SAPIENZA
UNIVERSITÀ DI ROMA

Real-Time Nonlinear Model Predictive Control for Motion Generation in Robotic Systems

Sapienza University of Rome

Dottorato di Ricerca in Automatica, Bioingegneria e Ricerca Operativa – XXXIII Ciclo

Candidate

Bárbara Barros Carlos
ID number 1794988

Thesis Advisor

Prof. Giuseppe Oriolo

March 2021

Thesis defended on July 13, 2021
in front of a Board of Examiners composed by:

Prof. Fabio Tardella (chairman)

Prof. Giuseppe Baselli

Prof. Stefano Panzieri

Real-Time Nonlinear Model Predictive Control for Motion Generation in Robotic Systems

Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: barros@diag.uniroma1.it

A France
per tutta la pazienza, l'incoraggiamento e l'amore.

Abstract

Robotics has revolutionized several industries across the globe through technologies never seen before. However, society still expects more than what today's robots are really capable of. To deal with such expectations, innovative algorithms must be translated into viable solutions so that robots can continue to improve labor in workplaces and ordinary people's lives. Among the most important are motion generation algorithms, whose advancements can solve some of the biggest barriers towards more flexible and safe human-robot interactions and operations in dynamic environments. The developments in optimization algorithms and computer processor technology have led *nonlinear model predictive control* (NMPC) to gain popularity in robotics as a motion generation technique. The main reason for that is its ability to minimize a cost function while respecting a set of constraints that typically represent the system's physical and operational limitations. Ultimately, these are the elements that grant NMPC an improved performance compared to classic approaches. Despite being a promising technique, the non-negligible computational burden associated with the online solution of the underlying optimal control problems has decisively limited its roll-out to robotic systems subject to short sampling times and resource-constrained hardware.

This thesis proposes multiple tailored algorithms for real-time motion generation in robotic systems based on high-performance implementations of NMPC. The primary computational bottlenecks concern the numerical simulation of the continuous-time nonlinear dynamic models and the online solution of the stemming large yet well-structured nonlinear program. The thesis shows that it is possible to achieve significant speed-ups in solution times while preserving nonlinearities through efficient software implementations that cover standard building blocks from nonlinear programming, tailored quadratic programming solvers, and fast approximate schemes for NMPC. Additionally, a discussion is provided in terms of dynamic modeling. It encompasses the design decisions required to create a model that exposes the system limitations so that high-quality motions can be attained due to a more accurate representation.

Among the ever-growing plethora of robotic systems, the thesis focuses on the motion generation of a double inverted pendulum and a quadrotor. In particular, two numerical simulations addressing human-robot interaction and operation in dynamic environments and two real-world applications dealing with position control demonstrate a significant improvement in control performance, with solution times in the range of micro- and milliseconds.

Acknowledgments

I am writing this thesis with a heart full of mixed feelings. It has been three and a half years of intense work, but more importantly, tremendous personal growth. I set off this journey to explore my hopes and dreams, but never actually knowing how it would unfold, how I would be changed by it, and those I would encounter. As I go ahead, I realize that this journey takes me along winding yet beautiful paths, where people seem as taken by me as I am by them. It is to them that I owe my sincere gratitude.

First and foremost, I would like to thank my advisor Giuseppe Oriolo for his incredible support over the last three and a half years. Oriolo's unique view in robotics and control theory made me see from another perspective what constitutes an acceptable solution to primary problems in robotic control systems. His extraordinary ability to ask the right questions, at the right time, in the right way taught me that who masters the definition of a problem masters its solution. I can never thank him enough for giving me the freedom to choose the academic fields of my interest and helping me to pursue my goals. He has always been a constant source of advice that I have valued greatly.

I would like to thank Marco Ferro, Paolo Ferrari, Valerio Modugno, Giulio Turrisi, Spyros Tarantos, Nicola Scianca, Massimo Cefalo, Daniele De Simone, Claudio Roberto Gaz, Khaled Al Khudir, and Maram Khatib, who have been sources of support and encouragement throughout my studies at Sapienza.

I wish to give my very special thanks to Moritz Diehl, who welcomed me so warmly at syscop and introduced me to the world of numerical methods. He has always amazed me with his brilliant view of embedded numerical methods and optimal control theory. Thank you for being a great teacher and giving me invaluable advice as my academic advisor in Freiburg.

Many thanks to Andrea Zanelli and Tommaso Sartor for our long hours of discussions and their valuable technical comments that have enhanced my understanding of numerical methods and optimal control theory. They gave me many examples of techniques to solve robotic problems that I had not thought of myself. I also would like to thank Katrin Baumgärtner, Gianluca Frison, Matilde Gargiani, Jonathan Frey, Jonas Koenemann, Jochem De Schutter, Tobias Schoels, Rachel Leuthold, Benedikt Schleusener, Maximilian Ernertus, and Cristopher Sieg at syscop and Kiteswarms for their support and the fun times we shared during my period in Freiburg.

I want to take this opportunity to thank Wolfram Burgard for his help, which otherwise this work would certainly not have been possible in its fullness. I also wish to thank Johan Vertens and Oier Meers at AIS for their support, especially during the stressful deadline periods.

I owe my sincere gratitude to Antonio Franchi for enthusiastically inviting me to research with his team at LAAS in Toulouse, which contributed significantly to a fruitful and lasting collaboration. I thank him not only for the many inspiring discussions about quadrotors and safety of human-in-the-loop robotics but also for his academic guiding in Toulouse. I so much appreciate all the Bossa Nova sessions with him and the LAAS researchers, who were also very passionate about Brazilian music.

I would like to extend my thanks to Simon Lacroix for his heartfelt support towards my research at LAAS and Anthony Mallet for the many valuable discussions about the implementation details of my NMPC within the GenoM3 architecture and his generous willingness to help with several software problems.

I wish to thank Gianluca Corsini, Dario Sanalitro, Andrea De Maio, Enrico Ferrentino, Antonio Enrique Jimenez Cano, Quentin Sablé, Enrica Rossi, Martin Jacquet, Marco Tognon, and Giuseppe Silano for their help and all the memorable moments we created in la Ville Rose. Special thanks to those who were my piece of Brazil in Toulouse, Heitor Savino and Luiza de Aguiar. There are no words for the joy they brought me.

The past years in Europe have been a wonderful opportunity for me to realize that, through language barriers, an exchange of knowledge is shared. I wish to express my gratitude to my Roman friends Mara, Fabrizio, Viviana, Valeria, Sabrina, Gabu, Tib, Silvia and, Federica, who never failed to inspire me and make me feel loved. They have always been very enthusiastic about teaching me Roman, not Italian, as well as many recipes from Italian cuisine. To them, I humbly say «veni, vidi, daje». I also wish to thank the expats that the Eternal City gave me as friends and with whom I shared many great moments: Lauren, Nicolas, Juan Pablo, Izabel, Tiago, Camila, and Laurent. Cheers to all the fun we have had, which has kept me away from science once in a while.

I heartily thank Eddie, Leia, Thérèse, and Nadia for having shared with me the multidimensional experience of everyday life in Freiburg, Toulouse, and Rome. The sum of all these experiences has decisively influenced my self-development on the very basis of my human nature, as in my interpretation of various cultural forms.

I am enormously grateful to my family. Although I had just one chance to see my family during my three and a half years of study in Europe, my heart has always been with them. They never stopped believing in me, and that was my driving force.

At last, my most special thanks go to Francesca. Her emotional support was the light on my pathway. I am incredibly grateful for her fortitude in the face of countless distress moments and her patience in listening to my everlasting conversations about papers and drones, which were most likely dull. An important journey ends and another equally important begins with you. Eu sei e você sabe, já que a vida quis assim, que nada nesse mundo levará você de mim.

Rome, Italy
March, 2021

Bárbara Barros Carlos

List of Abbreviations

A/D	Analog-to-Digital
AGV	Autonomous Guided Vehicles
API	Application Programming Interface
ARM	Advanced RISC Machines
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BLAS	Basic Linear Algebra Subprograms
BLASFEO	BLAS for Embedded Optimization
BLDC	Brushless Direct Current
C/GMRES	Continuation/Generalized Minimal Residual
CGT	Code Generation Tool
CoG	Center of Gravity
CoM	Center of Mass
CPU	Central Processing Unit
CQ	Constraint Qualification
CRTP	Crazy RealTime Protocol
D/A	Digital-to-Analog
DAE	Differential-Algebraic Equations
DC	Direct Current
DDPG	Deep Deterministic Policy Gradient
DP	Dynamic Programming
DWA	Dynamic Window Approach
EM	Euler Method
EMF	Electromotive Force
ERK4	Explicit Runge-Kutta of 4th Order
FONC	First-Order Necessary Conditions
GGN	Generalized Gauss-Newton

GNSF-IRK	Generalized Nonlinear Static Feedback-Implicit Runge-Kutta
HJB	Hamilton-Jacobi-Bellman
HPIPM	High-Performance Interior-Point Method
I/O	Input-Output
IMU	Inertial Measurement Unit
IP	Interior-Point
IPOPT	Interior-Point OPTimzer
IR	Infrared
IRK	Implicit Runge-Kutta
IVP	Initial Value Problem
KKT	Karush-Kuhn-Tucker
LA	Linear Algebra
LAPACK	Linear Algebra Package
LCL	Least Conservative Linearized
LICQ	Linear Independence Constraint Qualification
LIMP	Linear Inverted-Pendulum Model
LMPC	Linear Model Predictive Control
LP	Linear Programming
LQR	Linear Quadratic Regulator
MI	Mixed-Initiative
MOCAP	Motion Capture
MPBVP	Multi-Point Boundary Value Problem
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
NWU	North, West, Up
OCP	Optimal Control Problem
ODE	Ordinary Differential Equations
PCI	Peripheral Component Interconnect
PH	Predicted Healthiness
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
QP	Quadratic Programming
RK	Runge-Kutta
RL	Reinforcement Learning

ROS	Robot Operating System
RTI	Real-Time Iteration
RTT	Round-Trip Time
SCQP	Sequential Convex Quadratic Programming
SDRE	State-Dependent Riccati Equation
SOSC	Second-Order Sufficient Conditions
SQP	Sequential Quadratic Programming
TEB	Timed Elastic Band
UPM	Uninterruptible Power Module
ZOH	Zero-Order Hold

List of Symbols

Direct Optimal Control

N	Number of shooting intervals
Δt	Discrete-time step
t_N	Time horizon
x_i	Discrete-time differential states
s_i	Discrete-time differential states per shooting interval
u_i	Discrete-time control inputs
q_i	Piecewise affine control inputs per shooting interval
G_i	Discrete-time path constraints
G_N	Discrete-time terminal constraints

Dynamic Models

$f_p(\cdot)$	Continuous-time differential equations for the Pendubot
$f_c(\cdot)$	Continuous-time differential equations for the Crazyflie
$f_m(\cdot)$	Continuous-time differential equations for the MikroKopter
$F_p(\cdot)$	Discrete-time dynamics of the Pendubot
$F_c(\cdot)$	Discrete-time dynamics of the Crazyflie
$F_m(\cdot)$	Discrete-time dynamics of the MikroKopter

Nonlinear Programming

$\varphi(\cdot)$	Cost function
$G(\cdot)$	Equality constraints
$H(\cdot)$	Inequality constraints
$\mathcal{L}(\cdot)$	Lagrangian function
$\mathcal{A}$	Active set

Optimal Control

$L(\cdot)$	Stage cost
$E(\cdot)$	Terminal cost
$\dot{x}(t)$	Continuous-time differential states derivatives
$x(t)$	Continuous-time differential states
$u(t)$	Continuous-time control inputs
$g(\cdot)$	Continuous-time path constraints
$g_f(\cdot)$	Continuous-time terminal constraints
t_f	Horizon length
t_s	Sampling time

Quadratic Programming Solvers

B	Hessian matrix
φ	Cost function gradient
G	Equality constraint matrix
H	Inequality constraint matrix
g	Equality constraint gradient
h	Inequality constraint gradient
$\mathcal{W}$	Working set

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since Karel Čapel coined the word *robot*, breakthroughs in technology have accelerated the creation of robots that now fan out into everyday life. Once rare, these devices have been migrating steadily into the daily grind through existing service industries such as hospitality, retailing, security, in-home assistance, and entertainment (see Figures 1.1, 1.2, and 1.3). Furthermore, driverless cars are set to change the personal transport industry's future in the same way as delivery robots are setting the tone for the sector for years to come (see Figures 1.4 and 1.5). These exciting new applications result from the automation escalation outside the factories, giving rise to so-called *service robots*. Despite the technological barriers for today's service robots, there is a high degree of enthusiasm about prospects and opportunities for them. There are a few technical challenges to be tackled to secure their enduring growth such that a widespread deployment is achieved. Chief among them are trajectory planning and closed-loop control, which united synthesize the *motion generation* research field. Although the resources provided by research in the last decade have not yet made service robots exactly conventional, several segments are already leveraging them.

Autonomous guided vehicles (AGV) are paving their way around factories, warehouses, and distribution centers, carrying out tasks that involve material handling. Thanks to the advances in autonomous driving technology, the segment has been boosted so that companies could attain safety ratings for their vehicles. Yet, there are problems AGVs are expected to solve, such as random pick-and-place tasks. Similarly, retail stores have had their shelves scanned by mobile robots that cap-

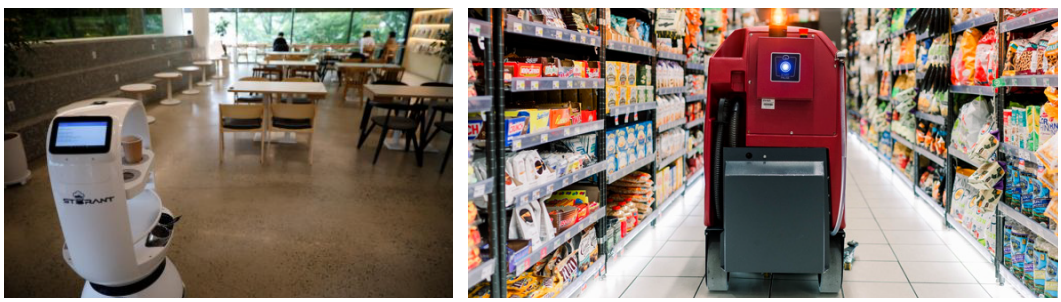


Figure 1.1. (left) robotic barista in a cafe; (right) cleaning robot in a retail store.

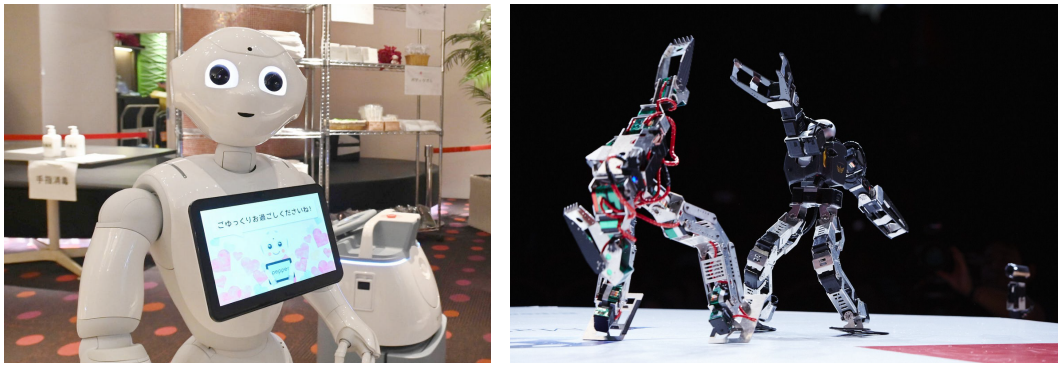


Figure 1.2. (left) receptionist robot in a hotel; (right) fighting robots in a live battle.

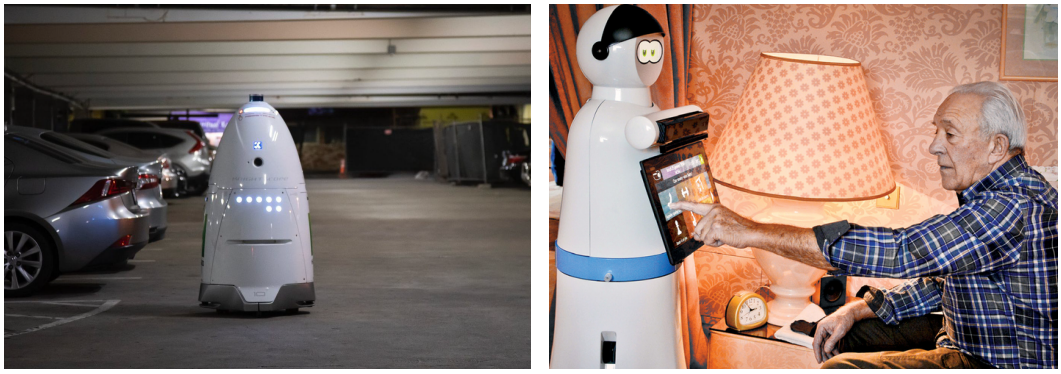


Figure 1.3. (left) security robot platform; (right) care robot to keep track of shopping lists, play music, and work as a videoconference system.

ture each product's data, which is then used to optimize the way workers unload delivery trucks and get products from the truck to the shelves faster.

In recent years, drones have expanded their addressable activities tremendously. As the fast development of component technology has made them affordable to most end-users, the segment's growth falls on value-added services. Among others, drones can execute 2D and 3D mapping, which has already been used in construction, agriculture, and mining. In construction and mining, drones are typically deployed to perform volumetric measurements on stockpiles or provide a visual assessment of haul roads state. However, to achieve the accuracy required for the final estimations, the vehicle must discharge high-precision flights using algorithms capable of maintaining a constant altitude and velocity while images are being gathered. Agriculture is no different. For instance, multispectral photos are combined into a map that provides farmers with an understanding of the crop state, stressors, diseases, and pests. Moreover, drones have shown great potential in long-range delivery. Successful examples already include blood and vaccine delivery in hard-to-reach areas in Africa and consumer goods' last-mile delivery. Limitations defined in terms of maximum payload (different weights and types of goods) and collision avoidance with electrical wires, birds, buildings, and trees represent the most significant challenges to these robots' broad and ambitious dissemination. At last, drones are particularly valuable for autonomous inspections of large geographical

Figure 1.4. Autonomous cars navigate through Pittsburgh (left) and San Francisco (right).

areas. They are dramatically altering how monitoring activities are carried out on telecommunication towers, wind turbines, solar panels, electric grids, pipelines, and so on. Breakthrough solutions are sought to achieve energy-efficient flights and also enhance collision avoidance features.

Field robotics is also taking hold in the agricultural segment. As crops have always been subject to weather vagaries, soil conditions, and pests, the demand for robotic solutions to help farmers in duties such as weed control, plant nursing, and feeding has been high. Motivated by a micro-level understanding of agricultural practices and a way to redress labor shortages, field robots are now blossoming into businesses with the help of motion generation techniques. Because there is no popular algorithm at the moment, there is plenty of room for innovative solutions that can handle monitoring, precision spraying, and harvesting. The most viable improvements when looking ahead are related to safe motions in steep slope terrains, where the robot has a limited roll, pitch, and yaw angles; complete coverage on uneven terrains without overlap; collision avoidance; and effective gripping on fruit-picking robots.

Research suggests that robotics is also making strides in healthcare spaces. Surgical robots have aided in minimally invasive procedures, making them easy and accurate to reduce the risks of infections and other complications. Shortly, technological furtherance will enable those robots to move through certain body regions while avoiding nerves and other obstacles. Rehabilitation robots are helping patients recover from strokes, paralysis, or traumatic brain injuries. They provide an assessment of prescribed exercises, measuring degrees of movement in distinct positions and tracking progress more accurately than the human eye. In the future,

Figure 1.5. (left) drone delivery; (right) six-wheeled delivery robot.

these features are likely to make robot-assisted rehabilitation more effective compared to conventional rehabilitation, which represents a massive opportunity for the development of novel motion generation methods.

Humanoid robots have been gradually introduced into social interaction contexts, albeit their primary use remains the research. This trend stems from the fact that as robots get more lifelike, people start investing in them with more trust. Conceptually, the physical acceptance experienced by humanoids aper to play a significant role in effective human-robot interactions. As opposed to other robots, humanoids' grounded acceptance is already a solid reason to support their technological development. Thus far, they have been used in research, personal assistance and caregiving, education, entertainment, search and rescue, and public relations. Even though engineering has made unprecedented progress in reproducing humans' grasp and gait patterns, today's robots cannot match humans' hands or legs.

After long years of limited interaction with humans, traditional robots finally broke out their cages and brought forth the idea of collaborative robots or *cobots*. Unlike the robots programmed to do repetitive, assembly-line tasks, cobots were conceived to meet today's industrial challenges, which demand higher versatility. As this idea reshapes the traditional industry, one can anticipate that the near future workplace will inevitably be an environment where humans and robots will work together to maximize each other's capabilities. This forethought also holds for every segment outside of the manufacturing/industrial domain. Regardless of the application, it will be necessary to take the time upfront to comprehend the workflow and ensure a robotic solution suits well in that environment to achieve this harmonious future. While flexibility is the word that will describe the most important characteristic of such solutions, defining the limits of that flexibility will be just as important. It becomes much easier to deploy and adapt a robotic solution to similar environments or workpieces when it has well-defined boundaries where small changes can take place.

Advancing motion generation algorithms is critical to ensuring the continued success of all robotic systems mentioned. Through them, robots will generate complex motions that will result, on one side, in more reliability when navigating in human spaces, and on the other side, in more fluid human-robot interactions. Therefore, research should not be limited to designing algorithms that result in robots outperforming nature only in force, speed, and precision. Instead, as robots will likely blend into our lives as the years go by, research should also look for algorithms that allow them to interact with humans or operate in dynamic environments. This thesis believes that algorithms must meet three desiderata to reach a broader repertoire of robot movements:

First, they must embody models that expose the limitations of dynamic primitives. As seen in natural systems, motor control often uses dynamic primitives (e.g., submovements and oscillations) to compose complex motions. For instance, the act of hovering in mid-air experienced by bees involves a series of forward-, mid-, and rearward-strokes at extremely high wingbeat frequencies. While bees' wing angle of attack ranges from 41.1 to 50.5 degrees at mid-stroke to prevent a stall, their forward-to-rearward stroke range of 90 degrees creates an imaginary plane whose tilting enables them to move forward and in reverse. These findings taught researchers how some aerial robots could perform vertical take-off, hovering,

and steering. Likewise, several studies are still underway to learn more about how humans move in order to achieve stable skip, gallop or skitter in walking robots and sit-to-stand in exoskeletons. The importance of bringing limitations to light is because robotic systems, for the most part, are reproductions of natural systems, which in turn are endowed with limited dynamic primitives. For this reason, motion generation algorithms ought to use models that take into account such limitations to avoid setting unrealistic standards of what robots can do.

Second, they must make the best of the synergy between robots and their computers. Most of the robotic applications previously mentioned are compute-intensive. As a result, their algorithms have traditionally had to grapple with trade-offs between motion quality and efficiency. This observation reveals that researchers have so far given scant attention to performance-oriented architectural algorithms that can leverage such trades more aggressively. As each computer system has a different architecture and approach to trade-offs, exposing information from hardware to software allows, for example, better hardware throughput. Therefore, motion generation algorithms shall exploit the instructions that all machines implementing a particular architecture must support, such as arithmetic, floating-point, vector operations, and several possible extensions as a direct means to improve overall performance.

Third, they must obey a strict *real-time* limit. Several motion generation problems in robotics are tackled by addressing trajectory planning and closed-loop control separately. In a traditional algorithmic framework, a trajectory planner generates a feasible trajectory in the configuration space, and a feedback control policy faithfully tracks it. Ideally, online trajectory planning is preferable to offline solutions because, in this way, the robot can react to dynamic obstacles and prior miscalculations. However, the computational burden to converge to a feasible and optimal solution under real-time constraints ultimately limits online trajectory planning approaches. Computational complexity can be alleviated if one uses simplified dynamic models rather than highly accurate ones but at the price of overloading the control policy that now needs to reject disturbances and deal with model mismatches. Whether coupling or decoupling trajectory planning and control, an algorithmic framework must terminate whatever computations are carried out within a real-time limit given a priori, returning a sufficiently accurate solution.

Problem statement The problem addressed in this thesis concerns real-time motion generation in robotic systems. The goal is to develop high-performance solutions in terms of resource-efficiency and motion quality that allow a robot to move either autonomously or semi-autonomously in predictable environments where static or dynamic obstacles may occasionally be present.

Real-time optimal control is a promising approach for solving the class of motion generation problems that this thesis is interested in. It allows one to minimize online a given *cost function* while satisfying a set of *constraints* that typically arise from operational or physical limitations. It is a sophisticated way of formulating a real-life problem through an *optimal control problem* (OCP) where cost, system dynamics, and constraints are explicitly taken into account. The online solution of an OCP outlines the elegant principle of *model predictive control* (MPC). The idea is simple: i) estimate the current state, ii) solve an OCP formulated over a time horizon using

the current state as the initial value, iii) get the first control of the trajectory that results from the numerical solution of the OCP, iv) apply it to the system for one time step, v) shift the horizon forward in time and repeat.

Models are the foundation of any predictive controller. *Linear* MPC (LMPC) is characterized by the use of linear dynamic models in the prediction, whereas *nonlinear* MPC (NMPC) uses nonlinear ones. Although both require the iterative solution of OCPs on a finite-horizon, the first striking difference is that in LMPC, these problems are convex, while in NMPC not necessarily. This trait poses significant challenges for both numerical solution and real-time feasibility, reasons why NMPC was restricted to slow dynamics systems for a long time. Given the context seemingly discouraging, a legitimate question to raise is: is there a place for NMPC? This question lies at the heart of what may be called "*the paradox of accurate representation of a process model*." It refers to the challenge in selecting a model, be it linear or nonlinear, that can appropriately describe a process's evolution. Although linear models have proven to be an educated approximation of real processes on many occasions, nonlinear models have become more valuable as researchers and practitioners seek to produce testable characterizations of nonlinearities that can guarantee superior performance in terms of motion quality. Besides, nonlinear models with a reasonable complexity are less prone to cause dynamic systems to exhibit non-physical behaviors.

The applicability of NMPC strongly relies on efficient optimization routines for solving the underlying OCPs. Since in NMPC, the problem that is repeatedly solved has a fixed structure, one can use this information to build tailored strategies to exploit sparsity and knowledge about the matrices size. To this end, much effort has been put into developing structure-exploiting algorithms that can reduce computational complexity and memory footprint through efficient and scalable factorization routines. These efforts have inspired other algorithmic breakthroughs that include linear algebra kernels for small-to-midsize matrices, optimized for different CPU architectures. Combined, these ideas enabled the implementation of high-performance MPC solvers that guarantee significant speed-ups in solution times.

Among the advantages of NMPC compared to classical control techniques is the unification of trajectory planning and control problems. Assuming that a trajectory can be quickly optimized, it can be used as a feedback control policy for a closed-loop system, yielding robust feedback against unforeseen disturbances and errors due to model-plant-mismatch. However, the main computational bottleneck in NMPC is typically the numerical solution of the (potentially) nonconvex OCPs. Hence, researchers have been devoted to finding remedies to this limitation. One attractive alternative relies on direct optimal control methods using Newton-type optimization algorithms to cast *real-time NMPC controllers*. At the core of it, instead of iterating to convergence an OCP that becomes more and more outdated as the system evolves, one can use an approximate feedback policy that, under proper initialization, runs within real-time limits and, thereby, reacts to disturbances considerably faster.

For all these reasons, this thesis considers that real-time NMPC uniquely meets all desiderata sought. Furthermore, among all the robotic systems mentioned earlier, this thesis will be committed to solving motion generation problems of only two: i) a double inverted pendulum whose stabilization principle typically describes complex motions in walking or self-balancing robots, and ii) a quadrotor.

1.1 Contributions and overview

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

Chapter 2 – Real-time nonlinear model predictive control This chapter introduces the class of dynamic systems and optimal control problems dealt with throughout the thesis. It then provides a detailed overview on direct optimal control, including a discussion on the most successful shooting methods for nonlinear optimal control problems in robotics. It presents the fundamental concepts and numerical methods for nonlinear programming. In particular, *sequential quadratic programming* (SQP) and Newton-type optimization, crucial for the efficient implementation of real-time optimal control. Besides, it presents state-of-the-art tailored *quadratic programming* (QP) solvers for the solution of the resulting QP subproblems. In particular, two methods are presented and analyzed: active-set methods and interior-point methods. The analysis is further leveraged by introducing two condensing approaches and the situations in which they may be favorable. The chapter shows how one can exploit the particular structure of the subproblems to achieve significant speed-ups in solution times. The chapter also provides a concise survey of the most recent real-time optimization algorithms, followed by a detailed description of the *real-time iteration* (RTI) scheme, the algorithm underlying the real-time NMPC controllers in the thesis. Finally, it describes the open-source software packages that implement the RTI scheme used in this thesis.

Chapter 3 – Dynamic models This chapter is divided into two parts. The first part quickly goes through the derivation of a nonlinear dynamic model of a double inverted pendulum called Pendubot. The second part proposes a nonlinear dynamic model of a quadrotor, including an initial discussion on the trade-off between the level of model complexity and the simplicity needed to design a controller. It proceeds by stating some assumptions and then continues with a brief presentation of the coordinate system transformation, followed by explaining the forces and moments acting on the quadrotor. It describes the rotor dynamic model, which is crucial for building up two different sets of ordinary differential equations: one for the Crazyflie nano-quadrotor and one for the MikroKopter quadrotor. In reliance on the concepts built in the previous chapter and current one, all subsequent chapters present the thesis's main contributions.

Massimo Cefalo kindly provided the parameters for the Pendubot model, while the parameters for the MikroKopter model were generously given by Gianluca Corsini.

Chapter 4 – Real-time NMPC for motion generation of a double inverted pendulum The inverted pendulum stabilization principle has been translating a myriad of motion generation problems in robotics up to now. These problems drive some of the technological trends that, in turn, reveal where the use of this principle may eventually lead. On these grounds, high-performance control in terms of computational efficiency and motion quality becomes the key when trying to resize the solution for any similar but high-dimensional system, such as walking or self-balancing robots. To this end, this chapter describes the design and implementation

of an efficient real-time NMPC to solve the swing-up problem of a double-inverted pendulum, a more challenging robotic variant. The controller is implemented using `ACADO`, a software package for dynamic optimization and control, implementing the RTI scheme. The QP subproblems arising from the tailored OCP are solved with the active-set method solver `qpOASES`, devised for real-time optimal control. The NMPC controller is validated through real-world experiments, and the results reveal that it is efficient enough to run in the microsecond range.

This chapter is based on the work published in [1]. The experiments were carried out jointly with Giulio Turrisi, Valerio Modugno, and Massimo Cefalo in the DIAG Robotics laboratory at Sapienza University of Rome.

Chapter 5 – An efficient real-time NMPC for quadrotor position control under communication time-delay The advances in computer processor technology have enabled the application of NMPC to agile systems, such as quadrotors. These systems are characterized by their underactuation, nonlinearities, bounded inputs, and time-delays. This chapter presents the design and implementation of an efficient position controller for quadrotors based on real-time NMPC with time-delay compensation and bounds enforcement on the actuators. To deal with the limited computational resources onboard, an offboard control architecture is proposed. It is implemented using the high-performance software package `acados`, which solves optimal control problems and implements an RTI variant of an SQP scheme with Gauss-Newton Hessian approximation. The quadratic subproblems in the SQP scheme are solved with `HPIPM`, an interior-point method solver built on top of the linear algebra library `BLASFEO`, finely tuned for multiple CPU architectures. Solution times are further reduced by reformulating the QPs using the efficient partial condensing algorithm implemented in `HPIPM`. The capabilities of the architecture are demonstrated experimentally using the Crazyflie nano-quadrotor.

This chapter is based on the work published in [2]. Besides the detailed comparisons shown in the chapter, the author released an open-source ROS stack with an efficient real-time NMPC as an additional contribution. It is readily available at [3]. The experiments were carried out jointly with Tommaso Sartor and Andrea Zanelli and with the generous help of Maximilian Ernestus, Benedikt Schleusener, and Johan Vertens in the AIS arena at the University of Freiburg.

Chapter 6 – Real-time NMPC for quadrotor motion generation in dynamic environments Today's robotics has shown many successful strategies to solve several navigation problems. However, moving into a dynamic environment is still a challenging task. This chapter presents a novel method for motion generation in dynamic environments based on real-time NMPC. At the core of the approach is a least conservative linearized constraint formulation built upon the RTI scheme with Gauss-Newton Hessian approximation. It is demonstrated that the proposed constraint formulation is less conservative for planners based on Newton-type method than for those based on a fully converged NMPC method. The approach is validated in two simulated scenarios. First, the Crazyflie nano-quadrotor avoids balls and reaches its desired goal despite the uncertainty about when the balls will be thrown. Second, it successfully avoids all obstacles in a cluttered environment, reaching the

goal without any constrain violation. The numerical results prove the theoretical findings and illustrate the computational efficiency of the proposed scheme.

This chapter is based on the work published in [4].

Chapter 7 – Mixed-initiative control based on real-time NMPC for safe human-quadrotor interaction This chapter presents a novel algorithm for blending human inputs and automatic controller commands, guaranteeing safety in mixed-initiative interactions between humans and quadrotors. The algorithm is based on NMPC and involves using the state solution to assess whether safety- and/or task-related rules are met to mix control authority. The mixing is attained through the convex combination of human and actual robot costs and is driven by a continuous function that measures the rules' violation. To achieve real-time feasibility, the algorithm relies on the RTI scheme to cast the mixed-initiative controller. Its effectiveness is shown through numerical simulations, where a second autonomous algorithm is used to emulate the behavior of pilots with different skill levels. Simulations show that the scheme provides suitable assistance to pilots, especially novices, in a workspace with obstacles while underpinning computational efficiency.

This chapter is based on the work accepted for publication in [5].

Chapter 8 – Conclusions and future research The remarks on the given contribution and the directions for future research are summarized.

1.2 Notation

In this thesis, we use $\mathbb{R}$ for the set of real numbers, and $\mathbb{R}^+$ for the non-negative ones. The set of real-valued vectors of dimension n is denoted by $\mathbb{R}^n$. Analogously, we write $\mathbb{R}^n_{>0}$ to denote the set of positive real-valued vectors of dimension n. The space of quaternions is denoted by $\mathbb{H}$. The set of matrices with n rows and m columns is denoted by $\mathbb{R}^{n\times m}$. We use round brackets when presenting vectors and matrices. Vectors are column vectors and we use the notation $(x; y)$ to denote concatenation. The operator $x \leq y$ is understood to be element-wise. We denote the gradient vector $\nabla f(x) \in \mathbb{R}^n$ as column vector. Sometimes, for a symmetric positive-definite matrix, we write $A \succ 0$. It is denoted by $\mathbf{I_n}$ the set of $n \times n$ identity matrices. An all-ones vector of dimension n is denoted by $\mathbf{1_n}$, whereas $\mathbf{0_n}$ denotes a n-vector of zeros. When dealing with norms of vectors $x \in \mathbb{R}^n$, we denote by $\|x\|$ the Euclidean norm. Denoted by $\|x\|^2_Q = x^T Q x$ is the weighted ℓ_2^2-norm with positive-definite weighting matrix $Q \in \mathbb{R}^{n\times n}$. The weighted ℓ_1-norm is defined by $\mu\|x\|_1$, where $\mu \in \mathbb{R}^n$.

Chapter 2

Real-time nonlinear model predictive control

MPC is an advanced control method that involves the solution of a finite-dimension optimization problem where the current state is used as the initial value to produce the optimal control action at a given instant. The optimization problem is solved using a dynamic model to forecast system behavior over a horizon shifted forward in time in an online procedure that grants convenient robustness against disturbances. Models are, therefore, crucial to every form of MPC [6]. For systems that can be adequately described by linear models, LMPC has become the prevailing advanced control method in the process industries [7]. Arguably this is due to its ability to handle large-scale systems subject to physical and operational constraints with reasonable computational complexity. As the optimization problem is parametric on the initial value and the controls are piecewise affine, there is an affine dependence between control and parameter. This allows one to precompute the feedback policy for every point in the feasible parameter space, reducing the computational cost to a search in a mapping table for the suitable *critical region* [8]. Although systems in the robotics domain are most likely to have a nonlinear model, LMPC has still been widely applied to various robots under the umbrella of simplified models (see, e.g., [9, 10, 11, 12, 13, 14, 15]). The method press ahead in this domain mainly because the underlying problems are typically quadratic or linear programs, known to be *convex*, and for which there exist a plethora of mature numerical methods.

While the computational complexity of LMPC poses a modest challenge for current processor technology, the complexity of NMPC undoubtedly limits its impact across multiple domains due to the challenges associated with the online solution of *nonlinear nonconvex* programs. Despite the difficulties, nonlinear models are quickly becoming common as researchers hanker for high-accuracy control performance, catapulting new software implementations. The results obtained in the early years of nonlinear optimal control theory (from the 1950s to the 1980s), with the well-founded formalizations [16] and [17], set off several academic studies in NMPC, for instance, [18, 19, 20, 21, 22, 23, 24, 25]. There have been some attempts to advocate NMPC as a viable control method in the process industries (see, e.g., [7, 26]). In particular, domains characterized by sufficiently long sampling times, such as chemical processing and oil refineries, have seen successful industrial applications of

NMPC since the 1980s. As the computation times associated with the solution of the underlying optimization problems can be rather long, the employment of NMPC has only recently been extended to applications where shorter sampling times are required [2]. The advances in numerical methods have enabled the development of a mature class of optimization algorithms such as SQP and *interior-point* (IP) methods. However, the real-time dilemma lies in the fact that while the SQP/IP iterations are being performed, the physical system evolves, and the information used to compute the control action becomes outdated [27]. This observation gives rise to recent NMPC schemes that trade accuracy for speed to achieve real-time feasibility. Underpinned by the progress in numerical methods that go back to the 1970s with LMPC, these approximate schemes can solve NMPC problems online efficiently, exploiting the inherent problem structure and reducing the computational burden.

As the lag in computer processor technology was never a long-lasting issue to stop new algorithms from being developed, this thesis aims at demonstrating that NMPC is ready for use in the context of complex robotic systems. Due to their strong real-time requirements, we rely on a successful approximate scheme known as the RTI scheme to present practical realizations of NMPC for motion generation in robotic systems. This chapter shows in detail the theoretical concepts needed for applying real-time NMPC to this class of systems.

2.1 Preliminaries

In this thesis, we consider continuous-time dynamic models described by a first-order *ordinary differential equation* (ODE)

$$\dot{x}(t) = f(x(t), u(t)), \quad \forall t \in [t_0, \infty), \tag{2.1}$$

with initial condition

$$x(t_0) = x_0. \tag{2.2}$$

Here, $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ denote the states and control inputs of the system whose dynamics are described by $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, and whose initial state is $x_0 \in \mathbb{R}^{n_x}$.

For this ODE setting, we will state a continuous-time infinite-dimensional OCP that minimizes a cost function of *Bolza* type

$$\int_{t_0}^{t_f} L(x(t), u(t))dt + E(x(t_f)), \tag{2.3}$$

where $L(x, u)$ is the *stage cost*, and $E(x(t_f))$ is the *terminal cost*. The state and control trajectories are defined on the time horizon of interest, i.e., $\forall t \in [t_0, t_f]$, where t_f is the horizon length. The OCP will also take into account the *path constraints*

$$g^l(t) \leq g(x(t), u(t)) \leq g^u(t), \quad \forall t \in [t_0, t_f) \tag{2.4}$$

and the *terminal constraint*

$$g_f^l(t) \leq g_f(x(t_f)) \leq g_f^u(t), \tag{2.5}$$

which must be satisfied for system (2.1) over the horizon.

This thesis aims at solving OCPs that typically arise in MPC where the *current state estimate* $\bar{x}_0$ is considered a parameter. Thus, the OCP that we address is:

$$\begin{aligned}
\min_{u(\cdot)} \quad & \frac{1}{2}\int_{t_0}^{t_f} \|\eta(x(t),u(t)) - \tilde{\eta}\|^2_{W_s} \mathrm{d}t + \frac{1}{2}\|\eta_f(x(t_f)) - \tilde{\eta}_f\|^2_{W_f} && \text{(2.6a)}\\
\text{s.t.} \quad & x(t_0) - \bar{x}_0 = 0, && \text{(2.6b)}\\
& \dot{x}(t) - f(x(t),u(t)) = 0, \qquad \forall t \in [t_0, t_f), && \text{(2.6c)}\\
& x^l(t) \leq x(t) \leq x^u(t), \qquad \forall t \in [t_0, t_f), && \text{(2.6d)}\\
& u^l(t) \leq u(t) \leq u^u(t), \qquad \forall t \in [t_0, t_f), && \text{(2.6e)}\\
& g^l(t) \leq g(x(t),u(t)) \leq g^u(t), \qquad \forall t \in [t_0, t_f), && \text{(2.6f)}\\
& g_f^l(t) \leq g_f(x(t_f)) \leq g_f^u(t). && \text{(2.6g)}
\end{aligned}$$

Therein, the stage and terminal cost terms are of *least squares* type, characterizing the so-called *tracking objective*. The positive-definite matrices $W_s \in \mathbb{R}^{n_\eta \times n_\eta}$ and $W_f \in \mathbb{R}^{n_{\eta,f} \times n_{\eta,f}}$ are called *weighting matrices*. The output functions in the stage and terminal cost terms are denoted by $\eta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$, while $\eta_f : \mathbb{R}^{n_x} \to \mathbb{R}$ and variables $\tilde{\eta} \in \mathbb{R}^{n_\eta}$ and $\tilde{\eta}_f \in \mathbb{R}^{n_{\eta,f}}$ denote the time-varying references. For simplicity, we explicitly divide the path constraints into: i) bounds on the states (2.6d) and controls (2.6e), with given lower an upper bound values; ii) nonlinear path constraints (2.6f).

To implement OCP (2.6) in an MPC fashion, one must also decide on a *sampling time* t_s with which it has to be solved. This choice must be a compromise between a t_s that is short enough to capture the evolution of the system's fast dynamics and long enough so that the necessary computations are carried out in time. In the following, we wrap up the principle of MPC:

1. get the current state estimate $\bar{x}_0$ from measurements
2. solve in real-time the parametric optimization problem (2.6)
3. inject the first control u_0 for time t_s into the system
4. repeat 1-3 after sampling time t_s

This principle is additionally illustrated in Figure 2.1, where the predicted state trajectories and the optimized control inputs at a certain sampling time are shown.

2.2 Direct optimal control

As mentioned earlier, NMPC is a feedback control strategy based on the solution of finite-horizon OCPs. There are three main classes of methods that one can use to address continuous-time OCPs: (a) *dynamic programming* (DP), (b) indirect methods, and (c) direct methods. Despite seemingly unrelated, these three classes have much more in common than was initially thought. As we will see throughout this chapter, the optimality conditions for many direct methods have a deep-seated relationship with the early two. In the following, we briefly outline their derivation.

