Doctoral Thesis

Low-Complexity Numerical Methods for Nonlinear Model Predictive Control

Author(s):
Torrisi, Giampaolo

Publication Date:
2017

Permanent Link:
https://doi.org/10.3929/ethz-b-000225851

Rights / License:
In Copyright - Non-Commercial Use Permitted
Low-Complexity Numerical Methods for Nonlinear Model Predictive Control

A dissertation submitted to attain the degree of

Doctor of Sciences of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

Giampaolo Torrisi
Dottore Magistrale in Ingegneria Elettrica e dell’Automazione
Università degli Studi di Firenze

born 10.12.1988
citizen of Italy

accepted on the recommendation of

Prof. Dr. Manfred Morari (examiner)
Prof. Dr. Roy S. Smith (co-examiner)
Prof. Dr. Melanie N. Zeilinger (co-examiner)
Prof. Dr. Lorenz T. Biegler (co-examiner)

2017
Acknowledgment

Many people have helped me reach this important milestone in my life and I will try to thank them in a few words.

I am indebted to Prof. Manfred Morari for giving me the opportunity of joining the Automatic Control Laboratory (IfA) first for a Master Thesis and then for a Doctorate. His capability of getting to the essence of the problems and quickly assessing the impact of an idea has been inspiring to me. His mentoring has allowed me to get the academic education I originally hoped for.

Prof. Sébastien Mariéthoz has created my job position via the Marie Curie European project Energy Smartops – probably joining IfA would have been much more complicated otherwise. During the initial time of my doctoral studies, I have learned from him the basics of power electronics and he let me find an organized laboratory at IfA.

I am grateful to Prof. Roy Smith for welcoming me to his group even when, at the beginning, my research topics were far from his area of expertise. During these years he helped me out in my research with many fruitful discussions at any stage of my Doctorate.

I would like to thank Prof. Lorenz T. Biegler and Prof. Melanie N. Zeilinger for accepting to be examiners of my Doctoral dissertation and for their valuable suggestions that allowed me to improve the results presented in this thesis.

The greatest power of IfA is the people. The brilliant researchers that have been attracted to the Institute and with whom any student may have the privilege to interact allow ideas to become projects and interesting pieces of research. Dr. Sergio Grammatico has taught me how to organize my work, critically read and deeply understand technical research. He gave me the basics on how to write papers in a clear and intelligible structure. I am grateful to Damian Frick for teaching me how to write clean and structured code. His patience to listen to problems and help troubleshooting a piece of code is probably unbounded. During Prof. Smith’s group and the applied optimization group meetings I had the opportunity to share opinions on recent findings with experts, receiving interesting feedback. I thank Dr. Juan L. Jerez, Dr. Alexander Domahidi and Dr. Andreas B. Hempel of Embotech for their
help (not only) when using the software FORCES Pro. I also thank the undergraduate students that have developed their theses under my supervision, in particular Adrian Zeberli, Alexandros Vouzas, Tommaso Robbiani (twice!), Francesco Seccamonte and Xinyue Li.

During my Doctorate I have also had the possibility to interact with the industry and gain insight into some of their control challenges. During my secondment period in the ABB corporate research center in Cracow, Poland and later in the several visits to the offices in Baden, Switzerland, I have worked on specific case studies such as compression systems and power electronics problems. The profitable interaction has given rise to a good part of my research and I especially thank Andrea Cortinovis, Dr. Mehmet Mercangöz, Dr. Stefan Almér, Victor Jaramillo and Dr. James R. Ottewill. I am grateful for having worked with Dr. Eduardo Gallestey and Dr. Peter Al-Hokayem as a head teaching assistant for the course Nonlinear Control Systems.

During the Energy Smartops project I also had the opportunity to interact with scientists having the most diverse backgrounds and competences. The many interesting discussions with Prof. Nina F. Thorhill, Prof. Ricardo Martinez-Botás, Dr. James R. Ottewill, Dr. Sara Budinis, Victor Jaramillo and all of the other scientists participating in the project gave me the possibility to quickly learn the basics about compressor control.

In my years at IfA as a Master and Doctoral student I have always been sharing a flat with other IfA people. Robert Nguyen, Bart Van Parys and Marcello Colombino have been nice flatmates with whom I have enjoyed sharing my spare time. I hope that they will bring along some of my Italian recipes in their new adventures.

Many good friends at IfA made the time fly. I first encountered Marcello Colombino during my Master and I shared much more than a flat with him throughout these years. I was happy to interview Basilio Gentile and give a positive feedback about him. We have become good friends, apart from the very different opinions about Italian soccer. They both came to visit me during my stay in Cracow and I still have very happy memories of the time spent there together. I thank Dario Paccagnan, Francesca Parise and Tony Wood for the fun moments and the trip together to Japan, as well as Dominic Gross and Tobias Sutter to the United States. I would like to thank Paul Beuchat and Nicoló Pagan for the pleasant discussions that we had (not only) during lunch breaks, often sweetened by some offered desserts. It was my pleasure to interact with Tommaso Novi, that let me remind some of the Florentine
expressions that I had forgotten after some years abroad. It was fun playing indoor volleyball for four years in a Nationalliga B team with David Sturzenegger, who used to be my Master Thesis supervisor and with whom I also had the pleasure to win the championship.

My last thoughts go to my family for their unconditional support in this four-year journey. Probably I would not have even considered going to Switzerland for a Master and Doctorate without their precious advice. Today the benefits of this choice are uncountable. Despite the distance separating Zurich and Como, my girlfriend Eleonora has been a close companion for this journey, working at the same time on her Doctorate in Philosophy and Neuroscience. She has helped me recover from stress and has also been inspiring in my Doctorate. The idea behind Algorithm 2.1 was devised on the train back to Zurich after one of the beautiful weekends spent together.
Abstract

This thesis presents a novel low-complexity method for the solution of the nonlinear optimization programs arising in Model Predictive Control problems. The optimization method consists of taking a gradient step and projecting it onto a linearization of the constraints around the current iterate. Since this approach does not fall into any of the standard optimization methods, we develop the convergence theory and design numerical algorithms to solve a generic optimization problem.

We present several theoretical results. The conceptually simplest implementation of the proposed method is only locally convergent to a local minimum under some technical assumptions. A modification to the method is considered in order to obtain global convergence to a critical point of the problem, i.e., a point satisfying first-order optimality conditions. This is achieved by considering merit functions, whose optimality conditions can be related to the optimality conditions of the original optimization problem.

The method introduced in this thesis is particularly efficient for nonlinear Model Predictive Control problems. In fact, for a specific class of nonlinearly constrained optimization programs and via the introduction of appropriately defined slack variables, the projection onto the linearization of the constraints becomes computationally cheap to evaluate. As a result, the complexity of the method is lower than other typical optimization methods applied to the same problem, since it is only linear in the prediction horizon and quadratic in the state and input dimension. We provide the automatic C code generation tool FalcOpt (First-order Algorithm via Linearization of Constraints for OPTimization) to solve custom Model Predictive Control problems via the proposed numerical algorithms.

