to send control commands to the robot. Robotino network parameters are given in Table 7.

Table 7
Robotino network parameters

Name	Robotino
Password	RobotinoV4
IPv4 address	192.168.0.5
Mask	255.255.255.0
Gateway	192.168.0.1

- MATLAB-SIMILINK integration

In practice, the control algorithms were designed and implemented in MATLAB, which communicated with Robotino through the REST API using the Robotino MATLAB toolbox. MATLAB reads the real-time data (position, speed, sensor readings) from Robotino's IP address. Using this data, it computes the necessary control inputs and sends them back to Robotino to adjust its movement. Specific Simulink blocks were developed using S-functions to facilitate communication with Robotino from Simulink:

- Odometry block: Takes the IP address as inputs and returns the robot's current position and velocities.
- Omnidrive block: Takes the IP address and desired speeds as inputs and sends these commands to Robotino.
- Bumper block: Inputs the IP address and reads the data from the bumper sensors to detect obstacles or collisions.

- Implementation workflow

The workflow involved reading the real-time odometry and sensor data from Robotino, processing this information in MATLAB-Simulink to compute the necessary control inputs, and then sending these inputs back to the robot as illustrated in Figure 31. This loop of data retrieval, processing, and command transmission was crucial for effective trajectory tracking and control.

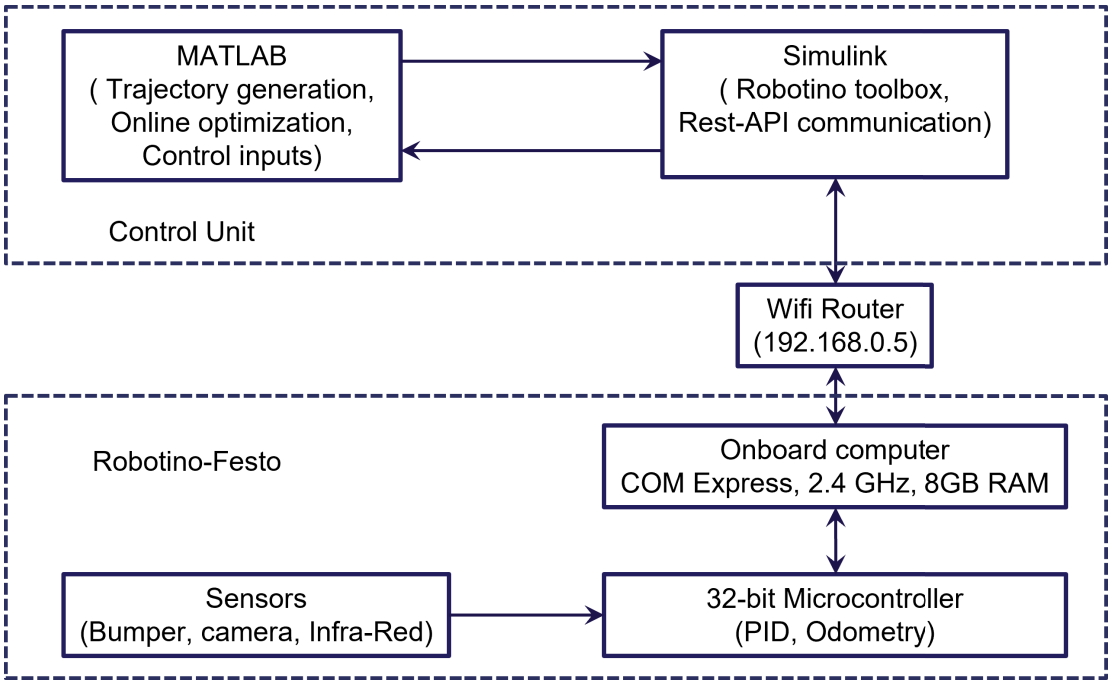


Figure 31
Communication between control unit and Robotino

REFERENCES

[1] E. E. Turhanlar, B. Y. Ekren, and T. Lerher, "Autonomous mobile robot travel under deadlock and collision prevention algorithms by agent-based modelling in warehouses," *International Journal of Logistics Research and Applications,* pp. 1-20, 2022.

[2] A. J. Sathyamoorthy, U. Patel, M. Paul, Y. Savle, and D. Manocha, "COVID surveillance robot: Monitoring social distancing constraints in indoor scenarios," *Plos one,* vol. 16, no. 12, p. e0259713, 2021.