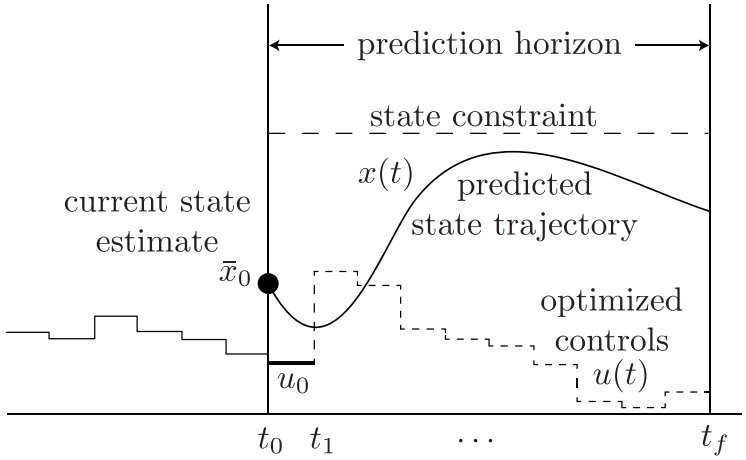


Figure 2.1. Illustration of the model predictive control principle.

DP seeks a solution to the *Hamilton-Jacobi-Bellman* (HJB) *equation* [28], which provides the necessary optimality conditions for continuous-time OCPs. The solution is related to the optimal cost of a *value function*, or *cost-to-go function*, which satisfies the HJB partial differential equation but requires the enumeration of the continuous state and control spaces for each sampled state. The solution is iteratively approached by first performing a backward recursion to compute all functions, starting from the end of the sampled horizon, reducing the step by one until the beginning of it, and then performing a forward recursion that selects the optimal control action to the given state. This process allows the reconstruction of the optimal trajectory through forward simulation, starting from the current state estimate, thanks to *Bellman's principle of optimality*, which alleges that each subtrajectory of an optimal trajectory is an optimal trajectory as well. Advantages of DP include the solution of the OCP up to global optimality, cheap forward simulations, and the computation of a feedback control law in closed form and not just a trajectory. On the other hand, if it shall be applied to systems with continuous state and control of moderate dimensions – as commonly found in robotics – computational complexity and memory requirements increase exponentially due to the large-scale state-space tabulation. This issue is what Bellman called *the curse of dimensionality*.

Indirect methods use *Pontryagin's maximum principle* [16] (also referred to as the *minimum principle* when convenient) to set up the necessary optimality conditions for continuous-time OCP. These conditions lead to the definition of a *multi-point boundary value problem* (MPBVP), which is, in other words, a set of *differential-algebraic equations* (DAEs) that are iterativelly solved, typically by a Newton method. Here, the optimal trajectory can be reconstructed through forward simulation once the initial value for the adjoint equations is known, a guesstimate that is rather non-trivial. A great advantage of indirect methods compared to direct methods (described next) is that they offer solutions with high accuracy, where accuracy refers to the selected accuracy region of the applied integration method [29]. However, many drawbacks arise when they are employed in a practical context in robotics. Some examples are: i) the need for expert knowledge on indirect methods;

ii) the derivation of the MPBVP is often analytically tedious or intractable [30], iii) the solution of the higher index DAE system itself is problematic; iv) the elimination of all controls through algebraic manipulation before solving the MPBVP may not always be straightforward or even possible.

Direct methods transcribe the continuous-time, infinite-dimensional OCP into a finite-dimensional *nonlinear program* (NLP) for which the necessary optimality conditions are set up and numerically solved using an NLP solver. While indirect methods are sketched as *first-optimize-then-discretize* methods since the optimality conditions are first written in continuous-time and then discretized, direct methods are instead sketched as *first-discretize-then-optimize* methods. In general, they all rely on a piecewise constant parametrization for the control trajectory, while the state trajectory is treated according to the discretization method employed. Among the advantages of direct methods are their flexible implementation to different robotic systems classes and their rather upfront handling of inequality constraints. Disadvantages include that solutions are only locally optimal, and only open-loop controls can be determined [29]. Two major discretization approaches can be distinguished within the direct methods class: (a) sequential and (b) simultaneous. The *direct single shooting* is a sequential approach where the variable space of the NLP is reduced by eliminating all states, except the initial one, through a forward simulation. In this way, the optimization problem is solved in the space of control parameters and initial state only [6], which confers an advantage to this approach as the reduced problem has much fewer variables. However, because of the usual long time-span associated with the numerical integration (from the beginning until the end of the time horizon), the nonlinearity propagation tends to increase, making direct single shooting not ideal for highly nonlinear or unstable systems. To overcome this issue, *direct multiple shooting* is proposed by Bock and Plitt [31]. At the price of a heavier computational demand, the time horizon is split into a set of smaller intervals in a time grid (usually equidistant). Each interval can be considered an independent single shooting approach with the addition of continuity constraints. Direct multiple shooting is a simultaneous approach that generally shows a much faster convergence. It is preferred for highly nonlinear or unstable systems [32] as it avoids creating a very nonlinear map in the NLP optimality conditions. Another successful simultaneous approach is *direct collocation*. The approach's key idea is to include the variables of all mesh points involved in the numerical integration (states, their derivatives, and controls) as NLP variables. In most cases, this idea is implemented using the orthogonal collocation method. In particular, each state's trajectory per interval, therein called collocation interval, is approximated via Lagrange interpolation polynomial [33].

There are several reasons why this thesis believes in the employment of direct methods to solve constrained OCPs arising in (N)MPC formulations for real-time motion generation in robotic systems:

1. There is ample room for online strategies with direct methods. The necessary optimality conditions provide a lot of information on how to approach the solution. This information can be used to build online strategies that benefit from the plethora of algorithms developed for real-time optimal control.

2. Shooting methods render the initial NLP a block-structured NLP. Structure-exploiting NLP solvers can leverage this particular structure to provide speed-ups in solution times.

3. Direct methods tend to be more accurate as the grid becomes progressively denser, making them suitable for applications requiring a moderate accuracy solution.

   Remark 1. *In such applications, it is of paramount importance to find a trade-off between computational efficiency and accuracy, as denser grids can lead to unrealistic solution times.*

For all reasons above, this thesis focuses on direct methods for addressing real-time optimal control. More specifically, direct multiple shooting is the chosen approach.

2.2.1 Direct multiple shooting

The previous section established that this thesis would employ direct multiple shooting to address continuous-time OCPs. This means that OCP (2.6) will be transcribed into a finite-dimensional NLP, which stands for a discrete-time approximation of the original problem. This section will focus on the parameterizations required by direct multiple shooting, which will be used throughout this thesis.

Shooting intervals Let us first define an equidistant grid with $N+1$ *shooting nodes*

$$t_0 < t_1 < \cdots < t_N = t_f, \tag{2.7}$$

which split the prediction horizon into N *shooting intervals*, such that we have the discrete-time step $\Delta t = (t_N - t_0)/N$. Note that one can also consider a non-equidistant grid.

Control parameterization As in most direct methods, the direct multiple shooting requires a finite-dimensional vector parameterization $q \in \mathbb{R}^{n_q}$ for the control trajectory $u(t)\,\forall t \in [t_0, t_N)$ defined on the time horizon of interest. To preserve the inherent sparsity of the problem, we use a simple piecewise constant control parameterization

$$u(\pi) = q_i, \quad \pi \in [t_i, t_{i+1}). \tag{2.8}$$

State parameterization Direct multiple shooting considers the states $s_i \approx x(t_i)$ at the shooting intervals' boundary points as NLP decision variables. Thus, the approximation of the state trajectory $x(\pi)$ at each shooting interval is the numerical solution of the *initial value problems* (IVPs)

$$\dot{x}(\pi) = f(x(\pi), q_i), \quad x(t_i) = s_i, \quad \pi \in [t_i, t_{i+1}], \tag{2.9}$$

for $i = 0, \ldots, N-1$ and starting at $x(0) = \bar{x}_0$. To constraint the artificial degrees of freedom introduced by the time horizon splitting, *continuity constraints* are imposed

$$s_{i+1} = x_{i+1}(t_i; s_i, q_i), \quad i = 0, \ldots, N-1. \tag{2.10}$$

Therein, $x_{i+1}(\cdot)$ denotes the numerical solution of function (2.9) given the value of $x(t_i) = s_i$. For convenience, we rewrite function (2.10) as

$$s_{i+1} = F(s_i, q_i), \quad i = 0, \dots, N-1, \tag{2.11}$$

where $F(\cdot)$ represents the discrete-time system dynamics that will be used throughout this thesis as the *integrator*. In particular, we use one step of *explicit Runge Kutta 4th order* (ERK4) per shooting interval. Despite the practical appeal of the *Euler method* (EM) due to its simplicity, high-order methods are preferred in this thesis, as they *converge* faster given the moderate accuracy required by real-time NMPC algorithms. Convergence is the property that indicates how fast an integrator converges as N increases. A method is said to be convergent with order m if $s_i \to x(t_i)$ as $N \to \infty$. In other words, for every ODE (2.1), with a Lipschitz function f, and every $t > 0$

$$\lim_{N\to\infty} \max_{i=0,\dots,N} \|s_i - x(t_i)\| = \mathcal{O}(\Delta t^m). \tag{2.12}$$

Suppose we want to simulate an ODE with a certain desired accuracy ε, considering first- and fourth-order methods (e.g., EM and ERK4). It follows that to achieve the accuracy ε, the fourth-order method needs four times less number of steps N compared to the first-order method, even if a fourfold cost per RK step is required which turns out to be significantly cheaper than the first-order method. This example is also illustrated in Figure 2.2. This desirable property makes ERK4 a fast and numerically stable candidate for an integrator in direct multiple shooting algorithms if one works with short simulation times and nonstiff systems of ODEs. For problems in which the system is known to be described by stiff ODEs a priori, the reader is advised to use *implicit integrators*. We refer to the studies in [34, 35] for detailed information about them.

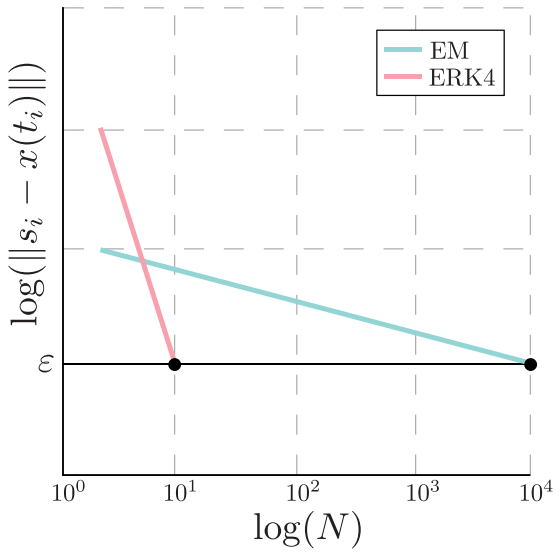


Figure 2.2. Illustration showing that ERK4 is cheaper than EM for desired accuracy ε.

Path constraints and terminal constraint For simplicity, the linear and nonlinear path constraints are defined on the same refined shooting grid

$$x_i^l \leq s_i \leq x_i^u, \qquad i = 1, \ldots, N-1, \tag{2.13}$$
$$u_i^l \leq q_i \leq u_i^u, \qquad i = 0, \ldots, N-1, \tag{2.14}$$
$$g_i^l \leq G_i(s_i, q_i) \leq g_i^u, \qquad i = 0, \ldots, N-1, \tag{2.15}$$

which, in practice, often guarantees constraint satisfaction in-between shooting nodes if a reasonable choice of the discretization step size is made. Moreover, the terminal constraint is defined as

$$g_N^l \leq G_N(s_N) \leq g_N^u. \tag{2.16}$$

Nonlinear programming formulation After discretizing the cost function (2.6a) on the same shooting grid, we obtain the *large-scale* but *sparse* NLP:

$$\min_{\substack{s_0,\ldots,s_N,\\ q_0,\ldots,q_{N-1}}} \quad \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(s_i, q_i) - \tilde{\eta}\|_W^2 + \frac{1}{2}\|\eta_N(s_N) - \tilde{\eta}_N\|_{W_N}^2 \tag{2.17a}$$
$$\text{s.t.} \quad s_0 - \bar{x}_0 = 0, \tag{2.17b}$$
$$s_{i+1} - F(s_i, q_i) = 0, \qquad i = 0, \ldots, N-1, \tag{2.17c}$$
$$x_i^l \leq s_i \leq x_i^u, \qquad i = 1, \ldots, N-1, \tag{2.17d}$$
$$u_i^l \leq q_i \leq u_i^u, \qquad i = 0, \ldots, N-1, \tag{2.17e}$$
$$g_i^l \leq G_i(s_i, q_i) \leq g_i^u, \qquad i = 0, \ldots, N-1, \tag{2.17f}$$
$$g_N^l \leq G_N(s_N) \leq g_N^u, \tag{2.17g}$$

with state trajectory $s := (s_0, \ldots, s_N)$ and input trajectory $q := (q_0, \ldots, q_{N-1})$, where $s_i \in \mathbb{R}^{n_x}$ and $q_i \in \mathbb{R}^{n_u}$. For convenience, variables W_s, W_f, g_f^l, g_f^u, $\tilde{\eta}_f$, t_f of OCP (2.6) are renominated to W, W_N, g_N^l, g_N^u, $\tilde{\eta}_N$, N in NLP (2.17) respectively. Figure 2.3 illustrates a continuous-time OCP solution using a discretization based on the direct multiple shooting approach. The state trajectory becomes continuous only when the NLP solution is reached, where the continuity constraints are enforced.

2.3 Nonlinear programming

To fully understand how to deal with the most general forms of real-time optimal control problems, it is essential to have a thorough grasp of nonlinear optimization's theoretical concepts. To that end, we stack the optimization variables of NLP (2.17) into a vector

$$w = (s_0; q_0; \ldots; q_{N-1}; s_N), \tag{2.18}$$

such that we have the following compact NLP:

$$\min_w \quad \varphi(w) \tag{2.19a}$$
$$\text{s.t.} \quad G(w) = 0, \tag{2.19b}$$
$$H(w) \leq 0, \tag{2.19c}$$

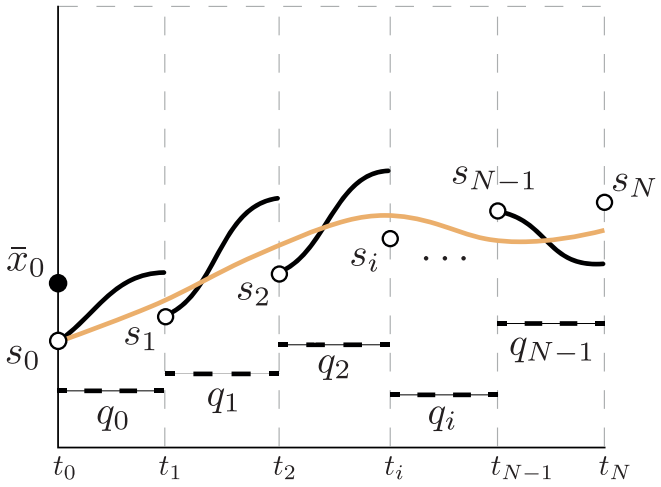


Figure 2.3. Illustration of the direct multiple shooting approach. A numerical integrator builds the simulations $x_{i+1}(t_i; s_i, q_i)$ for each interval $[t_i, t_{i+1}]$. In yellow the state trajectory at the solution, where the continuity constraints are satisfied.

where $\varphi : \mathbb{R}^{n_w} \to \mathbb{R}$ is the cost function, $G : \mathbb{R}^{n_w} \to \mathbb{R}^{n_g}$ and $H : \mathbb{R}^{n_w} \to \mathbb{R}^{n_h}$ represent the vectors that group equality and inequality constraints respectively. Furthermore, we assume the following:

Assumption 1. *Functions $\varphi(w)$, $G(w)$ and $H(w)$ are twice continuously differentiable.*

2.3.1 Optimality conditions

When dealing with nonlinear programming for real-time optimal control problems, one wants to know if a feasible point $\bar{w} \in \Omega$ satisfies the necessary optimality conditions. In case it does satisfy these conditions, $\bar{w}$ is a candidate local minimizer for NLP (2.19). We proceed with some basic definitions.

Definition 1 (Feasible set)**.** *The feasible set Ω is the set that collects all points satisfying the constraints*

$$\Omega := \{w \in \mathbb{R}^{n_w} : G(w) = 0, H(w) \leq 0\}.$$

Definition 2 (Feasible point)**.** *The point $\bar{w} \in \mathbb{R}^{n_w}$ is a feasible point iff the constraints*

$$G(\bar{w}) = 0, \quad H(\bar{w}) \leq 0$$

are satisfied.

The points we are interested in are the feasible points that locally minimize the cost function $\varphi(w)$.

Definition 3 (Local minimizer)**.** *The point $w^\star \in \mathbb{R}^{n_w}$ is a local minimizer iff $w^\star \in \Omega$ and there exists a neighborhood $\mathcal{N}$ of $w^\star$ (i.e., an open ball around $w^\star$), so that*

$$\forall w \in \Omega \cap \mathcal{N} : \varphi(w) \geq \varphi(w^\star),$$

where $\varphi(w^\star)$ is a local minimum.

To verify whether $w^\star$ is a local minimizer or not, we need to characterize the set $\mathcal{N}$. This set, however, depends on the inequality constraints active locally.

Definition 4 (Active constraint)**.** *An inequality constraint is termed active at a feasible point $\bar{w} \in \Omega$ iff*

$$H_i(\bar{w}) = 0.$$

Definition 5 (Active set)**.** *The active set is an index set*

$$\mathcal{A}(\bar{w}) \subset \{1, \ldots, n_h\},$$

where the indices specify the inequality constraints that are active.

The concepts above are important because one can only state the *first-order necessary conditions* (FONC) for optimality if a *constraint qualification* (CQ) is established. Normally, the *regularity concepts* are referred to as QC and are formally stated as follows.

Definition 6 (LICQ)**.** *The linear independence constraint qualification (LICQ) holds at a feasible point $\bar{w} \in \Omega$ iff all vectors $\nabla G_i(\bar{w})$ for $i \in 1, \ldots, n_g$ and $\nabla H_i(\bar{w})$ for $i \in \mathcal{A}(\bar{w})$ are linearly independent.*

If we stack all equality and inequality functions in a column vector, then the LICQ becomes equivalent to the full-row rank Jacobian matrix

$$\nabla \tilde{G}_i(\bar{w}) = \begin{pmatrix} \nabla G_i(\bar{w}) \\ \nabla H_i(\bar{w})(i \in \mathcal{A}(\bar{w})) \end{pmatrix}. \tag{2.20}$$

We now define the Lagrangian, an essential function in nonlinear optimization.

Definition 7 (Lagrangian function)**.** *The Lagrangian function for NLP* (2.19) *is defined as*

$$\mathcal{L}(w, \lambda, \mu) := \varphi(w) + \lambda^T G(w) + \mu^T H(w).$$

We call the $\lambda \in \mathbb{R}^{n_g}$ and $\mu \in \mathbb{R}^{n_h}$ above Lagrange multipliers, or dual-variables.

To give further meaning to the Lagrangian, let us now formulate the famous KKT conditions, derived initially by Karush [36], and Kuhn and Tucker [37].

Theorem 1 (Karush-Kuhn-Tucker (KKT) conditions)**.** *Let $w^\star$ be a local minimizer of NLP* (2.19) *for which the LICQ holds. Then, there must exist Lagrange multiplies $\lambda^\star$ and $\mu^\star$ for which*

$$\nabla_w \varphi(w^\star) + \nabla_w G(w^\star)\lambda^\star + \nabla_w H(w^\star)\mu^\star = 0 \qquad =: \nabla_w \mathcal{L}(w, \lambda, \mu) \tag{2.21a}$$

$$G(w^\star) = 0 \qquad =: \nabla_\lambda \mathcal{L}(w, \lambda, \mu) \tag{2.21b}$$

$$H(w^\star) \leq 0 \qquad =: \nabla_\mu \mathcal{L}(w, \lambda, \mu) \tag{2.21c}$$

$$\mu^\star \geq 0 \tag{2.21d}$$

$$\mu_i^\star H_i(w^\star) = 0, \quad i = 1, \ldots, n_h, \tag{2.21e}$$

holds. If some $(w^\star, \lambda^\star, \mu^\star)$ satisfy the KKT conditions, then it is also a solution to the primal and dual problems.

Proof. See Section 12.3 in Nocedal and Wright [38]. □

In nonlinear optimization, Eq. (2.21a) denotes the *stationarity condition*, Eq. (2.21b) and (2.21c) the *primal feasibility*, Eq. (2.21d) the *dual feasibility*, and Eq. (2.21e) the *complementary slackness*. In particular, one obtains the so-called *complementarity conditions* by lumping together the last three KKT conditions, i.e., (2.21c)–(2.21e). They define a nonsmooth L-shaped manifold in which μ_i and H_i are allowed to live at the solution. When

$$\mathcal{A}_0(w^\star) = \{i : H_i(w^\star) = 0, \mu_i^\star = 0\}, \tag{2.22}$$

we have the case of *weakly active constraints*, which is often not desirable due to the nondifferentiability at the origin. Conversely, when

$$\mathcal{A}_+(w^\star) = \{i : H_i(w^\star) = 0, \mu_i^\star > 0\}, \tag{2.23}$$

we have the case of *strictly active constraints*, a vital notion in nonlinear optimization. Note that the active set at the solution is defined by the union of both sets, i.e., $\mathcal{A}(w^\star) = \mathcal{A}_0(w^\star) \cup \mathcal{A}_+(w^\star)$. Figure 2.4 illustrates all these cases.

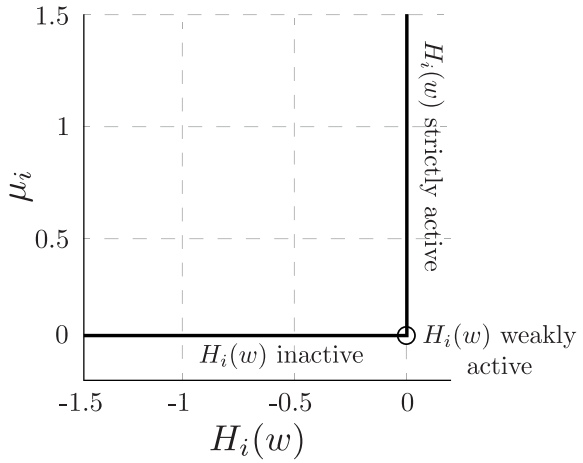


Figure 2.4. Illustration of the nonsmooth $\mu_i H_i(w) = 0$ manifold resulting from the KKT conditions.

Strict complementarity is, as we shall see in Section 2.3.3, a crucial concept to ensure some key properties of numerical methods. It can be formally define as follows.

Definition 8 (Strict complementarity)**.** *Let $(w^\star, \lambda^\star, \mu^\star)$ be a KKT point. Strict complementarity holds at this KKT point iff all active constraints are strictly active.*

This thesis is interested in solving strictly convex NLPs. In this particular case, the KKT conditions are *necessary* and *sufficient* to ensure that there exists at most one local minimizer of the optimization problem at hand. We refer to excellent numerical optimization textbooks [39, 38, 40] for more details about general non-convex problems, where complex *second-order sufficient conditions* (SOSC) can be formulated.

2.3.2 Sequential quadratic programming

An OCP with a quadratic cost function and linear constraints is called a *quadratic program* (QP). These programs are of particular interest since they typically arise as subproblems in many nonlinear optimization methods, such as in the SQP method. The SQP method approximates the general NLP (2.19) by a QP through successive linearizations of functions $\varphi(w)$, $G(w)$, and $H(w)$. Starting with an initial guess (w_0, λ_0, μ_0), a full step SQP iteration is performed at each iteration k as follows:

$$w_{k+1} = w_k + \alpha \Delta w_k, \quad \lambda_{k+1} = \lambda_k + \alpha(\lambda_k^{\mathrm{QP}} - \lambda_k), \quad \mu_{k+1} = \mu_k + \alpha(\mu_k^{\mathrm{QP}} - \mu_k), \tag{2.24}$$

where $\alpha \in (0, 1]$ is the *step length* determined using a globalization strategy [38], and $(\Delta w_k, \lambda_k^{\mathrm{QP}}, \mu_k^{\mathrm{QP}})$ is the solution of the QP subproblem:

$$\begin{aligned}
\min_{\Delta w} \quad & \nabla_w \varphi(w_k)^T \Delta w + \frac{1}{2} \Delta w^T B_k \Delta w && && \text{(2.25a)}\\
\text{s.t.} \quad & G(w_k) + \nabla_w G(w_k)^T \Delta w = 0, && |\lambda^{\mathrm{QP}} && \text{(2.25b)}\\
& H(w_k) + \nabla_w H(w_k)^T \Delta w \leq 0. && |\mu^{\mathrm{QP}} && \text{(2.25c)}
\end{aligned}$$

Therein, $\nabla_w G(w_k)$ and $\nabla_w H(w_k)$ denote the constraint Jacobians, the matrix B_k denotes the *exact Hessian* of the Lagrangian $\nabla_w^2 \mathcal{L}(w_k, \lambda_k, \mu_k)$, and $\nabla \varphi(w_k)$ is the gradient of the cost function. This thesis will focus on SQP methods to solve strictly convex problems resulting from real-time optimal control applied to robotic systems. This is motivated by their warm-starting capabilities and because SQP methods can satisfy the linearized versions of the inequality constraints (2.19c).

Assumption 2. *Because in real-time optimization, the previous iterate generally provides a sufficiently good initial guess to warm-start the subsequent problem, globalization strategies such as trust-region [41], line-search [42], or filter methods [43] are hereafter neglected. Therefore, we assume $\alpha = 1$. This assumption will prove of particular importance in the context of real-time NMPC algorithms in Section 2.5.*

2.3.3 Newton-type optimization

Newton-type optimization can be interpreted as an SQP method where at each iteration, the primal-dual iterate $(\Delta w, \lambda_k^{\mathrm{QP}}, \mu_k^{\mathrm{QP}})$ is found and used to warm-start the subsequent QP subproblem. To comprehend this meaningful parity, let us first regard the KKT conditions for the equality-constrained version of NLP (2.19) based on Theorem 1

$$\begin{aligned}
\nabla_w \varphi(w) + \nabla_w G(w) \lambda = 0 \quad & =: \nabla_w \mathcal{L}(w, \lambda) && \text{(2.26)}\\
G(w) = 0. \quad & =: \nabla_\lambda \mathcal{L}(w, \lambda) && \text{(2.27)}
\end{aligned}$$

Given a continuously differentiable function $R : \mathbb{R}^{n_r} \to \mathbb{R}^{n_r}$, $z : (w, \lambda) \mapsto R(z)$, we aim to solve the nonlinear system of equations $R(z) = 0$ using Newton's method. We proceed by recursively generating a sequence of iterates $\{z\}_{k=0}^{\infty}$, starting from an initial guess z_0. An *exact Newton iteration*, or *Newton direction*, at the current iterate can be explicitly written as

$$\nabla_z R(z_k) \overbrace{(z - z_k)}^{\Delta z} = -R(z_k). \tag{2.28}$$

Applying (2.28) to the KKT conditions (2.26)–(2.27), we obtain the following exact Newton iteration

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}(w_k, \lambda_k) & \nabla_w G(w_k) \\ \nabla_w G(w_k)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta \lambda \end{pmatrix} = - \overbrace{\begin{pmatrix} \nabla_w \mathcal{L}(w_k, \lambda_k) \\ \nabla_\lambda \mathcal{L}(w_k, \lambda_k) \end{pmatrix}}^{R(w_k, \lambda_k)}. \tag{2.29}$$

Using the definitions in (2.26), the above system is equivalent to

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}(w_k, \lambda_k) & \nabla_w G(w_k) \\ \nabla_w G(w_k)^T & 0 \end{pmatrix} \begin{pmatrix} w - w_k \\ \lambda - \lambda_k \end{pmatrix} = - \begin{pmatrix} \nabla_w \varphi(w_k) + \nabla_w G(w_k) \lambda_k \\ G(w_k) \end{pmatrix}. \tag{2.30}$$

Observe that contributions depending on the current multiplier λ_k cancel each other, yielding

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}(w_k, \lambda_k) & \nabla_w G(w_k) \\ \nabla_w G(w_k)^T & 0 \end{pmatrix} \begin{pmatrix} w - w_k \\ \lambda \end{pmatrix} = - \begin{pmatrix} \nabla_w \varphi(w_k) \\ G(w_k) \end{pmatrix} \tag{2.31}$$

and, hence,

$$\begin{pmatrix} w_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} w_k \\ 0 \end{pmatrix} - \begin{pmatrix} \nabla_w^2 \mathcal{L}(w_k, \lambda_k) & \nabla_w G(w_k) \\ \nabla_w G(w_k)^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla_w \varphi(w_k) \\ G(w_k) \end{pmatrix}. \tag{2.32}$$

Note that, from one side, one can now compute the next multiplier λ_{k+1} directly. However, from another side, the exact Hessian matrix still carries the dependency on λ_k. The last observation has motivated the development of a broader class of Newton-type optimization approaches, where methods to approximate the Hessian matrix $B_k \approx \nabla_w^2 \mathcal{L}(w_k, \lambda_k)$ are considered. Using such an approximation, one obtains a *general Newton-type iteration* or *search direction* as

$$\begin{pmatrix} w_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} w_k \\ 0 \end{pmatrix} - \begin{pmatrix} B_k & \nabla_w G(w_k) \\ \nabla_w G(w_k)^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla_w \varphi(w_k) \\ G(w_k) \end{pmatrix}. \tag{2.33}$$

Newton-type optimization algorithms differ in how approximate the Hessian matrix B_k in (2.33), a choice that has profound implications for local convergence and computation time, as we will briefly describe below. In summary, we can observe that Newton-type iterations in the KKT conditions (2.26)–(2.27) can be interpreted as solving equality-constrained QP approximations of the original NLP for a given fixed active set [38].

Hessian approximations

The difference between the exact Hessian and the Hessian approximation in the KKT matrix, i.e., $\delta b_k = B_k - \nabla_w^2 \mathcal{L}(w_k, \lambda_k)$, determines the local convergence rate on a Newton-type optimization algorithm based on the SQP method. Exact Hessian-based SQP methods rely on the computation of second-order derivatives, i.e., $B_k := \nabla_w^2 \mathcal{L}(w_k, \lambda_k)$, which can be rather expensive [44]. Their main advantage is that, because of the more sophisticated Hessian matrix, they have a *quadratic convergence rate* in a neighborhood of the optimal solution $w^\star$, when the iterate is sufficiently close to $w^\star$.

Definition 9 (Q-quadratic convergence rate)**.** *Assume $w_k \in \mathbb{R}^{n_w}$, $w_k \to w^\star$ for $k \to \infty$. The sequence w_k converges Q-quadratically if there exists a constant $C_q \in (0, \infty)$ such that*

$$\lim_{k\to\infty} \frac{\|w_{k+1} - w^\star\|}{\|w_k - w^\star\|^2} \leq C_q.$$

Exact Hessian-based SQP was successfully employed to solve a time-optimal race car driving problem in [45]. Other exact Hessian-based applications in the context of robot motion generation are found in [46, 47].

Another widely used technique is based on Quasi-Newton Hessian update, particularly the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) update [38]. In this case, B_k is iteratively approached by evaluating the Jacobian $\nabla_w \mathcal{L}$ at distinct points so that the curvature information is retrieved, enabling the estimation of $\nabla_w^2 \mathcal{L}(w_k, \lambda_k)$ at a considerably reduced computational cost. As the iterations proceed, the Hessian approximation B_k becomes increasingly similar to the exact Hessian, i.e., $B_k \to \nabla_w^2 \mathcal{L}(w_k, \lambda_k)$, in the relevant directions. In principle, the BFGS update technique has a *superlinear convergence rate* in a neighborhood of the optimal solution $w^\star$.

Definition 10 (Q-superlinear convergence rate)**.** *Assume $w_k \in \mathbb{R}^{n_w}$, $w_k \to w^\star$ for $k \to \infty$. The sequence w_k converges Q-superlinearly if*

$$\lim_{k\to\infty} \frac{\|w_{k+1} - w^\star\|}{\|w_k - w^\star\|} = 0.$$

However, in practice, the authors in [48] point out that this technique suffers from fluctuating convergence rates due to jumps in the state, associated with external noise. In these circumstances, the Hessian approximation proves to be poor, so that many BFGS updates are needed to recover a reasonable convergence rate, making it unadvisable in the context of real-time optimal control. Despite the drawback, it is possible to find recent studies in which the BFGS update performed very favorably, for example, in [49, 50, 51, 52].

In addition to the SQP methods based on the exact Hessian and BFGS update, another widespread technique is the *Generalized Gauss-Newton* (GGN) method, which we employ in this thesis. This technique grants a multiplier-free Hessian approximation within the Newton-type optimization algorithm, relinquishing the need for a good initialization. The GGN method is applicable when the cost function explicitly takes the form of (nonlinear) least squares. To illustrate, let us rewrite the cost function of the equality-constrained version of NLP (2.19) in a more general form

$$\varphi(w) = \frac{1}{2}\|\mathcal{F}(w)\|_2^2, \tag{2.34}$$

where $\mathcal{F} : \mathbb{R}^{n_w} \to \mathbb{R}^m$ is a smooth function called *residual*.

In the GGN method, the Hessian B_k is approximated by first-order derivatives, i.e., $B_k := \nabla_w \mathcal{F}(w_k) \nabla_w \mathcal{F}(w_k)^T \approx \nabla_w^2 \mathcal{L}(w_k, \lambda_k)$, underpinned by the assumption that nonlinearities are sufficiently small. In other words, given the exact second-order derivative

$$\nabla_w^2 \mathcal{L}(w, \lambda) = \nabla_w \mathcal{F}(w_k) \nabla_w \mathcal{F}(w_k)^T + \sum_{i=1}^{m} \mathcal{F}_i(w) \nabla_w^2 \mathcal{F}_i(w) + \sum_{i=1}^{n_g} \lambda_i \nabla_w^2 G_i(w), \tag{2.35}$$

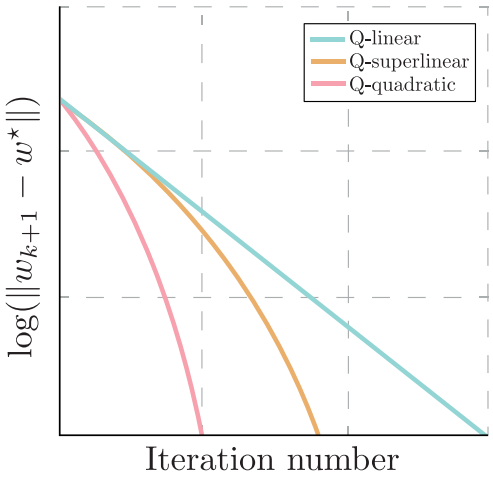


Figure 2.5. Illustration showing the convergence rates of Newton-type optimization algorithms in the neighborhood of the local minimizer $w^\star$.

the terms $\nabla_w^2 \mathcal{F}_i(w)$, $\nabla_w^2 G_i(w)$ are ignorably small in practice. For a linear regression model $\mathcal{F}(w)$ or small residual problems (with small residuals at the solution), the GGN method performs well and with a *linear convergence rate* once the iterate w_k has fallen into a small neighborhood around the solution $w^\star$.

Definition 11 (Q-linear convergence rate)**.** *Assume $w_k \in \mathbb{R}^{n_w}$, $w_k \to w^\star$ for $k \to \infty$. The sequence w_k converges Q-linearly if there exists a constant $C_l \in (0, 1)$ such that*

$$\lim_{k \to \infty} \frac{\|w_{k+1} - w^\star\|}{\|w_k - w^\star\|} \leq C_l.$$

This is typically the case in tracking formulations of optimal control problems, where the residuals at the solution tend to zero [48]. In principle, exact Hessian-based SQP methods are expected to perform better than those with a Gauss-Newton Hessian approximation due to the former's sophisticated Hessian matrix. However, in practical applications, where the current state estimate is highly corrupted by noise, the Gauss-Newton Hessian approximation is preferred to the exact Hessian. It turns out that, due to the increased noise, the inclusion of the second-derivative term can be destabilizing if the model fits badly [53].

Because of its properties, much interest in the GGN method has been roused in the robotics community; see, e.g., [54, 55, 56]. In [57, 4, 2], the GGN method was considered in the underlying optimization problems arising in real-time NMPC controllers. The interested reader can find a comprehensive review of the Gauss-Newton Hessian approximation in [58]. Moreover, Figure 2.5 illustrates the outline on local convergence rates in Newton-type optimization algorithms based on SQP method. Detailed definitions are provided in [38].

Inequality constrained optimization

Newton-type optimization algorithms can be extended to the case where inequality constraints need to be considered if one establishes a way to treat them in the KKT

conditions (2.21) [48]. The advances in optimization strategies have enabled the development of a mature class of optimization algorithms for this purpose. Among others, two main classes of algorithms stand out: IP methods and SQP methods.

In IP methods, the idea is to substitute the complementarity condition (2.21e) with a smooth approximation, typically a hyperbola $\mu_i^\star H_i(w^\star)+\tau = 0$, $i = 1, \ldots, n_h$. The *barrier parameter* $\tau > 0$ is introduced in the KKT conditions so that for a small value of τ, the L-shaped set is closely approximated by the hyperbola and vice-versa. The resulting KKT system can be seen as a root-finding problem under the form of $R(z, \tau) = 0$ where the primal-dual solution $(w^\star(\tau), \lambda^\star(\tau), \mu^\star(\tau))$ is iteratively approached by generating a sequence of positive barrier parameters for which $\tau \to 0$. One can also interpret this problem as solving the KKT conditions of a barrier problem, typically referred to as the *primal barrier method* or *log-barrier method*. As the name suggests, the inequality constraints are handled by adding a *log-barrier term* to the cost function, i.e., $\varphi(w) - \tau \sum_{i=1}^{n_h} \log(-H_i(w))$ where τ penalizes the inequality constraints as they approach the boundaries of the feasible set. IP methods have been successfully applied to solve motion generation problems, including moving through narrow passages [59], motion refinement [60, 61], engineering design optimization for biomimetic robotics [62], and grasping force optimization of multi-fingered robotic hands [63, 64]. One widespread and successful implementation of an IP method is in the open-source code `IPOPT` [65], which is used to solve general NLPs.

Conversely, SQP methods solve an inequality-constrained QP at each iteration, resulting from the linearization of the original objective and constraint functions. From the current perspective, to view this method as a Newton-type method, we need to assume that strict complementarity holds in a neighborhood $\mathcal{N}$ of $w^\star$. Then, if the SQP iterates w_k are sufficiently close to $w^\star$, the solution of the equality-constrained NLP with active constraints also satisfies the sufficient conditions of the QP [66]. The fast local convergence, typical of Newton-type methods, can also be observed in SQP methods. Unlike IP methods, which have not shown any substantial benefit from a good initial guess as they work by following the *central path* [67], the natural warm-starting capabilities of SQP methods make them especially popular for embedded numerical optimization [68]. For these reasons, the method remains a prominent candidate to solve many motion generation problems such as gait generation for bipedal robots [69, 12], target chasing mission in cluttered environments [70, 71], dynamic locomotion for quadruped robots [72, 15, 73], and multi-robot motion optimization [74, 75].

The following section will provide a compact and, thereby, an incomplete overview of how an inequality-constrained QP is actually solved. Although incomplete, the section will present methods that will cover most cases of interest.

2.4 Tailored quadratic programming solvers for optimal control

The fact that many robotics problems can be formulated as a QP has spurred widespread use and development of SQP methods. However, the methods' success critically depends on efficient and reliable QP solvers, capable of capitalizing on

warm-starts and handling QPs with an indefinite Hessian matrix. In this spirit, researchers built up a substantial class of numerical algorithms for solving QP subproblems, such as *dual Newton strategy*, *active-set method*, and *structure-exploiting interior-point method*. Yet, QP solvers are still a very active research area, and the situation is not at all clear [66]. This is especially true in the context of real-time optimal control, where the time available to solve a QP subproblem is rather tight.

This thesis is interested in fast solutions for QP subproblems arising from applying an SQP method to the direct multiple shooting-type NLP (2.17) with the Gauss-Newton Hessian approximation. Given these considerations, the resulting *large* and *sparse* QP subproblem reads as follows:

$$
\begin{aligned}
\min_{\substack{\Delta s_0,\dots,\Delta s_N,\\ \Delta q_0,\dots,\Delta q_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1}\begin{pmatrix}\Delta s_i\\ \Delta q_i\end{pmatrix}^T B_i \begin{pmatrix}\Delta s_i\\ \Delta q_i\end{pmatrix} + \varphi_i^T \begin{pmatrix}\Delta s_i\\ \Delta q_i\end{pmatrix} + \\
& \frac{1}{2}(\Delta s_N^T B_N \Delta s_N) + \varphi_N^T \Delta s_N && && \text{(2.36a)}\\
\text{s.t.} \quad & \Delta s_0 = \bar{x}_0 - s_0^k, && && \text{(2.36b)}\\
& \Delta s_{i+1} - A_i \Delta s_i - b_i \Delta q_i - c_i = 0, && i = 0,\dots,N-1, && \text{(2.36c)}\\
& x_i^l - s_i \le \Delta s_i \le x_i^u - s_i, && i = 1,\dots,N-1, && \text{(2.36d)}\\
& u_i^l - q_i \le \Delta q_i \le u_i^u - q_i, && i = 0,\dots,N-1, && \text{(2.36e)}\\
& g_i^l \le G_i^s \Delta s_i + G_i^q \Delta q_i \le g_i^u, && i = 0,\dots,N-1, && \text{(2.36f)}\\
& g_N^l \le G_N^s \Delta s_N \le g_N^u, && && \text{(2.36g)}
\end{aligned}
$$

where the state and control deviations are defined as $\Delta s_i = s_i - s_i^k$, $\Delta u = u_i - u_i^k$ and the superscript k refers to the linearization point of the k^{th} SQP iteration. The linearization of the equality constraints in (2.17c) is defined by the *first-order sensitivities* $A_i := \nabla_s F(s_i^k, u_i^k)^T$, $b_i := \nabla_u F(s_i^k, u_i^k)^T$ and the affine term $c_i := F(s_i^k, u_i^k) - A_i s_i^k - b_i s_i^k$, responsible for triggering a first-order correction in the QP (2.36). Similarly, Eqs. (2.36d)–(2.36f) denote the linearized path constraints, while Eq. (2.36g) denotes the linearized terminal constraint. Finally, the stage Gauss-Newton Hessian blocks are denoted by B_i for $i = 0,\dots,N-1$, while the terminal one is denote by B_N.

The QP subproblem (2.36) has a particular block-structure that can be exploited, yielding a reduced complexity solution in the number shooting intervals [32]. It is of paramount importance to exploit this structure to obtain fast software implementations that may need to cope with resource-constrained hardware. For instance, one can use structure-exploiting algorithms to reduce the memory footprint of general QP solutions, especially as part of NMPC solutions.