Several applications are presented for the proposed method, ranging from industrial centrifugal compression systems to power electronics. We observe that a small number of iterations is typically sufficient to yield an acceptable level of suboptimality. Therefore, the method is particularly aimed at time-critical applications. Furthermore, its capability to deal efficiently with single precision makes it suited for use on embedded devices.
Prefazione

Questa tesi presenta un metodo a bassa complessità per risolvere i problemi di ottimizzazione non lineari tipici dei controlli predittivi basati sul modello. Il metodo di ottimizzazione consiste nel calcolare un passo di gradiente e di proiettarlo sulla linearizzazione locale dei vincoli, calcolata attorno all’iterazione corrente. Questo metodo non ricade tra i classici metodi di ottimizzazione, perciò ne sviluppiamo sia la teoria di convergenza che metodi numerici per problemi di ottimizzazione generali.

Questa tesi contiene molteplici risultati teorici. L’implementazione più semplice dal punto di vista concettuale converge localmente a un minimo locale, assumendo regolarità del problema. Una variazione di tale metodo è considerata per ottenere convergenza globale ad un punto critico del problema, cioè un punto che soddisfa le condizioni di ottimalità del primo ordine. Questo è ottenuto tramite le funzioni di merito, le cui condizioni di ottimalità possono essere correlate alle condizioni di ottimalità del problema di ottimizzazione originale.

Il metodo introdotto in questa tesi è particolarmente efficace per problemi di controllo predittivo non lineare basato sul modello. Infatti, per una classe specifica di problemi di ottimizzazione vincolati nonlineari e tramite l’introduzione di opportune variabili di slack, la proiezione sulla linearizzazione dei vincoli diventa semplice da calcolare. Di conseguenza, la complessità computazionale del metodo è minore di altri metodi classici di ottimizzazione applicati allo stesso problema, risultando essere solo lineare nell’orizzonte di predizione e quadratica nella dimensione dello stato e degli ingressi. Il tool di generazione automatica di codice C FalcOpt (First-order Algorithm via Linearization of Constraints for OPTimization) consente di risolvere problemi di controllo predittivo basati sul modello specificati dall’utente tramite gli algoritmi numerici proposti in questa tesi.

Il metodo proposto ha molteplici possibili applicazioni, in particolare per il controllo di sistemi di compressione centrifughi industriali e di sistemi elettronici di potenza. In pratica si osserva che un numero limitato di iterazioni è in genere sufficiente per ottenere una soluzione vicina ad un punto critico desiderato. Quindi, il metodo è efficace per
applicazioni con basso tempo di campionamento. Inoltre, il metodo è particolarmente indicato per un utilizzo in dispositivi integrati, poiché l’utilizzo di variabili a precisione singola non compromette il corretto funzionamento degli algoritmi.
Notation

Sets

\( \mathbb{N} \) \quad \text{the natural numbers}
\( \mathbb{Z} \) \quad \text{the integer numbers}
\( \mathbb{R} \) \quad \text{the real numbers}
\( \mathbb{R}^n \) \quad \text{the space of vectors of length } n
\( \mathbb{R}^n_{>0} \) \quad \text{the set of element-wise positive vectors in } \mathbb{R}^n
\( \mathbb{R}^n_{\geq 0} \) \quad \text{the set of element-wise non-negative vectors in } \mathbb{R}^n
\( \mathbb{R}^{m \times n} \) \quad \text{the space of matrices of size } m \text{ rows by } n \text{ columns}
\( \mathbb{B} \) \quad \text{unit ball centered in the origin in Euclidean space:}
\[ \mathbb{B} = \{ x \in \mathbb{R}^n \mid x^\top x \leq 1 \} \]
\( x + \varepsilon \mathbb{B} \) \quad \text{ball of radius } \varepsilon \text{ centered in } x \in \mathbb{R}^n \text{ in Euclidean space}

Vectors and Matrices

\( x_j \) \quad j\text{-th component of a vector}
\( x^+ \) \quad \text{positive part of a vector } x \in \mathbb{R}^n:\n\quad x_j^+ = \max(x_j, 0) \quad \forall j \in \{1, \ldots, n\}
\( \text{diag}(x) \), or \( \text{diag}_j(x_j) \) \quad \text{the diagonal matrix whose } (j, j) \text{ element is the } j\text{-th component of the vector } x
\( I \) \quad \text{identity matrix}
\( A^{-1} \) \quad \text{inverse of the matrix } A
\( A_j \) \quad j\text{-th column of the matrix } A
\( \text{blockdiag}_j(A_j) \) \quad \text{the block diagonal matrix whose } (j, j) \text{ block is the matrix } A_j
Notation

Definitions and (In-)equalities

\( A := B \) \hspace{1em} A is defined by \( B \)
\( A \approx B \) \hspace{1em} A is approximately equal to \( B \)
\( A > B \) \hspace{1em} strict element-wise inequality between \( A \) and \( B \)
\( A \geq B \) \hspace{1em} element-wise inequality between \( A \) and \( B \)
\( A \succ B \) \hspace{1em} strict element-wise inequality between symmetric matrices: \( A - B \) is positive definite
\( A \succeq B \) \hspace{1em} strict element-wise inequality between symmetric matrices: \( A - B \) is positive semidefinite

Set Operations

\( A \setminus B \) \hspace{1em} difference of sets \( A \) and \( B \):
\( A \setminus B = \{ a \in A \mid a \notin B \} \)
\( A \times B \) \hspace{1em} Cartesian product of sets \( A \) and \( B \):
\( A \times B = \{ (a, b) \mid a \in A \text{ and } b \in B \} \)

Operators

\( \Pi_C (x) \) \hspace{1em} Euclidean projector of the vector \( x \) on the set \( C \)
\( \|x\|_p \) \hspace{1em} \( \ell_p \) norm of the vector \( x \):
\[ \|x\|_p := \left( \sum_j |x_j|^p \right)^{\frac{1}{p}} \]
\( \|x\|_P \) \hspace{1em} weighted \( \ell_2 \) norm with weight \( P \succ 0 \):
\[ \|x\|_P^2 = x^\top Px \]

Acronyms

CL \hspace{1em} Choke Line
CLF \hspace{1em} Control Lyapunov Function
FLOPS \hspace{1em} Floating Point Operations
IPM \hspace{1em} Interior-Point Method
KKT \hspace{1em} Karush-Kuhn-Tucker
LICQ \hspace{1em} Linear Independence Constraint Qualification
LP \hspace{1em} Linear Program
MFCQ \hspace{1em} Mangasarian-Fromovitz Constraint Qualification
MPC \hspace{1em} Model Predictive Control
NLP \hspace{1em} Nonlinear Program
QCQP \hspace{1em} Quadratically Constrained Quadratic Program
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>QP</td>
<td>Quadratic Program</td>
</tr>
<tr>
<td>RTI</td>
<td>Real-Time Iteration</td>
</tr>
<tr>
<td>SD (SCD)</td>
<td>Surge (Control) Distance</td>
</tr>
<tr>
<td>SL (SCL)</td>
<td>Surge (Control) Line</td>
</tr>
<tr>
<td>SQP</td>
<td>Sequential Quadratic Programming</td>
</tr>
</tbody>
</table>
Contents