[3] H. Hewawasam, M. Y. Ibrahim, and G. K. Appuhamillage, "Past, Present and Future of Path-Planning Algorithms for Mobile Robot Navigation in Dynamic Environments," *IEEE Open Journal of the Industrial Electronics Society,* vol. 3, pp. 353-365, 2022.

[4] F. Tian *et al.*, "Trajectory planning for autonomous mining trucks considering terrain constraints," *IEEE Transactions on Intelligent Vehicles,* vol. 6, no. 4, pp. 772-786, 2021.

[5] A. Saad *et al.*, "Advancing ocean observation with an ai-driven mobile robotic explorer," *Oceanography,* vol. 33, no. 3, pp. 50-59, 2020.

[6] M. Luperto *et al.*, "Towards long-term deployment of a mobile robot for at-home ambient assisted living of the elderly," in *2019 European Conference on Mobile Robots (ECMR)*, 2019: IEEE, pp. 1-6.

[7] W. Houtman, C. L. Martinez, S. Wang, A. Ketels, H. P. Bruyninckx, and M. van de Molengraft, "Dynamic control of steerable wheeled mobile platforms applied to an eight-wheeled RoboCup Middle Size League soccer robot," *Mechatronics,* vol. 80, p. 102693, 2021.

[8] D. Chatziparaschis, M. G. Lagoudakis, and P. Partsinevelos, "Aerial and ground robot collaboration for autonomous mapping in search and rescue missions," *Drones,* vol. 4, no. 4, p. 79, 2020.

[9] R. Raj and A. Kos, "A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives," *Applied Sciences,* vol. 12, no. 14, p. 6951, 2022.

[10] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox, "Mona: an affordable open-source mobile robot for education and research," *Journal of Intelligent & Robotic Systems,* vol. 94, pp. 761-775, 2019.

[11] L. Tagliavini, G. Colucci, A. Botta, P. Cavallone, L. Baglieri, and G. Quaglia, "Wheeled Mobile Robots: State of the Art Overview and Kinematic Comparison Among Three Omnidirectional Locomotion Strategies," *Journal of Intelligent & Robotic Systems,* vol. 106, no. 3, p. 57, 2022.

[12] D. Topolsky *et al.*, "Development of a Mobile Robot for Mine Exploration," *Processes,* vol. 10, no. 5, p. 865, 2022.

[13] H. Kim and Y. Choi, "Autonomous driving robot that drives and returns along a planned route in underground mines by recognizing road signs," *Applied Sciences,* vol. 11, no. 21, p. 10235, 2021.

[14] F. Günther, H. Mischo, R. Lösch, S. Grehl, and F. Güth, "Increased safety in deep mining with iot and autonomous robots," in *Mining Goes Digital*: CRC Press, 2019, pp. 603-611.

[15] J. Szrek, J. Jakubiak, and R. Zimroz, "A Mobile Robot-Based System for Automatic Inspection of Belt Conveyors in Mining Industry," *Energies,* vol. 15, no. 1, p. 327, 2022.

[16] A. Bonci, P. D. Cen Cheng, M. Indri, G. Nabissi, and F. Sibona, "Human-robot perception in industrial environments: A survey," *Sensors,* vol. 21, no. 5, p. 1571, 2021.

[17] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?," *Annual Reviews in Control,* vol. 50, pp. 233-252, 2020.

[18] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[19] W. A. Blyth, "Robotic pipe inspection: system design, locomotion and control," 2017.

[20] S. G. Tzafestas, *Introduction to mobile robot control*. Elsevier, 2013.

[21] Cyberbotics. "Pioneer Documentation." https://www.cyberbotics.com/doc/guide/pioneer-3dx?version=R2021a (accessed.

[22] K. Osborne. "Allen School releases MuSHR robotic race car platform to drive advances in AI research and education." https://news.cs.washington.edu/2019/08/21/allen-school-releases-mushr-robotic-race-car-platform-to-drive-advances-in-ai-research-and-education/ (accessed.

[23] H. Taheri and C. X. Zhao, "Omnidirectional mobile robots, mechanisms and navigation approaches," *Mechanism and Machine Theory,* vol. 153, p. 103958, 2020.

[24] S. G. Tzafestas, "Mobile robot control and navigation: A global overview," *Journal of Intelligent & Robotic Systems,* vol. 91, no. 1, pp. 35-58, 2018.

[25] M. A. Al Mamun, M. T. Nasir, and A. Khayyat, "Embedded system for motion control of an omnidirectional mobile robot," *IEEE Access,* vol. 6, pp. 6722-6739, 2018.