This section reviews two different algorithms directly used in state-of-the-art QP solvers targeting real-time optimal control. We present some approaches for condensing the sparse QP and the situations where they may be favorable. We show that these approaches are competitive within the context of real-time motion generation, as they both benefit from rather recent research.

2.4.1 Active-set quadratic programming solvers

Suppose that we are interested in solving a strictly convex QP written in the following compact form

$$\min_{w \in \mathbb{R}^{n_w}} \quad \frac{1}{2} w^T B w + \varphi^T w \tag{2.37a}$$

$$\text{s.t.} \quad G^T w + g = 0, \tag{2.37b}$$

$$H^T w + h \leq 0, \tag{2.37c}$$

where $B \succ 0$ is the Hessian matrix, and $\varphi \in \mathbb{R}^{n_w}$ is a gradient vector. Active-set methods were originally developed as extensions of the simplex method with the aim of solving *linear programming* (LP) problems [76]. The main idea is to approximate the subset of the active constraints $\mathcal{A}(w^\star)$, if we were to know it in advance, to a *working set* $\mathcal{W}(w^\star)$ and directly solve the equality-constrained version of QP (2.37)

$$\min_{w \in \mathbb{R}^{n_w}} \quad \frac{1}{2} w^T B w + \varphi^T w \tag{2.38a}$$

$$\text{s.t.} \quad \tilde{g}(w) = 0. \tag{2.38b}$$

Here, for convenience the affine constraints are lumped together

$$\tilde{g}(w) = \begin{pmatrix} G^T w + g \\ H_{\mathcal{W}}^T w + h_{\mathcal{W}} \end{pmatrix}, \tag{2.39}$$

and the dependency on $w^\star$ is dropped. Since $\nabla \tilde{g}(w)$ is LICQ, there must exist a regular point w satisfying the equality constraints (2.38b). This point is the solution sought by the convex solver. Typically, the KKT conditions for this QP are set up

$$B w + \varphi + G\lambda + H_{\mathcal{W}} \mu_{\mathcal{W}} = 0 \tag{2.40}$$

$$G^T w + g = 0 \tag{2.41}$$

$$H_{\mathcal{W}}^T w + h_{\mathcal{W}} = 0 \quad \text{with } \mu = 0,\ (H^T w + h) < 0 \tag{2.42}$$

and solved using a Newton-type algorithm that generates a sequence of iterates where the search direction is

$$\begin{pmatrix} B & G & H_{\mathcal{W}} \\ G^T & 0 & 0 \\ H_{\mathcal{W}}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta \lambda \\ \Delta \mu_{\mathcal{W}} \end{pmatrix} = - \begin{pmatrix} \varphi \\ g \\ h_{\mathcal{W}} \end{pmatrix}. \tag{2.43}$$

Suppose the working set is unknown a priori (which is often the case). If so, the method iteratively refines it by changing one constraint at a time until the correct active-set is estimated and the optimal solution is reached [32]. In fact, $\mathcal{A}$ is uniquely defined at any feasible w, whereas there may be many choices for $\mathcal{W}$. The delimitation of the working set depends directly on the active-set method being employed [77]. Most existing methods fall into three classes: *primal*, *dual*, and *parametric methods*.

A conventional active-set method has two phases. In the first phase, a feasible point is found while the cost function is ignored; in the second phase, the cost

is minimized while feasibility is maintained [78]. The primal method (see, e.g., [38, 79, 78]) generates a sequence of feasible iterates for the primal inequalities, i.e., always satisfies (2.37c), until the dual inequalities are steered to zero, and an optimal solution is obtained. Conversely, the dual method (see, e.g., [80, 78, 81]) maintains the feasibility of the dual inequalities across iterations until primal inequalities are satisfied, and hence an optimal solution is obtained.

Differently from its counterparts, the parametric method is a real-time variant of the active-set method built on the expectation that the active set does not change much from one QP solve to the other, which is similar to the idea in SQP methods. More precisely, it is based on an affine-linear homotopy between a QP with a known solution and the QP to be solved [82]. This method took inspiration from the explicit or offline QP solution, but it was synthesized for the online context, e.g., real-time NMPC applications, ducking the prohibitive offline computational cost. For this reason, we now focus on its brief presentation.

Let $\tau \in [0, 1]$ parameterize the homotopy. We then rewrite QP (2.37) as the one-parametric family of QP we want to solve

$$\min_{w(\tau) \in \mathbb{R}^{n_w}} \quad \frac{1}{2} w^T(\tau) B w(\tau) + \varphi^T(\tau) w(\tau) \tag{2.44a}$$

$$\text{s.t.} \quad G^T w(\tau) + g(\tau) = 0, \tag{2.44b}$$

$$H^T w(\tau) + h(\tau) \leq 0. \tag{2.44c}$$

Affine-linear gradients and constraints on the homotopy path are defined starting at $\tau_0 \in [0, \tau_1]$ and ending at τ_1 such that

$$\varphi(\tau) = (1 - \tau)\varphi(\tau_0) + \tau\varphi(\tau_1) \tag{2.45}$$

$$g(\tau) = (1 - \tau)g(\tau_0) + \tau g(\tau_1) \tag{2.46}$$

$$h(\tau) = (1 - \tau)h(\tau_0) + \tau h(\tau_1) \tag{2.47}$$

hold. The central idea is to move along a homotopy path that traverses critical regions towards the current QP subproblem solution. Advantages of using this approach for real-time optimal control include: the first phase is unnecessary; the sequence of optimal solutions for QPs can be prematurely terminated after every partial step; the homotopy can be started at the current iterate towards the next [83]. These features will prove particularly valuable in Section 2.5. We refer the interested reader to [83] for a detailed presentation of a parametric active-set method.

As our ultimate goal is to achieve real-time capabilities for NMPC controllers, this thesis will use the open-source implementation of the parametric active-set method `qpOASES` [84], which employs the null-space method and dense linear algebra routines, in some of the proposed applications. It is worth mentioning that because of the dense factorizations implemented in `qpOASES`, the solver is not ideal for directly addressing the sparse QP (2.37). However, by eliminating the state variables using the so-called *condensing approach* [31], we can hand over a considerably smaller but dense QP to the solver, resulting in significant speed-ups.

Full condensing

When condensing is employed to a QP subproblem resulting from the application of direct multiple shooting, all state variables are eliminated through the continuity

constraints. The condensed QP no longer has a multiple shooting block-structure. Instead, it becomes similar in terms of variable space dimension and sparsity pattern to the QP in the direct single shooting approach. At this point, one can use any standard dense method for QP to approach the solution. Although the condensed QP is solved in the reduced variable space Nn_u rather than in the original space $(N+1)n_x + Nn_u$, the solution yet needs to be expanded to yield the original QP solution. The fact that the iterations are still performed in the original QP space constitutes the fundamental difference between this approach and the direct use of single shooting in the control variables' space [48].

Within an NMPC scheme, condensing must be performed at each sampling instant, consuming a significant part of the available computation time. After many years of thinking that the procedure's computational complexity and memory requirements grow with the prediction horizon length N as $\mathcal{O}(N^3)$ [85], authors [86, 87, 88] have shown algorithms for which they grow as $\mathcal{O}(N^2)$. Yet condensing is favorable if the problem's horizon length N and control input dimension n_u are relatively small compared to the state dimension n_x. One can note this more easily through the block-structured Hessian matrix. In the original sparse QP, the Hessian has $N(n_x + n_u)^2 + n_x^2$ elements, while, in the condensed QP, it has $(Nn_u)^2$. By simple inspection, it becomes apparent that the sparse approach leads to a factorization cost that is linear in N ($\mathcal{O}(N)$), in contrast to the quadratic cost ($\mathcal{O}(N^2)$) exhibited by superior approaches. Therefore, employing condensing in unfavorable circumstances can lead to an increase in memory footprint and solution times. The reader is referred to [89] for a detailed comparison of condensing algorithms.

2.4.2 Structure-exploiting interior-point method solvers

The motivation for IP methods comes from the desire to find algorithms with better theoretical properties than the simplex method [38]. Depending on the problem at hand, the simplex method – here understood as the "active-set extension" – can lend a rather poor performance or even fail. Unlike active-set methods that bear down an optimal working set and, hence, a solution through many inexpensive iterations starting at the boundary of the feasible set, IP methods approach a solution from its interior. At the price of a more expensive iteration, the method works its way down to an optimum and across the feasible set but never reaching its boundary. They can be divided into two broad classes: *primal barrier* and *primal-dual IP methods*.

The primal barrier method, outlined in Section 2.3.3, consists of adding a logarithmic term to the cost function to prevent feasible iterates from moving too close to the feasible set boundary. In this way, iterates remain *strictly feasible* throughout the process. This can be achieved by minimizing the cost using some Newton-type algorithm that keeps the iterates strictly positive while ensuring a decrease at each search direction. Such a method has been recently revisited in [90, 91, 92]. On the other hand, the primal-dual method has distinguished itself as the most efficient practical approach [38], becoming the basis for much of the current generation of structure-exploiting QP solvers. Therefore, we focus on its concise presentation in what follows.

Let us consider a generic sparse QP in the form

$$\min_{w \in \mathbb{R}^{n_w}} \quad \frac{1}{2} w^T B w + \varphi^T w \tag{2.48a}$$
$$\text{s.t.} \quad Gw = g, \tag{2.48b}$$
$$Hw \geq h. \tag{2.48c}$$

In the primal-dual IP method, the inequality constraints (2.48c) are relaxed by introducing *slack variables* s, leading to

$$s = Hw - h \geq 0. \tag{2.49}$$

Thus, the first-order necessary KKT optimality conditions read as

$$Bw + \varphi - G^T \lambda - H^T \mu = 0 \tag{2.50a}$$
$$-Gw + g = 0 \tag{2.50b}$$
$$-Hw + h + s = 0 \tag{2.50c}$$
$$\mu_i s_i = \tau \quad i = 0, \ldots, n_h \tag{2.50d}$$
$$(\mu, s) \geq 0. \tag{2.50e}$$

At every iteration, the primal-dual method finds the solution for KKT system (2.50) through the employment of a Newton-type algorithm for which the search direction is found by solving

$$\begin{pmatrix} B & -G^T & -H^T & 0 \\ -G & 0 & 0 & 0 \\ -H & 0 & 0 & \mathbf{I} \\ 0 & 0 & M & T \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta \lambda \\ \Delta \mu \\ \Delta s \end{pmatrix} = - \begin{pmatrix} \varphi \\ g \\ h \\ Ms - \tau \mathbf{1} \end{pmatrix} \tag{2.51}$$

such that the inequalities $(\mu, s) \geq 0$ are strictly satisfied. Therein, $M = \text{diag}(\mu)$ and $T = \text{diag}(s)$.

Let us recall that, in the primal barrier method, the barrier parameter τ is related to the log-barrier term, and it is steered towards zero as the iterations approach the solution of (2.51) (see Section 2.3.3). In the primal-dual method, the parameter $\tau > 0$ defines the arc

$$\mathcal{C} = \{(w, \lambda, \mu, s) : \tau > 0\} \tag{2.52}$$

of strictly feasible points referred to as the *central path*. In a nutshell, it guides the algorithm towards a solution along a track that, at the same time as enforces $(\mu, s) > 0$, steers the pairwise product $\mu_i s_i$, $i = 0, \ldots, n_h$ to zero [38]. Unfortunately, for IP methods, the previous optimization problem's solution is not a good initialization to the next because it typically lies at the boundary of the feasible set [93]. These methods prefer starting points that enforce strictly feasibility of the inequality constraints, which means starting sufficiently close to the central path. For warm-start strategies in IP methods, we refer the reader to [67].

Most IP method solvers eliminate the slacks s and the Lagrange multipliers μ in a preprocessing before the actual factorization [94]. The resulting system is nothing

but the equality-constrained version of the KKT system (2.51)

$$\begin{pmatrix} B + H^T T^{-1} M H & -G^T \\ -G & 0 \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta \lambda \end{pmatrix} = - \begin{pmatrix} \varphi - H^T (T^{-1} M h - T^{-1} (M s - \tau \mathbf{1}) \\ g \end{pmatrix}. \tag{2.53}$$

The KKT system (2.53) can be interpreted as an equality-constrained QP for which one can employ structure-exploiting routines on the dense submatrices to approach the solution. For instance, [93] shows how an efficient Riccati block-elimination can be applied to a suitably reduced version of system (2.53). An alternative approach is presented in [95]. The Schur complement is used to procure a block-tridiagonal matrix, with dense blocks of size $n_x \times n_x$, which are later exploited using dense linear routines. Later work proposes an efficient variant to the classical Riccati recursion method that proves to be three times faster than the classical version, maintaining the same accuracy [88].

IP method solvers' exploitation capabilities make them great candidates to address the robotic challenges of this thesis. The advances in high-performance solvers, such as the *high-performance interior-point method* (`HPIPM`) [94], shed new light on efficient real-time optimal control. `HPIPM` is an open-source code that implements an efficient primal-dual IP method in a modular fashion. The main feature that distinguishes it from existing high-performance implementations of BLAS– and LAPACK–like routines for embedded applications is the use of the hardware-tailored *linear algebra* (LA) subroutines implemented in `BLASFEO` [96], optimized for small- to medium-scale matrices. In addition to the solver itself, `HPIPM` implements condensing and *partial condensing* routines. The second approach is of particular interest in this thesis due to the potential performance improvement when combined with the structure-exploiting solver in `HPIPM` for block-banded QPs.

Partial Condensing

A natural extension to the sparse and condensed approach is to combine the advantages of both. To this end, [97] proposes a technique to control the level of sparsity in MPC problems specifically. It consists of introducing a new degree of freedom in the MPC formulation to find the optimal sparsity level to the QP solver at hand by trading-off horizon length for input vector size at each shooting interval. Oppositely, the input vector size can be reduced at the price of a larger horizon length through an operation called *sequential update* [89]. The motivation behind partial condensing was the break in the sparse/dense dichotomy and the reduction in flop counts within the QP solver by changing the problem dimensions [97]. For the Riccati-based solver considered therein, it turns out that if roughly $n_x > n_u$, reducing the horizon length leads to a reduction in the flop count, whereas if roughly $n_x < n_u$, increasing the horizon length leads to a reduction in the flop count. In the practical context of MPC, this means significant reductions in solution times. Further investigations have found additional advantages in partial condensing that cover speed-ups in convergence for some algorithmic schemes [98], and improved efficiency of LA routines [99].

As in full condensing, partial condensing must be performed at each sampling instant when applied to NMPC schemes. However, in the present case, solution times

are expected to be reduced even further due to tailored LA subroutines. Particularly in `HPIPM`, the solver performance increases rapidly for small- to medium-scale matrices thanks to specific implementations in `BLASFEO`. The benefits realized from the combination of the LA subroutines with the partially condensed QP arising from an MPC formulation are significant. It turns out that there is a great advantage in replacing many operations in very small matrices with fewer operations with larger matrices, where LA subroutines can attain better performance [88]. As a result, the computational complexity grows with the prediction horizon N as $\mathcal{O}(N^2)$. Therefore, in this context, partial condensing allows better to exploit hardware throughput for problems of such size.

2.5 Real-time iteration scheme for nonlinear model predictive control

Thus far, we know how to formulate continuous-time optimal control problems and transcribe them into structured nonlinear programs that can be solved efficiently. When dealing with NLPs such as (2.17) within a nonlinear optimization scheme, the feasible state and control trajectory is found only at convergence. If these schemes are solved in an NMPC fashion, one can call them a *fully converged NMPC*. However, due to the non-negligible computational burden associated with their solution, it is often necessary to use approximate feedback policies to achieve real-time feasibility. This section will show that exploiting convex structures in nonlinear problems is the key to reliable and fast algorithms. On a concise note, we will outline a few algorithmic approaches in the domain of real-time optimization for NMPC that have gained significant attention from researchers and practitioners. Immediately after, we focus on describing the RTI scheme in detail.

Among many others, the *continuation/generalized minimal residual* (C/GMRES) method by Ohtsuka [100] implements Newton's method on the single shooting formulation while treating inequalities in an IP-like style with a fixed barrier parameter [101]. The *advanced-step NMPC controller* by Zavala and Biegler [102] performs IP iterations until convergence and then implements a tangential predictor to compensate for the computational delay. The algorithm continues with a correction step based on the solution manifold's linearization before applying the computed control input to the system. Explicit MPC approaches use multiparametric programming techniques to compute offline control actions given all possible states' enumeration so that online operations boil down to simple function evaluations [103, 104, 105]. Similar to DP (see Section 2.2), this approach is more suitable for systems with a few states and controls. NLP sensitivity-based controllers represent an alternative [106, 107, 108, 109, 110]. Several Newton-type controllers have been proposed to reduce the feedback delay in NMPC [21, 111, 112, 113, 114, 115]. This thesis is particularly interested in an efficient Newton-type algorithm for real-time NMPC applications, originally developed by Diehl [116]: the so-called *real-time iteration* (RTI) scheme.

The RTI scheme is an SQP variant that relies on the solution of a limited number of QP subproblems arising from direct multiple shooting and possibly with a Hessian approximation, such as QP subproblem (2.36). More precisely, only one

Algorithm 1: RTI with Gauss-Newton for NMPC at discrete-time k

Input: initial guess $x^{[0]} = (x_0^{[0]}, \ldots, x_N^{[0]})$, $u^{[0]} = (u_0^{[0]}, \ldots, u_{N-1}^{[0]})$, $k = 0$

while *true* **do**

Preparation phase

1. call to the integrator (2.11) to obtain $F(x_i^{[k]}, u_i^{[k]})$ and the first-order sensitivities $A_i^{[k]} = \nabla_x F(x_i^{[k]}, u_i^{[k]})^T$, $b_i^{[k]} := \nabla_u F(x_i^{[k]}, u_i^{[k]})^T$ for $i = 0, \ldots, N-1$.
2. linearize path and terminal constraints as in (2.36d), (2.36e), (2.36f) and (2.36g).
3. evaluate the Gauss-Newton Hessian blocks $B_i^{[k]} = \nabla \mathcal{F}(x_i^{[k]}, u_i^{[k]}) \nabla \mathcal{F}(x_i^{[k]}, u_i^{[k]})^T$ presented in Section 2.3.3.
4. *optional*: condense the sparse block-structured QP into a smaller but dense QP using one of the condensing approaches discussed in Section 2.4.1 and 2.4.2.
5. build QP (2.36) omitting the current state estimate $\bar{x}_0$.
6. wait until $\bar{x}_0$ is available.

Feedback phase

1. introduce $\bar{x}_0$ into the (possibly condensed) QP subproblem (2.36) and solve it by computing the search direction $(\Delta x, \Delta u)$ using a tailored QP solver (see Section 2.4).
2. apply one full step SQP iterate $x^{[k+1]} = x^{[k]} + \Delta x$, $u^{[k+1]} = u^{[k]} + \Delta u$ (see Section 2.3.2), and send the new control input $u_0^{[k+1]}$ to the system.
3. increment k and shift the trajectories x and u forward in time.

end

linearization and QP solve are carried out per sampling instant, leading to an approximate feedback control policy. In this way, solution times can be drastically reduced. Although only an approximate solution for (2.36) is obtained, the parametric manifold that describes the optimal solution $q^\star(s)$ can be tracked, leading to a feedback policy that, under suitable assumptions [117], achieves bounded suboptimality and preserves attractivity of equilibria.

The primary motivation behind the RTI scheme is to avoid iterating a problem that becomes outdated as the system evolves but rather apply an approximate solution computed using the most up-to-date information. This motivation illustrates the alleged *real-time dilemma* [27]. An important element in the RTI scheme is to keep the current state estimate $\bar{x}_0$ as a constrained decision variable, which is often referred to as *initial value embedding*. As presented in Algorithm 1, this property

allows one to divide computations into a *preparation* and *feedback* phase, where the former is typically more expensive. Also, it enables the integration of the dynamics and the computation of first-order sensitivities without actually knowing the current state estimate, as it enters in the optimization problem only via the trivial equality constraint (2.36b) [32]. If the algorithm is initialized at a neighboring solution, this constraint is readily satisfied by the first iterate and provides a good (first-order) approximation of the solution manifold. One can show that when the initial value embedding is combined with the Gauss-Newton Hessian approximation, one obtains a *multiplier-free, generalized tangential predictor*, namely, one that can work across active set changes [118].

A final and crucial element in the RTI scheme is the *shifting initialization* or *warm-start strategy*. Suppose the horizon length is significantly short so that the principle of optimality of subarcs (outlined in Section 2.2) does not hold, i.e., in an NMPC framework. In that case, one can prove that subsequent optimization problems may have very similar solutions [116]. Therefore, a rather natural way to initialize subsequent problems is to warm-start them, that is, using the solution of the previous iteration to initialize the current one. This is built on the expectation that the actual state estimate $\bar{x}_0^{\text{new}}$ does not differ much from the first-order correction to the previous iterate's optimal solution $\bar{x}_0^{\text{new}} \approx s_1^{\star\,\text{old}}$. Thereby, this strategy has a very natural connection with the initial value embedding. Also, because of the warm-start capability, local convergence and contraction rates [119, 120] are much more relevant here than globalization strategies. For completeness, we refer to other variants of the RTI scheme such as the multi-level iteration scheme [121, 122] and the advanced step RTI [123].

Efficient implementations of real-time optimization methods have played a vital role in bringing NMPC to real-time applications. In this spirit, this thesis will use the efficient implementations of the RTI scheme found in the open-source software packages `ACADO Toolkit` [85] and `acados` [124]. Other successful examples of real-time optimization packages with different methods are `VIATOC` [125], Control Toolbox [126], `MATMPC` [127], and `GRAMPC` [128]. The remainder of this chapter will present the features considered important in `ACADO` and `acados`. They will allow us to write the OCPs associated with the motion generation problems we will deal with. It is worth mentioning that the QP reformulations in the following sections are introduced for notational convenience. Note that we rely on an explicit multiple shooting notation to write QP subproblem (2.36). However, the use of shooting techniques is typically taken for granted within the presented software packages. Therefore, the QPs hereinafter are written directly in terms of states x, controls u, and slack variables s in one of the cases.

2.5.1 `ACADO Toolkit`

The open-source software package `ACADO Toolkit`[1] is dedicated to solving offline dynamic optimization, parameter and state estimation problems, and online estimation and MPC problems. It is written in `C++` and implements the scheme in Algorithm 1. It accounts with the so-called `ACADO Code Generation` Tool (CGT),

[1] `http://acado.github.io`

a feature to export an efficient, self-contained `C` code for real-time optimal control problems, which we use in this thesis. In `ACADO`, the OCP can be discretized using the single or multiple shooting approach. The solution of the underlying QP subproblems can be approached using one of the available solvers: `qpOASES`, `qpDUNES` [129], `FORCES` [130] or `HPMPC` [131]. Condensing options include either the classical $\mathcal{O}(N^3)$ [31] or the superior $\mathcal{O}(N^2)$ [87] condensing technique. There is a large class of integrators for code generation, comprising *implicit* RK (IRK) and ERK methods. Also, the RTI algorithm can be either based on the Gauss-Newton Hessian approximation or the exact Hessian, as pointed out in [44]. We refer the interested reader to the documentation [132] and some works related to the `ACADO Toolkit` software [133, 32, 134] for more details. One can find many successful implementations of real-time NMPC controllers for robotic applications using `ACADO` in [135, 136, 137, 55, 138, 139, 140, 141, 142, 143, 1]. Alternatively, the authors in [144] placed another layer on top of the `C` code generated by `ACADO` to make it directly compatible with ARM-based embedded platforms.

Within this thesis' scope, we deal with QP subproblems arising from applying an SQP method to the direct multiple shooting type NLP (2.17) with Gauss-Newton Hessian approximation and least squares tracking objective. Based on these choices, the MPC-related QP formulation in `ACADO` reads as follows:

$$\min_{\substack{\Delta x_0,\dots,\Delta x_N,\\ \Delta u_0,\dots,\Delta u_{N-1}}} \quad \frac{1}{2}\sum_{i=0}^{N-1}\begin{pmatrix}\Delta x_i\\ \Delta u_i\end{pmatrix}^T\begin{pmatrix}Q_i & S_i\\ S_i & R_i\end{pmatrix}\begin{pmatrix}\Delta x_i\\ \Delta u_i\end{pmatrix}+\begin{pmatrix}q_i\\ r_i\end{pmatrix}^T\begin{pmatrix}\Delta x_i\\ \Delta u_i\end{pmatrix}+ \tag{2.54a}$$

$$\frac{1}{2}\Delta x_N^T Q_N \Delta x_N + q_N^T \Delta x_N \tag{2.54b}$$

$$\text{s.t.} \quad \Delta x_0 = \bar{x}_0 - x_0^k, \tag{2.54c}$$

$$\Delta x_{i+1} - A_i\Delta x_i - B_i\Delta u_i - c_i = 0, \qquad i = 0,\dots,N-1, \tag{2.54d}$$

$$d_i^l \le G_i^x\Delta x_i + G_i^u\Delta u_i \le d_i^u, \qquad i = 0,\dots,N-1, \tag{2.54e}$$

$$d_N^l \le G_N^x\Delta x_N \le d_N^u. \tag{2.54f}$$

Therein, state and control deviations are defined as $\Delta x_i = x_i - x_i^k$, $\Delta u_i = u_i - u_i^k$ and superscript k denotes the linearization point for the states and inputs obtained at the previous QP iteration. The sensitivities and the linearized equality and inequality constraints are defined as

$$\begin{aligned}
A_i &:= \nabla_x F(x_i^k, u_i^k)^T, & B_i &:= \nabla_u F(x_i^k, u_i^k)^T,\\
G_i^x &:= \nabla_x g(x_i^k, u_i^k)^T, & G_i^u &:= \nabla_u g(x_i^k, u_i^k)^T,\\
G_N^x &:= \nabla_x g_N(x_N^k)^T, & c_i &:= F(x_i^k, u_i^k) - A_i x_i^k - B_i u_i^k,\\
d_i^l &:= g_i^l - G_i^x x_i^k - G_i^u u_i^k, & d_i^u &:= g_i^u - G_i^x x_i^k - G_i^u u_i^k,\\
d_N^l &:= g_N^l - G_N^x x_N^k, & d_N^u &:= g_N^u - G_N^x x_N^k.
\end{aligned} \tag{2.55}$$

The quadratic cost uses the exact gradients

$$\begin{aligned}
q_i &= \nabla_x(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)\\
r_i &= \nabla_u(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)\\
q_N &= \nabla_x(\eta_N(x_N^k) - \tilde{\eta}_N^k)(\eta_N(x_N^k) - \tilde{\eta}_N^k)
\end{aligned} \tag{2.56}$$

and the Gauss-Newton Hessian approximation:

$$\begin{aligned} Q_i &= \nabla_x(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)W^x\nabla_x(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)^T \\ R_i &= \nabla_u(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)W^u\nabla_u(\eta(x_i^k, u_i^k) - \tilde{\eta}_i^k)^T \\ Q_N &= \nabla_x(\eta_N(x_N^k) - \tilde{\eta}_N^k)W_N\nabla_x(\eta_N(x_N^k) - \tilde{\eta}_N^k)^T \end{aligned} \tag{2.57}$$

with W^x and W^u denoting the blocks in W related to states and inputs respectively.

2.5.2 `acados`

The high-performance software package `acados`[2] is the successor of `ACADO`. It has been completely rewritten to include, as the name suggests, *high-performance* implementations and state-of-the-art QP solvers. Unlike its predecessor, the automatic code generation is not necessary anymore. Yet efficiency is not affected as computationally intensive LA operations are performed using the high-performance BLAS/LAPACK library `BLASFEO`. Moreover, the `CasADi` [145] modeling language has been incorporated into the high-level interfaces of `acados` to generate nonlinear functions and sensitivities conveniently.One additional benefit of using `CasADi` is that it generates shorter instruction sequences and smaller (typically faster) code, more suitable for embedded applications [124]. The `acados` core library is written in `C` and implements the scheme in Algorithm 1 and other SQP-like methods in a modular fashion, as we will see next.

Nonlinear OCPs are discretized using multiple shooting to avoid the convergence problems common in single shooting formulations. The resulting QP subproblems can be solved using one of the several QP solvers present in the *OCP QP* and *Dense QP modules*, say: `qpDUNES`, `HPMPC`, `OSQP` [146], `qpOASES`, `HPIPM`, or `OOQP` [147]. The *Condensing module* provides efficient implementations of full and partial condensing routines $\mathcal{O}(N^2)$ [99]. The *Simulation module* contains IRK and ERK integrators, as well as novel implementations that cover lifted collocation integrators [148] and the structure-exploiting IRK algorithm, called GNSF-IRK scheme [149]. As for the Hessian approximations, a full-step SQP method is available in the *OCP NLP module* with different algorithmic choices, such as Gauss-Newton Hessians, *sequential convex quadratic programming* (SCQP) [150], and exact Hessians with regularization/convexification. We refer the reader to [124] for more detailed information about the `acados` software.

At the time of writing, `acados` was heavily under development to include, as mentioned earlier, the latest algorithmic and software implementations, e.g., new integration, condensing and Hessian approximation algorithms, as well as QP solvers. As a result, few studies have effectively implemented real-time NMPC controllers tailored to robotic applications [151, 2, 4, 5] with the package. In particular, the authors in [152] deployed the NMPC controller directly on a Cortex-A9 CPU running at 800 MHz onboard a human-sized quadrotor. The reader should be aware that the majority of these studies come from the applications presented in Chapters 5, 6 and 7 of this thesis.

[2] `https://github.com/acados/acados`

Under the same considerations for the QP subproblems in ACADO, the equivalent QP formulation in acados reads as follows:

$$
\begin{aligned}
\min_{\substack{\Delta x_0,\dots,\Delta x_N,\\ \Delta u_0,\dots,\Delta u_{N-1},\\ s_0,\dots,s_N}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \begin{pmatrix}\Delta x_i\\ \Delta u_i\end{pmatrix}^T \overbrace{\begin{pmatrix}Q_i & S_i\\ S_i & R_i\end{pmatrix}}^{H_i} \begin{pmatrix}\Delta x_i\\ \Delta u_i\end{pmatrix} + \begin{pmatrix}q_i\\ r_i\end{pmatrix}^T \begin{pmatrix}\Delta x_i\\ \Delta u_i\end{pmatrix} + && \text{(2.58a)}\\
& \frac{1}{2}\Delta x_N^T Q_N \Delta x_N + q_N^T \Delta x_N + && \text{(2.58b)}\\
& \frac{1}{2}\sum_{i=0}^{N-1} \begin{pmatrix}s_i^l\\ s_i^u\\ 1\end{pmatrix}^T \begin{pmatrix}Z_i^l & 0 & z_i^l\\ 0 & Z_i^u & z_i^u\\ z_i^{lT} & z_i^{uT} & 0\end{pmatrix} \begin{pmatrix}s_i^l\\ s_i^u\\ 1\end{pmatrix} + && \text{(2.58c)}\\
& \frac{1}{2}\begin{pmatrix}s_N^l\\ s_N^u\\ 1\end{pmatrix}^T \begin{pmatrix}Z_N^l & 0 & z_N^l\\ 0 & Z_N^u & z_N^u\\ z_N^{l\,T} & z_N^{u\,T} & 0\end{pmatrix} \begin{pmatrix}s_N^l\\ s_N^u\\ 1\end{pmatrix} && \text{(2.58d)}\\
\text{s.t.} \quad & \Delta x_0 = \bar{x}_0 - x_0^k, && \text{(2.58e)}\\
& \Delta x_{i+1} - A_i \Delta x_i - B_i \Delta u_i - c_i = 0, \qquad i = 0,\dots,N-1, && \text{(2.58f)}\\
& s_i + G_i^x \Delta x_i + G_i^u \Delta u_i \le 0, \qquad i = 0,\dots,N-1, && \text{(2.58g)}\\
& s_N + G_N^x \Delta x_N \le 0, && \text{(2.58h)}\\
& 0 \le s_i, \qquad i = 0,\dots,N. && \text{(2.58i)}
\end{aligned}
$$

The linearized dynamics and inequality constraints use the following matrices:

$$
\begin{aligned}
A_i &:= \nabla_x F(x_i^k, u_i^k)^T, \quad B_i := \nabla_u F(x_i^k, u_i^k)^T,\\
G_i^x &:= \nabla_x g(x_i^k, u_i^k)^T, \quad G_i^u := \nabla_u g(x_i^k, u_i^k)^T,\\
G_N^x &:= \nabla_x g_N(x_N^k)^T,\\
c_i &:= F(x_i^k, u_i^k) - A_i x_i^k - B_i u_i^k,\\
s_i &:= g(x_i^k, u_i^k) - G_i^x x_i^k - G_i^u u_i^k,\\
s_N &:= g_N(x_N^k) - G_N^x x_N^k.
\end{aligned}
\tag{2.59}
$$

Similarly, the quadratic cost uses the exact gradients q_i, r_i, q_N and the Gauss-Newton Hessian approximation H_i as defined in Eqs. (2.56) and (2.57). Different from ACADO, slack variables s are introduced and associated with the path constraints s_i^l, $s_i^u \in \mathbb{R}^{n_s}$, and the terminal constraints s_N^l, $s_N^u \in \mathbb{R}^{n_{s,N}}$ in acados. The symmetric ℓ_2 slack penalties for the stage and terminal cost are denoted by Z_i^l, Z_i^u and Z_N^l, Z_N^u, respectively. Likewise, the symmetric ℓ_1 slack penalties are denoted by z_i^l, z_i^u and z_N^l, z_N^u. Note that not all QP solvers present in acados can directly deal with slack variables. For those that do not, slack variables are reformulated as extra input variables [124].

2.6 Chapter summary

This chapter gave a detailed overview of the necessary theoretical foundations for real-time NMPC. It began by stating the class of dynamic systems and the corresponding optimal control problem formulation this thesis addresses. The treatment

of the resulting optimization problems was considered next, introducing the theory related to direct methods that recall concepts of nonlinear programming. Such problems are, in general, difficult to solve in real-time. This observation motivated the discussion about structure exploitation on QP subproblems, followed by a presentation of the numerical algorithms embedded in two efficient QP solvers. Afterward, the chapter provided a survey on online algorithms for real-time NMPC, describing the chosen method in detail – the well-established RTI scheme. It finished with a discussion of the RTI scheme's implementation as part of the software packages `ACADO Toolkit` and `acados`, forming the algorithmic frameworks for the subsequent chapters.

Chapter 3

Dynamic models

3.1 Double inverted pendulum

This section presents the dynamic model of a double inverted pendulum called *Pendubot*, schematically illustrated in Figure 3.1. It is a two-link underactuated planar robot that has been widely used across the field of nonlinear control. Unlike the *Acrobot*, which has the actuation at its elbow, the Pendubot is driven at its shoulder joint. Let us now go through the derivation of its dynamic model, reaching the ODEs that will be the basis for the controller design.

3.1.1 Dynamic model

The Pendubot dynamic model can be derived through the Euler-Lagrange equations of motion, which leads to a set of second-order ODEs of the form

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau, \tag{3.1}$$

where $q = (q_1, q_2) \in \mathbb{R}^2$ is the vector containing the joint angle positions. Eq. (3.1) makes use of the symmetric positive-definite generalized mass matrix

$$M(q) = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \tag{3.2}$$

$$\begin{aligned} m_{11} &= m_1 a_1^2 + m_2(l_1^2 + a_2^2 + 2l_1 a_2 \cos(q_2)) + J_1 + J_2 + J_m \\ m_{12} &= m_{21} = m_2(a_2^2 + l_1 a_2 \cos(q_2)) + J_2 \\ m_{22} &= m_2 a_2^2 + J_2, \end{aligned} \tag{3.3}$$

generalized stiffness vector

$$C(q,\dot{q}) = \begin{pmatrix} -m_2 l_1 a_2 \sin(q_2)\dot{q}_2^2 - 2m_2 l_1 a_2 \sin(q_2)\dot{q}_1\dot{q}_2 \\ m_2 l_1 a_2 \sin(q_2)\dot{q}_1^2 \end{pmatrix}, \tag{3.4}$$

generalized gravitational vector

$$G(q) = \begin{pmatrix} (m_1 a_1 + m_2 l_1) g \cos(q_1 + \pi/2) + m_2 a_2 g \cos(q_1 + \pi/2 + q_2) \\ m_2 a_2 g \cos(q_1 + \pi/2 + q_2) \end{pmatrix}, \tag{3.5}$$

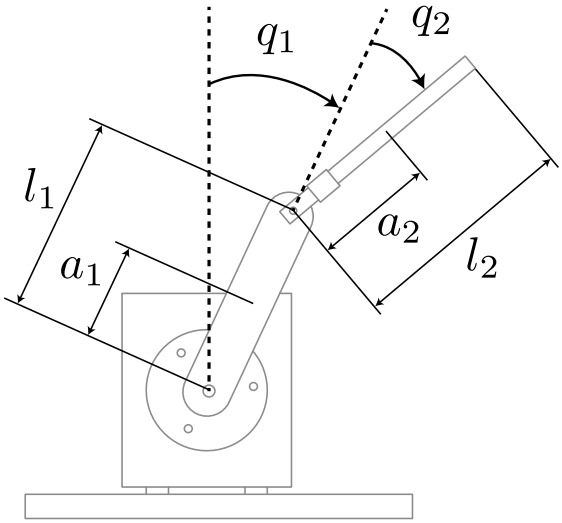


Figure 3.1. Schematic of the Pendubot coordinate system.

and generalized input vector

$$\tau = \begin{pmatrix} u \\ 0 \end{pmatrix} \tag{3.6}$$

due to an external input torque u. Therein, $g \in \mathbb{R}^+$ denotes the gravitational acceleration, $J_i \in \mathbb{R}^+$ and $m_i \in \mathbb{R}^+$ are the inertia moment and the mass of the i^{th} link, respectively, $l_i \in \mathbb{R}^+$ represents the length of the i^{th} link, and a_i is the distance between the *center of mass* (CoM) of the i^{th} link and the center of the i^{th} joint. Denoted by $J_m \in \mathbb{R}^+$ is the rotor inertia. Note that the joint position q_1 is measured with respect to the upwards vertical axis, while the angle q_2 is defined relative to the first joint, as depicted in Figure 3.1.

Nonlinear dynamics

If we solve Eq. (3.1) for $\ddot{q}$, we obtain

$$\begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} = M(q)^{-1}\tau - M(q)^{-1}C(q,\dot{q}) - M(q)^{-1}G(q), \tag{3.7}$$

which corresponds to the Pendubot state equations whose frictionless nonlinear dynamics we describe as

$$\dot{x} = f_p(x,u) = \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix}. \tag{3.8}$$

The state is defined by $x := (q_1, q_2, \dot{q}_1, \dot{q}_2) \in \mathbb{R}^4$, and the control input is the rotor torque $u \in \mathbb{R}$ at the shoulder joint.

3.1.2 Physical parameters

Accurate values for physical parameters are the key to creating a model that correctly describes a system's behavior and allows for superior control performance. The Pendubot model uses the parameters of the Quanser[1] platform available in the DIAG Robotics lab. They are presented in Table 3.1.

Table 3.1. Physical parameters for the Pendubot

Parameter	*Value*	*Description*
m_1	0.193 kg	total mass of link 1
l_1	0.1483 m	length of link 1
a_1	0.1032 m	distance to the center of mass of link 1
J_1	$3.54 \cdot 10^{-4}$ kg·m^2	inertia moment of link 1
m_2	0.073 kg	total mass of link 2
l_2	0.1804 m	length of link 2
a_2	0.1065 m	distance to the center of mass of link 2
J_2	$2 \cdot 10^{-4}$ kg·m^2	inertia moment of link 2
J_m	$4.74 \cdot 10^{-4}$ kg·m^2	rotor inertia

3.2 Quadrotor

This section introduces the dynamic model of a quadrotor. Several models have been proposed to describe a mathematical function that accurately captures the physical reality and, at the same time, can be used in the controller design procedure. But as the quadrotor dynamics uncover different characteristics according to the flight mission, – e.g., hovering, vertical/longitudinal/lateral motion, acrobatic maneuvers, etc. – one has to evaluate such a trade-off from the task's perspective. This assessment dictates the initial assumptions for the model derivation.

According to the literature, these models fall into two popular categories: *force-moment approach* and *single-rotor thrust approach*. They both have in common a system with four control inputs whose manipulation allows the control of output states resulting from translational and rotational dynamics. The force-moment approach defines the relationship between the output states and the total thrust force, rolling moment, pitching moment, and yawing moment. Authors [153, 154] use this approach in their modeling. Simplistic assumptions proved approximately sound as they deal with trajectory tracking at low speeds. Bounds on forces and moments were also considered, reflecting the control input limitations. In the single-rotor thrust approach, the output states relate to the balance of forces and moments around the quadrotor's CoM, which typically depends on gravity and the individual thrust force generated by each rotor. This approach was employed in the modeling

[1] `https://www.quanser.com/solution/robotics/`

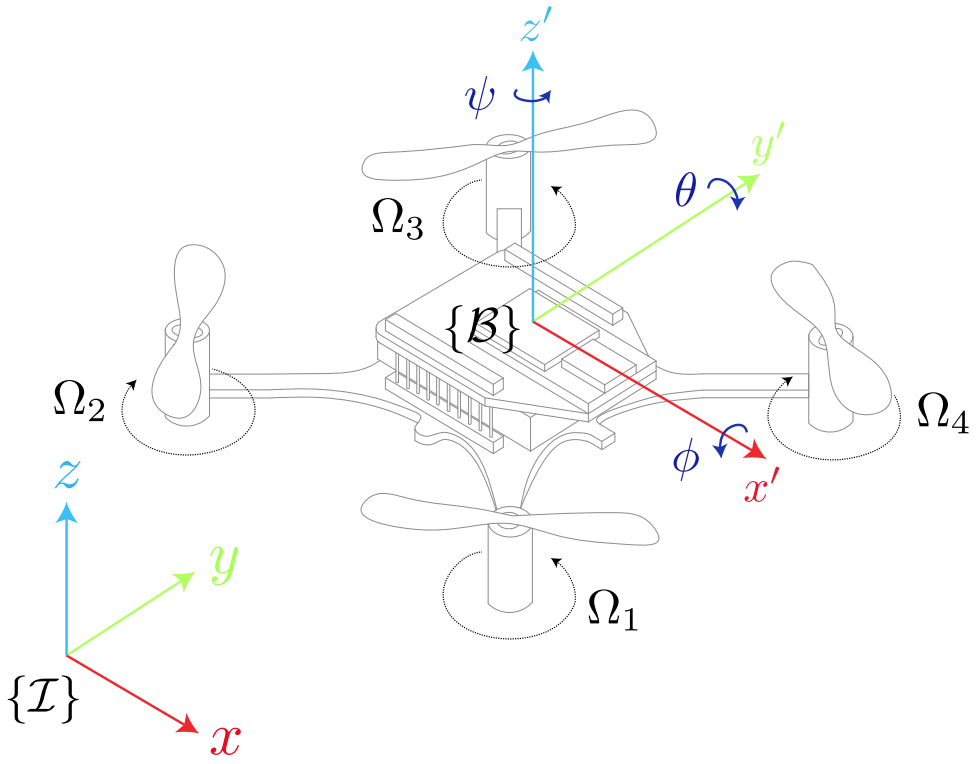


Figure 3.2. Schematic of the Crazyflie coordinate system.

of [46], where the task therein involves a high-speed 3D motion in an unknown environment, thus requiring the incorporation of aerodynamic effects. Analogously, [155] use this approach to model a quadrotor whose task is to move in a cluttered and unknown environment where physical and operational constraints are also considered.