Acknowledgment i

Abstract v

Prefazione vii

Notation ix

1 Introduction 1

1.1 Contribution of this thesis . . . . . . . . . . . . . . . . . 5

1.1.1 Theoretical contribution . . . . . . . . . . . . . . . . . 6

1.1.2 Practical implementation . . . . . . . . . . . . . . . . . 7

1.1.3 Control applications . . . . . . . . . . . . . . . . . . . 9

1.2 Publications . . . . . . . . . . . . . . . . . . . . . . . . . 10

I Theory 13

2 A projected gradient and constraint linearization method 15

for nonlinear programs

2.1 Literature review . . . . . . . . . . . . . . . . . . . . . . . 15

2.2 Statement of the nonlinear optimization problem . . . . . 17

2.3 The projected gradient and constraint linearization algo-

rithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.4 Proof of convergence of the proposed algorithm . . . . . 21

2.4.1 Local convergence . . . . . . . . . . . . . . . . . . . . . 23

2.4.2 Global convergence . . . . . . . . . . . . . . . . . . . . . 27

2.4.3 Asymptotic linear convergence . . . . . . . . . . . . . . 33

3 A non-smooth exact penalty function reformulation 37

3.1 Literature review . . . . . . . . . . . . . . . . . . . . . . . . 37

3.2 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . . 40

3.3 Global convergence via a non-smooth merit function . . . 41
4 Practical implementation for Model Predictive Control problems

4.1 General optimization problems .................................. 47
4.2 Model Predictive Control problems .............................. 51
  4.2.1 Quadratic terminal constraint .................................. 51
  4.2.2 Contractive constraint ......................................... 55
4.3 Computational considerations .................................... 56
  4.3.1 Tuning of the gradient step size $\alpha$ ......................... 56
  4.3.2 Stopping criteria ............................................. 58
  4.3.3 Automatic code generation with FalcOpt .................... 59

5 A heuristic to accelerate the practical convergence speed

5.1 Application to Model Predictive Control ....................... 62
  5.1.1 Tuning of the gradient step size $\alpha$ ......................... 65
5.2 Computational considerations .................................... 67
5.3 A further extension .............................................. 69
5.4 Benchmarks for comparison ...................................... 71

II Application: control of a centrifugal compression system

6 Compression systems for industrial applications ............... 75
  6.1 Literature review ................................................ 76
  6.2 The control problem ........................................... 78
    6.2.1 Mathematical model ....................................... 78
    6.2.2 Steady-state conditions ................................... 82
    6.2.3 Standard industrial control ............................... 83

7 Fast contractive MPC with stability and recursive feasibility guarantees

  7.1 Literature review ................................................ 85
  7.2 Formulation of the control problem ....................... 87
    7.2.1 Theoretical guarantees ................................. 87
  7.3 Numerical test of recursive feasibility .................. 89
  7.4 Simulation results ............................................ 93
    7.4.1 Computational times ................................... 95
  7.5 Conclusion ..................................................... 96
1 Introduction

Model Predictive Control (MPC) is an established method to control a dynamical system by using predictions of the state evolution into the future. The control input is designed as the solution to an optimization problem, solved in real-time. This control method has gained significant popularity in the last decades and it has been extensively analyzed in the literature. A selected number of publications include the following articles \[56, 92, 86, 9\] and books \[80, 106, 31, 23\]. If the dynamical system is linear and the constraints are polytopic, then the problem can be cast as a Quadratic Program (QP) or a Linear Program (LP), depending on the objective function. Nonlinear model dynamics or specific constraints can make the problem nonconvex. Thus, the control action of nonlinear Model Predictive Control problems is in general derived by solving a Nonlinear Program (NLP) at every time step. By considering the initialization $x_{k|k} := x_k$ given by the current state, a general nonlinear MPC can be written in the form:

$$\min_{(x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}} \sum_{j=0}^{N-1} \ell(x_{k+j|k}, u_{k+j|k}) + \ell_N(x_{k+N|k})$$

s.t.  
- $x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}) \quad \forall j \in \{0, \ldots, N - 1\}$
- $(x_{k+j|k}, u_{k+j|k}) \in (\mathcal{X} \times \mathcal{U}) \quad \forall j \in \{0, \ldots, N - 1\}$
- $x_{k+N|k} \in \mathcal{X}_N \quad \forall j \in \{0, \ldots, N - 1\},$

(1.1)

where the function $f$ represents the discretization of some dynamics, $\ell$ and $\ell_N$ are respectively a stage-cost and a terminal cost, and $\mathcal{X}, \mathcal{U}, \mathcal{X}_N$ are sets specifying some desired constraints. These problems are typically recast as a general NLP, obtaining the formulation:

$$\min_z J(z)$$

s.t.  
- $g(z) \leq 0$
- $h(z) = 0,$

(1.2)

with the functions $J, g$ and $h$ assumed to be twice continuously differentiable. We are interested in determining a critical point $z^*$, i.e.,
1 Introduction

a point that is feasible for problem (1.2) and that satisfies first-order optimality conditions. A critical point typically corresponds to a local minimum, or a saddle-point or even a local maximum of (1.2). We refer to [53] for a comprehensive introduction to the Karush-Kuhn-Tucker (KKT) optimality conditions.

The optimization problems cannot be solved analytically. In general, numerical methods generate a sequence of iterates \( (z^{(i)})_{i=1}^{\infty} \) to achieve convergence to a critical point \( z^* \) in the limit. The choice of the specific method clearly depends on the structure of the constraints.

In this introduction, we aim to give a general and far from exhaustive introduction of these methods and place our contributions in this framework. For the reader’s ease of understanding, we will focus on the concepts, rather than on the mathematical detail, and leave a rigorous explanation to the next chapters.

Historically, one of the earliest approaches to tackle such problems has been the Gradient (or Steepest Descent) Method [111, 120, 96]. Given a current iterate \( z^{(i)} \) which is feasible for Problem (1.2), we consider its derivative \( \nabla J(z^{(i)}) \), and determine the next iterate \( z^{(i+1)} := z^{(i)} + d^{(i)}_z \) as follows:

\[
d^{(i)}_z := \Pi \{ d_z \mid g(z^{(i)}+d_z) \leq 0, h(z^{(i)}+d_z) = 0 \} \left( -\alpha^{(i)} \nabla J(z^{(i)}) \right), \tag{1.3}
\]

where \( \Pi \) is the projection operator and \( \alpha^{(i)} \in \mathbb{R}_{>0} \) is a step size, to be designed in order to obtain a desired decrease of the objective \( J(\cdot) \).