[26] S. Jeong and D. Chwa, "Sliding-mode-disturbance-observer-based robust tracking control for omnidirectional mobile robots with kinematic and dynamic uncertainties," *IEEE/ASME Transactions on Mechatronics,* vol. 26, no. 2, pp. 741-752, 2020.

[27] J.-T. Huang, T. Van Hung, and M.-L. Tseng, "Smooth switching robust adaptive control for omnidirectional mobile robots," *IEEE Transactions on Control Systems Technology,* vol. 23, no. 5, pp. 1986-1993, 2015.

[28] Z. Xu, S. X. Yang, and S. A. Gadsden, "Enhanced bioinspired backstepping control for a mobile robot with unscented Kalman filter," *IEEE Access,* vol. 8, pp. 125899-125908, 2020.

[29] Z. Li and J. Zhai, "Super-twisting sliding mode trajectory tracking adaptive control of wheeled mobile robots with disturbance observer," *International Journal of Robust and Nonlinear Control,* vol. 32, no. 18, pp. 9869-9881, 2022.

[30] A. Khadhraoui, A. Zouaoui, and M. Saad, "Barrier Lyapunov function and adaptive backstepping-based control of a quadrotor UAV," *Robotica,* pp. 1-23, 2023.

[31] K. Shao, J. Zheng, R. Tang, X. Li, Z. Man, and B. Liang, "Barrier function based adaptive sliding mode control for uncertain systems with input saturation," *IEEE/ASME Transactions on Mechatronics,* vol. 27, no. 6, pp. 4258-4268, 2022.

[32] R. H. Abiyev, N. Akkaya, and I. Gunsel, "Control of omnidirectional robot using Z-number-based fuzzy system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 49, no. 1, pp. 238-252, 2018.

[33] B. Li, Y. Fang, G. Hu, and X. Zhang, "Model-free unified tracking and regulation visual servoing of wheeled mobile robots," *IEEE Transactions on Control Systems Technology,* vol. 24, no. 4, pp. 1328-1339, 2015.

[34] T. P. Nascimento, C. E. Dórea, and L. M. G. Gonçalves, "Nonholonomic mobile robots' trajectory tracking model predictive control: a survey," *Robotica,* vol. 36, no. 5, pp. 676-696, 2018.

[35] P. Harasim and M. Trojnacki, "State of the art in predictive control of wheeled mobile robots," *Journal of Automation, Mobile Robotics and Intelligent Systems,* pp. 34-42, 2016.

[36] M. T. Watson, D. T. Gladwin, T. J. Prescott, and S. O. Conran, "Dual-mode model predictive control of an omnidirectional wheeled inverted pendulum," *IEEE/ASME Transactions on Mechatronics,* vol. 24, no. 6, pp. 2964-2975, 2019.

[37] L. Grüne and J. Pannek, "Nonlinear model predictive control," in *Nonlinear model predictive control*: Springer, 2017, pp. 45-69.

[38] I. Aliskan, "Optimized inverse nonlinear function-based wiener model predictive control for nonlinear systems," *Arabian Journal for Science and Engineering,* vol. 46, no. 10, pp. 10217-10230, 2021.

[39] S. Chang *et al.*, "Model predictive control for seizure suppression based on nonlinear auto-regressive moving-average volterra model," *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 28, no. 10, pp. 2173-2183, 2020.

[40] Z. Zeng, H. Lu, and Z. Zheng, "High-speed trajectory tracking based on model predictive control for omni-directional mobile robots," in *2013 25th Chinese Control and Decision Conference (CCDC)*, 2013: IEEE, pp. 3179-3184.

[41] L. Armesto, V. Girbés, A. Sala, M. Zima, and V. Šmídl, "Duality-based nonlinear quadratic control: Application to mobile robot trajectory-following," *IEEE Transactions on Control Systems Technology,* vol. 23, no. 4, pp. 1494-1504, 2015.

[42] B. Zhang, C. Zong, G. Chen, and B. Zhang, "Electrical vehicle path tracking based model predictive control with a Laguerre function and exponential weight," *IEEE Access,* vol. 7, pp. 17082-17097, 2019.

[43] J. A. Rossiter and L. Wang, "Exploiting Laguerre functions to improve the feasibility/performance compromise in MPC," in *2008 47th ieee conference on decision and control*, 2008: IEEE, pp. 4737-4742.

[44] M. Ławryńczuk, "Nonlinear model predictive control for processes with complex dynamics: a parameterisation approach using Laguerre functions," *International Journal of Applied Mathematics and Computer Science,* vol. 30, no. 1, 2020.