Given these considerations, this thesis is interested in models that capture the quadrotor's physical limitations since they can be treated explicitly in the NMPC formulation. Thus, we opt for a quadrotor model that includes the rotor dynamics to impose more realistic constraints while maximizing gains due to a more accurate representation. This model also fits the type of tasks the thesis is interested in, mainly requiring low-speed 3D motions.

This section will use a top-down approach, starting with the rigid body dynamics, investigating the forces and moments acting on the quadrotor, and finally analyzing the rotor dynamics.

3.2.1 Coordinate frames

The first step to obtaining the quadrotor dynamic model is to define the coordinate frames. Two right-hand frames are used: an inertial frame, centered on $\{\mathcal{I}\}$ and pointing towards *North, West, and Up* (NWU); and a body-fixed frame $\{\mathcal{B}\}$ located at the quadrotor's CoM and aligned with $\{\mathcal{I}\}$, as shown in Figures 3.2 and 3.3. The quadrotor has position $p = (x, y, z) \in \mathbb{R}^3$ expressed in $\{\mathcal{I}\}$, attitude $q = (q_w, q_x, q_y, q_z) \in \mathbb{H}$ with respect to $\{\mathcal{I}\}$, linear velocity $v_b = (v_x, v_y, v_z) \in \mathbb{R}^3$

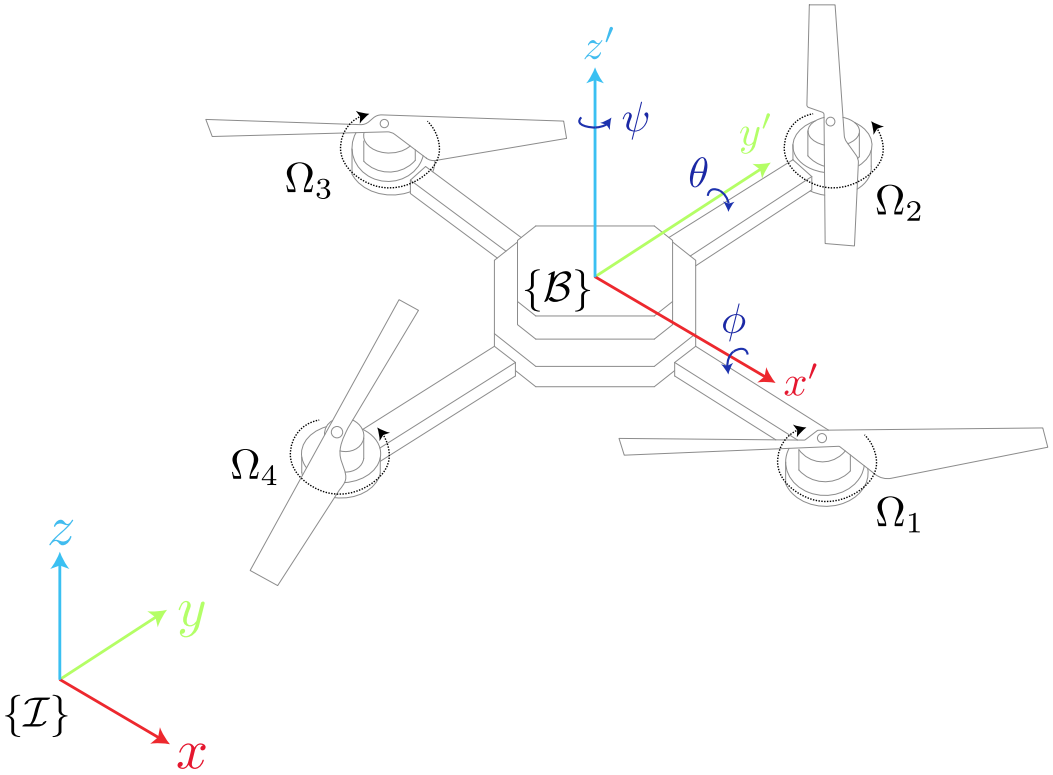


Figure 3.3. Schematic of the MikroKopter coordinate system

expressed in $\{\mathcal{B}\}$, and angular rate $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ expressed in $\{\mathcal{B}\}$. The four rotors are numbered as indicated in the figures. The rotational speed of the propellers is denoted by $\Omega = (\Omega_1, \Omega_2, \Omega_3, \Omega_4) \in \mathbb{R}^4$, while the rotor torques are denoted by $\tau = (\tau_1, \tau_2, \tau_3, \tau_4) \in \mathbb{R}^4$.

3.2.2 Dynamic model

Before proceeding, we first list several general assumptions that will allow us to simplify the model. Although they are not perfectly valid on the real platforms of this thesis, they provide sufficiently good approximations.

- The quadrotor is a rigid body.
- The quadrotor is symmetric in its geometry, mass, and propulsion subsystem.
- The rotor's flapping effect is neglected.
- Nonlinearities of the battery will be neglected.
- The ground effect is neglected during take-off and landing.
- All rotors have the same time constant.

When working with vectors expressed in different coordinate frames, it is important to pass from one frame to another. The rotation of any point in the Euclidian

3D space can be represented by a sequence of three single rotation matrices around the Euler angles,

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix}, \tag{3.9}$$

$$R_z(\psi) = \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Based on that, we can define a rotation matrix from $\{\mathcal{I}\}$ to $\{\mathcal{B}\}$ using their combination. This thesis uses the sequence $R_x(\phi) \to R_y(\theta) \to R_z(\psi)$, corresponding to the matrix product $R = R_z(\psi)R_y(\theta)R_x(\phi)$. The result is

$$R = \begin{pmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\phi\cos\theta \end{pmatrix}, \tag{3.10}$$

where ϕ, θ, and ψ represent the roll, pitch, and yaw angles, respectively. Note that due to the *orthogonality* property of R, its inverse is identical to its transpose, i.e., $R^{-1} = R^T$. On another note, when $\theta = \pi/2$, we lose one degree of freedom in the 3D space due to the *gimbal lock* phenomenon. In particular, the gimbals ϕ and ψ start spinning on the same plane, creating an ambiguity: one notation may represent two orientations. One way to dodge this ambiguity is to use unit quaternions to represent the attitude instead of Euler angles.

Quaternion is a quadrinominal expression composed of one term called the *real part* and the three others called the *imaginary part* [156]: $q = (q_w, q_x, q_y, q_z)$. Due to their "all-attitude" capability and numerical advantages [157], unit quaternions are now widely used in robotics to represent a rotating body's attitude. One can state their relation with the Euler angles representation as:

$$q = \begin{pmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ -(\cos(\psi/2)\cos(\theta/2)\sin(\phi/2) - \sin(\psi/2)\sin(\theta/2)\cos(\phi/2)) \\ -(\cos(\psi/2)\sin(\theta/2)\cos(\phi/2) + \sin(\psi/2)\cos(\theta/2)\sin(\phi/2)) \\ -(\sin(\psi/2)\cos(\theta/2)\cos(\phi/2) - \cos(\psi/2)\sin(\theta/2)\sin(\phi/2)) \end{pmatrix}, \tag{3.11}$$

where it should be noted that $\|q\| = 1$ at all times. One can also use quaternions to rotate a Euclidean vector, in the same way that one uses R, thanks to the following relation:

$$S = \begin{pmatrix} 2(q_w^2 + q_x^2) - 1 & 2(q_xq_y + q_wq_z) & 2(q_xq_z - q_wq_y) \\ 2(q_xq_y - q_wq_z) & 2(q_w^2 + q_y^2) - 1 & 2(q_yq_z + q_wq_x) \\ 2(q_xq_z + q_wq_y) & 2(q_yq_z - q_wq_x) & 2(q_w^2 + q_z^2) - 1 \end{pmatrix}. \tag{3.12}$$

The above expression corresponds to the rotation matrix from $\{\mathcal{I}\}$ to $\{\mathcal{B}\}$ in terms of unit quaternions, which inherits the same orthogonality properties of R. This thesis will mostly use unit quaternion for attitude representation. However, we will see in the coming chapters that, according to the motion generation problem we will be attempting to solve, the Euler angles representation can be more straightforward than unit quaternions when it comes to tracking the quadrotor's attitude.

Translational dynamics

According to Newton's 2nd law, the sum of forces in the inertial frame is

$$F_r = m\dot{v}, \tag{3.13}$$

where m is the quadrotor's mass, and v its linear velocity. To make the derivation more intuitive, let us now consider the sum of forces directly in the body-fixed frame, where the Coriolis effect appears, yielding the following equation:

$$F_r = m(\dot{v}_b + \omega \times v_b). \tag{3.14}$$

The resulting force F_r establishes the relation between the propellers' aerodynamic forces and the weight force projected onto the body-fixed frame. One can formally write this force balance as

$$\begin{pmatrix} 0 \\ 0 \\ F_z \end{pmatrix} - S \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = m \left(\begin{pmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{pmatrix} + \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \times \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \right), \tag{3.15}$$

where g is the gravitational acceleration. If we solve for $\dot{v}_b$, we obtain

$$\dot{v}_b = \begin{pmatrix} 0 \\ 0 \\ F_z/m \end{pmatrix} - S \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} - \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \times \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}, \tag{3.16}$$

which is the first set of state-space variables associated with the translational dynamics. To determine the second set, we need to project the vector v_b onto the inertial frame to obtain the quadrotor's velocity in this coordinate frame, which leads us to

$$\dot{p} = S^T \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}. \tag{3.17}$$

Each propeller produces a force along z' that is used for lift. The net force created by the propellers' rotation through the viscous air and applied to the quadrotor's CoM is

$$F_b = \begin{pmatrix} 0 \\ 0 \\ F_z \end{pmatrix}, \tag{3.18}$$

where F_z is referred to as the *total thrust*. It is widely known that, for a given rotor $j = 1, \ldots, 4$, the thrust force T_j depends quadratically on the propeller's rotational speed

$$T_j = C_T \Omega_j^2. \tag{3.19}$$

The *thrust coefficient* $C_T \in \mathbb{R}^+$ is usually large, indicating that the propeller turns much more slowly than the driving rotor. The total thrust can then be expressed in terms of the propeller's rotational speed as

$$F_z = \sum_{j=1}^{4} T_j = C_T \sum_{j=1}^{4} \Omega_j^2. \tag{3.20}$$

Rotational dynamics

Similar to the reasoning applied to the translational dynamics, the sum of moments (torques) in the inertial frame is the derivative of the angular momentum in the rotational case

$$M_r = \dot{L}, \tag{3.21}$$

with L being the angular momentum around the quadrotor's *center of gravity* (CoG). Expressing this sum of moments in the body-fixed frame and acknowledging the Coriolis effect gives us

$$M_r = J\dot{\omega} + \omega \times J\omega, \tag{3.22}$$

where $J \succ 0$ denotes the quadrotor's moment of inertia tensor. It is calculated with respect to its CoM and is given by

$$J = \begin{pmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{pmatrix}. \tag{3.23}$$

If we solve Eq. (3.22) for $\dot{\omega}$, we obtain

$$\dot{\omega} = J^{-1} \left(\begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} - \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \times J \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \right), \tag{3.24}$$

which stands for the first set of state equations in the rotational case. The second set comes from the derivative of a time-varying quaternion

$$\dot{q} = \frac{1}{2} \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix} \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} \tag{3.25}$$

or the relationship between the instantaneous rates of change of $\gamma := (\phi, \theta, \psi)$ and the instantaneous components of ω

$$\dot{\gamma} = \overbrace{\begin{pmatrix} 0 & 0 & 1/\cos(\theta) \\ 0 & 1/\cos\psi & -\tan\psi\tan\theta \\ 1 & \tan\psi & -\tan\theta/\cos\psi \end{pmatrix}}^{E} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}, \tag{3.26}$$

depending on the chosen attitude representation.

To calculate the net moment applied to the quadrotor's CoM, we need to know the rotation direction of each rotor. Figures 3.2 and 3.3 show that the Crazyflie and the MikroKopter have different arrangements for the rotors. Therefore, we explicitly regard two distinct net moments.

Due to the angular momentum's conservation, each propeller generates a moment that opposes its rotation direction. Thus, the net moment resulting from aerodynamics (the combination of produced rotor forces and air resistance) for the Crazyflie is

$${}^{c}M_b = \begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} = \begin{pmatrix} C_T \cdot l(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ C_T \cdot l(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \\ C_D(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{pmatrix}. \tag{3.27}$$

In contrast, for the MikroKopter is

$$^{m}M_b = \begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} = \begin{pmatrix} C_T \cdot l(\Omega_2^2 - \Omega_4^2) \\ C_T \cdot l(\Omega_3^2 - \Omega_1^2) \\ C_D(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{pmatrix}. \tag{3.28}$$

Therein, $l \in \mathbb{R}^+$ represents half of the distance between rotors for the Crazyflie, and the arm length for the MikroKopter, $C_D \in \mathbb{R}^+$ is the *drag coefficient*. The reaction moment, due to rotor drag, acting on the body-fixed frame and generated by each propeller can be modeled as

$$M_j = C_D \Omega_j^2. \tag{3.29}$$

This differential equation pits the moment (torque) delivered by the rotor against a quadratic-drag type loss, which depends on the propeller's rotational speed.

Rotors

From Eqs. (3.20), (3.27) and (3.28), one realizes that a high-performance attitude and position control directly implies a high-quality control at the propeller speed level, which in turn is related to the rotor torque. Most medium-sized quadrotors, such as the MikroKopter, are equipped with *brushless DC* (BLDC) motors whose *back electromotive force* (back EMF) is used to perform rotor commutation and high-frequency *pulse width modulation* (PWM) to control rotor voltage [158]. This thesis considers single-input control at the rotor torque level, similar to more sophisticated schemes

$$\tau = D\Omega + J_m\dot{\Omega} + \tau_L, \tag{3.30}$$

to achieve improved performance in medium-sized quadrotors. In Eq. (3.30), $\tau \in \mathbb{R}$ is the electromagnetic torque, $D \in \mathbb{R}^+$ is the damping coefficient, $J_m \in \mathbb{R}^+$ is the total moment of inertia that includes: the transmission system inertia, the load inertia w.r.t. the rotor shaft, and the rotor inertia. Finally, τ_L is the torque required to drive the load w.r.t. the rotor shaft, i.e., the torque of the j^{th} propeller $\tau_L = M_j$. If we plug Eq. (3.29) into (3.30) and solve for $\dot{\Omega}$, we obtain the ODE representing the rotor dynamics

$$\dot{\Omega} = \frac{1}{J_m}(\tau - C_D\Omega^2 - D\Omega). \tag{3.31}$$

The current supplied by the batteries ultimately limits control performance at the rotor level [158]. Classic PID and *linear quadratic regulator* (LQR) controls are agnostic to this physical limitation, requiring the introduction of saturation to keep the rotor torque within its operating range. Unfortunately, when a feedback control system's operating point happens to be in the saturation range, a change in the input does not cause any change in the output. In other words, the system behaves exactly as in *open-loop condition*, and no control is available [159]. This physical limitation does not represent a challenge for an NMPC controller since rotor torques end up being explicitly regarded as bounded control inputs in the optimization problem, as we will see next.

Nonlinear dynamics

Having already assessed the forces and moments acting on a quadrotor, we will now assemble the nonlinear models for the Crazyflie and the MikroKopter using the set of state equations retrieved previously. Considering the Crazyflie nano-quadrotor first, we assume that, due to its coreless DC motors, it is possible to change the rotational speed of the j^{th} propeller instantaneously. Thus, we define their set as the control input of the system

$$u := (\Omega_1, \Omega_2, \Omega_3, \Omega_4) \in \mathbb{R}^4. \tag{3.32}$$

The first-order ODEs then give the Crazyflie nonlinear dynamics:

$$\dot{\xi} = f_c(\xi, u) = \begin{pmatrix} S^T v_b \\ \frac{1}{2} q \otimes \omega \\ \frac{1}{m} F_b - S g \mathbf{1_z} - \omega \times v_b \\ J^{-1}({}^c M_b - \omega \times J\omega) \end{pmatrix}, \tag{3.33}$$

with state $\xi := (p, q, v_b, \omega) \in \mathbb{R}^{13}$. Here, $\mathbf{1_z} = (0, 0, 1)$ and denoted by $\otimes$, the quaternion product.

Let us now conseider the MikroKopter quadrotor. When dealing with a medium-sized quadrotor, J_m is not negligible since the radius of the rotor's axle is not small. Therefore, in this case, the rotor torques are the control inputs of the system

$$u := (\tau_1, \tau_2, \tau_3, \tau_4) \in \mathbb{R}^4. \tag{3.34}$$

Thus, the MikroKopter nonlinear dynamics are:

$$\dot{\vartheta} = f_m(\vartheta, u) = \begin{pmatrix} R^T v_b \\ E\omega \\ \frac{1}{m} F_b - R g \mathbf{1_z} - \omega \times v_b \\ J^{-1}({}^m M_b - \omega \times J\omega) \\ \frac{1}{J_m}(u - C_D \Omega \odot \Omega - D\Omega) \end{pmatrix}, \tag{3.35}$$

with state $\vartheta := (p, \gamma, v_b, \omega, \Omega) \in \mathbb{R}^{16}$. The element-wise product is denoted by $\odot$.

3.2.3 Physical parameters

Several studies provide reasonable approximations for the physical parameters of the Crazyflie 2.X, such as [160, 161, 162]. Our Crazyflie 2.1 model uses the parameters identified by [161], except for parameters l, m, and C_T that were identified experimentally. The values are summarized in Table 3.2. Likewise, our MikroKopter model uses the parameters of the RIS-LAAS[2] team platforms. Table 3.3 shows the values accordingly. Note that we perform model scaling as MPC uses normalized input/output variables. Other than equaling/approximating the variables' range, scaling provides further benefits. Among them, a more intuitive tuning of the weights, which allows the designer to emphasize the relative priority of each term in the cost function rather than the combination of priority and signal scale. It also offers improved numerical conditioning as optimization routines are less prone to round-off errors.

[2] `https://www.laas.fr/public/en/ris`

Table 3.2. Physical parameters for the Crazyflie 2.1

Parameter	*Value*	*Description*
m	0.33 kg	total mass
l	0.0325 m	arm length
J_{xx}	$1.395 \cdot 10^{-5}$ kg·m^2	inertia moment around x
J_{yy}	$1.395 \cdot 10^{-5}$ kg·m^2	inertia moment around y
J_{zz}	$2.173 \cdot 10^{-5}$ kg·m^2	inertia moment around z
C_D	$7.9379 \cdot 10^{-6}$ N·m/krpm^2	drag coefficient
C_T	$3.25 \cdot 10^{-4}$ N/krpm^2	thrust coefficient

Table 3.3. Physical parameters for the MikroKopter

Parameter	*Value*	*Description*
m	1.04 kg	total mass
l	0.23 m	arm length
J_{xx}	0.01 kg·m^2	inertia moment around x
J_{yy}	0.01 kg·m^2	inertia moment around y
J_{zz}	0.07 kg·m^2	inertia moment around z
C_D	10 N·m/kHz^2	drag coefficient
C_T	595 N/kHz^2	thrust coefficient
J_m	0.08 g·m^2	total rotor inertia
D	0.5 mN·m·s	damping coefficient

3.3 Chapter summary

This chapter introduced the nonlinear dynamic models of a Pendubot and a quadrotor. Among all trade-offs required for deriving a model, the most important is balancing the real system's complexity and simplicity for the controller's design. It is crucial that the outcome of this balance explicitly states what is known about the system's physical limitations so they can be addressed explicitly by the NMPC. To derive such a model for a Pendubot system, classical Euler-Lagrange equations of motion were used to draw out a set of second-order ODEs that sharply describe the relationship between the limited input torque of the first joint and the joint accelerations. The friction amongst joints was neglected, but this property may not be of great importance for most self-balancing control applications, as the results suggest. Concerning quadrotors, two different systems were considered. Through the Newton-Euler formalization, specific sets of first-order ODEs were derived for each case. They differ in whether the rotor inertia is small enough so that it is possible to change the limited propeller speeds instantly or if it is sufficiently big such that nonlinear rotor dynamics drive propeller speeds.

Chapter 4

Real-time NMPC for motion generation of a double inverted pendulum

4.1 Motivation and contribution

Several modern motion generation problems in robotics build up their foundation on the inverted pendulum stabilization principle. This happens because it can describe such problems quite naturally and concurrently explain countless biological processes that robots often attempt to mimic. For this reason, the rich properties of the inverted pendulum play a major role in inspiring technology trends as well as detecting abilities of innovative control methods dealing with nonlinearities, delays, oscillations, uncertainties, minimum phase dynamics, and so on [163]. These simple – and plausible – implications reveal that top-level control of the inverted pendulum benchmark yields first-rate solutions that may be easily translatable to intricate robotic systems.

Technology trends are heading towards wheeled self-balancing robots for personal urban mobility and humanoid robots as human collaborators. Hoverboards and one-wheel scooters are successful examples of the first case, while the T-HR3[1], the Digit[2], the Walker[3], and Sophia[4] are some cobots that illustrate the second one. At the same pace, innovative control methods have been pushing systems to their maximum performance while ensuring active trajectory tracking and stabilization to prevent overbalancing. In single and multi-linked inverted pendulum systems, the loci of equilibria lie on a nonlinear manifold. Therefore, an appropriate choice is to use nonlinear controllers. Despite the elegant solutions handed by classic nonlinear control methods, such as feedback linearization and Lyapunov-based methods, their design process becomes cumbersome when it comes to handling constraints, as they do not scale well for high-dimensional systems. Alternatively, when the NMPC's ability to explicitly handle constraints is combined with pow-

[1] https://www.toyota-europe.com/startyourimpossible/t-hr3
[2] https://www.agilityrobotics.com/meet-digit
[3] https://www.ubtrobot.com/collections/innovation-at-ubtech?ls=en
[4] https://www.hansonrobotics.com/sophia/

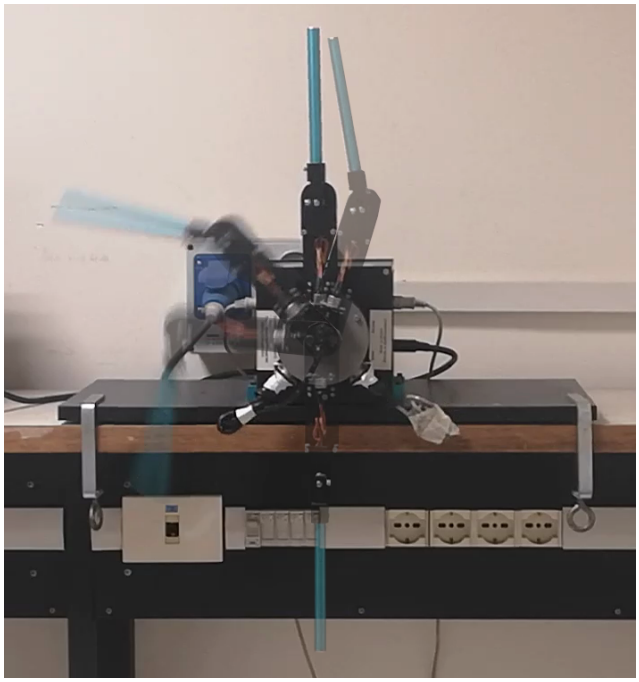


Figure 4.1. The Pendubot performing the swing-up task with the proposed NMPC controller.

erful structure-exploitation numerical methods, the scalability bottleneck can be mitigated.

From this perspective, this chapter aims at describing the design and implementation of a real-time NMPC controller to solve the swing-up problem of a double inverted pendulum (see Figure 4.1), a more challenging realization of the original benchmark. In particular, we benefit from the straightforward treatment of constraints in NMPC to enforce physical and operational limitations. The current work can be seen as part of a broader study on tracking controllers for applications other than the double inverted pendulum, such as motion generation in self-balancing mobile robots or (potentially) walking pattern generation in humanoid robots. In this sense, the main contributions of this chapter are:

- A real-time NMPC controller with constraints enforcement for tracking unstable equilibrium points in robotic systems characterized by balanced motion.
- An experimental validation that demonstrates the controller's capabilities and computational efficiency when employed to the Pendubot in a swing-up task.

4.2 Related works

Robotic benchmark systems have played a crucial role in teaching, research in control theory, and intelligent service robotics. Despite the simplified structures, they offer several attractive control challenges that include model validation, implementation of novel control strategies, and performance evaluation. Among many, the

most common robotic benchmarks are the Acrobot [164], the cart-pole [165], the Furuta pendulum [166], the reaction wheel pendulum [167], the ball-and-beam system [168] (see Figures 4.2 and 4.3), and the Pendubot [169]. They fostered scientific findings that helped researchers solve complex motion generation problems, enabling novel technologies. Remarkably, their simplified models typically contain the information needed to capture the dynamic essence of the desired motion. The 3D *linear inverted-pendulum models* (LIPM) [170] for walking pattern generation in humanoid robots and the wheeled self-balancing robots, such as the Segway [171] and the Ballbot [172], are notable examples. Presumably, high-quality control of such benchmarks implies high-quality control of complex robotic systems.

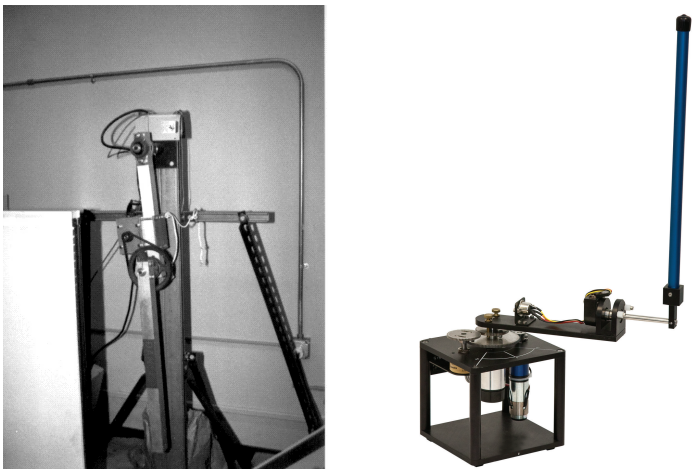


Figure 4.2. (left) Acrobot; (right) Furuta pendulum.

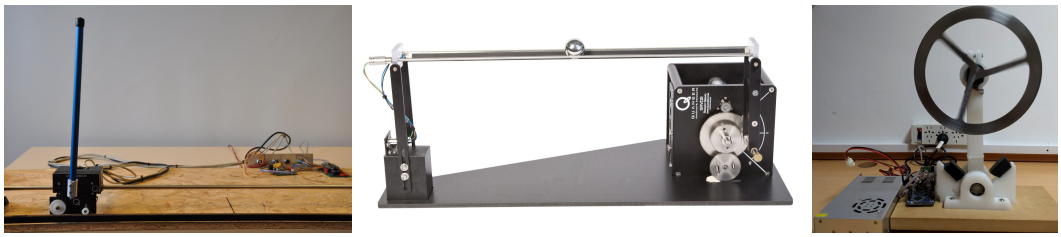


Figure 4.3. (left) cart-pole; (center) ball-and-beam system; (right) reaction wheel pendulum.

The Pendubot, as most other systems mentioned, constitutes one of the multiple facets of the inverted pendulum benchmark. Since its conception in the early 1970s, the double inverted pendulum (later called Pendubot by Spong and Block) has been used for nonlinear control research. The first control objectives were described by Sturgeon and Loscutoff in 1972 [173] and consisted of state transfer among equilibria and stabilization. In 1977, Kimura [174] pointed out observers' structure for feedback control laws, immediately followed by Furuta's stabilization controller with a CADO minicomputer in 1978 [175], later extended to the case where the system is on an inclined rail [176]. As early as the 1990s, when the now so-called Pendubot was settling itself in control engineering education, many works started

using this benchmark to validate different control schemes. Some examples are partial feedback linearization [177], fuzzy-PD control [178], *state-dependent Riccati equation* (SDRE) control [179], and Lyapunov stabilization [180].

Recent developments in nonlinear control theory have provided us with a rich collection of powerful and successful control schemes. They show that Pendubot continues to prove its usefulness in illustrating numerous new ideas in the field. A nonlinear tracking control with sliding modes was validated on the Pendubot in [181]. Similarly, an almost-global asymptotic tracking controller is described in [182]. The work in [183] considers the swing-up problem and proposes an efficient model-based policy search algorithm to solve it. Fourier transformation and particle swarm optimization are applied to the stabilization problem in [184]. Swing-up control and stabilization are considered in [185], where the virtual holonomic constraints approach is employed.

4.3 Control architecture

4.3.1 Problem formulation

Throughout this chapter, we will consider the following NMPC tracking formulation as the OCP that needs to be solved numerically:

$$\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1},\\ s_0,\dots,s_N}} \quad \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(x_i,u_i)-\tilde{\eta}_i\|_W^2 + \frac{1}{2}\|\eta_N(x_N)-\tilde{\eta}_N\|_{W_N}^2 + \frac{\rho}{2}\sum_{i=0}^{N}\|s_i\|_2^2 \tag{4.1a}$$

$$\text{s.t.} \quad x_0 - \bar{x}_0 = 0, \tag{4.1b}$$

$$x_{i+1} - F_p(x_i,u_i) = 0, \qquad i=0,\dots,N-1, \tag{4.1c}$$

$$u^l \le u_i \le u^u, \qquad i=0,\dots,N-1, \tag{4.1d}$$

$$\dot{q}_i \le \dot{q}^u + s_i, \qquad i=1,\dots,N, \tag{4.1e}$$

$$\dot{q}_i \ge \dot{q}^l - s_i, \qquad i=1,\dots,N, \tag{4.1f}$$

$$s_i \ge 0, \qquad i=0,\dots,N, \tag{4.1g}$$

where $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ represent the state and input of the Pendubot, respectively. Its discrete-time dynamics are described by $F_p : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$. The weighting matrices are denoted by $W, W_N \succ 0$, while $\rho \in \mathbb{R}^{n_s}$ denotes the real-valued vector associated with the penalty on the slack variables $s_i \in \mathbb{R}^{n_s}$. The input bounds $u^l < u^u \in \mathbb{R}^{n_u}$ and the joint velocity bounds $\dot{q}^l < \dot{q}^u \in \mathbb{R}^2$ represent the system's physical and operational limitations, respectively. The functions $\eta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\eta_N : \mathbb{R}^{n_x} \to \mathbb{R}$ represent the stage and terminal output functions, respectively. They are defined as

$$\eta(x_i,u_i) := \begin{pmatrix} x_i \\ u_i \end{pmatrix}, \quad \eta_N(x_N) := x_N. \tag{4.2}$$

Analogously, the time-varying references are denoted by $\tilde{\eta}_i$ and $\tilde{\eta}_N$. Finally, $\bar{x}_0$ is a parameter describing the current state estimate, and N is the horizon length.

Reference trajectory In the original work which this chapter describes, a learned policy based on the *deep deterministic policy gradient* (DDPG) by Lillicrap et al. [186] generates the reference trajectory. However, as *reinforcement learning* (RL) techniques are not in the scope of this thesis, we will limit ourselves to briefly describing the role of the NMPC in leveraging what is considered the most significant drawback in RL, its *unconstrained* nature. At each sampling instant, starting from $\bar{x}_0$, the learned policy is applied to the dynamic model (3.8) over a fixed prediction horizon. The resulting state-control trajectory is passed as a reference to the NMPC, which in turn enforces constraints (4.1b)–(4.1g) and, hence, guarantees safety. In particular, the trajectory is generated by forward integration through an efficient ERK4 scheme. The interested reader can refer to the work in [1] for more detail on the RL approach used to learn the offline policy that solves the desired task.

4.4 Experimental results

This section presents the experimental validation of the NMPC controller on the real Pendubot. We consider the problem of swinging up and balancing the robot around the up-up equilibrium, i.e., $x_f = (0, 0, 0, 0)$, starting from the stable down-down equilibrium, i.e., $x_0 = (-\pi, 0, 0, 0)$ and satisfying the imposed constraints. In principle, the swing-up can be performed either rapidly or with an energy pumping maneuver, where the desired final state x_f is reached through an oscillatory motion. In both cases, it is common to use a local controller when the current state estimate $\bar{x}_0$ is sufficiently close to x_f. The controller is usually based on a linearization of the system around the final equilibrium point, where the balancing phase starts. In this regard, we employ an LQR. Furthermore, as the robot only accounts with shaft encoders, angular velocities are numerically derived from position measurements.

LQR and NMPC controllers work with a sampling time of $t_s = 2$ ms. The NMPC considers $N = 10$ shooting intervals and an ERK4 integration scheme to discretize the dynamics (3.1). The weighting matrices and the slack penalty are $W = \text{diag}(3, 3, 1, 1, 5)$, $W_N = \text{diag}(3, 3, 1, 1)$, and $\rho = 40 \cdot \mathbf{1_2}$. Finally, the input bounds are $u^l = -0.4$ N·m, $u^u = 0.4$ N·m, while the joint velocity bounds are $\dot{q}^l = (-7.8, -5)$ rad/s, $\dot{q}^u = (7.8, 5)$ rad/s.

4.4.1 Hardware description

The proposed controller is experimentally validated using the Quanser Pendubot, whose physical parameters are summarized in Table 3.1. It has two incremental optical encoders, one per joint, with resolutions of 1/8192 (first joint) and 1/4096 (second joint) on lap angle. The actuator is a DC motor with permanent magnets driven in current by an UPM stage that uses PWM at 40 kHz as the input signal. It is well-known that PWM corresponds to the voltage applied to the motor, which in turn represents the setpoint for the associated torque. For the Pendubot, these voltages are internally limited to ±10 V. Moreover, the physical interface comprises two cards: one for data acquisition, and A/D or D/A conversion with low-pass filters, called MultiQ-PCI[5] (a branded version of the Sensoray Model 626 multifunction I/O

[5] `https://docs.quanser.com/quarc/documentation/multiq_pci.html`

card); and another with connectors for the input-output channels.

4.4.2 Software interface

One can control the system through software interfaces that include MATLAB/Simulink or plain `C/C++` code. In the first case, the manufacturer provides Simulink blocks that rely on proprietary software code to communicate with some MATLAB tools. In particular, the *Real-Time Workshop*, which performs automatic `C++` code generation from Simulink models, and the *Real-Time Windows Target*, which executes Simulink models in real-time. In the second case, application programs written in `C/C++` can be developed using the *Model 626 driver*[6], an API that serves as an interface between the application and the operating system. In practice, any application that accesses the functions in the API must dynamically link to the corresponding shared library before calls are made to any of its functions, e.g., enable communication with the board, read a counter's latch registers, write an analog output setpoint on a D/A converter channel, etc.

Despite the tremendous visual appeal and the potential real-time capabilities of Simulink, we opt for a control framework wholly written in `C`, prioritizing performance. The framework depends on some of the API functions to retrieve position measurements and command the actuator and a second plain `C` code that solves NLP (4.1) online. The code is generated by the `ACADO` CGT through its `C++` interface, and implements the RTI-based NMPC controller alongside the $\mathcal{O}(N^3)$ condensing technique. The QP subproblems are solved using the dense linear algebra QP solver `qpOASES`.

4.4.3 Performance analysis

Figure 4.4 shows that the NMPC successfully performs the swing-up task despite the velocity constraints. Let us recall that the LQR is used to stabilize the system around the final equilibrium point x_f, which explains the oscillatory behavior from roughly $t = 2$ s to $t = 4$ s. As mentioned earlier, the reference trajectory generated by the learned policy is kinodynamically infeasible. Yet, it is a reasonable warm-start for the SQP-based algorithm that, through an iterative refinement, generates a trajectory that is, in principle, feasible. The use of soft constraints in the NMPC formulation becomes necessary to recover the local feasibility, as model inaccuracies and noise may cause the opposite. Although this approach does not necessarily guarantee that the constraints will be satisfied, if possible, it does allow a new control input to be computed. In fact, we observe that the introduction of slack variables causes the minor constraint violations shown by the insets of Figure 4.4. The violations are about 0.1 rad/s and 1 rad/s, respectively, on the first and second joint velocities. As we will see in detail in Chapter 6, one can alleviate this effect by using the theory of exact penalty functions. This is done by introducing a non-smooth ℓ_1 penalty function whose value becomes greater than zero only when necessary.

[6] `https://github.com/jbrindza/Sensoray626-MATLAB-Interface/blob/master/doc/s626_programming_manual.pdf`

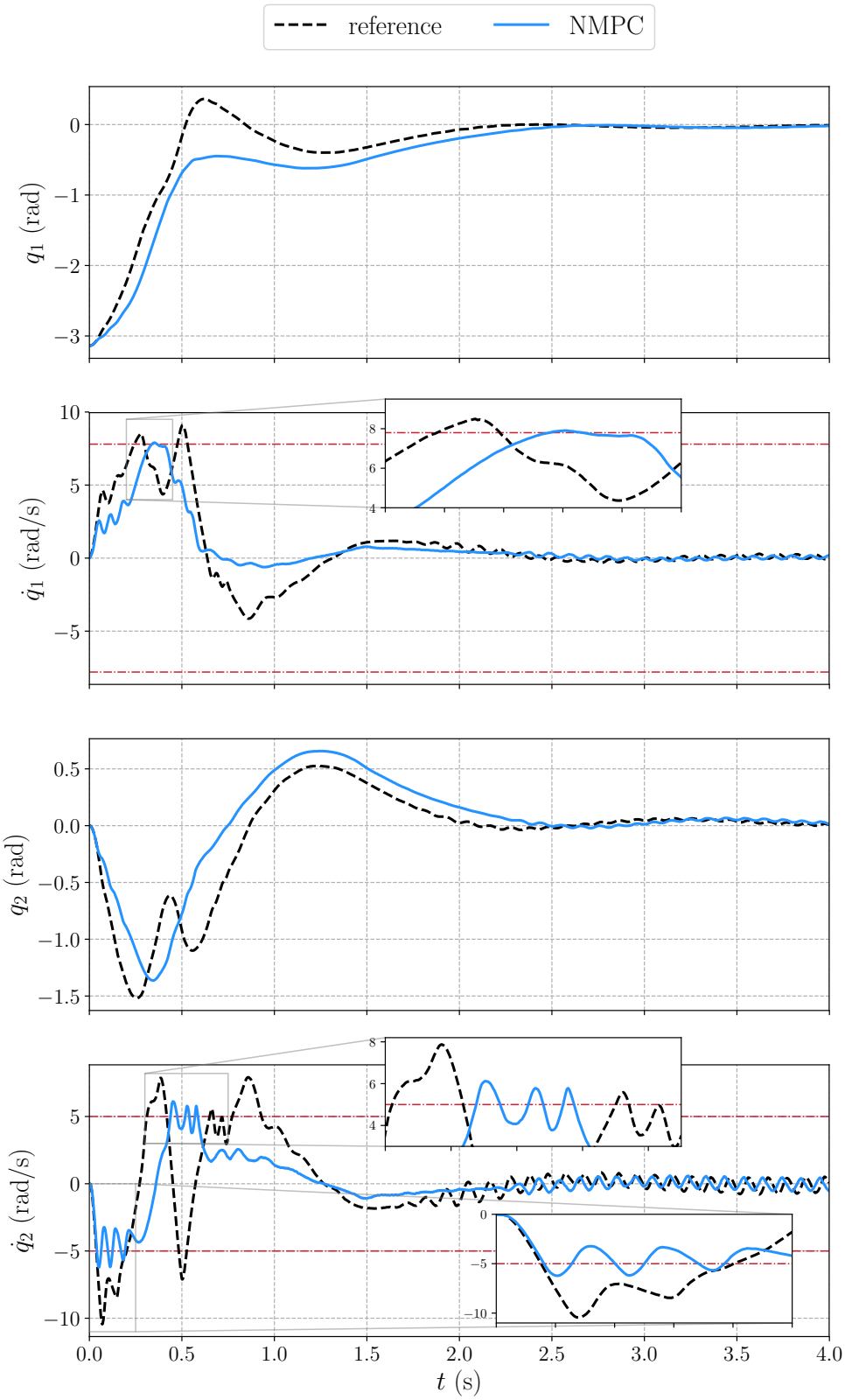


Figure 4.4. Generated trajectories during the swing-up and balancing task; red dashed lines represent the joint velocity bounds; insets show the moment when the bounds are violated.

Moreover, we evaluate the computational complexity of our NMPC controller. The results reported here are from a PC equipped with an Intel i7-4770 @ 3.40 GHz running Windows 7. The average solution time associated with the resulting RTI scheme in `ACADO` is 98.6 μs. This result shows the efficiency of the proposed NMPC. The overhead for non-MPC computations, i.e., data handling, state estimation, policy calculation, forward integration, and log-file writing, is fairly constant and is around 1.5 ms. Note that condensing is well suited for this application due to the small-scale dynamic system and the short prediction horizon. When employing this NMPC controller to high-dimensional robotic systems described by balancing motion, one shall consider either solving the sparse problem or the partially condensed one. This happens because, for a fixed prediction horizon N, an increase in the state/control ratio leads to an increase in the number of affine constraints related to the optimization variables vector in the condensed QP space, which directly implies that the problem is much more difficult to solve. In practice, `qpOASES` would possibly perform more working set recalculations, yielding longer solution times.

4.5 Chapter summary

This chapter presented the design and implementation of an NMPC controller with constraint enforcement for tracking unstable equilibrium points in self-balancing robotic systems. The controller was experimentally validated on a double inverted pendulum, named Pendubot, delivering solution times in the microsecond range. According to the results, the condensing technique proved to be competitive for short horizons, e.g., $N \leq 10$, and low state/control ratio, allowing computational efficiency. Due to model uncertainties and noisy data, the operational constraints were relaxed using slack variables to avoid feasibility issues in the practical context. Yet, bound violations were observed, implying that, for this setting, the penalty weight must be larger than the one considered. Ultimately, this work aims to motivate researchers to use NMPC to solve tracking problems in other robotic systems described by balanced motion.

Chapter 5

An efficient real-time NMPC for quadrotor position control under communication time-delay

5.1 Motivation and contribution

NMPC is an advanced control technique that takes into account nonlinear dynamics and constraints in the problem formulation, leading to significant performance improvements compared to classical control solutions. At the price of a heavier computational burden, one solves a parametric OCP at each sampling instant, which can still cause latency and throughput if the time scale for feedback is too short. Fortunately, the increasing available computational power and the advances in optimization strategies and algorithms have enabled the numerical solution of such OCPs by now. Further research has shed light on algorithms that approximate the optimal feedback policy so that real-time NMPC could also be viable and, hence, applicable to systems with fast dynamics.