Determining the projection onto the feasible set requires in general the solution of a new optimization problem, whose complexity is comparable to the original problem in (1.2). Therefore, the constraint set is typically required to be easy-to-project, e.g., a box or with other desirable geometrical properties. When the constraint set is given by a general nonlinear manifold, the gradient step in (1.3) is modified in a two step procedure [112]. First, a projection onto the tangent space is determined, then the next feasible point \( z^{(i+1)} \) is recovered in a correction phase, that consists of an iterative interpolation of the nonlinear constraint and thus it may require several steps (Figure 1.1). Therefore, when the constraints of the problem are not easy to project, the Gradient Method is typically no longer considered a suitable optimization method. In fact, each iterate of the Gradient Method needs to be feasible, and this limits its performance.

Instead, two large families of optimization methods consider possi-
Figure 1.1: Gradient method: (a) easy-to-project constraints let one compute the projection inexpensively, e.g., on box constraints it only consists of clipping the variable components; (b) should the constraints be general nonlinear manifolds, first the projection is determined onto the tangent space, then an iterative procedure is needed to determine a feasible point that decreases the objective function.

We start with the analysis of the former. The decrease direction $d_z^{(i)}$ is found as the solution of a Quadratic Program, obtained by approximating the nonlinear problem in (1.2) around the current iterate $z^{(i)}$ with a quadratic objective and linear constraints as follows:

$$d_z^{(i)} := \arg \min_{d_z} \frac{1}{2} d_z^\top H^{(i)} d_z + \nabla J(z^{(i)})^\top d_z$$

s.t.  
$$g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z \leq 0$$
$$h(z^{(i)}) + \nabla h(z^{(i)})^\top d_z = 0,$$

with the Hessian $H^{(i)}$ capturing the local second-order information of the problem [18].

It is clear that, even starting from a feasible iterate $z^{(i)}$, a new iterate $z^{(i+1)} := z^{(i)} + d_z^{(i)}$ will not be feasible for the original constraints. On the other hand, under some assumptions it is possible to prove local convergence to a critical point $z^*$, i.e., if the initial iterate $z^{(0)}$ is sufficiently close to $z^*$.

Convergence to a critical point $z^*$ for any initialization $z^{(0)}$, referred to in this thesis as global convergence, is proved by modifying the update step of the algorithm, i.e., $z^{(i+1)} := z^{(i)} + t^{(i)} d_z^{(i)}$, with $t^{(i)} \in (0, 1]$
a merit function step size to be determined. Then, a merit function is introduced, that weights optimality and feasibility of the iterate $z^{(i)}$, and with specific theoretical guarantees on its critical points, i.e., an equivalence with the critical points of the original nonlinear problem. By ensuring that the increment $d_{z}^{(i)}$ is a decrease direction for the merit function, it is possible to prove global convergence to a critical point, $z^*$, of (1.2).

In this case, there is no inexpensive way to solve the QP problem in (1.4). Thus, several minor steps are required to determine the solution to each QP. This limits the computational performance of the method, and has progressively opened the way to suitable approximations or to Interior-Point Methods.

A good approximation to a desired critical point, $z^*$, for control purposes is obtained in an SQP formulation by limiting the maximum number of iterations $i$ to a small value, e.g., to be at most 1, in what is commonly referred to as the Real-Time Iteration (RTI) \cite{45, 46, 48}. This is driven by control-based motivations: since the Model Predictive Control problem is solved at every time step with small sampling time, under some technical assumptions the error on the solution of the MPC problem vanishes asymptotically with time. In practice, the closed-loop performance of the RTI will be comparable to the MPC employing full NLP solvers. Additional details about the RTI can be found in Appendix B.

Interior-point methods for nonlinear programming have been intensively studied over the last decades. Here we briefly mention the idea underlying the barrier Interior-Point Method and refer to \cite{13, 12, 133, 29, 55} and the references therein for further details.

In the barrier Interior-Point Method, a parameter sequence $(\mu^{(i)})_i$ is introduced, with $\lim_{i \to \infty} \mu^{(i)} = 0$. Given a fixed $\mu^{(i)}$, a solution to the following problem is determined:

$$z(\mu^{(i)}) := \min_{z} \quad J(z) - \mu^{(i)} \sum_{j=1}^{m} \ln \left( -g_{j}(z) \right)$$

s.t. $h(z) = 0$, $\quad (1.5)$

where the second term of the objective is called the barrier function. Typically logarithmic functions are employed, since they are convex and go to infinity as the argument approaches 0.

Under some regularity assumptions, if we solve a sequence of problems (1.5) with $\mu^{(i)} \to 0$, then we obtain local convergence to a local
1.1 Contribution of this thesis

In this thesis we introduce a projected gradient and constraint linearization approach that can be seen as a modification of the Gradient Method in (1.3) as follows:

\[ d_z^{(i)} := \Pi_{C^{(i)}} \left( -\alpha^{(i)} \nabla J(z^{(i)}) \right), \]

with the set \( C^{(i)} \) defined as

\[ C^{(i)} := \{ d_z \in \mathbb{R}^n \mid g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z \leq 0, h(z^{(i)}) + \nabla h(z^{(i)})^\top d_z = 0 \}. \]

The gradient step is projected onto a linearization of the constraint, rather than on the nonlinear manifold. Then, the next iterate is computed as \( z^{(i+1)} := z^{(i)} + d_z^{(i)} \). Note that the set \( C^{(i)} \) in (1.7) coincides with the feasible set in (1.4). Thus, each \( z^{(i)} \) need not be feasible for the original problem (1.2). On the other hand, since the projection at each iterate is computed onto an affine subspace, its computation is in general easier, compared to (1.3).

As a result, the proposed method does not fall either into the projected Gradient Method approaches, because the projection is not performed onto the original nonlinear manifold, or into the standard SQP, since second-order information is not used.

This thesis has three main contributions. It develops the theory for the proposed method, guaranteeing its local and global convergence and assessing its rate. Then, it focuses specifically on nonlinear Model Predictive Control and presents how the proposed method can solve
its optimization problem efficiently and with low complexity. Finally, we demonstrate our theoretical results on several control applications and compare the computational performance with commercial software implementing SQP and IPM algorithms.

We now outline the content of the remaining chapters of this thesis.

1.1.1 Theoretical contribution

The projected gradient and linearization approach in (1.6), combined with the update rule \( z^{(i+1)} := z^{(i)} + d_z^{(i)} \) is proved to be locally convergent to a local minimum \( z^* \) when \( \alpha^{(i)} \) is taken small enough. We are actually able to quantify how small \( \alpha^{(i)} \) needs to be, based on second-order information of the problem at the local minimum \( z^* \). The convergence rate is linear, since only first-order information is used by the method. The proposed result is derived by using the standard assumptions and arguments of SQP.

In the practical application of the algorithm, such a bound on \( \alpha^{(i)} \) is not known a priori – as it requires knowledge of the local minimum \( z^* \) – and even computing an approximation based on the current iterate \( z^{(i)} \) would be overly expensive. In fact, the first-order method that we propose does not require such quantity anywhere else, except for computing the step size \( \alpha^{(i)} \). Further, we cannot guarantee that an initial iterate \( z^{(0)} \) be close enough to the critical point \( z^* \), thus a globally convergent approach is needed.