[45] J. A. Rossiter, L. Wang, and G. Valencia-Palomo, "Efficient algorithms for trading off feasibility and performance in predictive control," *International Journal of Control,* vol. 83, no. 4, pp. 789-797, 2010.

[46] J. Saeed, L. Wang, and N. Fernando, "Model Predictive Control of Phase Shift Full-Bridge DC-DC Converter Using Laguerre Functions," *IEEE Transactions on Control Systems Technology,* 2021.

[47] L. Wang, "Discrete model predictive controller design using Laguerre functions," *Journal of process control,* vol. 14, no. 2, pp. 131-142, 2004.

[48] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[49] G. Bai, Y. Meng, L. Liu, W. Luo, Q. Gu, and L. Liu, "Review and comparison of path tracking based on model predictive control," *Electronics,* vol. 8, no. 10, p. 1077, 2019.

[50] T. A. Teatro, J. M. Eklund, and R. Milman, "Nonlinear model predictive control for omnidirectional robot motion planning and tracking with avoidance of moving obstacles," *Canadian Journal of Electrical and Computer Engineering,* vol. 37, no. 3, pp. 151-156, 2014.

[51] Q. Hu, M. R. Amini, I. Kolmanovsky, J. Sun, A. Wiese, and J. B. Seeds, "Multihorizon Model Predictive Control: An Application to Integrated Power and Thermal Management of Connected Hybrid Electric Vehicles," *IEEE Transactions on Control Systems Technology,* 2021.

[52] M. Elsisi, "Optimal design of nonlinear model predictive controller based on new modified multitracker optimization algorithm," *International Journal of Intelligent Systems,* vol. 35, no. 11, pp. 1857-1878, 2020.

[53] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning," *IEEE Transactions on Automatic Control,* vol. 65, no. 2, pp. 636-648, 2019.

[54] C. Muller, D. E. Quevedo, and G. C. Goodwin, "How good is quantized model predictive control with horizon one?," *IEEE transactions on automatic control,* vol. 56, no. 11, pp. 2623-2638, 2011.

[55] M. H. Korayem, H. R. Adriani, and N. Y. Lademakhi, "Intelligent time-delay reduction of nonlinear model predictive control (NMPC) for wheeled mobile robots in the presence of obstacles," *ISA Transactions,* 2023.

[56] Y. Hamada, T. Tsukamoto, and S. Ishimoto, "Receding horizon guidance of a small unmanned aerial vehicle for planar reference path following," *Aerospace Science and Technology,* vol. 77, pp. 129-137, 2018.

[57] T. Ohtsuka, "A tutorial on C/GMRES and automatic code generation for nonlinear model predictive control," in *2015 European Control Conference (ECC)*, 2015: IEEE, pp. 73-86.

[58] Y. Hu *et al.*, "Nonlinear model predictive control for mobile medical robot using neural optimization," *IEEE Transactions on Industrial Electronics,* vol. 68, no. 12, pp. 12636-12645, 2020.

[59] Z. Li, C. Yang, C.-Y. Su, J. Deng, and W. Zhang, "Vision-based model predictive control for steering of a nonholonomic mobile robot," *IEEE Transactions on Control Systems Technology,* vol. 24, no. 2, pp. 553-564, 2015.

[60] Z. Li, J. Deng, R. Lu, Y. Xu, J. Bai, and C.-Y. Su, "Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 46, no. 6, pp. 740-749, 2015.

[61] H. Merabti, K. Belarbi, and B. Bouchemal, "Nonlinear predictive control of a mobile robot: a solution using metaheuristcs," *Journal of the Chinese institute of engineers,* vol. 39, no. 3, pp. 282-290, 2016.

[62] N. Ito, H. Okuda, and T. Suzuki, "Model predictive driving for tractor-trailer mobile robot with an omni-directional tractor," in *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2020: IEEE, pp. 1530-1533.

[63] R. G. Patel and J. J. Trivedi, "Nonlinear model predictive control of steam-assisted-gravity-drainage well operations for real-time production optimization," *SPE Production & Operations,* vol. 35, no. 03, pp. 564-578, 2020.

[64] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation,* vol. 11, pp. 1-36, 2019.

[65] R. Quirynen and S. Di Cairano, "PRESAS: Block-structured preconditioning of iterative solvers within a primal active-set method for fast model predictive control," *Optimal Control Applications and Methods,* vol. 41, no. 6, pp. 2282-2307, 2020.

[66] A. G. S. Conceição, C. E. Dórea, L. Martinez, and E. R. de Pieri, "Design and implementation of model-predictive control with friction compensation on an omnidirectional mobile robot," *IEEE/ASME Transactions On Mechatronics,* vol. 19, no. 2, pp. 467-476, 2013.