Quadrotors are well-known examples of agile systems. They are characterized by their underactuation, nonlinearities, bounded inputs, and, in some cases, communication time-delays. The development of their maneuvering capability poses some challenges that cover dynamics modeling, state estimation, trajectory generation, and control. The latter, in particular, must be able to exploit the system nonlinear dynamics to generate complex motions with high-accuracy (see, e.g., Figure 5.1).

Figure 5.1. Examples of high-accuracy motions on quadrotors.

However, the presence of communication time-delay is known to highly degrade control performance.

This chapter presents an efficient position control architecture based on real-time NMPC with time-delay compensation for quadrotors. The architecture consists of predicting the state over the delay time interval, granting a nominally delay-free model in the NMPC formulation, which takes into account the input bounds. The NMPC algorithm is built upon the RTI scheme with Gauss-Newton Hessian approximation, and it is implemented using the high-performance software package `acados`. We use the automatic differentiation and modeling framework `CasADi` to provide sensitivity calculations. The QP solver is the Riccati-based interior-point method `HPIPM`, which relies on the hardware-tailored linear algebra library `BLASFEO`. We show that by using a Hessian condensing algorithm particularly well-suited for a structure-exploiting QP solver, a considerable speed-up can be achieved with respect to a state-of-the-art NMPC solver. To experimentally validate the architecture, we use the Crazyflie 2.1 nano-quadrotor, an open-source off-the-shelf platform for research and education in robotics (see Figure 5.2 for an illustration). The main contributions of this work are:

- An experimental validation of our architecture.
- An efficient software implementation that needs to cope with resource-constrained hardware.

5.2 Related works

One important precondition for employing NMPC in the context of agile systems is the availability of approximate schemes that enable real-time feasibility. Particularly successful schemes trade control performance for speed and typically hinge on implementations of the: Newton-type method [187, 4], augmented Lagrangian method [188, 189], and C/GMRES method [190, 191]. Solution times can be further reduced by building efficient condensing approaches and structure-exploiting solvers. Examples of numerical algorithms for condensing linear algebra are presented in [31, 192, 98, 193, 194], while [195, 196, 101, 197] propose algorithms for structure-exploiting in optimal control problems.

The effects of delays on output feedback control of dynamical systems is an issue that has been extensively studied in the literature over the years. However, given the growing number of applications requiring collective interaction, the most recent efforts are in the field of multi-agent control. Particularly, [198] present the design and validation of a distributed backstepping controller for multi-quadrotor systems subject to network delays. The controller has a stable delay-independent structure under the influence of non-constant distributed delays. In [199], a time-delay tolerant control law is synthesized to comply with collective aerial interaction during a simulated transport operation. The interconnected dynamics feature a communication delay exposed in the time-delayed consensus protocol and handled by a PID controller. In [200], the authors combine a Lyapunov-based strategy with a linear matrix inequality technique to derive sufficient conditions for the control gain design that ensure asymptotic consensusability in the constant delay case and

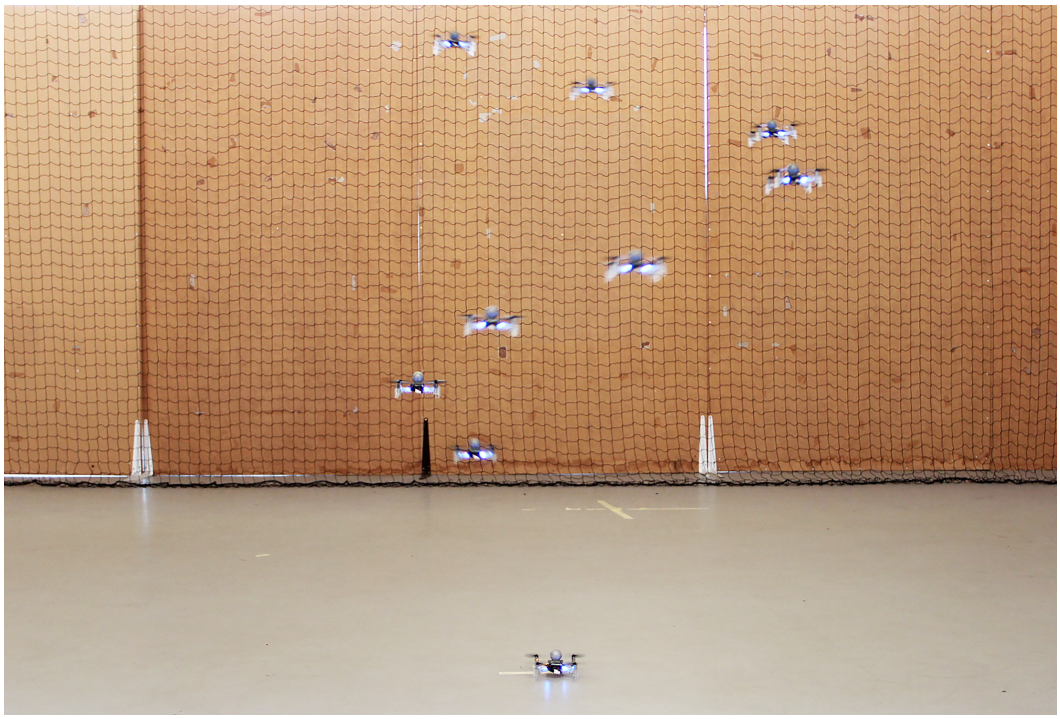


Figure 5.2. Composite image showing the real-time NMPC with time-delay compensation being used on the Crazyflie during the tracking of a helical trajectory. Given the current measurement, the state is predicted over the delay time interval using an integrator and then passed to the NMPC, which takes into account the input bounds. The efficiency of the proposed architecture leans on high-performance software implementations that span: RTI scheme to address the NMPC problem, Hessian condensing algorithm suited for partial condensing, structure-exploiting QP solver, and hardware-tailored LA library. A video of the experiments is available at `https://youtu.be/xZLVQ7BdUHA`.

consensus with bounded errors in the time-varying delay case. Although these strategies are encouraging, the resulting feedback control laws are not optimal.

5.3 Control architecture

5.3.1 State predictor

Because of the radio communication latency present in our aerial system, we have delays both in receiving measurements (τ_1) and sending control inputs (τ_2). Likewise, since we intend to use NMPC, the potentially high computational burden (τ_c) associated with its solution becomes an element that must also be taken into account to minimize the state prediction error.

One-way latency is inherently difficult to measure due to the clock synchronization dilemma, an issue that is usually solved by measuring the *round-trip time* (RTT). In this context, the sum of network latencies must be considered and can be deliberately placed where it is most convenient, as long as the RTT is not changed. Following these considerations, we propose a state predictor based on the RTT as a delay compensator. The prediction is computed by performing forward iterations of

system (3.33), starting from the current state estimate and over the RTT, through an ERK4 integrator. Due to this operation's independent nature, one can achieve perfect delay compensation by adjusting the integration step equal to the RTT. Thus, this chapter assumes that there is a fixed RTT, defined by $\tau_r := \tau_1 + \tau_2 + \tau_c$, to be compensated. Although this prediction is simple (and inaccurate), it will prove surprisingly effective in experimentation.

5.3.2 Problem formulation

Employing direct multiple shooting to discretize the underlying continuous-time OCP and assuming a linear least squares objective, the NMPC problem considered in this chapter is defined as the following constrained NLP:

$$\begin{aligned} \min_{\substack{\xi_0,\dots,\xi_N,\\ u_0,\dots,u_{N-1}}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(\xi_i,u_i)-\tilde{\eta}_i\|^2_W + \frac{1}{2}\|\eta_N(\xi_N)-\tilde{\eta}_N\|^2_{W_N} \\ \text{s.t.} \quad & \xi_0 = \hat{\xi}(k+\tau_r), \\ & \xi_{i+1} - F_c(\xi_i,u_i) = 0, \quad i=0,\dots,N-1, \\ & u^l \le u_i \le u^u, \qquad\qquad i=0,\dots,N-1. \end{aligned} \tag{5.1}$$

Therein, $\xi := (\xi_0,\dots,\xi_N)$ and $u := (u_0,\dots,u_{N-1})$ denote the state and input trajectories of the discrete-time system whose dynamics are described by $F_c : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\xi}$. The functions in the stage and terminal least squares terms are denoted by $\eta : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\eta_N : \mathbb{R}^{n_\xi} \to \mathbb{R}$, respectively. Variables $\tilde{\eta}_i \in \mathbb{R}^{(n_\xi+n_u)}$ and $\tilde{\eta}_N \in \mathbb{R}^{n_\xi}$ represent the time-varying references. The weighting matrices are denoted by $W, W_N \succ 0$. The inputs bounds are $u^l < u^u \in \mathbb{R}^{n_u}$. Finally, N denotes the horizon length, and $\hat{\xi}(k+\tau_r)$ represents the system's estimated state at the current time instant k.

5.3.3 Condensing approach and structure-exploiting QP solver

Among other methods to approximate the Hessian of the Lagrangian, a particularly successful one is the GGN method, which is applicable when the objective function is of least squares type, as in NLP (5.1). After applying it, the resulting QP has a sparse banded structure and can be rewritten as:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\sum_{i=0}^{N} w_i^T H_i w_i + h_i^T w_i \\ \text{s.t.} \quad & \Delta\xi_0 = \hat{\xi}(k+\tau_r) - \xi_0^n, \\ & E_{i+1}w_{i+1} - C_i w_i - d_i = 0, \quad i=0,\dots,N-1, \\ & g_i^l \le G_i w_i \le g_i^u, \qquad\qquad i=0,\dots,N-1, \end{aligned} \tag{5.2}$$

where $w_i = (\Delta\xi_i\,;\Delta u_i)$, $w_N = \Delta\xi_N$ with state and control deviations defined as $\Delta\xi_i := \xi_i - \xi_i^n$, $\Delta u_i := u_i - u_i^n$. The superscript n refers to the linearization points at the previous QP iteration. To match the QP formulation used in `acados`, we define: $C_i := (A_i\ B_i)$, $E_i := (\mathbf{I}\ 0)$, $G_i := (0\ \mathbf{I})$, for $i = 0,\dots,N-1$, and $E_N := \mathbf{I}$.

The matrices used in the linearized dynamics and the upper and lower bounds of the polyhedral constraints are defined as

$$\begin{aligned} A_i &:= \nabla_\xi F_c(\xi_i^n, u_i^n)^T, \quad B_i := \nabla_u F_c(\xi_i^n, u_i^n)^T, \\ d_i &:= F_c(\xi_i^n, u_i^n) - A_i \xi_i^n - B_i u_i^n, \\ g_i^l &:= u^l - \mathbf{I} u_i^n, \quad g_i^u := u^u - \mathbf{I} u_i^n. \end{aligned}$$

Finally, we have that the Hessian and the gradient terms for $i = 0, \ldots, N-1$ are defined as

$$H_i := W_i = \begin{pmatrix} Q_i & 0 \\ 0 & R_i \end{pmatrix}, \quad h_i := \begin{pmatrix} q_i \\ r_i \end{pmatrix},$$

where

$$\begin{aligned} Q_i &= \nabla_\xi(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n) W^\xi \nabla_\xi(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n)^T \\ R_i &= \nabla_u(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n) W^u \nabla_u(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n)^T \\ q_i &= \nabla_\xi(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n)(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n) \\ r_i &= \nabla_u(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n)(\eta(\xi_i^n, u_i^n) - \tilde{\eta}_i^n), \end{aligned}$$

with W^ξ and W^u denoting the blocks in W related to states and inputs respectively, and

$$\begin{aligned} Q_N &= \nabla_\xi(\eta_N(\xi_N^n) - \tilde{\eta}_N^n) W_N \nabla_\xi(\eta_N(\xi_N^n) - \tilde{\eta}_N^n)^T \\ q_N &= \nabla_\xi(\eta_N(\xi_N^n) - \tilde{\eta}_N^n)(\eta_N(\xi_N^n) - \tilde{\eta}_N^n). \end{aligned}$$

One way to address the QP (5.2) is to exploit its banded structure and apply a sparse QP solver. Another way is first to reduce or condense the variable space of the QP by eliminating all state deviations through the continuity constraint of (5.2) and then apply a dense QP solver. Differently, in this work, we employ a recently proposed approach, in-between the sparse and the condensed one, called partial condensing. In particular, we use a Hessian condensing algorithm where a state component is retained as an optimization variable at each partially condensed QP stage [99]. The algorithm finds the optimal level of sparsity for the QP solver at hand, trading horizon length for input vector size. Thus, partial condensing is employed as a preparation step before the call to `HPIPM`, the Riccati-based interior-point method solver we use, which has been tailored to exploit the particular structure of the obtained QP.

5.4 Simulation results

This section is divided into two parts. First, it assesses the performance of two optimal controllers in simulation. In particular, an LQR and an NMPC will be considered. Such comparison is valuable since the LQR can perform an optimal control law within short execution times and with a low computational burden, while the NMPC deals with constraints and predictions. The values of the parameters appearing in the model are listed in Table 3.2. Accounting for numerous computation time uncertainties, a sampling time of $\tau_s = 15$ ms is set for both controllers in this chapter. Second, it investigates the effect of the time-delay on the nano-quadrotor system's control performance, considering different RTTs.

5.4.1 LQR controller

The first controller considered is the LQR. During the LQR control design, the nonlinear dynamics (3.33) are linearized around the hovering steady state and input $(\bar{\xi}, \bar{u})$ and discretized using τ_s as sampling time, yielding:

$$\xi_{k+1} - \bar{\xi} = A(\xi_k - \bar{\xi}) + B(u_k - \bar{u}). \tag{5.3}$$

Due to the uncontrollable linearized quaternion dynamics, the local linear model (5.3) is also not controllable. To address this issue, we project the dynamics onto a controllable subspace as done in [201, 152], using the fact that the first component of the quaternion vector can be eliminated thanks to the relation $q_w = \sqrt{1 - q_x^2 - q_y^2 - q_z^2}$. Thus, we obtain the 12-state model needed to project the LQR static gain. A discrete-time LQR controller can be designed by finding the infinite horizon solution P associated with the discrete-time algebraic Riccati equation

$$A^T P A - P - (A^T P B)(B^T P B + R)^{-1}(B^T P A) + Q = 0,$$

which yields the static feedback law $u = K\xi + \bar{u}$, where

$$K = (B^T P B + R)^{-1}(B^T P A).$$

The positive-definite matrices Q and R are defined so that the stability of the closed-loop system is guaranteed.

5.4.2 NMPC controller

The second controller is the NMPC. When designing an NMPC, choosing the horizon length has profound implications for computational burden and tracking performance. For the former, the longer the horizon, the higher the computational burden. As for the latter, in principle, a long prediction horizon tends to improve the controller's overall performance. To select this parameter and achieve a trade-off between performance and computational burden, NLP (5.1) has been implemented in `acados` considering $N = \{10, 20, 30, 40, 50\}$, discretizing the dynamics (3.33) using an ERK4 integration scheme. Likewise, we compare the condensing approach with the state-of-the-art solver `qpOASES` against the partial condensing approach with `HPIPM` for the set of considered horizons. The weighting matrices W, W_N are the same in all simulations. The functions in the stage and terminal least squares terms are defined as

$$\eta(\xi_i, u_i) := \begin{pmatrix} \xi_i \\ u_i \end{pmatrix}, \quad \eta_N(\xi_N) := \xi_N. \tag{5.4}$$

Remark 2. *We regulate for $q^d = (1, 0, 0, 0)$ so that the geodesic arc can be locally approximated by a straight line.*

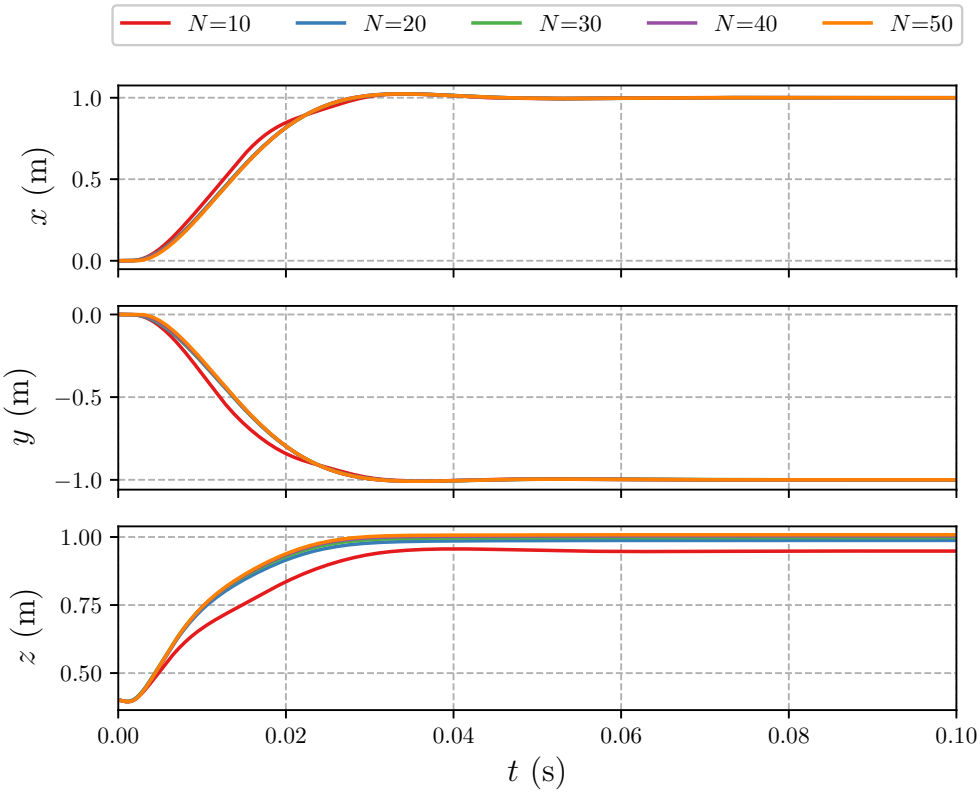


Figure 5.3. Closed-loop trajectories for different horizon lengths.

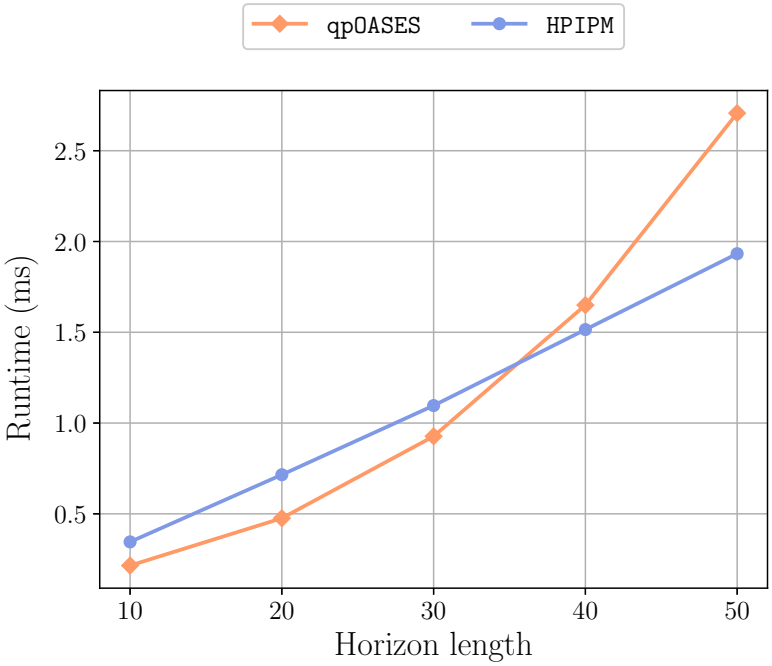


Figure 5.4. Average runtimes per SQP-iteration for five different horizons considering two distinct QP solvers for QP (5.2).

According to Figure 5.3, horizon $N = 50$ provides a higher tracking performance. Nevertheless, when considering a long horizon, the available computational power may not be sufficient to deliver a solution within the runtime requirements of an aerial system. In Figure 5.4, the average runtimes per SQP-iteration are reported. As `qpOASES` is a solver based on the active-set method and dense LA, it requires condensing to be computationally efficient. In line with the observations found in the literature that condensing is effective for short to medium horizon lengths, we note that `qpOASES` is competitive for horizons up to approximately $N = 30$ compared to `HPIPM`. The break-even point moves higher on the scale for longer horizons, mainly due to efficient software implementations that cover: i) Hessian condensing procedure tailored for partial condensing, ii) structure-exploiting QP solver based on novel Riccati recursion, iii) hardware-tailored linear algebra library.

Furthermore, when dealing with a nominal NMPC formulation, as in this chapter, the model uncertainties and their propagation also contribute to the horizon length's decision. The longer the prediction, the more the modeling errors accumulate, and the tracking performance can no longer be guaranteed. Uncertainty can be added to the system model to account for the noise associated with the onboard *inertial measurement unit* (IMU) and simulate the variability in the repetition of the task. However, this has not been included in the scope of this work.

Therefore, we can conclude that horizon $N = 50$ offers a reasonable trade-off between deviation from the reference trajectory and computational burden.

5.4.3 Comparison

This section is dedicated to compare and discuss the simulation results for LQR and NMPC. Initially, the LQR controller was tuned so that its control policy was locally equivalent to the NMPC one. However, as also experienced in the work of [152], the simulations showed the need to detune the controller so that the performance – specifically for the position in z in our case – was acceptable. Thus, the following weighting matrices are chosen:

$$\begin{aligned} Q = \text{blkdiag}(&12.24 \cdot 10^3, 10.2 \cdot 10^3, 9 \cdot 10^5, 0.102 \cdot \mathbf{I_3}, 71.4, \\ &102, 408, 1.02 \cdot 10^{-3} \cdot \mathbf{I_2}, 1.02 \cdot 10^3),\ R = 0.12 \cdot \mathbf{I_4}. \end{aligned}$$

For the NMPC, the weighting matrices are

$$\begin{aligned} W &= \text{blkdiag}(120, 100 \cdot \mathbf{I_2}, 1 \cdot 10^{-3} \cdot \mathbf{I_4}, 7 \cdot 10^{-1}, 1, 4, 1 \cdot 10^{-5} \cdot \mathbf{I_2}, 10, 6 \cdot 10^{-2} \cdot \mathbf{I_4}), \\ W_N &= 50 \cdot \text{blkdiag}(120, 100 \cdot \mathbf{I_2}, 1 \cdot 10^{-3} \cdot \mathbf{I_4}, 7 \cdot 10^{-1}, 1, 4, 1 \cdot 10^{-5} \cdot \mathbf{I_2}, 10), \end{aligned}$$

while the input bounds on the propellers' rotational speed are $u^l = \mathbf{0_4}$, $u^u = 22 \cdot \mathbf{1_4}$ krpm.

The simulation scenario consists of producing steep reference changes in position. Figure 5.5 shows the comparative results concerning the closed-loop tracking performances. We notice that the NMPC controller outperforms the LQR, considering that absence of overshoots and faster response to changes in references can be achieved. This may be due to the fact that nonlinearity and unmodeled constraints can degrade the LQR controller's performance for large reference changes.

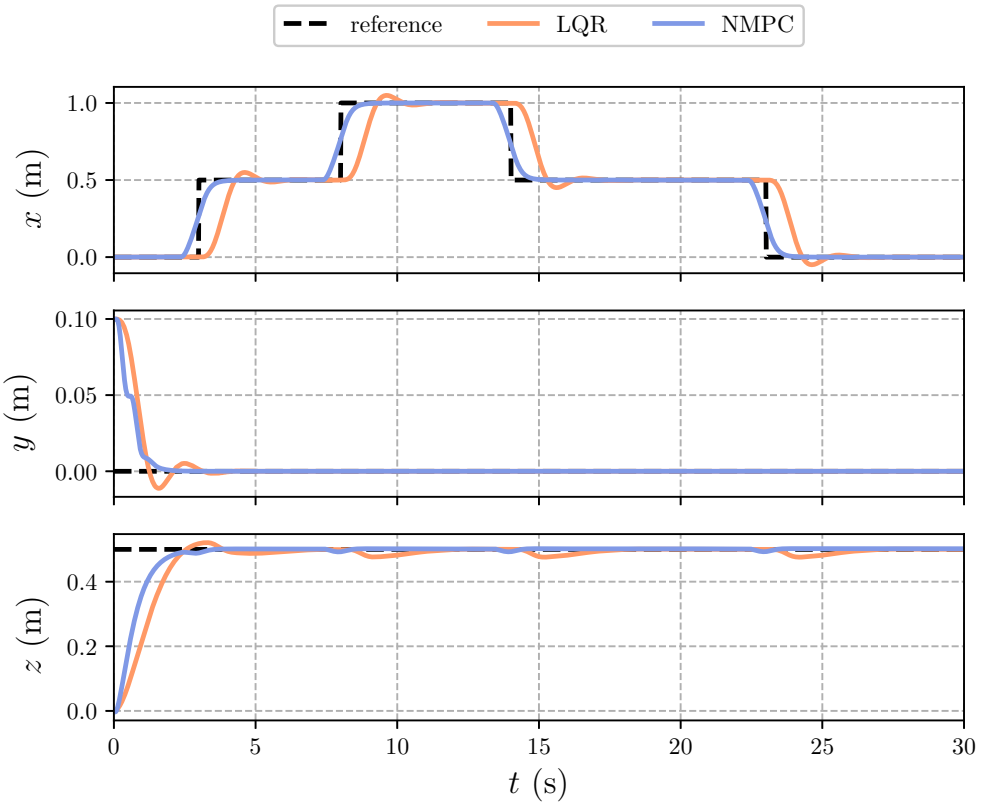


Figure 5.5. Simulation results comparing different controllers: closed-loop trajectories for position tracking.

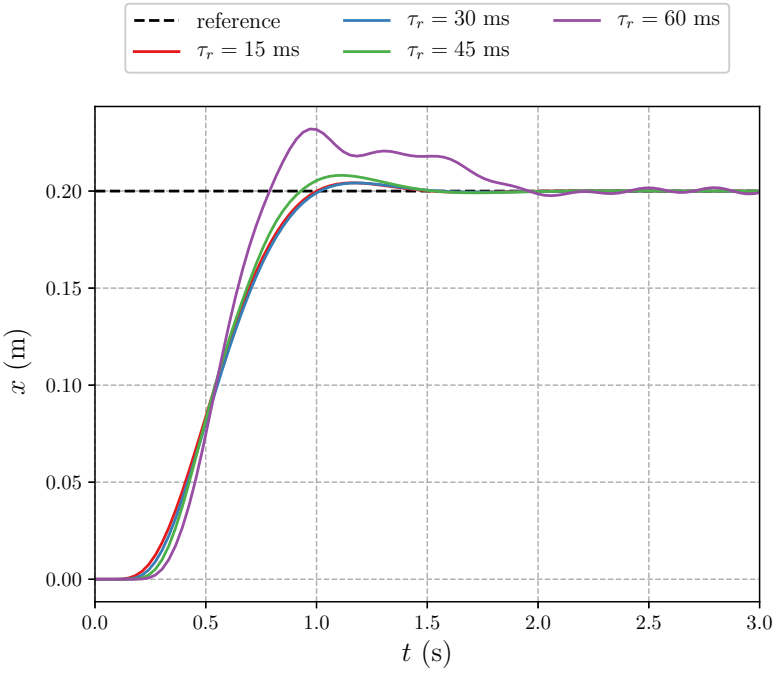


Figure 5.6. Step response of the Crazyflie NMPC without considering the delay compensation for four distinct RTT.

5.4.4 Time-delay

For the purposes of this analysis, we assume that $\tau_r = \lambda\tau_s$ for some nonnegative integer λ. However, we emphasize that the same analysis can be done considering any value of τ_r, as mentioned in Section 5.3.1. The results for the step response of the proposed NMPC without considering the time-delay compensation are shown in Figure 5.6. We observe that the greater the time-delay associated with the RTT, the greater the NMPC performance's degradation. This outcome is observed through the increase of overshoots and oscillations in the system's response, indicating that the proposed NMPC cannot control the system properly. Unlike in simulation, in a real-world context, where model uncertainties come into view, the overshoots and oscillations will tend to be even more significant. Besides, other simulations have shown that the system becomes unstable for greater RTTs. These observations indicate that delay compensation is of particular importance in this application.

5.5 Experimental results

5.5.1 Hardware description

The proposed architecture is experimentally validated on the Crazyflie 2.1 nano-quadrotor, developed by the Swedish company Bitcraze[1]. One can control the system through the onboard long-range radio receiver and its correspondent transmitter, the Crazyradio PA, or even through the Bluetooth LE (low-energy) connection. It has an open-source firmware based on FreeRTOS that allows the user to log sensor data at a maximum frequency of 100 Hz. The radio communication uses a protocol developed by Bitcraze, the *Crazy RealTime Protocol* (CRTP), which limits the size of the packages sent from the Crazyflie to the Crazyradio in up to 30 bytes (1-byte header, 29-bytes data payload).

The *motion capture* (MOCAP) system consists of ten Raptor-E cameras and the *Cortex* software, both from Motion Analysis. It provides the 3D global position of an IR-reflective marker placed on top of the Crazyflie (see Figure 5.7) at a frequency of 100 Hz. The entire setup has an estimated RTT of 60 ms to be compensated.

5.5.2 Software interface

The control architecture hinges upon a `ROS Kinetic` framework and runs at 66.67 Hz. The CRTP is used in combination with our `crazyflie_nmpc` stack [3] – based on the Crazyflie ROS interface [202] – to stream in runtime custom packages containing the required data to reconstruct the part of the measurement vector that depends on the IMU data. The logging subsystem is mapped into ROS messages, and, thus, it is possible to recover the current Euler angles and angular velocities of the nano-quadrotor. Likewise, the `cortex_ros` bridge streams the 3D global position of the Crazyflie from which linear velocities are estimated using a numerical derivation method.

Through the use of `acados` Python interface and `CasADi` automatic differentiation and modeling language, we export a plain `C`-code that corresponds to NLP

[1] `https://www.bitcraze.io`

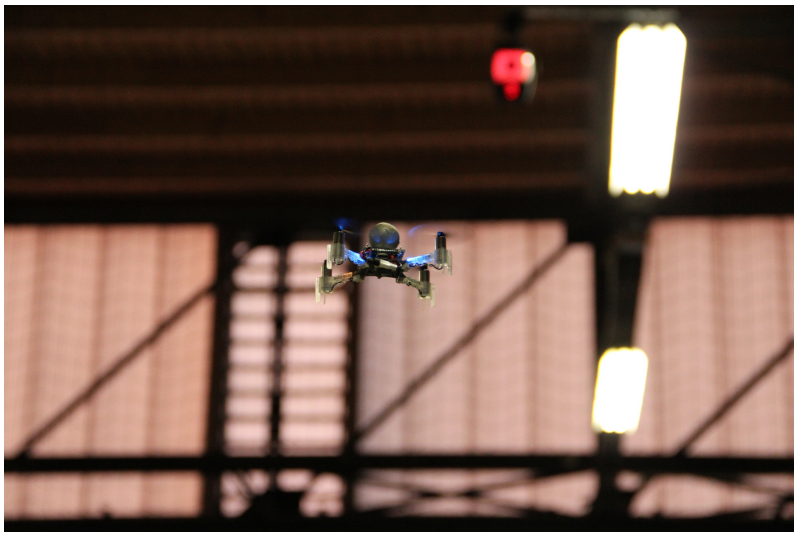


Figure 5.7. Crazyflie with an IR-marker during one of the experiments.

(5.1). The exported code is linked against the precompiled `acados` core library and then wrapped by the NMPC ROS node, written in `C++`. The QP subproblems arising in the SQP algorithm in `acados` are solved using the high-performance `HPIPM` solver, which implements an interior-point method. It is written entirely in `C` and is built on top of the high-performance linear algebra package `BLASFEO`. We use the `X64_INTEL_HASWELL` target in `BLASFEO`, which explicitly employs the powerful vector instructions provided by this computer architecture. In particular, the high-performance routines provided by `BLASFEO` outperform other state-of-the-art dense linear algebra libraries in the case of matrices of moderate size, as in the current application. Regarding the condensing approach, we use the partial condensing routines provided by `HPIPM`.

5.5.3 Implementation considerations

Although system (3.33) is used in our NMPC formulation, the standard *input commands* for the actual Crazyflie (indicated by the red variables in Figure 5.8) are:

- the desired roll ϕ^d and pitch θ^d angles, in degrees
- the yaw rate ω_z^d, in degrees per second
- the base PWM signal $\tilde{\Omega}^d$

It is important to clarify that $\tilde{\Omega}^d$ is a 16-bit integer number ranging from 0 to 65535 and stands for the base value of PWM applied to all motors to maintain altitude. Despite the name *thrust* in the official firmware[2], $\tilde{\Omega}^d$ does not directly represent the net force applied along the z body-fixed axis but is related to it. In principle, once the NMPC solution is obtained, the first element of the predicted control trajectory $u^\star(0|k)$ should be sent to the Crazyflie. However, there is an incompatibility between $\tilde{\Omega}^d$ and $u^\star(0|k)$.

[2] `https://github.com/bitcraze/crazyflie-firmware`

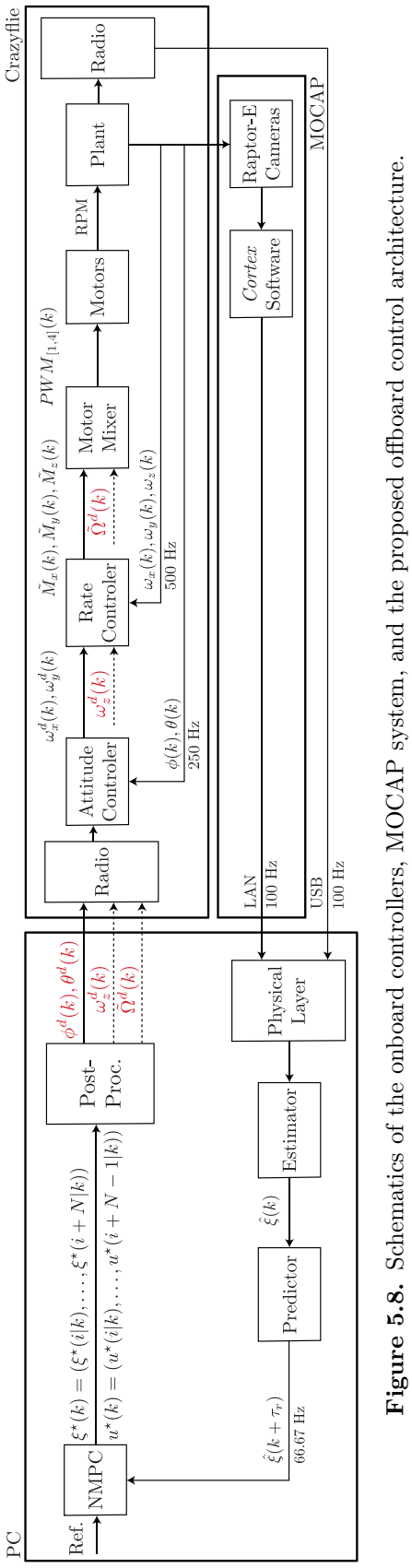


Figure 5.8. Schematics of the onboard controllers, MOCAP system, and the proposed offboard control architecture.

To translate the control inputs $\Omega_i^\star = u^\star(0|k)$, in krpm, to the base PWM signal $\tilde{\Omega}^d$, an integer, we need to establish some mappings. To that end, we use the following relations adapted from [161]:

$$\Omega^d = \frac{1}{4}\sum_{i=1}^{4} \Omega_i^\star \text{ [krpm]}, \quad \tilde{\Omega}^d = \frac{1000 \cdot \Omega^d - 4070.3}{0.2685} \text{ [int]}.$$

Similarly, at each new NMPC solution, we reconstruct the remaining input commands ϕ^d, θ^d, ω_z^d using part of the state solution $\xi^\star(i|k)$ at stage $i = 1$, as denoted in Figure 5.8. This approach, however, entails some distortions onto the onboard controllers (attitude and rate controllers) since the NMPC does not take them into account in its formulation. This effect will be described in the following subsection.

5.5.4 Onboard controller considerations

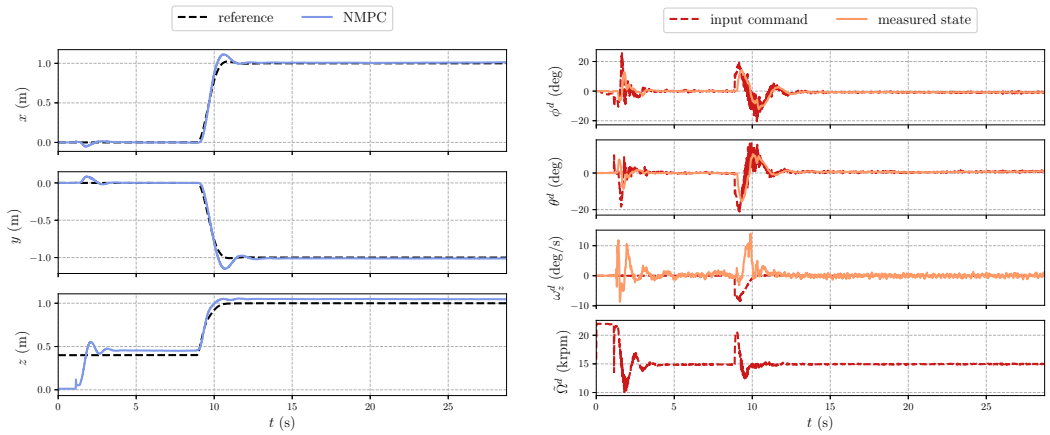
How the onboard controllers (PIDs) use the setpoints of the offboard controller (NMPC) in our architecture is not entirely conventional and, thereby, deserves some considerations. We must address two main points presented in Figure 5.8 must be addressed. First, the reference signals that the PID loops track do not fully correspond to the control inputs considered in the NMPC formulation. Instead, part of the state solution $\xi^\star(1|k)$ is used in conjunction with the control inputs $u^\star(0|k)$ to reconstruct the actual input commands passed as a setpoint to the Crazyflie. Second, a part of the reconstructed input commands is sent as a setpoint to the outer loop (attitude controller), and the other part is sent to the inner loop (rate controller).

Furthermore, as the NMPC model does not include the PID loops, it does not truly represent the real system, even in the case of perfect knowledge of the physical parameters. As a consequence, the optimal feedback policy is distorted in the real system by the PIDs. Note that intuitively reducing the NMPC discretization time mitigates the distortion. Another solution is to implement a linear interpolation onboard, considering the frequency of each loop. In this case, given a new input command from the NMPC, data points are constructed and passed as reference signals to the PIDs.

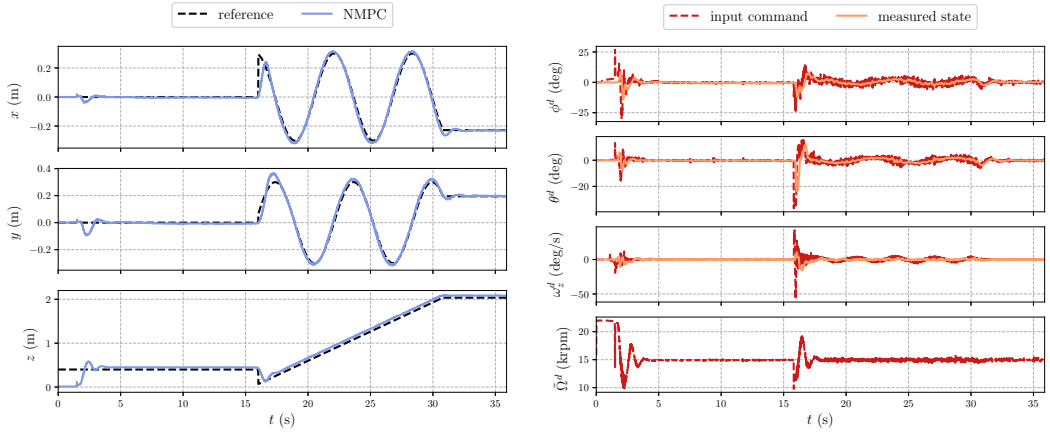
5.5.5 Experiments description

In every experiment, the Crazyflie takes off and hovers at $p^d = (0, 0, 0.4)$. This is done to avoid the unmodeled ground effect during the tracking phase. When the nano-quadrotor reaches the steady-state for hovering, the user triggers the tracking. The nano-quadrotor is then asked to hold the last point of the reference trajectory until the user triggers the landing command.

To validate the effectiveness of the control architecture, we ran two experiments. For each of them, we generate a reference trajectory on a base computer and pass it to the NMPC ROS node every $\tau_s = 15$ ms. We explicitly address the feasibility issue in the design process when generating the trajectories. In this spirit, two references are created: one feasible and one infeasible. In addressing this issue, we prove through experiments that the performance of the proposed NMPC is not



(a) Smooth step.



(b) Helical reference.

Figure 5.9. Experimental results: output trajectory and input commands.

degraded even when the nano-quadrotor attempts to track an infeasible trajectory, which could, in principle, make it deviate significantly or even crash. The reference trajectories for each experiment are:

Smooth Step For the first experiment, the reference is a dynamically feasible smooth step on (x, y, z) that drives the nano-quadrotor from the hover position to the final one, defined as $p^d := (1, -1, 1)$. To generate it, we implemented NLP (5.1) using `CasADi` with a time horizon of $T = 6$ s and $N = 400$ shooting intervals, discretizing the dynamics (3.33) using an ERK4 integration method. The resulting NLP is solved using the interior-point solver `IPOPT`, exploiting the *just-in-time* compiling function evaluations and alongside the linear solver `MA57` [203]. Figure 5.9a shows the closed-loop trajectory and input commands for this experiment.

Helical Reference For the second experiment, the reference is a cartesian helical trajectory generated with MATLAB. We imposed a radius of $r = 0.3$ m, initial height of $h_0 = 0.38$ m, height increments of $\Delta h = 0.002$ m per τ_s, and a duration

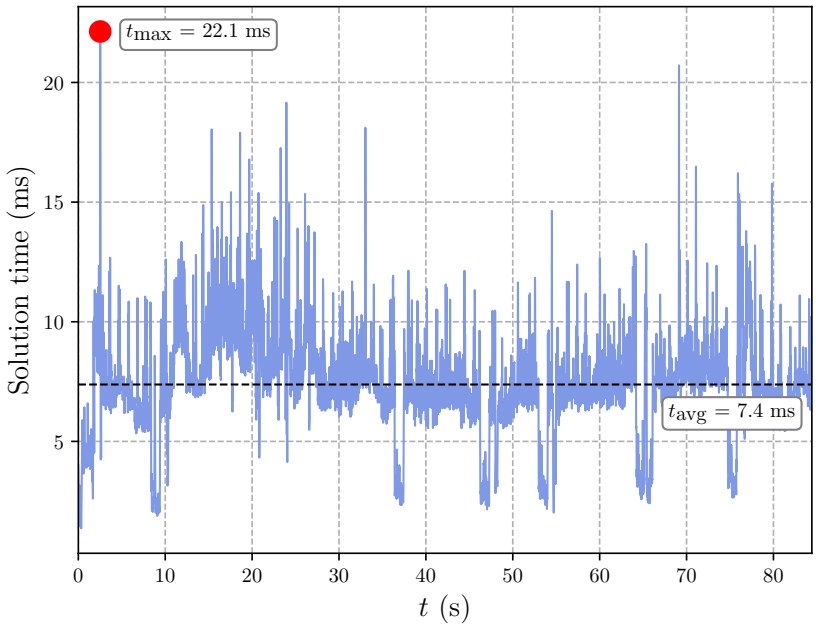


Figure 5.10. Solution times over time.

of $t_f = 15$ s, divided into $m = 1000$ intervals. Figure 5.9b presents the closed-loop trajectory and input commands for this experiment.