The local convergence results suggest that when the initial iterate is far away from a critical point \( z^* \), a smaller update step has to be taken, i.e., \( z^{(i+1)} := z^{(i)} + t^{(i)} d_z^{(i)} \) with step size \( t^{(i)} \in (0, 1] \). The value of \( t^{(i)} \) is determined via a merit function treatment, similarly to SQP. By ensuring that \( d_z^{(i)} \) is a decrease direction for the merit function, we prove global convergence to a critical point.

We show two different global convergence results, respectively considering two possible merit functions: an augmented Lagrangian and a non-smooth exact penalty function. The former is obtained by adding a penalty term to the Lagrangian of the problem, consisting of a penalty parameter and the square of the constraints violation. Therefore, it is continuously differentiable and it requires an estimate of the dual variables. In the latter the penalty parameter multiplies an \( \ell_p \) norm of the constraints violation, with \( p \geq 1 \), and hence it is not differentiable. The non-smooth exact penalty function has the computational advantage of not requiring any estimate of the dual variables and, under
some technical assumptions, is such that the penalty parameter stays bounded.

Chapter 2 In this chapter the proposed algorithm is presented in more detail and compared with SQP. We also present its local and global convergence properties, using an augmented Lagrangian as a merit function.

Chapter 3 The algorithm presented in Chapter 2 is modified to consider a non-smooth exact penalty function as a merit function. After showing the practical implications for the algorithm, we prove global convergence.

1.1.2 Practical implementation

Model Predictive Control problems naturally introduce sparsity in the NLP, because of the causality of the nonlinear dynamics and the stage-wise nature of the constraints, that can be actively exploited by the method. As in standard Gradient Method, the dynamics are embedded in the objective function.

Then, we need to make the projection in (1.6) easy to compute. Clearly, this cannot happen for any possible constraint set in (1.1). As in Gradient Method, the state constraints $x \in \mathcal{X}$ make the feasible set no longer easy-to-project.

The approach that we follow considers squared-slack variables. Under some assumptions, it is well known that the NLP in (1.2) is equivalent to the following:

$$\begin{align*}
\min_{z,y} & \quad J(z) \\
\text{s.t.} & \quad g(z) + \frac{1}{2} \text{diag}(y) y = 0 \\
& \quad h(z) = 0.
\end{align*} \tag{1.8}$$

This formulation presents a theoretical drawback that it may introduce new KKT points in the problem, which did not exist in the original problem in (1.2). However, we have the possibility to easily check this a-posteriori, namely by the sign of the dual variable. Convergence to undesired new KKT points can be avoided in practice by an appropriate initialization of the slack variables (more on this later).

Since the problem has only equality constraints, the projection in (1.6) is on an affine subspace, thus it can be written in closed-form by computing a matrix inversion. Instead of performing numerically
the inversion, which for dense matrices would have a complexity cubic on the matrix dimension, we propose to invert the matrix analytically. For a nonlinear MPC problem, this reduces the complexity of computing the projection (and the overall complexity of the algorithm) to $O(N(n_x^2 + n_x n_u))$ Floating-Point Operations (FLOPS), with $N$ being the prediction horizon and $n_x$ and $n_u$ respectively the state and input dimensions. This complexity is better than what achieved by SQP and IPM for the same problem, but also by QP methods such as the active set and the accelerated Gradient Method for the RTI approximation.

The constraint sets that allow us to analytically compute the matrix inversion need to be specifically sparse: besides the standard easy-to-project constraints, such as box input-constraints, we can efficiently include low-dimensional nonlinear input constraints and stability constraints (e.g., terminal constraints). The low-dimensional specification is required to be able to compute the matrix inversion analytically.

We provide the tool FalcOpt (First-order Algorithm via Linearization of Constraints for OPTimization) that automatically generates efficient C code to solve nonlinear MPC problems by using the algorithms introduced in Chapters 2 and 3.

Chapter 4 The NLP reformulation in (1.8) is presented in this chapter. We present the class of MPC problems that can be handled by the tool FalcOpt and show the computational cost of each iteration.

After having introduced the algorithm, the natural question that we raise is whether we can accelerate the convergence of the proposed method by extending the gradient step size. In particular, we propose to modify the gradient step in (1.3) as follows:

$$
d_z^{(i)} := \Pi_{C^{(i)}} \left( -\alpha^{(i)} \left( H^{(i)} d_{z,ini}^{(i)} + \nabla J(z^{(i)}) \right) \right),$$  

(1.9)

with $C^{(i)}$ defined as in (1.7), the Hessian $H^{(i)}$ as in (1.4) and $d_{z,ini}^{(i)}$ being either the previous iterate $d_z^{(i-1)}$, or 0 based on a specific test.

Chapter 5 In this chapter we investigate the approach in (1.9) and, based on a heuristic, determine a test for the variable $d_{z,ini}^{(i)}$, specifically tailored for a particular class of nonlinear MPC problems. This gives rise to a slightly different version of the algorithm for which we observe very fast convergence in practical problems.
1.1 Contribution of this thesis

1.1.3 Control applications

The proposed algorithm is applied to several case studies. The main application considered in this thesis are industrial compression systems, typically employed in gas plants and chemical industry to provide gas in pressure. We also provide some examples in power electronics, where the computational requirements are very stringent, typically in the order of hundreds of microseconds.

The presented algorithm is particularly effective when considering a stability constraint, e.g., in contractive MPC [41]. This control scheme ensures closed-loop stability by adding a Lyapunov-based constraint to the MPC problem.

Following this line, in this thesis we develop a numerical method to ensure recursive feasibility of such stability constraint, hence of the contractive MPC problems. Assuming that the dynamics $f$ in (1.1) are too complicated to determine analytically a control Lyapunov function (CLF) in a contractive set, we then impose a quadratic Lyapunov function $V(x) := x^T P x$ with a $P > 0$, and check numerically that it is a CLF in a grid of the state space [15, 16]. Then, we extend this to the neighboring points - via an additional test, exploiting Lipschitz continuity. The points that satisfy these tests form a contractive set, where the stability constraint is always feasible. We show an example of this method on the control of the compression system and report on the computational performance of the proposed algorithm.

Another control tool that is particularly useful for the control of compression systems is the offset-free MPC [100, 81, 93]. This allows reference tracking also in presence of disturbance or model mismatch. Since the dimension of the desired reference to be tracked matches the number of manipulated inputs, linear and nonlinear offset-free MPC schemes allow the compression system to be robust to several possible disturbances. The heuristic proposed in Chapter 5 solves the underlying optimization problem efficiently. The remaining chapters are organized as follows:

**Chapter 6** The control problem of the compression system is presented here, including model, goal and state-of-art-control. An introduction of the relevant literature on the control of compression system is included, which recently turned into MPC based approaches both in academic and industrial research.

**Chapter 7** In this chapter we review the contractive MPC scheme and
1 Introduction

present the numerical method to ensure recursive feasibility of the contractive constraint, together with a numerical example.