[67] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *IEEE international conference on neural networks*, 1993: IEEE, pp. 586-591.

[68] T. P. Nascimento, A. P. Moreira, and A. G. S. Conceição, "Multi-robot nonlinear model predictive formation control: Moving target and target absence," *Robotics and Autonomous Systems,* vol. 61, no. 12, pp. 1502-1515, 2013.

[69] T. P. Nascimento, C. E. T. Dórea, and L. M. G. Gonçalves, "Nonlinear model predictive control for trajectory tracking of nonholonomic mobile robots: A modified approach," *International Journal of Advanced Robotic Systems,* vol. 15, no. 1, p. 1729881418760461, 2018.

[70] T. Nascimento and M. Saska, "Embedded Fast Nonlinear Model Predictive Control for Micro Aerial Vehicles," *Journal of Intelligent & Robotic Systems,* vol. 103, no. 4, pp. 1-11, 2021.

[71] T. M. Bailey, "Convergence of Rprop and variants," *Neurocomputing,* vol. 159, pp. 90-95, 2015.

[72] A. D. Anastasiadis, G. D. Magoulas, and M. N. Vrahatis, "New globally convergent training scheme based on the resilient propagation algorithm," *Neurocomputing,* vol. 64, pp. 253-270, 2005.

[73] F. Corporation. "Festo." https://www.festo.com/us/en/ (accessed.

[74] D. Pršić, V. Stojanović, and V. Đorđević, "A Constructive Approach to Teaching with Robotino," in *7th International Scientific Conference Technics and Informatics in Education*, Serbia, 2018.

[75] F. Corporation. "Robotino Wiki." https://wiki.openrobotino.org/index.php?title=Robotino3#Hardware (accessed.

[76] E. F. Camacho and C. Bordons, *Model predictive controllers*. Springer, 2007.

[77] M. El-Sayyah, M. R. Saad, and M. Saad, "Enhanced MPC for omnidirectional robot motion tracking using laguerre functions and non-iterative linearization," *IEEE Access,* vol. 10, pp. 118290-118301, 2022.

[78] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous driving motion planning with constrained iterative LQR," *IEEE Transactions on Intelligent Vehicles,* vol. 4, no. 2, pp. 244-254, 2019.

[79] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[80] E. Todorov, "General duality between optimal control and estimation," in *2008 47th IEEE Conference on Decision and Control*, 2008: IEEE, pp. 4286-4292.

[81] C. Igel and M. Hüsken, "Empirical evaluation of the improved Rprop learning algorithms," *Neurocomputing,* vol. 50, pp. 105-123, 2003.

[82] M. El-Sayyah, M. R. Saad, and M. Saad, "NMPC for Trajectory Tracking of Omnidirectional Robot using Resilient Propagation," 2024.

[83] N. Guo, B. Lenzo, X. Zhang, Y. Zou, R. Zhai, and T. Zhang, "A real-time nonlinear model predictive controller for yaw motion optimization of distributed drive electric vehicles," *IEEE Transactions on Vehicular Technology,* vol. 69, no. 5, pp. 4935-4946, 2020.

[84] C. Rajhans, D. W. Griffith, S. C. Patwardhan, L. T. Biegler, and H. K. Pillai, "Terminal region characterization and stability analysis of discrete time quasi-infinite horizon nonlinear model predictive control," *Journal of Process Control,* vol. 83, pp. 30-52, 2019.

[85] D. Q. Mayne and P. Falugi, "Stabilizing conditions for model predictive control," *International Journal of Robust and Nonlinear Control,* vol. 29, no. 4, pp. 894-903, 2019.

[86] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica,* vol. 36, no. 6, pp. 789-814, 2000.

[87] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica,* vol. 34, no. 10, pp. 1205-1217, 1998.

[88] R. R. Bitmead, M. Gevers, and V. Wertz, "Adaptive optimal control the thinking man's GPC," 1990.

[89] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," in *Proceedings of the 27th IEEE Conference on Decision and Control*, 1988: IEEE, pp. 464-465.

[90] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE transactions on automatic control,* vol. 38, no. 11, pp. 1623-1633, 1993.

[91] L. F. Lupián and J. R. Rabadán-Martin, "LQR control methods for trajectory execution in omnidirectional mobile robots," *Recent Advances in Mobile Robotics,* pp. 385-400, 2011.

[92] D.-H. Lee and J. Hu, "A study of the duality between kalman filters and lqr problems," 2016.