5.5.6 Performance analysis

From Figure 5.9, we note that the PID distortion on the NMPC performance is relatively small. As expected, the most challenging setpoints to be tracked are the positions in which, given a change in the motion, the Crazyflie has to pitch/roll in the opposite direction quickly. These are the setpoints where the distortion has the most significant influence on the system, causing the small overshoots in position.

To assess our algorithm's computational complexity, we gather the average and maximum solution times of the tailored RTI scheme using `acados`. The results were obtained on an Intel Core i5-8250U @ 3.4 GHz running Ubuntu. We had $t_{\text{avg}} = 7.4$ ms and $t_{\max} = 22.1$ ms, where the computational peak was during take-off (see Figure 5.10). Throughout the tracking phase, the computational workload offered a reasonable safety margin within the sampling time. These results show the efficiency of the proposed scheme.

5.6 Chapter summary

This chapter presented the design and implementation of a novel position controller based on NMPC for quadrotors. The control architecture incorporates a predictor as a delay compensator for granting a delay-free model in the NMPC formulation, which in turn enforces bounds on the actuators. It was experimentally validated on the Crazyflie 2.1 nano-quadrotor in two different tracking tasks. The results prove

that the efficient RTI-based scheme, exploiting the full nonlinear model, achieves a high-accuracy tracking performance and is fast enough for real-time deployment.

Chapter 6

Real-time NMPC for quadrotor motion generation in dynamic environments

6.1 Motivation and contribution

Autonomous navigation in dynamic environments still poses an important challenge for robotics research. In contrast to static scenarios, where global path planning strategies are well-suited, in dynamic environments, the decision about motion must be based on the world's online perception and the fast reaction-time behavior (see, e.g., Figure 6.1). From a technical point of view, dynamic obstacles give rise to new facets in motion planning problems. For instance, one may wish to find a path where energy consumption is minimal, thus requiring extra care with the robot's velocity and acceleration. Also, clearance from obstacles is a critical issue in this context. It becomes even more critical when dealing with cluttered environments. A path with a high clearance, notably a safer route, turns out to be over-conservative. Consequently, the robot ends up taking longer routes or suffering from the overly constrained obstacle fields to the point where no viable paths remain.

There has been a wealth of research on algorithms for motion planning in dynamic environments. While they may differ in terms of the applied technique, they all strive for the simultaneous computation of a collision-free path from start to goal and a velocity profile along the path, which can satisfy both system dynamics and actuator limits. These computations are also known as *trajectory planning*, which, as seen before, is at the heart of all motion generation algorithms. Ensuring that trajectories have these properties is difficult for several reasons: robots are typically

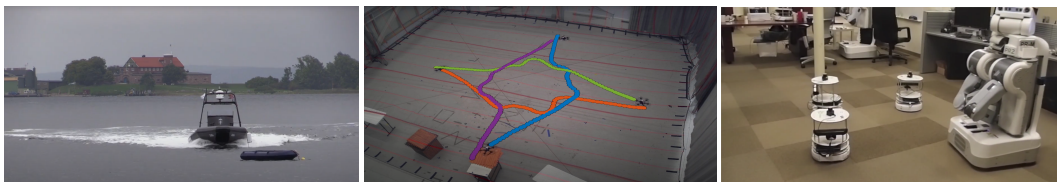


Figure 6.1. Examples of robots navigating autonomously in dynamic environments.

described by nonlinear dynamics, and it may be hard for an algorithm to cope with that; it is difficult to attest that an algorithm can achieve a tractable computation time when considering a high-fidelity model and many arbitrary obstacles.

A prominent approach to generate kinodynamically feasible trajectories in dynamic environments is NMPC. Typically, this requires solving a (possibly) nonlinear nonconvex NLP at each planning iteration, which may be intractable in principle. Indeed, a great challenge associated with the employment of NMPC schemes for motion generation has been the availability of fast and reliable algorithms for online implementations. Among other approaches, the RTI scheme is proposed in [119] to trade speed for accuracy while incorporating "globalization features". Its principle hinges upon approximate feedback policies to meet the required real-time constraints. This attribute has led to NMPC gradually become a viable solution for applications with high sampling rates, such as quadrotors.

Motivated by these observations, this chapter presents a novel approach for real-time motion generation for quadrotors in dynamic environments (see an example in Figure 6.2). More precisely, we exploit the algorithmic ideas of the RTI scheme with inexact Hessian to formulate a least conservative linearized collision avoidance constraint. We provide a theoretical analysis proving that the proposed constraint formulation is less conservative for planners based on Newton-type method than those based on a *fully converged NMPC* approach. This analysis is further leveraged to overcome the numerical difficulties associated with the (local) feasibility of the optimization problem. The framework uses the RTI strategy implementation through the high-performance software package `acados`. The QP solver is `HPIPM`, which is based on `BLASFEO`. We use the Crazyflie nano-quadrotor as a relevant example to assess the planner's performance and implementability. Concerning the challenges at hand, the main contributions of this chapter are twofold:

- An NMPC-based algorithmic framework for real-time motion generation that is reliable and guarantees less conservative optimal trajectories for quadrotors.
- A numerical validation that shows the efficiency of the proposed approach.

6.2 Related works

Traditional motion planning methods typically rely on graph-search methods [204, 205], combinatorial methods [206], or sampling-based methods [207, 208]. Despite the effectiveness of those algorithms in finding a path between two given points, the quality of the path obtained may be far from optimal. Thus, the research to improve the path refinement led to a common subdivision of the problem into a *global* and *local* planner.

The local planner accounts for the dynamic constraints and generates sets of feasible local trajectories. Popular algorithms used in this layer are *dynamic window approach* (DWA) [209] and *timed elastic band* (TEB) [210]. More recent motion planning methods focus, instead, on optimization-based strategies. The authors in [211] develop the concept of linear interval programming functions, which is based on multi-objective optimization. In this case, the collision avoidance constraint is defined as a shrinking convex polygon around the obstacle. As the optimization prob-

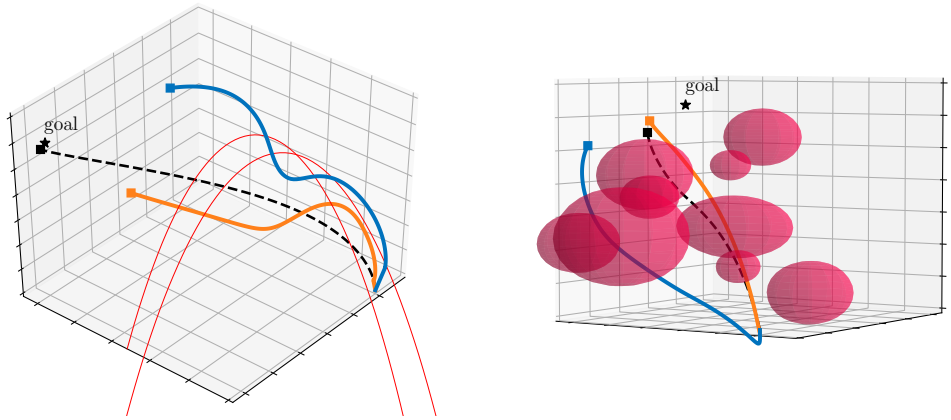


Figure 6.2. Example trajectories found by our NMPC-based local planner: with proposed constraint formulation (orange), with its counterpart (blue); reference is dashed; the red lines (left) are the moving obstacles' trajectories; the red spheres (right) are the static obstacles. A video of the simulations is available at `https://youtu.be/ZRbGyikvsxw`.

lem seeks the set of so-called Pareto optimal solutions – which is rather demanding – its application in systems with tight runtime requirements is usually not possible. Another approach is risk-constrained model predictive control [212], where expectation constraints are defined within a linearly constrained mixed-integer convex program. However, the efficiency of this method highly depends on the tightness of the continuous linear program relaxations. Other techniques involve manifold trajectory optimization [213] and convex decomposition through interval allocation [214], both exploiting the inflated ellipsoids concept to ensure collision-free trajectories. These approaches have obtained promising results, but the solutions achieved for local replanning often lead to optimal trajectories that are far too conservative.

6.3 Control architecture

6.3.1 Collision avoidance constraint

To ensure kinodynamically feasible and collision-free trajectories, we consider a constrained OCP. Having the dynamic model of the robot as a constraint provides kinodynamically feasible state trajectories. On the other hand, collision avoidance is granted by constraining the robot's distance with respect to an obstacle $o \in \mathbb{R}^n$ in the *workspace* $\mathcal{W} \in \mathbb{R}^n$ to be larger than the *clearance* $d_\mathcal{C} \in \mathbb{R}^+$. The obstacles, here considered dynamic, are described as point-mass models. Furthermore, we assume a single-body robot, which can be well approximated by a point if one defines the clearance to take into account: i) the actual size of the robot, ii) the actual size of the obstacle, iii) a safety margin between them. When extending our approach to a multi-body robot system, one needs to consider the enlargement of the clearance. Ultimately, one must include an additional collision avoidance constraint in the OCP for each obstacle present in the workspace.

To formulate the collision avoidance constraint, let us consider the *distance function* as the Euclidean distance between the robot and an obstacle in the workspace,

i.e., $d_{\mathcal{O}} : \mathcal{W} \to \mathbb{R}$. One can formally define it as the following:

Definition 12 (Distance Function)**.** *Let us define the distance function for an arbitrary point $p \in \mathcal{W}$ with respect to an obstacle as*

$$d_{\mathcal{O}}(p) = d_{\mathcal{O}}(p; o) = \|p - o\|_2,$$

and consider bounds, i.e. $d_{\mathcal{C}} \leq d_{\mathcal{O}}(p)$ and $\exists\, d_{\mathcal{M}}$ such that $d_{\mathcal{O}}(p) \leq d_{\mathcal{M}} \forall p \in \mathcal{W}$.

Based on the above definition, the obstacle avoidance constraint can be concisely written as

$$d_{\mathcal{C}} \leq \|p - o\|_2. \tag{6.1}$$

Note that (6.1) is not differentiable at $p = o$ because we can get different limits by choosing different paths to the origin. To tackle this problem, one can square both sides of the inequality,

$$d_{\mathcal{C}}^2 \leq \|p - o\|_2^2 \tag{6.2}$$

and recover the desirable property of being continuously differentiable everywhere. However, depending on the optimization algorithm behind the planner at hand, one constraint formulation can be more advantageous than the other.

For a planner whose optimization algorithm is based on a Newton-type method, e.g., with `acados`, note that only one system linearization and one QP solve are performed per sampling time. Each QP corresponds to a linear approximation of the original NLP along a time-varying trajectory. In this setting, squaring the Euclidean norm, as in (6.2), provides local trajectories that are over-conservative. In a case where linearization points are far away from the boundaries of the original NLP's linearized feasible set, the use of (6.1) leads to linear constraints that are "least" conservative. We will formalize this statement in Proposition 1. In contrast, for a planner whose optimization algorithm is based on a fully converged NMPC method, e.g., with `IPOPT`, in general, no constraint formulation has a distinct advantage over the other. As in this case, the iterations continue until convergence or certain termination conditions are reached, the feasible sets for (6.1) and (6.2) are the same.

Thus, in view of the algorithmic ideas of the RTI scheme, our main theoretical result for the *least conservative linearized* (LCL) constraint formulation is the following:

Proposition 1. *Let $H_1(\bar{p}) := \{p : c_1(\bar{p}) + \nabla_p c_1(\bar{p})^T (p - \bar{p}) \geq 0\}$ and $H_2(\bar{p}) := \{p : c_2(\bar{p}) + \nabla_p c_2(\bar{p})^T (p - \bar{p}) \geq 0\}$, where $c_1(p) := \|p - o\|_2 - d$ and $c_2(p) := \|p - o\|_2^2 - d^2$, denote the half-spaces defined by the linearization of the constraints $c_1(p) \geq 0$ and $c_2(p) \geq 0$ at any feasible point $\bar{p}$, respectively. Moreover, let $S := \{p : c_1(p) \geq 0\}$ denote the set defined by the original nonlinear constraints. Then, $H_1(p) \subset S$ and $H_2(p) \subset S$. Moreover, we have that $H_2(p) \subseteq H_1(p)$.*

Proof. Define $c_{1,\text{lin}}(\bar{p}, p) := c_1(\bar{p}) + \frac{\partial c_1}{\partial p}(p - \bar{p})$ and $c_{2,\text{lin}}(\bar{p}, p) := c_2(\bar{p}) + \frac{\partial c_2}{\partial p}(p - \bar{p})$. Due to convexity of $c_1(p)$ and $c_2(p)$, we have that, for any $\bar{p}$ we have that

$$0 \leq c_{1,\text{lin}}(\bar{p}, p) \leq c_1(p)$$

and

$$0 \leq c_{2,\text{lin}}(\bar{p}, p) \leq c_2(p)$$

such that $p \in H_1(\bar{p}) \implies p \in S$ and $p \in H_2(\bar{p}) \implies p \in S$, which proves the first statement. In order to prove that $H_2(p) \subseteq H_1(p)$, we proceed as follows. Define the auxiliary functions $\hat{c}_2, \hat{c}_1 : \mathbb{R} \to \mathbb{R}$ as $\hat{c}_2(v) := v^2 - d^2$ and $\hat{c}_1(v) := v - d$. Moreover, define their linearizations $\hat{c}_{1,\text{lin}}(\bar{v}, v) = \hat{c}_1(\bar{v}) + \frac{\partial \hat{c}_1}{\partial v}(v - \bar{v})$ and $\hat{c}_{2,\text{lin}}(\bar{v}, v) = \hat{c}_2(\bar{v}) + \frac{\partial \hat{c}_2}{\partial v}(v - \bar{v})$. Due to convexity in v of $\hat{c}_{1,\text{lin}}$ and affinity in v of $\hat{c}_{2,\text{lin}}$ we have that, for any d, and any $\bar{v}$, the following holds:

$$\min_{v \geq d} \hat{c}_{1,\text{lin}}(\bar{v}, v) = \min_{v \geq d} \hat{c}_1(v) = 0$$

and

$$\min_{v \geq d} \hat{c}_{2,\text{lin}}(\bar{v}, v) \leq \min_{v \geq d} \hat{c}_2(v) = 0.$$

Define $r(p) := \|p - o\|$. Then, we can state the following:

$$\begin{aligned} \min_{p \in H_1(p)} c_{2,\text{lin}}(\bar{p}, p) &= \min_{p \ \text{s.t}\ c_{1,\text{lin}}(\bar{p},p) \geq 0} c_{2,\text{lin}}(\bar{p}, p) \\ &= \min_{p \ \text{s.t}\ \hat{c}_{1,\text{lin}}(r(\bar{p}), r(\bar{p}) + \frac{\partial r}{\partial p}(p - \bar{p})) \geq 0} \hat{c}_{2,\text{lin}}(r(\bar{p}), r(\bar{p}) + \frac{\partial r}{\partial p}(p - \bar{p})). \end{aligned}$$

Introducing the change of variables $v = r(\bar{p}) + \frac{\partial r}{\partial p}(p - \bar{p})$, and using the fact that $\hat{c}_{1,\text{lin}}(r(\bar{p}), r(\bar{p}) + \frac{\partial r}{\partial p}(p - \bar{p})) \geq 0$ if and only if $v \geq d$, we obtain

$$\min_{p \in H_1(p)} c_{2,\text{lin}}(\bar{p}, p) = \min_{v \geq d} \hat{c}_{2,\text{lin}}(r(\bar{p}), v) \leq \min_{v \geq d} \hat{c}_2(v) = 0.$$

This last inequality shows that $H_2(\bar{p}) \subseteq H_1(\bar{p})$ concluding the proof. □

Figure 6.3 shows the boundaries of the linearized feasible set considering three distinct regular points $p^\star$ for constraint formulations (6.1) and (6.2). Note that the linearized constraint formulation defined by (6.1) provides solutions that are closer to the lower bound $d_{\mathcal{C}}$.

6.3.2 Constraint violation

The constraint presented in (6.1) guarantees collision avoidance instantaneously since its evaluation occurs only at t_i and t_{i+1}. This, however, does not ensure that there are no collisions in-between shooting nodes, which may result in a constraint violation. As a result, the predicted trajectory between time points may be seen penetrating the obstacle.

One can address this issue by expanding the obstacle, in all directions, by the maximum incursion distance [215], or by using the hyperplane separation theorem as in [216]. This chapter does not use any particular method to ensure collision-free trajectories in-between the robot's successive positions, as it comes with a higher computational burden. Accordingly, it relies on a reasonable choice for the size of the discretization time step.

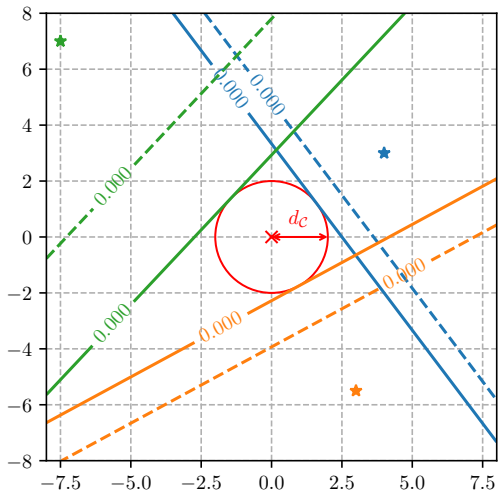


Figure 6.3. Boundaries of the linearized feasible set – a comparison between constraint formulations: linearization points $p^\star$ are represented as $\star$, in red the clearance around an obstacle marked by $\times$, $\|\cdot\|_2^2$ in dashed line, and $\|\cdot\|_2$ in solid line. The contour lines at zero are shown for half-spaces $H_1(\bar{p})$ and $H_2(\bar{p})$.

6.3.3 Feasibility and soft constraints

One major drawback regarding the use of hard constraints is that they may render the optimization problem infeasible: this is especially true in the case of state constraints [217], such as the one in (6.1). If the input constraints represent a physical limitation, they have to be enforced all the time. State constraints, instead, are only often desired – although they are not always physically necessary – and hence should be satisfied whenever possible.

A more systematic approach for dealing with infeasibility is to modify the cost function to include a penalization on the violation of the collision avoidance constraint. This strategy is usually achieved by introducing *slack variables* associated with the soft constraint and heavily penalizing them. The optimization algorithm searches for a solution that minimizes the original cost while keeping the slack variables equal to zero whenever possible.

The authors in [218] discuss the inclusion of an ℓ_1-norm $\mu\|\epsilon\|_1$ penalization term of the slack in order to obtain an exact penalty function, i.e., the controller violates the constraints only when necessary. A hazard with ℓ_1 penalty functions is their inherent non-smoothness at optimal points, which may degrade the final solution accuracy. On the other hand, the ℓ_2^2-norm $\mu\|\epsilon\|_2^2$ benefits of being smooth and having "simple" derivatives. A major drawback to the ℓ_2^2 penalty function is the uneven way that it penalizes constraints, such that μ has to be very large to enforce asymptotic feasibility [219]. Given the potential benefits of an exact penalty function, this work opts to use an ℓ_1-norm.

6.3.4 Problem formulation

The tailored soft-constrained problem with LCL constraint formulation reads as follows:

$$\min_{\substack{\xi_0,\dots,\xi_N,\\ u_0,\dots,u_{N-1},\\ \epsilon_0,\dots,\epsilon_N}} \quad \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(\xi_i,u_i)\|_W^2 + \frac{1}{2}\|\eta_N(\xi_N)\|_{W_N}^2 + \frac{\mu}{2}\sum_{i=0}^{N} \|\epsilon_i\|_1 \tag{6.3a}$$

$$\text{s.t.} \quad \xi_0 - \bar{\xi}_0 = 0, \tag{6.3b}$$

$$\xi_{i+1} - F_c(\xi_i,u_i) = 0, \quad i=0,\dots,N-1, \tag{6.3c}$$

$$u_i \in \mathcal{U}, \quad i=0,\dots,N-1, \tag{6.3d}$$

$$\epsilon_i + d_{\mathcal{O}}(p_i) \geq d_{\mathcal{C}} \quad i=0,\dots,N, \tag{6.3e}$$

$$\epsilon_i \geq 0, \quad i=0,\dots,N, \tag{6.3f}$$

where $\xi := (\xi_0,\dots,\xi_N)$, $u := (u_0,\dots,u_{N-1})$, and $\epsilon := (\epsilon_0,\dots,\epsilon_N)$ denote the state, input, and slack trajectories, respectively. The discrete-time quadrotor dynamics is denoted by $F_c : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{\xi}$. The residual functions $\eta : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\eta_N : \mathbb{R}^{n_\xi} \to \mathbb{R}$ represent the stage and terminal cost, respectively. They are weighted by the symmetric matrices W, $W_N \succ 0$. Similarly, the slack penalty is weighted by the real-valued vector $\mu \in \mathbb{R}^{n_\epsilon}$. Eq. (6.3d) implements the input box constraints $\mathcal{U} = \{u \in \mathbb{R}^{n_u} : u_{\min} \leq u \leq u_{\max}\}$, representing the actuator limits. Finally, N and $\bar{\xi}_0$ denote the horizon length and the current state estimate, respectively.

6.4 Simulation results

This section presents a motion generation benchmark to validate the local planner related to NLP (6.3). It particularly uses the Crazyflie nano-quadrotor dynamic model as an example and whose physical parameters are summarized in Table 3.2.

6.4.1 Quadrotor benchmark

We consider a sampling time of $t_s = 15$ ms and $N = 50$ shooting intervals in the simulations. We use an ERK4 integration scheme through the automatic differentiation and modeling framework `CasADi` to discretize the dynamics (3.33). Moreover, the least squares cost residuals are described as

$$\eta(\xi,u) := \begin{pmatrix} \xi_i - \xi_i^r \\ u_i - u_i^r \end{pmatrix}, \quad \eta_N(\xi_N) := \xi_N - \xi_N^r, \tag{6.4}$$

where the quantities in (6.4) with the r superscript stand for the precomputed desired references.

In our approach, for each obstacle in the workspace, we explicitly add new constraints (6.3e)–(6.3f) to the NLP (6.3). In this chapter, two dynamic obstacles are considered, thus $n_\epsilon = 2$. Therefore, the gradients with respect to the lower (μ_l) and upper (μ_u) slack penalty values are

$$\begin{aligned} \mu_l &= (14.5 \cdot 10^3, 14.5 \cdot 10^3), \\ \mu_u &= (14.2 \cdot 10^3, 14.2 \cdot 10^3). \end{aligned} \tag{6.5}$$

The weighting matrices are

$$\begin{aligned} W &= \text{blkdiag}(30 \cdot \mathbf{I_2}, 60, 1 \cdot 10^{-1} \cdot \mathbf{I_4}, 2, 3, 5, 3 \cdot \mathbf{I_3}, 5 \cdot 10^{-2} \cdot \mathbf{I_4}), \\ W_N &= \text{blkdiag}(30 \cdot \mathbf{I_2}, 60, 1 \cdot 10^{-1} \cdot \mathbf{I_4}, 2, 3, 5, 3 \cdot \mathbf{I_3}). \end{aligned} \tag{6.6}$$

For the robotic system at hand, the input bounds are $u_{\min} = 0$, $u_{\max} = 22$ krpm, while the bounds for the slack variables are $\epsilon_{\min} = 0$, $\epsilon_{\max} = 100 \cdot 10^3$. Finally, the bounds for the collision avoidance constraint are $d_{\mathcal{C}} = 0.2$ m, $d_{\mathcal{M}} = 100 \cdot 10^3$ m.

Reference trajectory Two different global reference trajectories are regarded. They have been generated offline by solving the following NLP:

$$\min_{\substack{\xi_0,\dots,\xi_N,\\ u_0,\dots,u_{N-1}}} \quad \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(\xi_i, u_i)\|_W^2 + \frac{1}{2}\|\eta_N(\xi_N)\|_{W_N}^2 \tag{6.7a}$$

$$\text{s.t.} \quad \xi_0 - \bar{\xi}_0 = 0, \tag{6.7b}$$

$$\xi_{i+1} - F_c(\xi_i, u_i) = 0, \quad i = 0, \dots, N-1, \tag{6.7c}$$

$$u_i \in \mathcal{U}, \quad i = 0, \dots, N-1. \tag{6.7d}$$

We assume a time horizon of $T = 3$ s, $N = 200$ shooting intervals, discretizing the dynamics (3.33) via an ERK4 integration method. The NLP solution provides the optimal trajectories that drive the nano-quadrotor from the hovering state at $p = (0, 0, 0.4)$ to the final desired state at positions: $p_1^d = (1, -1, 1)$ and $p_2^d = (0.5, -0.5, 1)$. We regulate for $q^d = (1, 0, 0, 0)$ so that a straight line can approximate the geodesic arc without significant problems. Note that, as the obstacles are dynamic, the global references are unaware of them. Finally, we use `IPOPT` and the *just-in-time* compiling function evaluations to solve the NLP up to convergence.

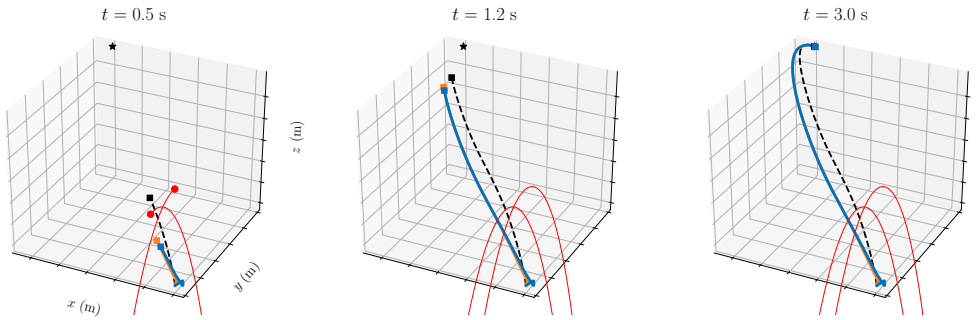
6.4.2 Software interface

The NLP results were obtained on an Intel Core i5-4288U @ 2.6 GHz running macOS Catalina. Similar to Chapter 5, we use the `acados` Python template-based interface to generate the library that implements the tailored problem formulation (6.3). The QPs arising in this NMPC formulation are solved using the solver `HPIPM`, built upon `BLASFEO`. The `X64_INTEL_HASWELL` implementation of the `BLASFEO` package has been used. We have also employed the partial condensing routines implemented in `HPIPM`.

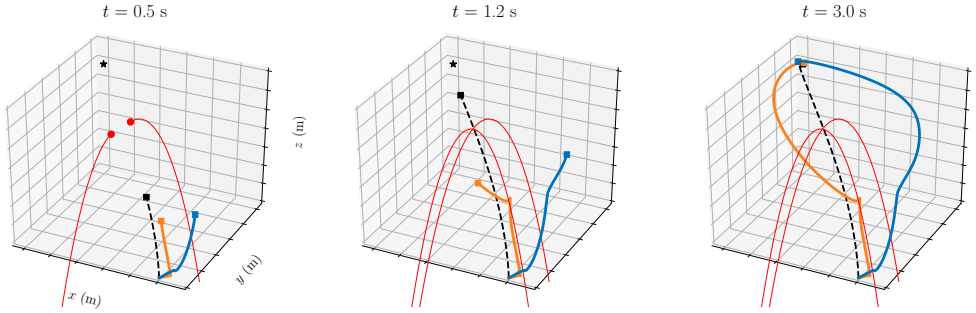
6.4.3 Performance analysis

To evaluate the trajectory conservativeness introduced by the constraint approximation, we perform simulations comparing our real-time planner with LCL constraint formulation to its counterpart. The latter refers to a planner where constraint (6.3e) is defined by $\|\cdot\|_2^2$. Both use the parameterizations presented in Section 6.4.1, including the slack penalties values (6.5) and weighting matrices (6.6).

We now test the proposed NMPC under two simulated scenarios. First, the Crazyflie must travel from a hovering configuration to another while avoiding balls thrown at it. In this case, we assume that the obstacle's movement between two



(a) Final desired position p_1^d



(b) Final desired position p_2^d

Figure 6.4. Generated trajectories among moving balls (red): real-time planner with LCL constraint formulation (orange), real-time planner with $\|\cdot\|_2^2$ constraint formulation (blue); reference trajectory is dashed. The $\star$ is the final goal.

shooting nodes is ballistic. Second, the Crazyflie must perform a similar task but in a cluttered environment with fixed obstacles. These scenarios will help to illustrate the NMPC performance. The results are presented in Figures 6.4, 6.5, and 6.6.

As expected, the planner with LCL constraint formulation generates trajectories significantly closer to the minimum clearance $d_{\mathcal{C}}$ when compared to its counterpart. Besides, we note that there is no constraint violation. However, the effort required to tune the slack penalties directly impacts the amount of constraint violation. Much of the difficulty exists because, in this particular problem, the optimal solution will frequently lie on the boundary of the feasible set. Therefore, imposing stringent penalties – in the case of soft constraints – increases the difficulty in driving the solution towards the optimum while distancing the local trajectories from the lower bound.

Conversely, if the penalty is not strict enough, then the search will tend to stall outside the feasible region and thereby violate the constraint. Although the tuning considered in this work has provided a suitable closed-loop performance for both real-time planners – with no constraint violation – there is no guarantee that the violation will not occur. This is especially true in a real-world scenario, where the state estimation deals effectively with the uncertainty due to noisy sensor data and, to some extent, with random external factors, i.e., wind. One way to avoid this shortcoming and recover feasibility is to retune the slack penalties of the soft-constrained problem.

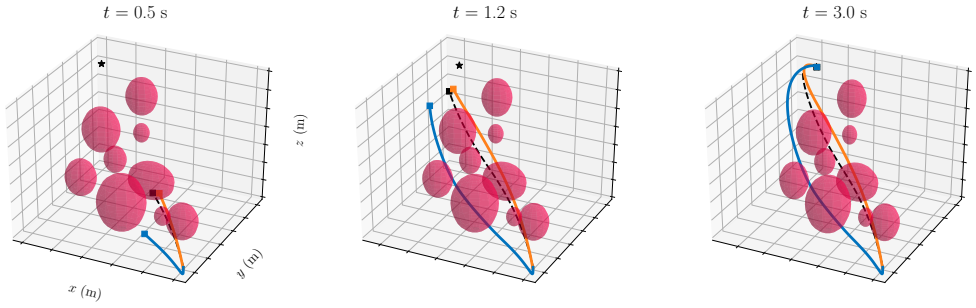
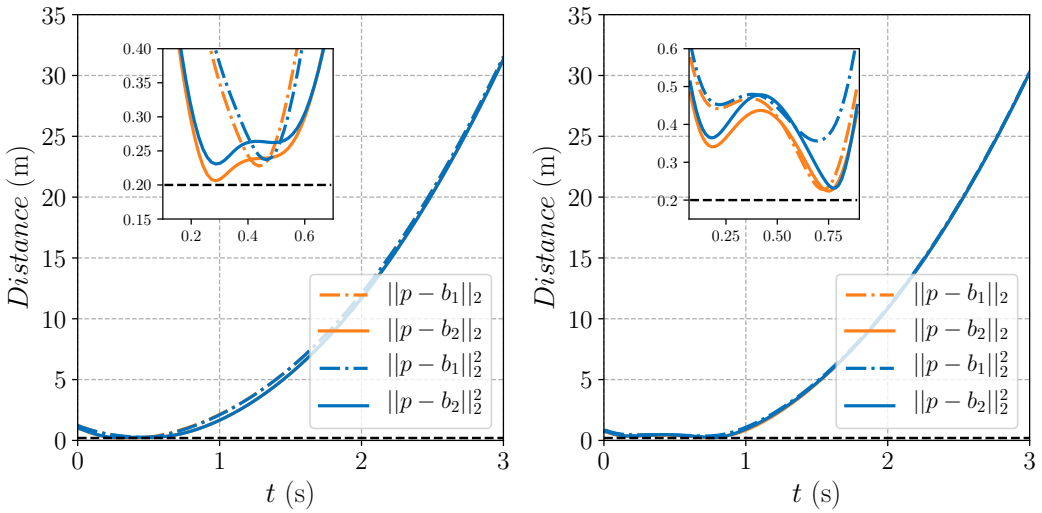


Figure 6.5. Generated trajectories among static obstacles (red): real-time planner with LCL constraint formulation (orange), real-time planner with $\|\cdot\|_2^2$ constraint formulation (blue); reference trajectory is dashed. The $\star$ is the final goal.



(a) Final desired position p_1^d **(b)** Final desired position p_2^d

Figure 6.6. Distance between the Crazyflie and the balls b_i with LCL constraint formulation (orange) and $\|\cdot\|_2^2$ constraint formulation (blue). Inset shows the time instant of closest proximity to the minimum clearance.

Lastly, we gathered the average solution time in `acados` to assess the computational complexity of the planner's algorithm. The values for final desired positions p_1^d and p_2^d were 4.76 ms and 7.97 ms, respectively, while for the cluttered environment, it was 8.5 ms. These results show the computational efficiency of the proposed scheme.

6.5 Chapter summary

This chapter presented a new real-time motion generation scheme for quadrotors in dynamic environments. The key concepts contributing to the method's effectiveness stem from the algorithmic ideas tied to the RTI scheme to formulate a least conservative linearized constraint. The chapter provided a reformulation that guarantees less conservative trajectories for planners whose optimization algorithm is based on

the Newton-type method. Strategies to ensure both constraint satisfaction and feasibility of the optimization problem have also been discussed. Finally, the approach was tested in simulation on a challenging example involving a high-dimensional quadrotor system, showing efficient computational performance.

Chapter 7

Mixed-initiative control via real-time NMPC for safe human-quadrotor interaction

7.1 Motivation and contribution

Quadrotors have been widely used across various applications, resulting in a significant market expected to grow exponentially. Due to this increasing demand, one can envisage that, in the future, learning to fly a quadrotor will become a taken-for-granted skill, just as learning to drive a car nowadays. A first step towards making this idea a realistic prospect could be a mixed-initiative control approach that guarantees the aerial system's "working conditions" while the novice operator learns to fly it. In other words, as long as some safety- and/or task-related rules are met, the operator commands are obeyed. Once novices start working in the context of complex tasks, it is essential to provide them with assistance that gradually decreases as their skill increases – here, skill refers to the knowledge and ability that allow the use of the robot to accomplish a task. Conversely, one could think of an experienced operator who already has the skill level that enables him/her to fly a quadrotor. But as his/her cognitive load is primarily focused on the short-term aspect of the task, he/she cannot consider other underlying factors in the long run (e.g., safety). It is then essential to have a control approach that supervises the long-term task to prevent the operator from being overwhelmed by (too) high engagement.

To fill these gaps, this paper proposes an efficient mixed-initiative controller based on NMPC to enforce safety in human-quadrotor interactions. The primary objective of this study is to provide a "safety-layer" controller over which one may define metrics to assess pilots' learning curves. The technical difficulties associated with the proposed controller are twofold: first, satisfy both the existing constraints and the human intentions; second, solve the underlying optimization problems that include a high-dimensional quadrotor system under the available computation time. We demonstrate how to handle these difficulties using concepts of zone MPC [220] and online weight adaptation within the RTI scheme. We exploit high-performance numerical optimization algorithms to further speed up solution times, including a

structure-exploiting convex solver and linear algebra kernels for small-to-midsize matrices. The mixed-initiative controller is validated in simulation against a second autonomous algorithm that emulates pilots with different skill levels. The main contributions of this chapter are:

- A mixed-initiative control scheme for human-quadrotor systems based on NMPC with safety guarantees.
- Numerical simulations that show the effectiveness and computational efficiency of the proposed algorithm.

The proposed *mixed-initiative* (MI) control approach for human-quadrotor interaction leads to major advantages compared to existing methods if applied to our scenario. Unlike [221, 222, 200, 223], our approach offers a rigorous predictive diagnosis of interaction degradation so that most of the control authority is allocated to the most capable agent, ultimately yielding a time-varying synergy between the human and the automatic controller. While the predictive behavior is relevant in a highly nonlinear context (disregarded in previous works), the dynamic synergy is particularly important to manage the control authority in conflict situations, i.e., when human and automatic controller commands are rather different. To the best of our knowledge, this is the first work that subsumes the advantages of the RTI scheme, zone MPC, and high-performance algorithms to synthesize a nonlinear MI controller cognizant of how to mix control authority to enforce safety and with real-time capabilities. For this reason, this work makes a significant contribution to human-robot interaction literature.

7.2 Related works

One way robots and humans can interact is to put the human in the loop with some blending scheme [221]: the human and the automatic controller have control over the robot, and both systems must adapt to ensure task completion. Figure 7.1 illustrates a couple of cases. For example, in exoskeleton systems, the interplay between robot and human may be characterized as an interaction between teacher and student in a learning process. The teacher (robot) tries to minimize the student (human) error, applying a minimal effort [222, 224]. In the context of autonomous cars, the driver interacts with the automatic controller that aims at reducing the driver's workload and, at the same time, taking prompt actions in case of human failure [200, 223]. For human-swarm systems, the interaction paradigm is less obvious. As the size of the swarm increases, control should become more focused on the swarm as a whole rather than on the individuals, given the human's limited capacity to multitask. In this case, the interaction is more concerned with the human-swarm ability to accomplish a particular task than with the swarm spatial positioning [225, 226].

Other interactive approaches rely on methods designed from the human perspective, where, in general, the presence of haptic cues (e.g., force feedback) increases situational awareness. These methods lean heavily in favor of perceiving the human as the source of action and the robot as the passive collaborator. Figure 7.2 depicts a pair of situations. For instance, virtual fixtures have been used to inform the

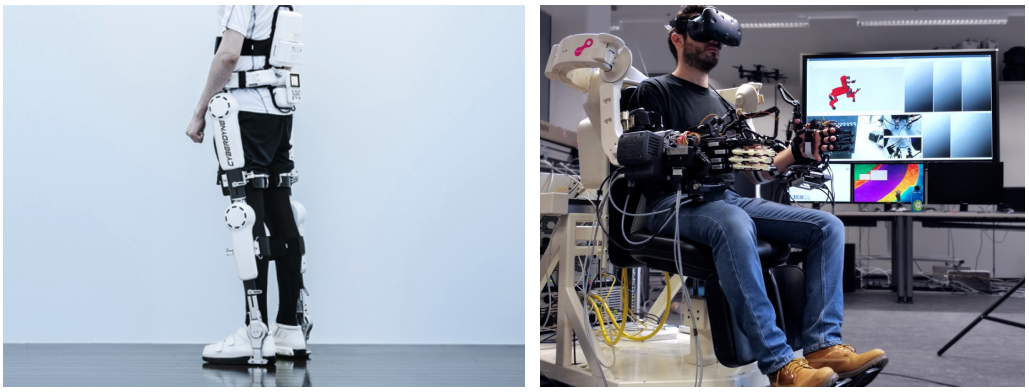


Figure 7.1. (left) Cyberdyne lower-limb exoskeleton for assisting patients with brain and mobility disabilities as well as non-medical purposes such as eldercare and worker assistance device; (right) bimanual teleoperation of the Centauro robot in a scenario with rough terrains and austere conditions, typical characteristics of disasters.

operator of the highest comfort position, distance from the target position, proximity to unsafe kinematic configurations, or misalignment in contact-driven surface conditioning tasks [227, 228, 229]. Obstacle avoidance is one of the most critical requirements to be met in many robotics scenarios. In this matter, researchers have been using haptic feedback to warn the operator about instantaneous collisions [230, 231, 232, 233]. Several human-collaborative schemes are compared in [234] where it is shown that haptic feedback is one of the main aspects. Among other considerations, it should be noted that the method used to generate assistive haptic cues strongly determines the usefulness of kinesthetic guidance to a large extent. For this reason, considerable effort has been put into introducing new assistive methods to improve task performance explicitly. The authors in [235] propose a force-feedback telepresence method based on an external wrench estimation that enables the operator to feel, e.g., wind and contact forces. In [236], a human-teacher receives haptic feedbacks from a robot-learner through virtual fixtures that produce assistive forces and torques toward the learned kinematic behavior. Analogously, [237] use the gradient of an artificial potential field to generate repulse forces that help humans to avoid touching adjacent vessels while operating surgical robots. Interested readers are referred to [238, 239, 240, 241] for the introduction of other noteworthy methods.

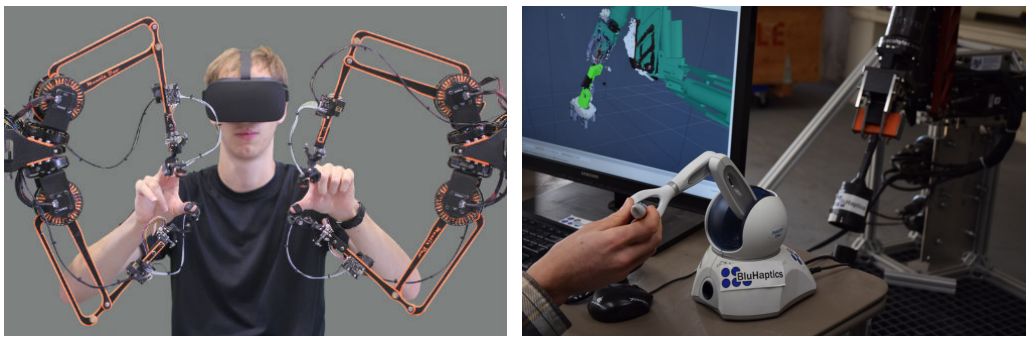


Figure 7.2. (left) Mantis, a lightweight, affordable force feedback device; (right) BlueHaptics framework providing haptic feedback in a manipulator guidance task.

7.3 Problem statement

The fundamental problem in mixed-initiative control is how to blend human inputs and automatic controller commands to realize the former as much as possible while always enforcing safety. In this context, the formulation of our problem hinges upon the following given components:

- A *robot* represented by a certain nonlinear, time-varying dynamic system

$$\dot{\vartheta} = f(\vartheta, u), \tag{7.1}$$

  subject to a set of constraints, where states, control inputs, and dynamics map are denoted by $\vartheta \in \mathbb{R}^{n_\vartheta}$, $u \in \mathbb{R}^{n_u}$ and $f : \mathbb{R}^{n_\vartheta} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\vartheta}$, respectively.