Chapter 8 The offset-free MPC scheme for linear and nonlinear MPC is reviewed here. We present how to cast it for the specific compressor example and then show extensive simulation results. The computational performance of the proposed algorithm is presented and compared to the other commercial solvers.

Chapter 9 Further control applications are presented here, such as power electronics examples (a converter and an induction motor) and the control of an inverted pendulum.

1.2 Publications

The work in this thesis is based on the following publications.

Chapter 2: A projected gradient and constraint linearization method for nonlinear programs

This chapter is based on the following work:


Chapter 3: A non-smooth exact penalty function reformulation

This chapter is based on the following work:


Chapter 4: Practical implementation for Model Predictive Control problems

This chapter is based on sections of the papers referenced in Chapters 2 and 3 and on the following software:
1.2 Publications


Chapter 5: A heuristic to accelerate the practical convergence speed

This chapter is based on:


Chapter 6: Compression systems for industrial applications

Chapter 7: Fast contractive MPC with stability and recursive feasibility guarantees

This chapter is based on sections of the following paper:


Chapter 8: Offset-free MPC for the compression system

This chapter is based on sections of the papers mentioned in Chapter 7. Preliminary results can be found in:

Chapter 9: Further applications

The inverted pendulum example is contained in the paper referenced in Chapter 2. The power electronics applications are currently unpublished.

Other publications

The following publications were published during the Doctorate but are not featured in this thesis:


Part I

Theory
2 A projected gradient and constraint linearization method for nonlinear programs

In this chapter we present in detail the proposed projected gradient and constraint linearization algorithm, also analyzing its similarities with SQP. The convergence of the algorithm is proved in Section 2.4; in particular, the local convergence results are contained in Theorem 2.1. The global convergence properties are instead presented in Theorem 2.3. Next, we show that in the final iterates of the algorithm, linear convergence rate is obtained asymptotically.

2.1 Literature review

The projected Gradient Method is an established approach for solving convex optimization problems. The subject has been extensively investigated over the last decades, developing algorithms that guarantee best performance for convex and strongly convex problems, see [96, 12]. Recently, Nesterov’s accelerated Gradient Method has been applied to linear Model Predictive Control, and a priori worst-case bounds for finding a solution with prespecified accuracy have been derived [95, 108]. When the optimization problem is a general nonlinear program, the Gradient Method can still be used for finding a critical point [112]. As shown in Chapter 1 for general nonconvex constraints in the form of a nonlinear manifold, the projection onto the feasible set is performed in two stages. First, the projection is derived onto the tangent space to the nonlinear manifold, which in general is a polyhedron. Then, the determined point is projected again onto the original nonlinear manifold, via some strategy guaranteed to determine a feasible point that improves the objective function. While ensuring convergence, this second projection is in general computationally expensive, hence the method is not recommended in practice for solving nonlinear MPC problems.

In this chapter we analyze a Gradient Method for Nonlinear Programs that only requires projections onto the tangent space, obtained
by linearization of the nonlinear manifold around the current iterate. Note that this determines a sequence of points that are not necessarily feasible for the original NLP. Thus, standard projected Gradient Method results do not apply to prove convergence.

Linearized constraints are instead considered in Sequential Quadratic Programming, which is an established method to determine a local solution to a smooth nonconvex NLP. The solution is determined via a sequence of iterates, each obtained as the solution to a Quadratic Program, that are usually called the major iterations. In turn, each QP is solved via so-called minor iterations using available convex optimization methods [99, 110, 18]. Typically, each QP has as objective function a second-order approximation of the Lagrangian function of the nonlinear problem and as constraints the linearization of the nonlinear manifold, both computed at the current iterate. The solution of the QP updates the current iterate and then the next QP is formulated.

The basic SQP method is equivalent to Newton’s method applied to the first-order optimality conditions of the original nonlinear optimization problem, thus it is locally quadratically convergent [116]. Since the required Hessian of the Lagrangian is expensive to compute and it is not guaranteed to be positive definite on every subspace far from the solution, a suitable approximation of the Hessian of the Lagrangian is typically used, e.g., quasi-Newton or Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates, that guarantee local superlinear convergence [26, 19, 34, 97]. Global convergence is usually obtained via a line search approach. Merit functions are considered that comprise both the objective and the constraint functions, e.g., in the form of an augmented Lagrangian [54, 60, 102]. Then by appropriate tuning of some penalty parameters, the solution to the QP is proved to be a descent direction for the merit function. A line search then determines a merit function step size for the convergence of the method. Several contributions in the literature have discussed different reformulations that trade off theoretical convergence guarantees and computational complexity, see the brief review of Sequential Quadratic Programming in Appendix [18, 104, 68, 103, 60]. Indeed, commercial numerical solvers use this technique for approaching a critical point of an NLP [59]. An alternative approach for establishing global convergence is the trust-region method, where additional constraints are included in the optimization program [18, 44, 119].

The proposed gradient algorithm in this chapter can be seen as an incomplete SQP where, instead of solving each generated QP, only one
gradient step is computed for the QP and then projected onto the linearized constraint. Some literature has proposed solving the QP program inexactly, e.g., by bounding the suboptimality of the estimated solution to the QP to recover some rate of convergence for the SQP [42, 76, 94]. However, our approach does not fall into this class of methods, because we cannot guarantee to satisfy the tolerated level of suboptimality by computing only one gradient step for each QP. Moreover, for these approaches second-order information on the Lagrangian is necessary, which in contrast is not required for our proposed algorithm.

In this chapter we show local and global convergence properties of the proposed gradient algorithm, leveraging established SQP results in the literature. Local conditions are derived when each gradient step is directly employed to update the current iterate. As in standard Gradient Method, ensuring convergence of the algorithm requires that some conditions have to be set on the gradient step size, typically depending on second-order information of the considered problem – in our case, the Lipschitz constant of the gradient of the Lagrangian function. Under a particular assumption on the Hessian, the algorithm converges with linear rate, as expected for first-order methods for NLPs [57]. This is guaranteed to work close to a local optimum only. For the practical use of the algorithm, global convergence is required instead. Since updating the current iterate with this gradient step might not guarantee convergence, a variable merit function step size is considered. Analogously to SQP, a merit function in the form of an augmented Lagrangian function weights the optimality and infeasibility of the iterates and is employed in the line search for determining the merit function step size.

### 2.2 Statement of the nonlinear optimization problem

We consider the nonlinear optimization problem (NLP)

\[
\min_{z \in \mathbb{R}^n} J(z)
\]

\[
\text{s.t. } g(z) \leq 0, \\
\quad h(z) = 0,
\]

(2.1)

where the functions \( J : \mathbb{R}^n \to \mathbb{R} \), \( g : \mathbb{R}^n \to \mathbb{R}^m \) and \( h : \mathbb{R}^n \to \mathbb{R}^p \) are twice continuously differentiable functions, possibly nonconvex.

Let us define the Lagrangian function of the NLP in (2.1) as

\[
\mathcal{L}(z, \lambda, \nu) := J(z) + g(z)^{\top} \lambda + h(z)^{\top} \nu,
\]

(2.2)
with Lagrange multiplier vector \( \lambda \in \mathbb{R}^m_{\geq 0} \) and \( \nu \in \mathbb{R}^p \).

We call \( z^* \) a critical point of (2.1) if it satisfies the first-order Karush-Kuhn-Tucker (KKT) optimality conditions with strict complementarity, i.e., there exist \( \lambda^* \in \mathbb{R}^m_{\geq 0} \) and \( \nu^* \in \mathbb{R}^p \) such that

\[
\begin{align*}
\nabla L(z^*, \lambda^*, \nu^*) &= \nabla J(z^*) + \nabla g(z^*) \lambda^* + \nabla h(z^*) \nu^* = 0 \\
\text{diag}(\lambda^*) g(z^*) &= 0 \\
\lambda^*_j > 0 & \text{ if } g_j(z^*) = 0 \\
g(z^*) &\leq 0 \\
h(z^*) &= 0.
\end{align*}
\]

The strict complementarity assumption is required in order to obtain the continuity properties of the KKT triple in Section 2.4.1.

Let us assume that the NLP in (2.1) has a finite number of critical points, where additionally the second-order sufficiency conditions hold:

\[
\tilde{z}^T \nabla^2 L(z^*, \lambda^*, \nu^*) \tilde{z} > 0
\]

for all \( \tilde{z} \neq 0 \) such that

\[
\begin{align*}
\nabla g_j(z^*)^T \tilde{z} &= 0 \forall j \in \{1, \ldots, m\} \text{ s.t. } g_j(z^*) = 0 \\
\nabla h(z^*)^T \tilde{z} &= 0.
\end{align*}
\]

Iterative methods generate a sequence \( (z^{(i)})_{i=1}^\infty \) to determine a critical point \( z^* \). In Section 2.3, we present the proposed algorithm to determine a critical point \( z^* \). At the \( i \)-th major iteration, \( z^{(i+1)} \) is directly derived via a projected gradient step onto a linearization of the constraint around the current iterate \( z^{(i)} \).

### 2.3 The projected gradient and constraint linearization algorithm

We propose determining \( d_z^{(i)} \) through one projected gradient step onto the linearization of the constraints around \( z^{(i)} \). This is formalized by

\[
d_z^{(i)} := \Pi_{C^{(i)}} \left( -\alpha^{(i)} \nabla J(z^{(i)}) \right),
\]

with bounded gradient step size \( \alpha^{(i)} \in \mathbb{R}_{>0} \), and \( \Pi_{C^{(i)}} (\cdot) : \mathbb{R}^n \to C^{(i)} \subseteq \mathbb{R}^n \) being the Euclidean projection onto the set

\[
C^{(i)} := \{ d_z \in \mathbb{R}^n | g(z^{(i)}) + \nabla g(z^{(i)})^T d_z \leq 0, h(z^{(i)}) + \nabla h(z^{(i)})^T d_z = 0 \}.
\]
The gradient step size \( \alpha^{(i)} \) influences the observed convergence speed of the algorithm. Here we only assume \( \alpha^{(i)} \) to be bounded to prove the convergence properties of the method and we leave to Section 4.3.1 the practical discussion on how to set this parameter.

Note that the algorithm step in (2.4) is equivalent to computing only one gradient step of the QP in (1.4) with null initialization \( d_{z,ini}^{(i)} \):

\[
d_z^{(i)} = \Pi_{C^{(i)}} \begin{pmatrix} d_{z,ini}^{(i)} - \alpha^{(i)} \left( H^{(i)} d_{z,ini}^{(i)} + \nabla J(z^{(i)}) \right) \right) = \Pi_{C^{(i)}} \left( -\alpha^{(i)} \nabla J(z^{(i)}) \right).
\]  

(2.6)

This allows us to use some of the results of Sequential Quadratic Programming to characterize the convergence properties of the algorithm.

Based on the solution \( d_z^{(i)} \) in (2.4), the sequence is updated as

\[
z^{(i+1)} := z^{(i)} + t^{(i)} d_z^{(i)},
\]

(2.7)

where \( t^{(i)} \in (0, 1] \) is a merit function step size to be determined.

Note that the projection in (2.4) is equivalent to solving the following optimization problem:

\[
d_z^{(i)} = \arg \min_{d_z \in \mathbb{R}^n} \frac{1}{2\alpha^{(i)}} \left\| d_z + \alpha^{(i)} \nabla J(z^{(i)}) \right\|^2_2
\]

\[
\text{s.t. } g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z \leq 0 \\
  h(z^{(i)}) + \nabla h(z^{(i)})^\top d_z = 0.
\]

(2.8)

Under regularity assumptions (see Section 2.4) there exist dual multipliers \( \lambda_G^{(i)} \in \mathbb{R}_+^m, \nu_G^{(i)} \in \mathbb{R}_+^p \) such that

\[
\frac{1}{\alpha^{(i)}} d_z^{(i)} + \nabla J(z^{(i)}) + \nabla g(z^{(i)}) \lambda_G^{(i)} + \nabla h(z^{(i)}) \nu_G^{(i)} = 0
\]

(2.9a)

\[
\text{diag}(\lambda_G^{(i)}) \left( g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z^{(i)} \right) = 0
\]

(2.9b)

\[
g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z^{(i)} \leq 0
\]

(2.9c)

\[
h(z^{(i)}) + \nabla h(z^{(i)})^\top d_z^{(i)} = 0,
\]

(2.9d)

which are then used to define the dual variables increments:

\[
d_{\lambda}^{(i)} := \lambda_G^{(i)} - \lambda^{(i)} \\
d_{\nu}^{(i)} := \nu_G^{(i)} - \nu^{(i)}.
\]

(2.10)
To ensure global convergence to a critical point, a merit function in the form of an augmented Lagrangian is considered:

\[
L_{\text{aug}}(z, \lambda, \nu, s, \rho) := J(z) + (g(z) + s)^\top \lambda + h(z)^\top \nu + \frac{\rho}{2} \|g(z) + s\|^2 + \frac{\rho}{2} \|h(z)\|^2,
\]

(2.11)

where \(\rho \in \mathbb{R}_{\geq 0}\) is a penalty parameter to be determined and \(s \in \mathbb{R}^m_{\geq 0}\) is a vector of slack variables, defined at the beginning of each iteration \(i\) such that its \(j\)-th component satisfies the following equation [60, Equation (2.8)]:

\[
s_j^{(i)} := \begin{cases} 
\max \left( -g_j(z^{(i)}), 0 \right) & \text{if } \rho = 0 \\
\max \left( -g_j(z^{(i)}) - \frac{\lambda_j}{\rho}, 0 \right) & \text{otherwise}
\end{cases}
\]

(2.12)

When the parameter \(\rho\) is nonzero, the vector \(s\) as defined in (2.12) yields the value of \(L_{\text{aug}}\) minimized with respect to the slack variables alone, i.e., \(\partial L_{\text{aug}}/\partial s = 0\), subject to the non-negativity constraint \(s \in \mathbb{R}^m_{\geq 0}\).