- A *motion generator* providing a *reference trajectory* η^r for the *task variables* $\eta = c(\vartheta)$, which take values in $\mathbb{R}^{n_\eta}$. Typical examples of task variables are a subset of the state variables or the position of some relevant point of the robot. As for the motion generator, it can be either an offline planner (such as a motion planner among obstacles) or an online controller.

- A *human* assigning reference values ν^r to some subset $\nu \in \mathbb{R}^{n_\nu}$ of the state variables through a control interface. These signals are henceforth called *human inputs* and convey human intentions.

- A set of *working conditions* defining requirements on the state for the robot to operate healthily. These can include both safety rules (e.g., maximum kinetic energy, maximum dissipation) and task-related rules (e.g., the accuracy of end-effector positioning). We assume that working conditions are such that the associated feasible set is convex.

For this setting, we want to devise a *mixed-initiative controller* that will attempt to execute human inputs as much as possible without compromising both working conditions and constraints inherent to the robot's physical limitations. As we will see, in addition to the mixed-initiative controller itself, the control algorithm also includes a *blending mechanism* that predicts the violation of working conditions and lends most of the control authority to the most capable agent at any time.

7.4 Control architecture

The mixed-initiative controller considers the human operator and the motion generator as two different agents. The blending mechanism assesses the working conditions' violation and distributes control authority accordingly. For example, working conditions may include a requirement of collision-free motion. If human inputs ignore this, a violation will be predicted, and the blending mechanism will shift more control authority to the motion generator. Overall, this controller can be seen as an assistive scheme for novice and experienced pilots. For the former, it allows them to learn to fly a quadrotor without crashing it. For the latter, it introduces some level of disengagement from which they can profit to perform manual flights without focusing on avoiding obstacles. In the following, we describe in detail the proposed algorithm.

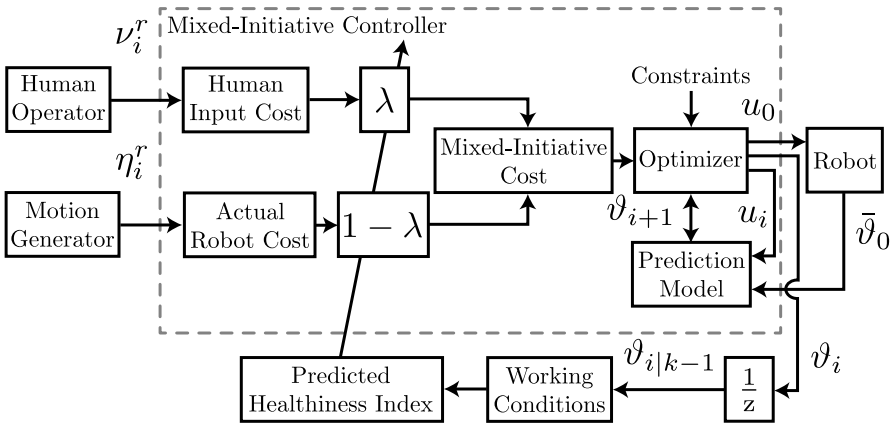


Figure 7.3. A block diagram of the proposed mixed-initiative control algorithm.

7.4.1 Algorithm overview

Mixed control is accomplished through a *mixed-initiative NMPC controller* whose cost function – the *mixed-initiative cost* – is a convex combination between the *actual robot cost* and the *human input cost*. The latter is a cost term that penalizes the deviation from the human inputs. The proposed blending mechanism is a continuous function between 0 and 1, named *predicted healthiness index*, that drives the convex combination. It is computed in this way: when the index tends to 0, the working conditions' violation is too close and, hence, the mixed-initiative cost tends to the actual robot cost. When the index tends to 1, the violation is far enough and, therefore, the mixed-initiative cost tends to the human input cost.

Based on the state solution of the NMPC controller, at each time instant, the predicted healthiness index computes how far the robot is from violating these conditions within a *virtual horizon* whose length is defined by the designer. It then selects a value between 0 and 1 and blends the individual costs. The idea is that the closer the index to 0, the more troublesome it will be to keep the system within the working conditions in the next virtual horizon, even if full control is given to the motion generator (and human inputs are heavily ignored from that moment on). Vice-versa, the closer the index to 1, the easier the motion generator's job will be if it takes full control of the system. Figure 7.3 shows a block diagram of the proposed algorithm.

7.4.2 Mixed-initiative controller

When using NMPC to control a system, a nonlinear nonconvex program is solved using the current state as the initial value at each sampling instant. However, as the computational burden associated with the solution can be rather long, the employment of NMPC has only recently been extended to applications where shorter sampling times are required [2]. Typically, a continuous-time, infinite-dimensional OCP is tailored according to the problem at hand, discretized into N intervals using a time step Δt over a fixed time horizon t_N, and then solved. In doing so, the tailored OCP is transcribed into a discrete-time, finite-dimensional NLP, now defined over a refined coarse grid $[t_0, t_N]$, for which the KKT conditions are set up

and solved at each sampling interval $[t_k, t_{k+1}]$. In our approach, we cast the MI controller as a constrained NLP with linear least-squares tracking cost, formulated as follows:

Problem 1 (Mixed-Initiative Controller)**.**

$$\min_{\substack{\vartheta_0,\dots,\vartheta_N,\\ u_0,\dots,u_{N-1}}} \quad \sum_{i=0}^{N-1} L(\eta_i, \nu_i, e_i, u_i) + M(\eta_N, \nu_N, e_N) \tag{7.2a}$$

$$\text{s.t.} \quad \vartheta_0 - \bar{\vartheta}_0 = 0, \tag{7.2b}$$

$$\vartheta_{i+1} - F(\vartheta_i, u_i) = 0, \qquad i = 0, \dots, N-1, \tag{7.2c}$$

$$\vartheta_i \in \mathcal{X}, \qquad i = 1, \dots, N-1, \tag{7.2d}$$

$$u_i \in \mathcal{U}, \qquad i = 0, \dots, N-1, \tag{7.2e}$$

where

$$L(\cdot) = \frac{1}{2}((1-\lambda)\Delta\eta_i^T Q_\eta \Delta\eta_i + \lambda\Delta\nu_i^T Q_\nu \Delta\nu_i + (1-\lambda)\Delta e_i^T Q_e \Delta e_i + (1-\lambda)u_i^T R u_i)$$

$$M(\cdot) = \frac{1}{2}((1-\lambda)\Delta\eta_N^T Q_{\eta_N} \Delta\eta_N + \lambda\Delta\nu_N^T Q_{\nu_N} \Delta\nu_N + (1-\lambda)\Delta e_N^T Q_{e_N} \Delta e_N).$$

Here, the functions L and M represent the stage and terminal cost terms, respectively, which are weighted by the positive-definite matrices Q_η, $Q_{\eta_N} \in \mathbb{R}^{n_\eta \times n_\eta}$, Q_ν, $Q_{\nu_N} \in \mathbb{R}^{n_\nu \times n_\nu}$, Q_e, $Q_{e_N} \in \mathbb{R}^{n_h \times n_h}$, and $R \in \mathbb{R}^{n_u \times n_u}$. The convex polytopic sets $\mathcal{X}$ and $\mathcal{U}$ implement the state and input constraints associated with the physical limitations of the robot. The horizon length is denoted by N while $\bar{\vartheta}_0$ denotes the current state estimate.

Moreover, we denote the task variables tracking error as $\Delta\eta_i = \eta_i - \eta_i^r$, $\Delta\eta_N = \eta_N - \eta_N^r$, and the human inputs' tracking error as $\Delta\nu_i = \nu_i - \nu_i^r$, $\Delta\nu_N = \nu_N - \nu_N^r$. The cost function also includes other penalty terms that may be relevant to the specific application. Their tracking errors are denoted as $\Delta e_i = h(\vartheta_i) - e_i^r$, $\Delta e_N = h_N(\vartheta_N) - e_N^r$, where the output functions $h(\vartheta_i)$, $h_N(\vartheta_N) \in \mathbb{R}^{n_h}$ and their respective references e_i^r, $e_N^r \in \mathbb{R}^{n_h}$ are defined by the relative complement $\mathbb{R}^{n_\vartheta} \setminus (\mathbb{R}^{n_\eta} \cup \mathbb{R}^{n_\nu})$.

The scaling factor $\lambda \in (0, 1)$ is the output of the blending mechanism and is used to determine how control authority should be mixed. Looking at the cost function, one can observe that increasing the value of λ will give more weighting to the human inputs than the motion generator commands. Note that we consider an open interval because we are interested in strictly convex problems. Finally, once the solution for interval $[t_k, t_{k+1}]$ is computed, the first element of the input trajectory u_0 is applied to the system before shifting the horizon forward in time, see Figure 7.3.

7.4.3 Working conditions

While the particular requirements that a robot has to obey may differ from one system to another, a generic approach is to express these requirements through suitable working conditions. Here, these conditions are designed to help pilots fly a

quadrotor along a collision-free trajectory in an arena with obstacles. To this end, we impose a maximum deviation $r \in \mathbb{R}_{>0}$ from the reference trajectory to enforce safety. In terms of Euclidean norm, this condition reads as follows:

$$d(\eta) = \|\eta - \eta^r\| \leq r. \tag{7.3}$$

Any point η satisfying (7.3) describes a convex free region $\mathcal{B}$, which we defined as

$$\mathcal{B} = \{\eta \in \mathbb{R}^{n_\eta} : \|\eta - \eta^r\| \leq r\}. \tag{7.4}$$

Note that $\mathcal{B}$ is a norm ball fully described by its center point η^r. This observation implies that the robot is softly constrained to move inside of a ball, where its actual trajectory is just one of the possible trajectories that avoid collisions, and at best, it is the reference trajectory itself. Although we rely on a map and a collision-free trajectory, one can tailor the method in two ways to deal with unstructured environments: (a) the motion generator can plan collision-free trajectories by taking into account the nearby obstacles detected by the robot sensors; (b) the MPC itself can do the same by incorporating an explicit collision avoidance constraint in the formulation and using the robot sensors to set up this constraint, e.g., see [242, 243].

Unlike the constraints in the MI controller, i.e., Eq. (7.2b)–(7.2e), the working conditions may be seen as "soft constraints" that facilitate the decoupled design of the safety- and/or task-related rules through the task variables. By defining them this way, we dovetail zone MPC ideas into our MI controller. Zone MPC is used when the specific setpoint of an output variable is of low relevance compared to a zone delimited by upper and lower bounds, typically read as a soft constraint [220]. This idea can be easily accommodated by choosing a suitable weighting matrix to penalize the output deviation in the cost function. For a linear least-squares tracking cost, an output deviation, which depends linearly on the states, causes the cost to strictly increase as one moves away from the reference. This increase is acceptable as long as the deviation rests inside the zone. As the deviation crosses or approaches the zone's boundary, large penalty weights are set. We revisit this concept by making the weighting matrices of human and actual robot costs antagonist: when the task variables' prediction lies outside $\mathcal{B}$, the human cost weighting gets closer and closer to zero, while the actual robot cost weighting tends to a high value, which brings back the prediction to the interior of $\mathcal{B}$. The self-optimizing output variables allow safety enforcement to the best possible degree. However, minor violations may still occur due to the soft constraint nature of this approach. In principle, one can enforce $\eta \in \mathcal{B}$ as a hard constraint (if deemed necessary) by defining a task-related working condition. Nevertheless, such a constraint will inevitably be made soft in a practical implementation.

7.4.4 Predicted healthiness index

As previously established, the robot must remain inside $\mathcal{B}$ to enforce safety. Based on that, at each sampling instant t_k, we use the state solution of Problem 1 at time t_{k-1} to compute the *predicted healthiness* (PH) index $\lambda : (0, r) \mapsto (0, 1)$, i.e., the blending mechanism. More precisely, we measure how far the robot is from violating the ball's boundary within a virtual horizon whose length is defined by $N_b \leq N$.

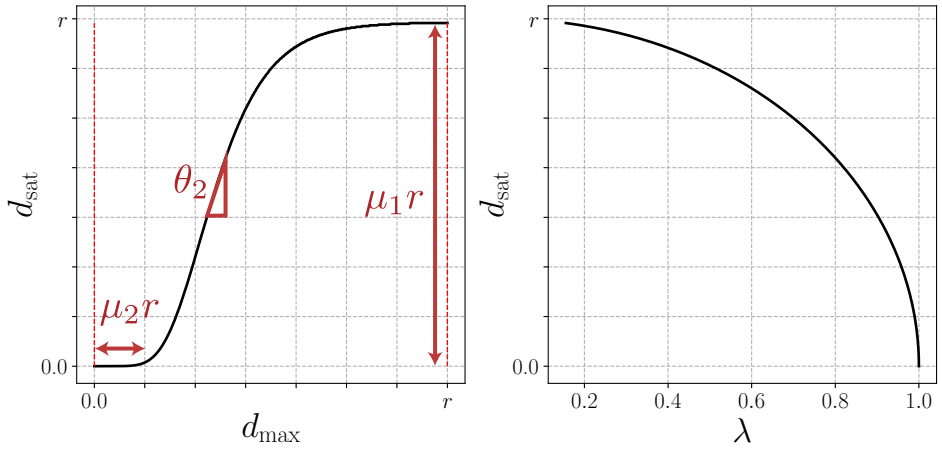


Figure 7.4. Left: pictorial description of the function (7.7). Dashed red lines indicate the bounds on $d_{\max}$. Particularly, when $d_{\max} = r$ then $d_{\text{sat}} = \mu_1 r$. Analogously, when $d_{\max} = 0$ then $d_{\text{sat}} = \mu_2 r$. Right: an illustration of the function defined in (7.8) showing a profile relatively quadratic for penalizing residuals inside $\mathcal{B}$.

Then, we select an index λ between 0 and 1 that is used to determine the weighting in Problem 1. The fact that N_b may differ from N allows the designer to choose the level of insight that the blending mechanism should have into the interaction. In what follows, we formally define the PH index.

Given a virtual horizon with N_b intervals and the predicted $\eta_{i|k-1}$ from the state solution of Problem 1 at time t_{k-1}, let us first select the largest residual within $\mathcal{B}$

$$d_{\text{res}} = \max_{i=0,\dots,N_b} \|\eta_{i|k-1} - \eta^r_{i|k-1}\| \tag{7.5}$$

and ensure that it does not outstrip the upper limit allowed

$$d_{\max} = \min(d_{\text{res}}, r). \tag{7.6}$$

Then, to make sure the PH index is a strictly positive function in the range of interest, let us consider a saturation function $d_{\text{sat}} : [0, r] \mapsto (0, r)$ defined by the following Richards curve:

$$d_{\text{sat}}(d_{\max}, \Lambda) = \frac{\mu_1 r}{(1 + \zeta \cdot \exp\{-\theta_2(d_{\max} - \mu_2 r)\})^{1/\zeta}} \in (0, r), \tag{7.7}$$

where $\Lambda = (\mu_1, \mu_2, \theta_2, \zeta)$ is a quadruple that parameterizes the curve (see Figure 7.4-left). In particular, μ_1 is a constant related to the the upper asymptote and whose value is close to 1 but strictly less than 1, μ_2 is a positive real number related to the lag phase, θ_2 is the growth rate, and ζ is a positive real number known as the shape parameter[1].

Definition 13 (PH index)**.** *The PH index is defined as*

$$\lambda(d_{\text{sat}}) = \frac{\sqrt{r^2 - d^2_{\text{sat}}}}{r} \in (0, 1). \tag{7.8}$$

[1]The shape of the d_{sat} curve depends on ζ. If $\zeta = 1$, one has the *logistic function*. If ζ tends to zero, the curve converges to the *Gompertz function*.

An illustration of function (7.8) is provided in Figure 7.4-right. Its profile indicates how the MI controller shapes human control authority. The outcome strongly depends on the combination of the r value and Λ tuning. For instance, one shall choose μ_1 as close to 1 as possible so that the extrema of λ get as close as possible to 0 and 1, expanding the range of human control authority. Factor μ_2 shall be selected relatively close to 0 as it influences the offset within $\mathcal{B}$ at which human control authority starts to decrease. The rate of decrease θ_2 is defined by the slope at the inflection point. The parameter $0 < \zeta \leq 1$ controls the sigmoid curvature. An in-depth analysis has shown that this parameter has little effect on the blending mechanism.

7.5 Simulation results

This section describes the numerical testbeds for studies on human-quadrotor interaction with emulated pilots. The results are discussed in terms of the effectiveness and efficiency of the MI controller.

7.5.1 Human-quadrotor benchmark

Robot Let us denote with $\{\mathcal{I}\}$ the inertial frame, and with $\{\mathcal{B}\}$ the body frame located at the quadrotor's CoM. The quadrotor state is composed of its position $p = (x, y, z) \in \mathbb{R}^3$ expressed in $\{\mathcal{I}\}$, orientation $\gamma = (\phi, \theta, \psi) \in \mathbb{R}^3$, linear velocity $v_b = (v_x, v_y, v_z) \in \mathbb{R}^3$ expressed in $\{\mathcal{B}\}$, angular rate $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ expressed in $\{\mathcal{B}\}$ and rotational speed of the propellers $\Omega = (\Omega_1, \Omega_2, \Omega_3, \Omega_4) \in \mathbb{R}^4$, bounded by $\mathcal{X} = \{\Omega \in \mathbb{R}^4 : \underline{\Omega} \leq \Omega \leq \bar{\Omega}\}$. The system control inputs are the rotor torques, defined by $u := (\tau_1, \tau_2, \tau_3, \tau_4) \in \mathbb{R}^4$ and constrained in magnitude $\mathcal{U} = \{u \in \mathbb{R}^4 : \underline{u} \leq u \leq \bar{u}\}$. Its nonlinear dynamics are then given by the first-order ODEs (3.35). As an example, we report in Table 3.3 the values of the dynamic parameters appearing in (3.35) corresponding to the MikroKopter quadrotor platform. Note that a more intuitive tuning of the weights associated with the human and actual robot costs is possible thanks to the model scaling.

Reference trajectory An offline planner provides a Cartesian helical trajectory η^r that avoids collisions with existing obstacles. This means that task variables are defined as $\eta := p \in \mathbb{R}^3$. Note that the reference trajectory implicitly enforces safety.

Pilot inputs Usually, pilots provide the quadrotor with the desired roll, pitch, z angular rate, and total thrust, i.e., $g = (\phi_h, \theta_h, \omega_z, F_{z,h}) \in \mathbb{R}^4$. For simplicity, we assume that the desired z angular rate is zero throughout the task, and the total thrust is mapped into the speed of the propellers, i.e., $\nu^r : g \mapsto \mathbb{R}^6$.

Literature suggests that MPC provides a favorable basis for shaping the human decision-making process (see [244] and references therein). Supported by these findings, we use a second NMPC controller to simulate the inputs of a human pilot.

Its finite-time OCP reads as follows:

$$\begin{aligned} \min_{\substack{\vartheta_0,\dots,\vartheta_N,\\ u_0,\dots,u_{N-1}}} \quad & \sum_{i=0}^{N-1} L_h(\eta_i,\gamma_i,v_{b,i},\omega_i,\Omega_i,u_i) + M_h(\eta_N,\gamma_N,v_{b,N},\omega_N,\Omega_N) \\ \text{s.t.} \quad & \text{(7.2b)} - \text{(7.2e)}, \end{aligned} \tag{7.9}$$

where

$$\begin{aligned} L_h(\cdot) &= \frac{1}{2}(\Delta\eta_i^T Q_1 \Delta\eta_i + \gamma_i^T Q_2 \gamma_i + v_{b,i}^T Q_3 v_{b,i} + \omega_i^T Q_4 \omega_i + \Delta\Omega_i^T Q_5 \Delta\Omega_i + u_i^T R_1 u_i) \\ M_h(\cdot) &= \frac{1}{2}(\Delta\eta_N^T Q_{1N} \Delta\eta_N + \gamma_N^T Q_{2N} \gamma_N + v_{b,N}^T Q_{3N} v_{b,N} + \omega_N^T Q_{4N} \omega_N + \\ &\qquad \Delta\Omega_N^T Q_{5N} \Delta\Omega_N). \end{aligned}$$

Therein, $\Delta\Omega_i = \Omega_i - \Omega_{\text{hov}}$ for $i = 0,\dots,N$ represents the propeller speed errors, with $\Omega_{\text{hov}} = \sqrt{mg/4C_T}$. The weighting matrices are denoted by Q_j, $Q_{jN} \succ 0$ for $j = 1,\dots,5$ and $R_1 \succ 0$. In particular, constraint (7.2d) represents the control interface's real limitation.

At each sampling interval $[t_k, t_{k+1}]$, the solution of (7.9) is computed and the predicted state at stage $i = 1$ is used as pilot inputs, namely

$$\phi_h = \phi_1^\star, \quad \theta_h = \theta_1^\star, \quad \Omega_h = \Omega_1^\star. \tag{7.10}$$

Remark 3. *In practice, the total thrust coming from the joystick can be easily mapped into the propellers' desired speed and passed to the MI controller if one uses Eq.* (3.20) *assuming hovering condition, i.e.,* $\Omega_1 = \Omega_2 = \Omega_3 = \Omega_4 = \Omega_{\text{ss}}$*, yielding*

$$\Omega_{\text{ss}} = \sqrt{\frac{F_{z,h}}{4C_T}}, \quad \Omega_h = \Omega_{\text{ss}} \cdot \mathbf{1_4}.$$

Different skill levels are obtained by shaping the weighting matrices of the NLP (7.9). Flying a quadrotor is less automated for novice pilots and, for this reason, requires more of their attention span than experienced ones. Due to their limited self-regulatory ability, their inputs tend to be oscillatory. On the contrary, experienced pilots' inputs tend to be more precise, presumably reflecting their ability to use sensory cues to support their actions. These behaviors were emulated considering distinct tuning for matrices Q_j, Q_{jN}. Experienced pilots are sketched by an improved tracking performance with a heavily weighted sum of squares. Oppositely, novice pilots underperform using relatively small weights that allow for larger reference deviations.

Assumption 3. *Since, in practice, we need to predict human inputs to solve Problem 1, we chose to use a zero-order hold (ZOH) method. This prediction method assumes that future human inputs will all be the same as current ones, an assumption that has proved effective in experimentation, [245].*

Additional penalty terms As previously hypothesized, pilots would typically perform continuous maneuvers by changing their inputs all the time. To preserve

smoothness in the generated trajectory, we consider additional penalty terms in the MI controller cost function. These terms rely on the following output functions:

$$h(\vartheta_i) = h_N(\vartheta_N) = (\psi; v_b; \omega) \in \mathbb{R}^7 \tag{7.11}$$

and their corresponding references

$$e_i^r = e_N^r = \mathbf{0_7}. \tag{7.12}$$

The vector of zeros implies two underlying assumptions: first, we are not dealing with aerobatic maneuvers; second, we assume it is an educated guess to initialize and, thereby, speed up the subsequent optimization algorithm.

NLPs parameterization Among several approaches available to discretize continuous-time OCPs, this work will use direct multiple shooting due to its convergence and initialization properties. Assuming an equidistant grid and piecewise constant control parameterization, the following initial value problem defines the state trajectory $\vartheta(\pi)$ at each shooting interval:

$$\dot{\vartheta}(\pi) = f_m(\vartheta(\pi), u_i), \quad \vartheta(t_i) = \vartheta_i, \quad \pi \in [t_i, t_{i+1}]. \tag{7.13}$$

Function (7.13) is evaluated numerically, i.e., $\vartheta(t_{i+1}) \approx F_m(\vartheta_i, u_i)$, using a single-step ERK4 per Δt. Additionally, we approximate the Hessian of the Lagrangian using the GGN method so that structure-exploiting convex solvers can make the most of the block structure of the resulting QP.

7.5.2 Software interface

In this chapter, we will be using the RTI method's implementation in `acados`. Through `acados` Python template-based interface, we generate the library that implements Problem 1, which is then wrapped by our framework written in Python. The QP subproblems arising in the SQP algorithm in `acados` are addressed using the `HPIPM` solver, and `BLASFEO`, which is hardware-optimized for the moderately sized matrices present in our mixed-initiative NMPC. We use the `X64_INTEL_HASWELL` implementation in `BLASFEO`. Solution times are further reduced by reformulating the QPs resulting from our NMPC using the efficient partial condensing algorithm implemented in `HPIPM`.

7.5.3 Performance analysis

The parameters used for all the testbeds in the blending mechanism and the mixed-initiative NMPC are in Table 7.1. In particular, we adopt a short virtual horizon N_b to compensate for the fact that with a ZOH method, pilot inputs would have a natural tendency to violate the norm ball's boundary in the long run. The chosen value is a compromise between the base level of "trust" in the human inputs and the switching frequency of control authority. Another assumption is that $\lambda = 0.5$ during the first iterations, i.e., control authority is equally distributed. As previously mentioned, the testbeds incorporate a second autonomous algorithm that emulates

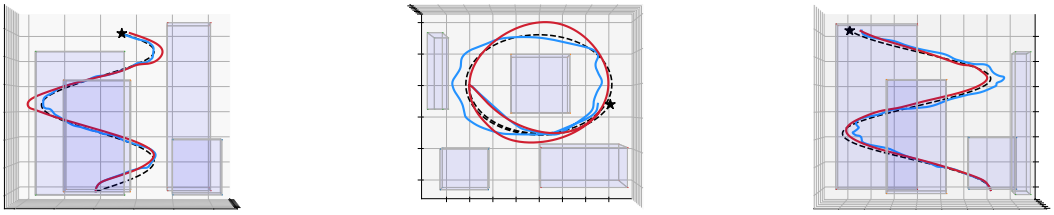


Figure 7.5. Generated trajectories among obstacles (purple) using our mixed-initiative NMPC: novice pilot (blue); experienced pilot (red); reference trajectory is dashed; the $\star$ marks the final goal.

a novice and an experienced pilot. This will enable a comparative analysis of our approach.

Figure 7.5 displays the trajectories generated by the MI control algorithm for both pilots. The results show that the controller maintained the quadrotor inside $\mathcal{B}$, successfully avoiding all obstacles in all cases (see also Fig. 7.7-left). In Figure 7.6, we observe that the mixed-initiative NMPC commands closely follow the pilot inputs when the safety constraint is unlikely to be violated. This outcome is more evident for the experienced pilot, where the lines overlap almost completely, implying that the inputs issued kept the quadrotor closer to the reference trajectory. From Figure 7.7-right, it is obvious that the blending mechanism begins to drop off rapidly the level of control authority for the novice. At the same time, it maintains a relatively high level for the experienced pilot. Overall, note that the control algorithm assists the novice through what is often a very challenging part of their learning journey, the damping in pitching-rolling motions. The algorithm can keep up considerably with the signals' pattern (preserving the pilot's primary intentions), but attenuates their magnitude to cope with the safety constraint. Also, as the physical constraints herein regarded are linear, they can be enforced very efficiently by the optimization algorithm (see Figure 7.8).

Including maximum deviations provides versatility in the quadrotor motion generation. First, it gives a certain level of spatial freedom so that the quadrotor can leverage the motion by exploiting its dynamics. Second, it similarly adds new degrees of freedom to the controller. From a theoretical point of view, the control objective of the mixed-initiative NMPC can be seen as a target set (in the space of the task variables) instead of a target point, since inside $\mathcal{B}$ there are no preferences between one point and another.

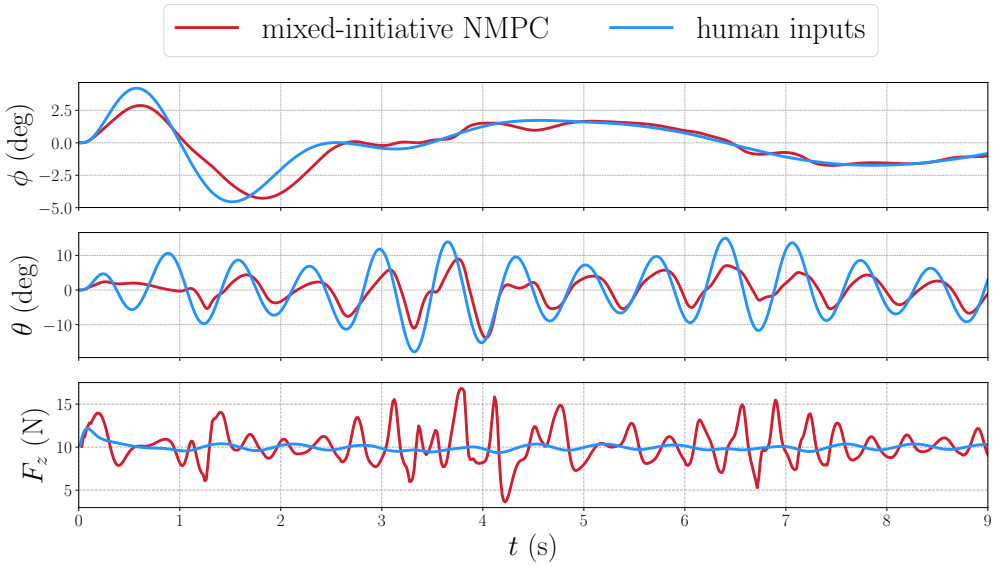
The findings here indicate that the proposed algorithm provides pilots with more control authority without sacrificing safety or even exceeding the robot's physical

Table 7.1. Parameters used by the blending mechanism and the mixed-initiative NMPC

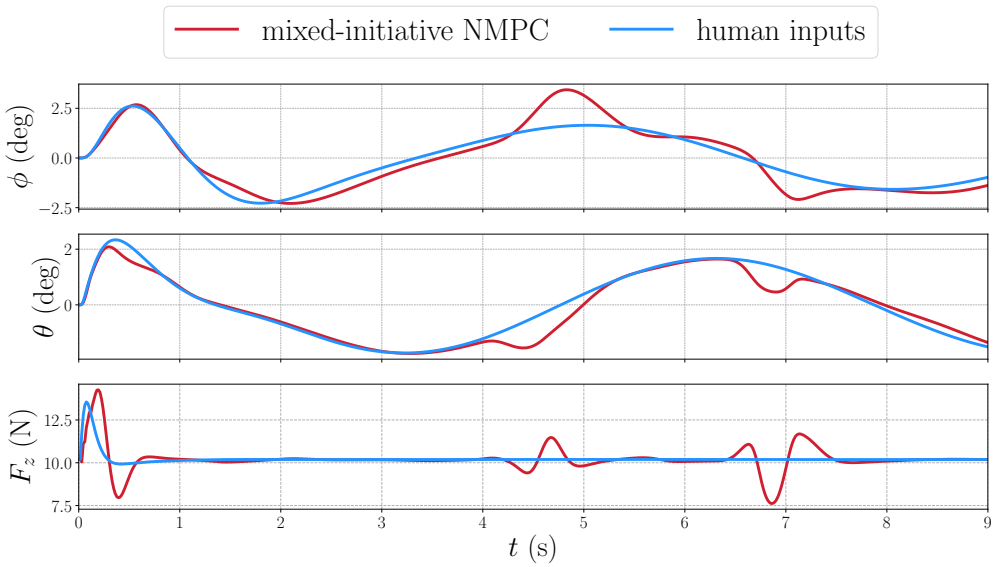
$\underline{\Omega}$	$\mathbf{0}_4$	$\bar{\Omega}$	$0.09 \cdot \mathbf{1}_4$ kHz	μ_1	0.99
$\underline{u}$	$-0.1285 \cdot \mathbf{1}_4$ Nm	$\bar{u}$	$0.1285 \cdot \mathbf{1}_4$ Nm	μ_2	0.3
t_N	0.75 s	N	50	θ_2	55
r	0.175 m	N_b	10	ζ	$1 \cdot 10^{-13}$

$Q_\eta = Q_{\eta_N} = \text{blkdiag}(100 \cdot \mathbf{I}_2, 200)$, $Q_\nu = Q_{\nu_N} = 800 \cdot \mathbf{I}_6$,
$Q_e = Q_{e_N} = \text{blkdiag}(10, 3 \cdot \mathbf{I}_5, 10)$, $R = 70 \cdot \mathbf{I}_4$

limits. Therefore, it has the potential to pave the way to the definition of a standard assistive scheme that can lead to a future improvement in pilot decision-making performance.



(a) Novice.



(b) Experienced.

Figure 7.6. Pilot inputs generated by an autonomous algorithm and mixed-initiative NMPC commands as a result of the blending mechanism.

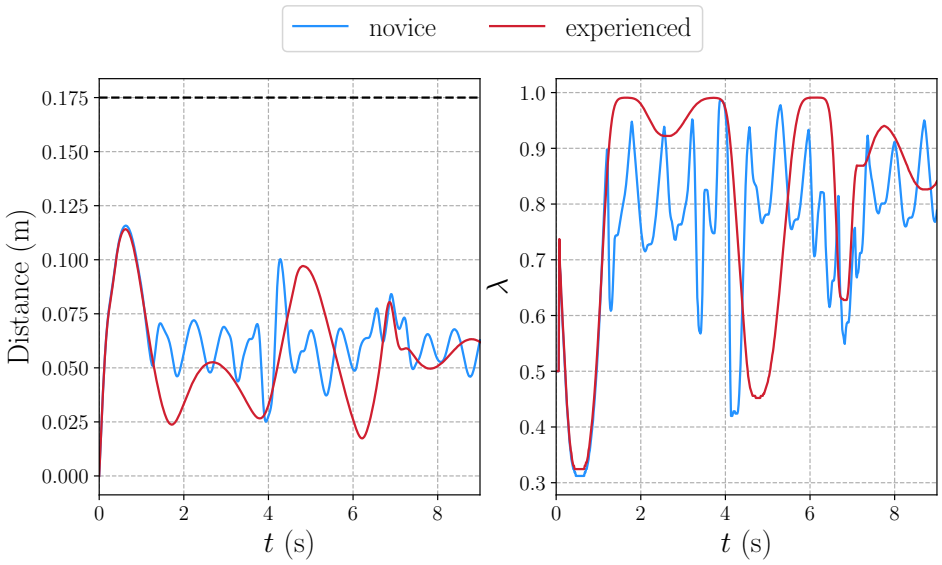
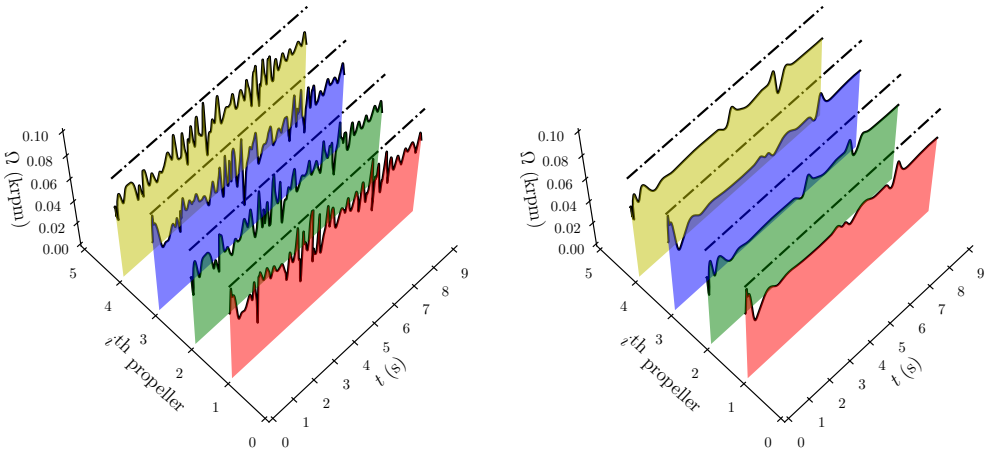


Figure 7.7. Left: distance between generated trajectories and reference; r is dashed. Right: predicted healthiness index profiles indicating the level of human control authority.



(a) Novice. **(b)** Experienced.

Figure 7.8. Generated propeller speeds; the upper bound $\bar{\Omega}$ is dashed.

7.5.4 Computational burden

Solution times reported are from an Intel Core i5-4288U @ 2.6 GHz running macOS Catalina. For a horizon of $N = 50$, the resulting QP has 1016 optimization variables. The testbeds showed that `HPIPM` is significantly faster in solving the partially condensed QP than the corresponding dense QP. In this context, computational efficiency is granted through partial condensing as it allows us to exploit hardware throughput better. The average times per SQP-iteration in `acados` were 4.27 ms for the novice and 3.87 ms for the experienced pilot. Interestingly, one can modify N_b and potentially improve the blending mechanism performance without affecting

solution times significantly. Figure 7.9 shows that changes in N_b affect marginally the solution times. Note that cases in which solution time is larger than Δt or the solver fails are handled using the previous feasible solution at stage $i = 1$.

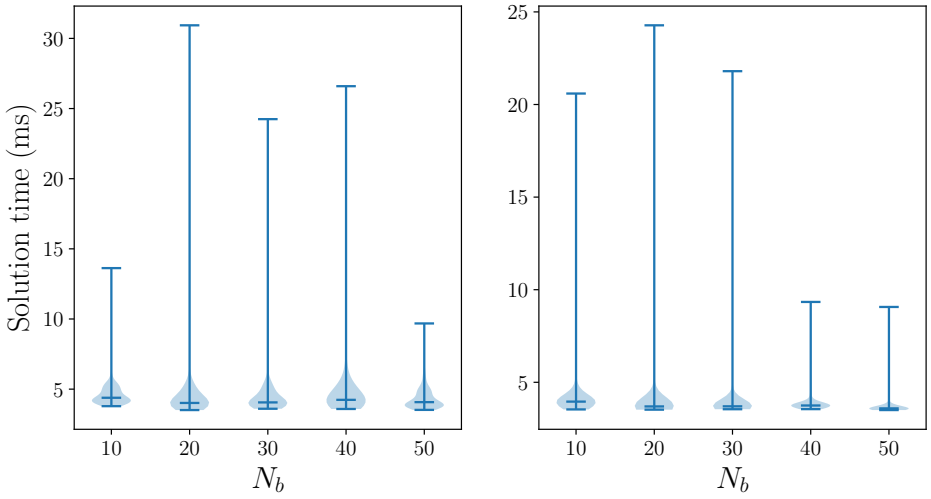


Figure 7.9. Violin plot representations of solution times associated with the mixed-initiative controller in simulations with a novice (left) and an experienced pilot (right). Central bars represent the mean values for each virtual horizon N_b.

7.6 Chapter summary

This chapter presented a novel mixed-initiative control algorithm based on NMPC to enforce safety in human-quadrotor interactions. The algorithm adopts a predict-then-blend approach to mix human inputs and motion generator commands during operation. It assesses whether safety rules are met to perform the control authority blending through the quadrotor's predicted positions. Then it uses an efficient RTI-based scheme that iteratively refines the combined costs so that all NMPC constraints are fulfilled within the available computation time. Simulations showed that the approach allows safety enforcement with specific benefits for useful assistance to pilots, especially novices, and low computational effort. Thanks to high-performance software implementations, we could further speed up solution times and achieve computational efficiency.

Chapter 8

Conclusions and future research

This thesis has contributed with an assortment of tailored algorithms for real-time motion generation in robotic systems capable of taking full advantage of NMPC potentialities. For this purpose:

Chapter 2 first introduced the numerical optimization algorithms that form the basis for implementing NMPC in real-time. Suitable parameterization of the optimization variables is an essential element when considering direct methods for transcribing OCPs that typically arise in MPC. The chapter presented and discussed the choices made concerning the control and state parameterizations that gave rise to a general multiple shooting-type NLP. These choices culminate into a large NLP but with highly structured linear subsystems, where structure-exploiting algorithms play a crucial role in meeting strict real-time limits. The chapter continued explaining SQP methods and their relation with Newton-type optimization, motivated by the fact that many motion generation problems are formulated using a quadratic interpretation of the resulting NLP. The chapter provided a broader view of how Newton-type optimization algorithms approximate the Hessian matrix in terms of convergence rate and computational complexity. In particular, it is shown that the GGN method's properties make it a suitable candidate when addressing the QPs resulting from the practical applications of interest in this thesis. The fast solution of the QP subproblems inspires a follow-up outline on efficient structure-exploiting algorithms. It is shown that one can condense the sparse QP into a smaller but dense one and solve it using, e.g., an efficient active-set method QP solver. Alternatively, one can introduce a new degree of freedom to the algorithm to find the optimal level of sparsity for the QP of interest and directly use a structure-exploiting QP solver. This outline is leveraged by showing the situations in which these condensing approaches may be favorable. The chapter finally binds together the various theoretical concepts on numerical optimization to present the well-established RTI scheme, the algorithm used by the real-time NMPC controllers in the thesis.

Chapter 3 provided the mathematical derivation of the nonlinear models of a double inverted pendulum – also known as the Pendubot – and a quadrotor. Unlike the Pendubot, whose most popular model already exposes existing rotor torque

limitations, the quadrotor model requires proper conjectures that force the modeler to state the system's dynamic limitations explicitly. The chapter showed that the chosen quadrotor model is a compromise between the amount of fidelity to the real system and the simplicity that allows for feasible solutions in real-time on resource-constrained hardware.

Chapter 4 described the design and implementation of a real-time NMPC controller tailored to the Pendubot and with the aim of tracking its unstable equilibrium points. The controller considers the bounds on the rotor torque and some operational constraints relevant to the task. However, because of model uncertainties and sensor noise, the NMPC solution turned out infeasible in practice. To overcome this shortcoming and, thereby, recover local feasibility, operational constraints were softened by including slack variables into the optimization problem. Thereupon, experimental results showed that the robot successfully performed the swing-up task, despite minor violations of operational constraints. Condensing proved favorable at the consider horizon, i.e., $N = 10$, making any solver based on dense linear algebra competitive in terms of fast solution times.

Future research: Exact penalty functions, such as ℓ_1 and ℓ_∞ norms, have a natural advantage that constraints are violated only when necessary. On the other hand, ℓ_2^2 penalty functions are known for placing extreme emphasis on constraint violations in a jagged way, requiring sufficiently large penalization parameters to enforce feasibility, which potentially leads to more steps and, hence, more function evaluations. Motivated by these problems, the reassessment of the controller's performance in terms of penalty functions could be a future study.

Chapter 5 presented an efficient position control architecture based on real-time NMPC with time-delay compensation for quadrotors. The architecture consists of predicting the state over the RTT using an ERK4 integration scheme during the estimation phase and then passing it to the NMPC, which considers the propeller speed limitations. Efficiency is attained through high-performance software implementation covering the RTI scheme, a tailored partial condensing algorithm, a structure-exploiting QP solver, and a hardware-optimized LA library. The proposed control architecture achieved high-accuracy tracking performance, computational efficiency, and constraint satisfaction, as demonstrated by the experimental results.

Future research: Communication time-delay has a deep-seated relationship with custom packages' payload size, meaning that time-delay is naturally time-varying. In this light, one can enhance the state predictor by considering an integrator where the RTT is estimated online. Since with the current approach, no clock synchronization is required, one can use the timer of the device in which all computations are performed to measure the current RTT and use it as the integration step. In this way, it is possible to take hold of time-varying delay cases and improve control performance due to more accurate delay measurements. One can further improve the performance by including two integration schemes onboard the Crazyflie. They should run at the frequency of the attitude and rate controllers so that NMPC commands are reachable within the respective sampling rates. This second enhancement is expected to reduce the existing position overshoots considerably.

Chapter 6 proposed a novel real-time motion generation approach for quadrotors in dynamic environments drawn upon the RTI scheme, where a least conservative linearized constraint was dovetailed to reduce conservativeness in the generated trajectories. The chapter included a theoretical analysis proving that the proposed constraint formulation is less conservative for planners based on the Newton-type method than those based on a fully converged NMPC approach. This was done by showing that the affine outer function in the proposed original nonlinear constraint formulation is identical to its linearization, while the outer strongly convex function of the alternative baseline, when linearized, leads to an underestimator. The chapter further leveraged this analysis to overcome the numerical difficulties associated with the (local) feasibility of the optimization problem. Simulations considering the Crazyflie nano-quadrotor dynamic model validated the approach and, at the same time, reassured the claim of efficient computational performance.

Future research: Future developments involve real-world implementation and investigation of the algorithm's sensitivity to initialization. In practice, when a high-quality initial guess (e.g., dynamically feasible, collision-free, etc.) is available, one should use it. However, a global planner capable of providing such an initial trajectory is often not available and may be expensive to design or time-consuming to run. An investigation in this direction can reveal how challenging it is for the proposed method to improve on, for instance, initial guesses going straight through multiple obstacles.

Chapter 7 introduced a new control algorithm based on real-time NMPC to enforce safety in human-quadrotor interactions. The algorithm uses the optimal state solution to assess whether safety- and/or task-related rules are fulfilled to blend human inputs and motion generator commands. The blending mechanism is a continuous function, named PH index, which is in charge of measuring how much the rules are being violated and lending the most control authority to the most capable agent at any time. The RTI-based mixed-initiative NMPC controller refines the combined commands to fulfill all intrinsic constraints within the available computation time. Numerical testbeds considering simulated pilots with different skill levels indicated that the algorithm could assist pilots, especially novices, and reduce the risk of crashes. The results also underpinned the MI controller's computational efficiency, even under the challenge of finding a solution for an NLP that contains a high-dimensional quadrotor system within restricted execution times.

Future research: One future development is the inclusion of an actual human in the loop by integrating a joystick and a graphical user interface to the testbed environment. A natural subsequent step is to upgrade the framework to integrate a real quadrotor. It is desirable to develop metrics that help assess pilots' learning curves and provide real-world evidence of improved proficiency. Additionally, it would be beneficial to investigate the stability of the MI controller.

Bibliography

[1] G. Turrisi, B. Barros Carlos, M. Cefalo, V. Modugno, L. Lanari, and G. Oriolo, "Enforcing constraints over learned policies via nonlinear MPC: Application to the Pendubot," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9502–9507, 2020, doi: https://doi.org/10.1016/j.ifacol.2020.12.2426.

[2] B. Barros Carlos, T. Sartor, A. Zanelli, G. Frison, W. Burgard, M. Diehl, and G. Oriolo, "An efficient real-time NMPC for quadrotor position control under communication time-delay," in *Proc. 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020, pp. 982–989, doi: https://doi.org/10.1109/ICARCV50220.2020.9305513.

[3] "crazyflie_nmpc," 2020. [Online]. Available: https://github.com/bcbarbara/crazyflie_nmpc

[4] B. Barros Carlos, T. Sartor, A. Zanelli, M. Diehl, and G. Oriolo, "Least conservative linearized constraint formulation for real-time motion generation," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9384–9390, 2020, doi: https://doi.org/10.1016/j.ifacol.2020.12.2407.

[5] B. Barros Carlos, A. Franchi, and G. Oriolo, "Towards safe human-quadrotor interaction: Mixed-initiative control via real-time NMPC," *IEEE Robotics and Automation Letters, Special Issue: Shared Autonomy for Physical Human-Robot Interaction*, 2021 (to appear), doi: https://doi.org/10.1109/LRA.2021.3096502.

[6] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[7] S. J. Qin and T. A. Badgwell, "An overview of industrial model predictive control technology," in *AIche symposium series*, vol. 93, no. 316. New York, NY: American Institute of Chemical Engineers, 1971-c2002, 1997, pp. 232–256.

[8] A. Bemporad, F. Borrelli, and M. Morari, "The explicit solution of constrained LP-based receding horizon control," in *Proc. 2000 IEEE 39th Conference on Decision and Control (CDC)*, vol. 1, 2000, pp. 632–637.

[9] R. Qi, M. Rushton, A. Khajepour, and W. W. Melek, "Decoupled modeling and model predictive control of a hybrid cable-driven robot (HCDR)," *Robotics and Autonomous Systems*, vol. 118, pp. 1–12, 2019.

[10] J. Wang, Z. Wu, M. Tan, and J. Yu, "Model predictive control-based depth control in gliding motion of a gliding robotic dolphin," *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 1–12, 2019.

[11] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the Koopman operator and model predictive control," 2019. [Online]. Available: https://arxiv.org/abs/1902.02827

[12] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "MPC for humanoid gait generation: Stability and feasibility," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1171–1188, 2020.

[13] X. Xiong, J. Reher, and A. Ames, "Global position control on underactuated bipedal robots: Step-to-step dynamics approximation for step planning," 2020. [Online]. Available: https://arxiv.org/abs/2011.06050

[14] F. Shi, M. Zhao, M. Murooka, K. Okada, and M. Inaba, "Aerial regrasping: Pivoting with transformable multilink aerial robot," in *Proc. 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 200–207.

[15] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-based controller with terrain insight for dynamic legged locomotion," in *Proc. 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2436–2442.

[16] L. S. Pontryagin, V. G. Boltyanski, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*. Interscience Publishes, New York, 1962.

[17] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[18] C. C. Chen and L. Shaw, "On receding horizon feedback control," *Automatica*, vol. 18, no. 3, pp. 349–352, 1982.

[19] S. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations," *Journal of optimization theory and applications*, vol. 57, no. 2, pp. 265–293, 1988.

[20] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," in *Proc. 1988 IEEE 27th Conference on Decision and Control (CDC)*, 1988, pp. 464–465.

[21] W. C. Li and L. T. Biegler, "A multistep, Newton-type control strategy for constrained, nonlinear processes," in *Proc. 1989 American Control Conference (ACC)*, 1989, pp. 1526–1527.

[22] W. C. Li and L. T. Biegler, "Newton-type controllers for constrained nonlinear processes with uncertainty," *Industrial & Engineering Chemistry Research*, vol. 29, no. 8, pp. 1647–1657, 1990.

[23] M. Alamir and G. Bornard, "Stability of a truncated infinite constrained receding horizon scheme: the general discrete nonlinear case," *Automatica*, vol. 31, no. 9, pp. 1353–1356, 1995.

[24] 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.

[25] G. De Nicolao, L. Magni, and R. Scattolini, "Stability and robustness of nonlinear receding horizon control," in *Nonlinear Model Predictive Control*, F. Allgöwer and A. Zheng, Eds. Birkhäuser Basel, 2000, pp. 3–22.

[26] G. C. Goodwin, M. G. Cea, M. M. Seron, D. Ferris, R. H. Middleton, and B. Campos, "Opportunities and challenges in the application of nonlinear MPC to industrial problems," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 39–49, 2012.

[27] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[28] P. L. Lions, *Generalized solutions of Hamilton-Jacobi equations*. London Pitman, 1982, vol. 69.

[29] B. Passenberg, "Theory and algorithms for indirect methods in optimal control of hybrid systems," Ph.D. dissertation, Technische Universität München, 2012.

[30] Y. Vincent, "An inverse dynamic-based dynamic programming method for optimal point-to-point trajectory planning of robotic manipulators," *International Journal of Systems Science*, vol. 26, no. 2, pp. 181–195, 1995.

[31] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *9th IFAC World Congress*, vol. 17, no. 2, pp. 1603–1608, 1984.

[32] M. Vukov, "Embedded model predictive control and moving horizon estimation for mechatronics applications," Ph.D. dissertation, KU Leuven, 2015.

[33] L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Computers & chemical engineering*, vol. 8, no. 3-4, pp. 243–247, 1984.

[34] R. Quirynen, M. Vukov, and M. Diehl, "Auto generation of implicit integrators for embedded NMPC with microsecond sampling times," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 175–180, 2012.

[35] J. Loffeld and M. Tokman, "Comparative performance of exponential, implicit, and explicit integrators for stiff systems of ODEs," *Journal of Computational and Applied Mathematics*, vol. 241, pp. 45–67, 2013.

[36] W. Karush, "Minima of functions of several variables with inequalities as side constraints," Master's thesis, Department of Mathematics, University of Chicago, 1939.

[37] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *(J. Neyman, ed.) Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley*, 1951.

[38] J. Nocedal and S. Wright, *Numerical optimization*, 2nd ed. Springer Science & Business Media, 2006.

[39] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, Belmont, Massachusetts, 1999.

[40] L. T. Biegler, *Nonlinear programming*, ser. Mos-Siam Series on Optimization. SIAM, 2010.

[41] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*. SIAM, 2000.

[42] A. L. Tits, A. Wächter, S. Bakhtiari, T. J. Urban, and C. T. Lawrence, "A primal-dual interior-point method for nonlinear programming with strong global and local convergence properties," *SIAM Journal on Optimization*, vol. 14, no. 1, pp. 173–199, 2003.

[43] R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," *Mathematical programming*, vol. 91, no. 2, pp. 239–269, 2002.

[44] R. Quirynen, B. Houska, M. Vallerio, D. Telen, F. Logist, J. Van Impe, and M. Diehl, "Symmetric algorithmic differentiation based exact hessian SQP method and software for economic MPC," in *Proc. 2014 IEEE 53rd Conference on Decision and Control (CDC)*, 2014, pp. 2752–2757.

[45] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm," in *Proc. 2016 15th European Control Conference (ECC)*, 2016, pp. 141–147.

[46] I. B. Nascimento, A. Ferramosca, L. C. Piment, and G. V. Raffo, "NMPC strategy for a quadrotor uav in a 3d unknown environment," in *Proc. 2019 19th International Conference on Advanced Robotics (ICAR)*, 2019, pp. 179–184.

[47] W. Huang, X. Huang, C. Majidi, and M. K. Jawed, "Dynamic simulation of articulated soft robots," *Nature communications*, vol. 11, no. 1, pp. 1–9, 2020.

[48] R. Quirynen, "Numerical simulation methods for embedded optimization," Ph.D. dissertation, KU Leuven, 2017.

[49] K. Erleben and S. Andrews, "Solving inverse kinematics using exact hessian matrices," *Computers & Graphics*, vol. 78, pp. 1–11, 2019.

[50] M. Khadem, J. O'Neill, Z. Mitros, L. Da Cruz, and C. Bergeles, "Autonomous steering of concentric tube robots for enhanced force/velocity manipulability," in *Proc. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2197–2204.

[51] K. Pfeiffer, "Efficient kinematic and algorithmic singularity resolution for multi-contact and multi-level constrained dynamic robot control," Ph.D. dissertation, Université de Montpellier, 2019.

[52] H.-T. Dang, L. Lapierre, R. Zapata, P. Lepinay, and B. Ropars, "Dynamic configuration for an autonomous underwater robot," in *Proc. 2020 28th Mediterranean Conference on Control and Automation (MED)*, 2020, pp. 520–525.

[53] S. A. T. W. T. V. William H. Press, Brian P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

[54] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[55] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, "Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2746–2753, 2019.

[56] S. Duenser, R. Poranne, B. Thomaszewski, and S. Coros, "RoboCut: hot-wire cutting with robot-controlled flexible rods," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 98:1–98:15, 2020.

[57] I. K. Erunsal, R. Ventura, and A. Martinoli, "Nonlinear model predictive control for 3D formation of multirotor micro aerial vehicles with relative sensing in local coordinates," 2019. [Online]. Available: https://arxiv.org/abs/1904.03742

[58] P. Chen, "Hessian matrix vs. Gauss-Newton hessian matrix," *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1417–1435, 2011.

[59] J. Mainprice, N. Ratliff, M. Toussaint, and S. Schaal, "An interior point method solving motion planning problems with narrow passages," in *Proc. 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2020, pp. 547–552.

[60] A. Kuntz, M. Fu, and R. Alterovitz, "Planning high-quality motions for concentric tube robots in point clouds via parallel sampling and optimization," in *Proc. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2205–2212.

[61] A. Kuntz, C. Bowen, and R. Alterovitz, "Fast anytime motion planning in point clouds by interleaving sampling and interior point optimization," in *Robotics Research*, 2020, pp. 929–945.

[62] C. Li, Q. Shi, Z. Gao, M. Ma, H. Ishii, A. Takanishi, Q. Huang, and T. Fukuda, "Design and optimization of a lightweight and compact waist mechanism for a robotic rat," *Mechanism and Machine Theory*, vol. 146, p. 103723, 2020.

[63] X. Mu and Y. Zhang, "Grasping force optimization for multi-fingered robotic hands using projection and contraction methods," *Journal of Optimization Theory and Applications*, vol. 183, no. 2, pp. 592–608, 2019.

[64] Z. Chen, Q. Wu, C. Hong, and X. Zhang, "Multi-fingered grasping force optimization based on generalized penalty-function concepts," *Robotics and Autonomous Systems*, p. 103672, 2020.

[65] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[66] R. Fletcher, "The sequential quadratic programming method," in *Nonlinear optimization*. Springer, 2010, pp. 165–214.

[67] E. A. Yildirim and S. J. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM Journal on Optimization*, vol. 12, no. 3, pp. 782–810, 2002.

[68] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, M. Raimondo, and F. Allgöwer, Eds. Springer, 2009, vol. 384, pp. 391–417.

[69] Z. Sun, B. Zhang, Y. Sun, Z. Pang, and C. Cheng, "A novel superlinearly convergent trust region-sequential quadratic programming approach for optimal gait of bipedal robots via nonlinear model predictive control," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 2, pp. 401–416, 2020.

[70] J. Chen, T. Liu, and S. Shen, "Tracking a moving target in cluttered environments using a quadrotor," in *Proc. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 446–453.

[71] B. F. Jeon and H. J. Kim, "Online trajectory generation of a MAV for chasing a moving target in 3D dense environments," 2019. [Online]. Available: https://arxiv.org/abs/1904.03421

[72] Y. Shi, P. Wang, M. Li, X. Wang, Z. Jiang, and Z. Li, "Model predictive control for motion planning of quadrupedal locomotion," in *Proc. 2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2019, pp. 87–92.

[73] S. Hong, J.-H. Kim, and H.-W. Park, "Real-time constrained nonlinear model predictive control on SO(3) for dynamic legged locomotion," in *Proc. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 972–979.

[74] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[75] B. Şenbaşlar, W. Hönig, and N. Ayanian, "Robust trajectory execution for multi-robot teams using distributed real-time replanning," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 167–181.

[76] P. Wolfe, "The simplex method for quadratic programming," *Econometrica: Journal of the Econometric Society*, pp. 382–398, 1959.

[77] E. L. S. Wong, "Active-set methods for quadratic programming," Ph.D. dissertation, UC San Diego, 2011.

[78] A. Forsgren, P. E. Gill, and E. Wong, "Primal and dual active-set methods for convex quadratic programming," *Mathematical programming*, vol. 159, no. 1-2, pp. 469–508, 2016.

[79] P. E. Gill, W. Murray, and M. A. Saunders, "User's guide for QPOPT 1.0: A fortran package for quadratic programming," 1995.

[80] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical programming*, vol. 27, no. 1, pp. 1–33, 1983.

[81] M. Fält and P. Giselsson, "QPDAS: Dual active set solver for mixed constraint quadratic programming," in *Proc. 2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 4891–4897.

[82] A. Potschka, C. Kirches, H. G. Bock, and J. P. Schlöder, "Reliable solution of convex quadratic programs with parametric active set methods," Interdisciplinary Center for Scientific Computing, Heidelberg University, Im Neuenheimer Feld, Heidelberg, Germany, Tech. Rep. 368, 69120, 2010.

[83] H. J. Ferreau, "An online active set strategy for fast solution of parametric quadratic programs with applications to predictive engine control," Ph.D. dissertation, University of Heidelberg, 2006.

[84] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[85] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[86] D. Axehill and M. Morari, "An alternative use of the Riccati recursion for efficient optimization," *Systems & Control Letters*, vol. 61, no. 1, pp. 37–40, 2012.

[87] J. A. Andersson, "General-purpose software framework for dynamic optimization," Ph.D. dissertation, KU Leuven, 2013.

[88] G. Frison and J. B. Jørgensen, "Efficient implementation of the riccati recursion for solving linear-quadratic control problems," in *Proc. 2013 IEEE International Conference on Control Applications (CCA)*, 2013, pp. 1117–1122.

[89] G. Frison, "Algorithms and methods for high-performance model predictive control," Ph.D. dissertation, Technical University of Denmark, 2016.

[90] A. G. Pandala, Y. Ding, and H. Park, "qpSWIFT: A real-time sparse quadratic program solver for robotic applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3355–3362, 2019.

[91] H. Jiang, T. Kathuria, Y. T. Lee, S. Padmanabhan, and Z. Song, "A faster interior point method for semidefinite programming," 2020. [Online]. Available: https://arxiv.org/abs/2009.10217

[92] M. O'Neill and S. J. Wright, "A log-barrier Newton-CG method for bound constrained optimization with complexity guarantees," *IMA Journal of Numerical Analysis*, p. drz074, 2020.

[93] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of optimization theory and applications*, vol. 99, no. 3, pp. 723–757, 1998.

[94] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," 2020. [Online]. Available: https://arxiv.org/abs/2003.02547

[95] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2009.

[96] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl, "BLASFEO: Basic linear algebra subroutines for embedded optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 44, no. 4, pp. 1–30, 2018.

[97] D. Axehill, "Controlling the level of sparsity in MPC," *Systems & Control Letters*, vol. 76, pp. 1–7, 2015.

[98] D. Kouzoupis, R. Quirynen, J. Frasch, and M. Diehl, "Block condensing for fast nonlinear MPC with the dual Newton strategy," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 26–31, 2015.

[99] G. Frison, D. Kouzoupis, J. B. Jørgensen, and M. Diehl, "An efficient implementation of partial condensing for nonlinear model predictive control," in *Proc. 2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 4457–4462.

[100] T. Ohtsuka, "A continuation/GMRES method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, 2004.

[101] D. Kouzoupis, "Structure-exploiting numerical methods for tree-sparse optimal control problems," Ph.D. dissertation, Albert-Ludwigs-Universität Freiburg im Breisgau, 2019.

[102] V. M. Zavala and L. T. Biegler, "The advanced-step NMPC controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[103] T. A. Johansen, "On multi-parametric nonlinear programming and explicit nonlinear model predictive control," in *Proc. 2002 IEEE 41st Conference on Decision and Control (CDC)*, vol. 3, 2002, pp. 2768–2773.

[104] E. N. Pistikopoulos, "Perspectives in multiparametric programming and explicit model predictive control," *AIChE journal*, vol. 55, no. 8, pp. 1918–1925, 2009.

[105] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications*. Springer Science & Business Media, 2012, vol. 429.

[106] C. Büskens and H. Maurer, "Sensitivity analysis and real-time control of parametric optimal control problems using nonlinear programming methods," in *Online Optimization of Large Scale Systems*. Springer, 2001, pp. 57–68.

[107] J. V. Kadam and W. Marquardt, "Sensitivity-based solution updates in closed-loop dynamic optimization," *IFAC Proceedings Volumes*, vol. 37, no. 9, pp. 947–952, 2004.

[108] V. M. Zavala, C. D. Laird, and L. T. Biegler, "Fast implementations and rigorous models: Can both be accommodated in NMPC?" *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, no. 8, pp. 800–815, 2008.

[109] E. Suwartadi, V. Kungurtsev, and J. Jäschke, "Sensitivity-based economic NMPC with a path-following approach," *Processes*, vol. 5, no. 1, pp. 1–18, 2017.

[110] E. Suwartadi and J. Jäschke, "Fast sensitivity-based nonlinear economic model predictive control with degenerate NLP," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 399–404, 2018.

[111] N. M. De Oliveira and L. T. Biegler, "An extension of Newton-type algorithms for nonlinear process control," *Automatica*, vol. 31, no. 2, pp. 281–286, 1995.

[112] J. T. Wen, S. Seereeram, and D. S. Bayard, "Nonlinear predictive control applied to spacecraft attitude control," in *Proc. 1997 American Control Conference (ACC)*, vol. 3, 1997, pp. 1899–1903.

[113] D. DeHaan and M. Guay, "A new real-time perspective on non-linear model predictive control," *Journal of process control*, vol. 16, no. 6, pp. 615–624, 2006.

[114] K. Butts, A. Dontchev, M. Huang, and I. Kolmanovsky, "A perturbed chord (Newton-Kantorovich) method for constrained nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 253–258, 2016.

[115] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *Proc. 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 1939–1944.

[116] M. Diehl, "Real-time optimization for large scale nonlinear processes," Ph.D. dissertation, Heidelberg University, 2001.

[117] M. Diehl, R. Findeisen, and F. Allgöwer, "A stabilizing real-time implementation of nonlinear model predictive control," in *Real-Time and Online PDE-Constrained Optimization*, L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, Eds. SIAM, 2007, pp. 23–52.

[118] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*. Springer, 2009, pp. 391–417.

[119] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[120] A. Zanelli, Q. T. Dinh, and M. Diehl, "A Lyapunov function for the combined system-optimizer dynamics in nonlinear model predictive control," 2020. [Online]. Available: https://arxiv.org/abs/2004.08578

[121] J. V. Frasch, L. Wirsching, S. Sager, and H. G. Bock, "Mixed-level iteration schemes for nonlinear model predictive control," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 138–144, 2012.

[122] L. Wirsching, "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control," Ph.D. dissertation, Heidelberg University.

[123] A. Nurkanović, A. Zanelli, S. Albrecht, and M. Diehl, "The advanced step real time iteration for NMPC," in *Proc. 2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 5298–5305.

[124] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl, "acados: a modular open-source framework for fast embedded optimal control," 2019. [Online]. Available: https://arxiv.org/abs/1910.13753

[125] J. Kalmari, J. Backman, and A. Visala, "A toolkit for nonlinear model predictive control using gradient projection and code generation," *Control Engineering Practice*, vol. 39, pp. 56–66, 2015.

[126] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli, "The Control Toolbox – An open-source C++ library for robotics, optimal and model predictive control," in *Proc. 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2018, pp. 123–129.

[127] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi, "MATMPC - a MATLAB based toolbox for real-time nonlinear model predictive control," in *Proc. 2019 18th European Control Conference (ECC)*, 2019, pp. 3365–3370.

[128] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen, "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)," *Optimization and Engineering*, vol. 20, no. 3, pp. 769–809, 2019.

[129] J. V. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 289–329, 2015.

[130] A. Domahidi and J. Jerez, "FORCES Pro User Manual," 2014–2019. [Online]. Available: https://embotech.com/FORCES-Pro

[131] G. Frison, "HPMPC - library for high-performance implementation of solvers for MPC," 2014–2019. [Online]. Available: https://github.com/giaf/hpmpc

[132] B. Houska, H. Ferreau, M. Vukov, and R. Quirynen, "ACADO Toolkit User's Manual," 2009–2013. [Online]. Available: http://www.acadotoolkit.org

[133] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit – An open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[134] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 685–704, 2015.

[135] E. Mikuláš, M. Gulan, and G. Takács, "Model predictive torque vectoring control for a formula student electric racing car," in *Proc. 2018 European Control Conference (ECC)*, 2018, pp. 581–588.

[136] J. Sacks, D. Mahajan, R. C. Lawson, and H. Esmaeilzadeh, "RoboX: An end-to-end solution to accelerate autonomous control in robotics," in *Proc. 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 479–490.

[137] T. Stastny and R. Siegwart, "Nonlinear model predictive guidance for fixed-wing UAVs using identified control augmented dynamics," in *Proc. 2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 432–442.

[138] B. B. Kocer, T. Tjahjowidodo, M. Pratama, and G. G. L. Seet, "Inspection-while-flying: An autonomous contact-based nondestructive test using UAV-tools," *Automation in Construction*, vol. 106, p. 102895, 2019.

[139] R. Ritschel, F. Schrödel, J. Hädrich, and J. Jäkel, "Nonlinear model predictive path-following control for highly automated driving," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 350–355, 2019.

[140] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3620–3625, 2020.

[141] N. Imanberdiyev and E. Kayacan, "Redundancy resolution based trajectory generation for dual-arm aerial manipulators via online model predictive control," in *Proc. 46th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2020, pp. 674–681.

[142] G. Zogopoulos-Papaliakos and K. J. Kyriakopoulos, "A flight envelope determination and protection system for fixed-wing UAVs," in *Proc. 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9599–9605.

[143] M. Faroni, M. Beschi, C. G. L. Bianco, and A. Visioli, "Predictive joint trajectory scaling for manipulators with kinodynamic constraints," *Control Engineering Practice*, vol. 95, p. 104264, 2020.

[144] S. Adhau, S. Patil, D. Ingole, and D. Sonawane, "Implementation and analysis of nonlinear model predictive controller on embedded systems for real-time applications," in *Proc. 2019 18th European Control Conference (ECC)*, 2019, pp. 3359–3364.

[145] 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, no. 1, pp. 1–36, 2019.

[146] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[147] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 1, pp. 58–81, 2003.

[148] R. Quirynen, S. Gros, B. Houska, and M. Diehl, "Lifted collocation integrators for direct optimal control in ACADO toolkit," *Mathematical Programming Computation*, vol. 9, no. 4, pp. 527–571, 2017.

[149] J. Frey, R. Ouirynen, D. Kouzoupis, G. Frison, J. Geisler, A. Schild, and M. Diehl, "Detecting and exploiting generalized nonlinear static feedback structures in DAE systems for MPC," in *Proc. 2019 18th European Control Conference (ECC)*, 2019, pp. 2756–2762.

[150] R. Verschueren, N. van Duijkeren, R. Quirynen, and M. Diehl, "Exploiting convexity in direct optimal control: a sequential convex quadratic programming method," in *Proc. 2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 1099–1104.

[151] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Frison, and M. Diehl, "NMPC for racing using a singularity-free path-parametric model with obstacle avoidance."

[152] A. Zanelli, G. Horn, G. Frison, and M. Diehl, "Nonlinear model predictive control of a human-sized quadrotor," in *Proc. 2018 17th European Control Conference (ECC)*, 2018, pp. 1542–1547.

[153] A. Eskandarpour and I. Sharf, "A constrained error-based MPC for path following of quadrotor with stability analysis," *Nonlinear Dynamics*, vol. 99, no. 2, pp. 899–918, 2020.

[154] M. Owis, S. El-Bouhy, and A. El-Badawy, "Quadrotor trajectory tracking control using non-linear model predictive control with ROS implementation," in *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*, 2019, pp. 243–247.

[155] X. Zhang, J. Ma, Z. Cheng, S. Huang, S. S. Ge, and T. H. Lee, "Trajectory generation by chance-constrained nonlinear MPC with probabilistic prediction," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[156] W. R. Hamilton, "Theory of quaternions," *Proceedings of the Royal Irish Academy (1836-1869)*, vol. 3, pp. 1–16, 1844.

[157] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems*, 3rd ed. John Wiley & Sons, 2016.

[158] R. Mahony, V. Kumar, and P. Corke, "Modeling, estimation and control of quadrotor aerial vehicles," *IEEE Robotics & Automation Magazine*, pp. 20–32, 2012.

[159] M. A. Haidekker, "Chapter 8 – Linearization of nonlinear components," in *Linear Feedback Controls*. Elsevier, 2013, pp. 113–120.

[160] J. Förster, "System identification of the Crazyflie 2.0 nano quadrocopter," Bachelor thesis, Institute for Dynamic Systems and Control, ETH Zurich, 2015.

[161] C. Luis and J. L. Ny, "Design of a trajectory tracking controller for a nanoquadcopter," 2016. [Online]. Available: https://arxiv.org/abs/1608.05786

[162] M. Greiff, "Modelling and control of the Crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation," Master's thesis, Department of Automatic Control, Lund University, 2017.

[163] O. Boubaker and R. Iriarte, *The inverted pendulum in control theory and robotics: From theory to new innovations*. IET, 2017, vol. 111.

[164] J. Hauser and R. M. Murray, "Nonlinear controllers for non-integrable systems: The acrobot example," in *Proc. 1990 American Control Conference*, 1991, pp. 669–671.

[165] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[166] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted pendulum using pseudo-state feedback," *Proc. Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 206, no. 4, pp. 263–269, 1992.

[167] M. W. Spong, P. Corke, and R. Lozano, "Nonlinear control of the reaction wheel pendulum," *Automatica*, vol. 37, no. 11, pp. 1845–1851, 2001.

[168] J. Hauser, S. Sastry, and P. Kokotović, "Nonlinear control via approximate input-output linearization: The ball and beam example," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 392–398, 1992.

[169] M. W. Spong and D. J. Block, "The pendubot: A mechatronic system for control research and education," in *Proc. 1995 IEEE 34th Conference on Decision and Control (CDC)*, vol. 1, 1995, pp. 555–556.

[170] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proc. 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2001, pp. 239–246.

[171] H. G. Nguyen, J. Morrell, K. D. Mullens, A. B. Burmeister, S. Miles, N. Farrington, K. M. Thomas, and D. W. Gage, "Segway robotic mobility platform," in *Proc. SPIE Mobile Robots XVII*, vol. 5609, 2004, pp. 207–220.

[172] T. Lauwers, G. Kantor, and R. Hollis, "One is enough!" in *Robotics Research*. Springer Berlin Heidelberg, 2007, pp. 327–336.

[173] W. R. Sturgeon and W. V. Loscutoff, "Application of modal control and dynamic observers to control of a double inverted pendulum," *Joint Automatic Control Conference*, vol. 10, pp. 857–865, 1972.

[174] H. Kimura, "Geometric structure of observers for linear feedback control laws," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 846–855, 1977.

[175] K. Furuta, T. Okutani, and H. Sone, "Computer control of a double inverted pendulum," *Computers & Electrical Engineering*, vol. 5, no. 1, pp. 67–84, 1978.

[176] K. Furuta, H. Kajiwara, and K. Kosuge, "Digital control of a double inverted pendulum on an inclined rail," *International Journal of Control*, vol. 32, no. 5, pp. 907–924, 1980.

[177] D. J. Block and M. W. Spong, "Mechanical design and control of the Pendubot," *SAE International*, vol. 104, pp. 36–43, 1995.

[178] C. Mélin and B. Vidolov, "A fuzzy PD-like scheme for two underactuated planar mechanisms," in *Proc. 2000 IEEE 9th International Conference on Fuzzy Systems*, vol. 2, 2000, pp. 792–797.

[179] E. B. Erdem and A. G. Alleyne, "Experimental real-time sdre control of an underactuated robot," in *Proc. 2001 IEEE 40th Conference on Decision and Control (CDC)*, vol. 3, 2001, pp. 2986–2991.

[180] I. Fantoni, R. Lozano, and A. M. Annaswamy, "Adaptive stabilization of underactuated flexible-joint robots using an energy approach and min-max algorithms," in *Proc. 2000 American Control Conference (ACC)*, vol. 4, 2000, pp. 2511–2512.

[181] S. Ramos-Paz, F. Ornelas-Tellez, and A. G. Loukianov, "Nonlinear optimal tracking control in combination with sliding modes: Application to the Pendubot," in *Proc. 2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, 2017, pp. 1–6.

[182] A. Nayak and R. N. Banavar, "Almost-global tracking of the unactuated joint in a pendubot," *IFAC-PapersOnLine*, vol. 51, no. 3, pp. 137–142, 2018.

[183] K. Chatzilygeroudis and J.-B. Mouret, "Using parameterized black-box priors to scale up model-based policy search for robotics," in *Proc. 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–9.

[184] J. Wu, Y. Wang, W. Ye, and C.-Y. Su, "Control strategy based on Fourier transformation and intelligent optimization for planar pendubot," *Information Sciences*, vol. 491, pp. 279–288, 2019.

[185] S. Sellami, S. Mamedov, and R. Khusainov, "A ROS-based swing up control and stabilization of the Pendubot using virtual holonomic constraints," in *Proc. 2020 International Conference Nonlinearity, Information and Robotics (NIR)*, 2020, pp. 1–5.

[186] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[187] H. Deng and T. Ohtsuka, "ParNMPC – a parallel optimization toolkit for real-time nonlinear model predictive control," *International Journal of Control*, pp. 1–32, 2020.

[188] T. Englert and K. Graichen, "Nonlinear model predictive torque control and setpoint computation of induction machines for high performance applications," *Control Engineering Practice*, vol. 99, p. 104415, 2020.

[189] A. S. Sathya, J. Gillis, G. Pipeleers, and J. Swevers, "Real-time robot arm motion planning and control with nonlinear model predictive control using

augmented Lagrangian on a first-order solver," in *Proc. 2020 19th European Control Conference (ECC)*, 2020, pp. 507–512.

[190] 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.

[191] S. Katayama, M. Doi, and T. Ohtsuka, "A moving switching sequence approach for nonlinear model predictive control of switched systems with state-dependent switches and state jumps," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 2, pp. 719–740, 2020.

[192] J. A. Andersson, J. V. Frasch, M. Vukov, and M. Diehl, "A condensing algorithm for nonlinear MPC with a quadratic runtime in horizon length," *Automatica*, pp. 97–100, 2013.

[193] A. Malyshev, R. Quirynen, and A. Knyazev, "Preconditioned Krylov iterations and condensing in interior point MPC method," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 388–393, 2018.

[194] D. Liao-McPherson and I. Kolmanovsky, "The FBstab quadratic programming method applied to model predictive control: An implicit condensing approach," in *Proc. 2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 3370–3376.

[195] I. Nielsen and D. Axehill, "A parallel structure exploiting factorization algorithm with applications to model predictive control," in *Proc. 2015 IEEE 54th Conference on Decision and Control (CDC)*, 2015, pp. 3932–3938.

[196] I. Washington and C. L. Swartz, "A parallel structure exploiting nonlinear programming algorithm for multiperiod dynamic optimization," *Computers & Chemical Engineering*, vol. 103, pp. 151–164, 2017.

[197] J. Kardoš, D. Kourounis, and O. Schenk, "Structure-exploiting interior point methods," in *Parallel Algorithms in Computational Science and Engineering*. Springer, 2020, pp. 63–93.

[198] Y. Kartal, K. Subbarao, N. R. Gans, A. Dogan, and F. Lewis, "Distributed backstepping based control of multiple UAV formation flight subject to time delays," *IET Control Theory & Applications*, vol. 14, no. 12, pp. 1628–1638, 2020.

[199] J. Alvarez-Muñoz, J. Castillo-Zamora, J. Escareno, I. Boussaada, F. Méndez-Barrios, and O. Labbani-Igbida, "Time-delay control of a multi-rotor VTOL multi-agent system towards transport operations," in *Proc. 2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019, pp. 276–283.

[200] Z. Huang, Y.-J. Pan, and R. Bauer, "Leader–follower consensus control of multiple quadcopters under communication delays," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 10, 2019.

[201] Y. Yang, "Spacecraft attitude determination and control: Quaternion based method," *Annual Reviews in Control*, vol. 36, no. 2, pp. 198–219, 2012.

[202] W. Hönig and N. Ayanian, "Flying multiple UAVs using ROS," in *Robot Operating System (ROS)*. Springer, 2017, pp. 83–118.

[203] I. S. Duff, "MA57 – a code for the solution of sparse symmetric definite and indefinite systems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 30, no. 2, pp. 118–144, 2004.

[204] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[205] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[206] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning-using the voroni diagram for a clearance-based shortest path," *IEEE Robotics & Automation Magazine*, vol. 15, no. 2, pp. 58–66, 2008.

[207] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[208] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. 2000 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2000, pp. 995–1001.

[209] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. 1999 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 1999, pp. 341–346.

[210] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. 1993 IEEE International Conference on Robotics and Automation (ICRA)*, 1993, pp. 802–807.

[211] M. R. Benjamin, M. Defilippo, P. Robinette, and M. Novitzky, "Obstacle avoidance using multiobjective optimization and a dynamic obstacle manager," *IEEE Journal of Oceanic Engineering*, vol. 44, no. 2, pp. 331–342, 2019.

[212] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using CVaR-constrained optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[213] M. Watterson, S. Liu, K. Sun, T. Smith, and V. Kumar, "Trajectory optimization on manifolds with applications to quadrotor systems," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 303–320, 2020.

[214] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "FASTER: Fast and safe trajectory planner for flights in unknown environments," in *Proc. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1934–1940.

[215] Y. Kuwata, "Real-time trajectory design for unmanned aerial vehicles using receding horizontal control," Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2003.

[216] S. Brossette and P.-B. Wieber, "Collision avoidance based on separating planes for feet trajectory generation," in *Proc. 2017 IEEE-RAS 17th International Conference on Humanoid Robotics*, 2017, pp. 509–514.

[217] G. Frison and J. B. Jørgensen, "Efficient solvers for soft-constrained MPC," in *Proc. 19th Nordic Process Control Workshop*. Norwegian University of Science and Technology, 2015.

[218] P. O. M. Scokaert and J. B. Rawlings, "Feasibility issues in linear model predictive control," *AIChE Journal*, vol. 45, no. 8, pp. 1649–1659, 1999.

[219] J. D. Griffin and T. G. Kolda, "Nonlinearly constrained optimization using heuristic penalty methods and asynchronous parallel generating set search," *Applied Mathematics Research eXpress*, vol. 2010, no. 1, pp. 36–62, 2010.

[220] J. M. Maciejowski, *Predictive control: with constraints*. Prentice Hall, 2002.

[221] N. Amirshirzad, A. Kumru, and E. Oztop, "Human adaptation to human–robot shared control," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 2, pp. 126–136, 2019.

[222] X. Bao, V. Molazadeh, A. Dodson, B. E. Dicianno, and N. Sharma, "Using person-specific muscle fatigue characteristics to optimally allocate control in a hybrid exoskeleton–preliminary results," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 2, pp. 226–235, 2020.

[223] Y. Lu, L. Bi, and H. Li, "Model predictive-based shared control for brain-controlled driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 630–640, 2019.

[224] W. M. dos Santos and A. A. Siqueira, "Optimal impedance via model predictive control for robot-aided rehabilitation," *Control Engineering Practice*, vol. 93, pp. 1–8, 2019.

[225] C. C. Ashcraft, M. A. Goodrich, and J. W. Crandall, "Moderating operator influence in human-swarm systems," in *Proc. 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 4275–4282.

[226] C. Nam, P. Walker, H. Li, M. Lewis, and K. Sycara, "Models of trust in human control of swarms with varied levels of autonomy," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 3, pp. 194–204, 2019.

[227] R. Rahal, G. Matarese, M. Gabiccini, A. Artoni, D. Prattichizzo, P. R. Giordano, and C. Pacchierotti, "Caring about the human operator: haptic shared control for enhanced user comfort in robotic telemanipulation," *IEEE Transactions on Haptics*, vol. 13, no. 1, pp. 197–203, 2020.

[228] F. Abi-Farraj, C. Pacchierotti, O. Arenz, G. Neumann, and P. R. Giordano, "A haptic shared-control architecture for guided multi-target robotic grasping," *IEEE Transactions on Haptics*, vol. 13, no. 2, pp. 270–285, 2020.

[229] V. Girbés-Juan, V. Schettino, Y. Demiris, and J. Tornero, "Haptic and visual feedback assistance for dual-arm robot teleoperation in surface conditioning tasks," *IEEE Transactions on Haptics*, vol. 14, no. 1, pp. 44–56, 2021.

[230] S. Parsa, D. Kamale, S. Mghames, K. Nazari, T. Pardi, A. R. Srinivasan, G. Neumann, M. Hanhaide, and A. Ghalamzan E., "Haptic-guided shared control grasping: collision-free manipulation," in *Proc. 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 1552–1557.

[231] R. Moccia, C. Iacono, B. Siciliano, and F. Ficuciello, "Vision-based dynamic virtual fixtures for tools collision avoidance in robotic surgery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1650–1655, 2020.

[232] M. Tannous, M. Miraglia, F. Inglese, L. Giorgini, F. Ricciardi, R. Pelliccia, M. Milazzo, and C. Stefanini, "Haptic-based touch detection for collaborative robots in welding applications," *Robotics and Computer-Integrated Manufacturing*, vol. 64, p. 101952, 2020.

[233] C. Masone, M. Mohammadi, P. Robuffo Giordano, and A. Franchi, "Shared planning and control for mobile robots with integral haptic feedback," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1395–1420, 2018.

[234] A. Franchi, "Human-collaborative schemes in the motion control of single and multiple mobile robots," in *Trends in Control and Decision-Making for Human–Robot Collaboration Systems*. Springer, 2017, pp. 301–324.

[235] A. Moortgat-Pick, A. Adamczyk, T. Tomić, and S. Haddadin, "Feeling the true force in haptic telepresence for flying robots," in *Proc. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9789–9796.

[236] D. Papageorgiou, F. Dimeas, T. Kastritsi, and Z. Doulgeri, "Kinesthetic guidance utilizing dmp synchronization and assistive virtual fixtures for progressive automation," *Robotica*, vol. 38, no. 10, pp. 1824–1841, 2020.

[237] T. Kastritsi, D. Papageorgiou, I. Sarantopoulos, Z. Doulgeri, and G. A. Rovithakis, "Stability of active constraints enforcement in sensitive regions defined by point-clouds for robotic surgical procedures," in *Proc. 2019 18th European Control Conference (ECC)*, 2019, pp. 1604–1609.

[238] B. Weber, R. Balachandran, C. Riecke, F. Stulp, and M. Stelzer, "Teleoperating robots from the international space station: Microgravity effects on performance with force feedback," in *Proc. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 8144–8150.

[239] G. Barnaby and A. Roudaut, "Mantis: A scalable, lightweight and accessible architecture to build multiform force feedback systems," in *Proc. 32nd Annual ACM Symposium on User Interface Software and Technology*, 2019, pp. 937–948.

[240] S. Mintchev, M. Salerno, A. Cherpillod, S. Scaduto, and J. Paik, "A portable three-degrees-of-freedom force feedback origami robot for human–robot interactions," *Nature Machine Intelligence*, vol. 1, no. 12, pp. 584–593, 2019.

[241] S. Fani, S. Ciotti, M. G. Catalano, G. Grioli, A. Tognetti, G. Valenza, A. Ajoudani, and M. Bianchi, "Simplifying telerobotics: Wearability and teleimpedance improves human-robot interactions in teleoperation," *IEEE Robotics & Automation Magazine*, vol. 25, no. 1, pp. 77–88, 2018.

[242] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments," *Vehicle System Dynamics*, vol. 56, no. 6, pp. 853–882, 2018.

[243] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[244] J. R. Anderson and B. Ayalew, "A cascaded optimization approach for modeling a professional driver's driving style," *Journal of Dynamic Systems, Measurement, and Control*, vol. 142, no. 9, p. 10, 2020.

[245] R. Chipalkatty, G. Droge, and M. B. Egerstedt, "Less is more: Mixed-initiative model-predictive control with human inputs," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 695–703, 2013.