The dual multipliers \(\lambda\) and \(\nu\) are considered as additional variables, updated with merit function step size \(t^{(i)}\) along with the primal sequence \((z^{(i)})_{i}\).

Then we define the function \(\phi : \mathbb{R} \to \mathbb{R}\) as

\[
\phi(t) := L_{\text{aug}}(z + td_z, \lambda + td_\lambda, \nu + td_\nu, s + td_s, \rho)
\]

(2.13)

to determine the merit function step size \(t\), e.g., via a backtracking line search starting from \(t = 1\), that satisfies the Wolfe conditions [44, 98]:

\[
\phi(t) - \phi(0) \leq \sigma_1 t \phi'(0)
\]

(2.14a)

\[
|\phi'(t)| \leq -\sigma_2 \phi'(0) \quad \text{or} \quad (t = 1 \text{ and } \phi'(1) \leq -\sigma_2 \phi'(0))
\]

(2.14b)

for some \(0 < \sigma_1 \leq \sigma_2 < \frac{1}{2}\). For ease of notation, we avoid making explicit the dependence of \(\phi\) on the arguments \(z, \lambda, \nu, s\) of the augmented Lagrangian function \(L_{\text{aug}}\).

Note that if the derivative \(\phi'(0)\) is negative, then there exists a merit function step size \(t^{(i)} \in (0, 1]\) such that the conditions in (2.14) hold. The condition on the derivative \(\phi'(0)\) is checked numerically at every iteration \(i \in \mathbb{N}\) via the inequality condition

\[
\phi'(0) \leq -\frac{1}{2\alpha^{(i)}(i)} \|d_z^{(i)}\|^2.
\]

(2.15)
2.4 Proof of convergence of the proposed algorithm

If this latter inequality does not hold true, then the parameter $\rho$ is adjusted. Later, in Lemma 2.4 we provide a lower bound $\hat{\rho} \in \mathbb{R}_{\geq 0}$ such that (2.15) holds for all $\rho \geq \hat{\rho}$.

For the practical implementation of the algorithm, the line search in (2.14) is typically simplified in order to check only the first condition in (2.14a) [43, 99]. This has the effect of reducing the computational burden required to compute the derivative $\phi'(t)$ and it does not impede convergence of the algorithm in practice. To derive the merit function step size $t$, a backtracking line search is employed with safeguarded polynomial interpolation [77].

We consider the iterative update

$$
\begin{bmatrix}
z^{(i+1)} \\
\lambda^{(i+1)} \\
\nu^{(i+1)} \\
s^{(i+1)}
\end{bmatrix}
:=
\begin{bmatrix}
z^{(i)} \\
\lambda^{(i)} \\
\nu^{(i)} \\
s^{(i)}
\end{bmatrix}
+ t^{(i)}
\begin{bmatrix}
d_z^{(i)} \\
d_\lambda^{(i)} \\
d_\nu^{(i)} \\
d_s^{(i)}
\end{bmatrix},
$$

where $d_z^{(i)}$ is from (2.4), $d_\lambda^{(i)} := \lambda^{(i)}_{QP} - \lambda^{(i)}$, $d_\nu^{(i)} := \nu^{(i)}_{QP} - \nu^{(i)}$ and the slack variation $d_s^{(i)}$ satisfies

$$g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z^{(i)} + s^{(i)} + d_s^{(i)} = 0. \quad (2.17)$$

Our proposed approach is summarized in Algorithm 2.1.

### 2.4 Proof of convergence of the proposed algorithm

In this section, we show the convergence properties of the proposed approach in Algorithm 2.1. We introduced the following assumptions, which are typically considered in the SQP method [60, Assumptions (i)-(iii)].

**Assumption 2.1.** For all $i \in \mathbb{N}$, the QP in (2.8) is feasible.

This assumption is typically satisfied in the practical problems that we aim to solve and it allows us to simplify the mathematical treatment (see Chapter 4). If the assumption does not hold true, then standard SQP arguments can be used to derive an iterate $d_z$ that solves a relaxed problem [13].

**Assumption 2.2.** There exists a compact set $\Omega \subset \mathbb{R}^n$, such that for all $i \in \mathbb{N}$, $z^{(i)}$, $z^{(i)} + d_z^{(i)} \in \Omega$. 

21
Algorithm 2.1  Projected gradient and constraint linearization method

\textbf{Initialize} \( i \leftarrow 0 \) and \( z(0) \in \mathbb{R}^n \) and \( \rho(0) = 0 \)

\textbf{repeat}

\hspace{1em} Compute \( d_z(i) \) with gradient step size \( \alpha(i) \) in (2.4)

\hspace{1em} Determine \( \lambda^{(i)}_G, \nu^{(i)}_G \) such that (2.9) holds

\hspace{1em} \textbf{if} \( d_z(i) = 0 \) then

\hspace{2em} Set \( z^* = z(i), \lambda^* = \lambda^{(i)}_G, \nu^* = \nu^{(i)}_G \) and \textbf{Stop}

\hspace{1em} \textbf{else}

\hspace{2em} \textbf{if} \( i = 0 \) then

\hspace{3em} Set \( \lambda^{(0)} = \lambda^{(0)}_G, \nu^{(0)} = \nu^{(0)}_G \)

\hspace{2em} \textbf{end if}

\hspace{2em} Set \( d^{(i)}_\lambda = \lambda^{(i)}_G - \lambda^{(i)}, d^{(i)}_\nu = \nu^{(i)}_G - \nu^{(i)} \)

\hspace{1em} \textbf{end if}

\hspace{1em} Determine \( s(i) \) and \( d_s(i) \) from (2.12) and (2.17)

\hspace{1em} Set \( \rho(i) \geq 0 \) such that (2.15) holds

\hspace{1em} Determine the merit function step size \( t(i) \) that satisfies (2.14), e.g., via line search

\hspace{1em} Update \( z(i+1), \lambda(i+1), \nu(i+1), s(i+1) \) in (2.16)

\hspace{1em} \( i \leftarrow i + 1 \)

\hspace{1em} \textbf{until} Convergence

\textbf{return} \( z^*, \lambda^* \) and \( \nu^* \)

This is a standard assumption in Sequential Quadratic Programming methods. It guarantees a number of properties, such as the existence of at least an accumulation point of the sequence \( (z(i))_i \). The next two assumptions consider boundedness of the objective and constraint functions in \( \Omega \) and constraint qualification of (2.8).

\textbf{Assumption 2.3.} The functions \( J, g, h \), and their first and second derivatives are uniformly bounded in norm in \( \Omega \).

\textbf{Assumption 2.4.} For all \( i \in \mathbb{N} \), and \( z^{(i)} \), let \( \mathcal{I}_G(d_z^{(i)}, z^{(i)}) \) denote the
2.4 Proof of convergence of the proposed algorithm

index set of the active constraints in (2.8), i.e.,

\[ I_G(d_z^{(i)}, z^{(i)}) = \left\{ j \in \{1, \ldots, m\} \mid g_j(z^{(i)}) + \nabla g_j(z^{(i)})^\top d_z^{(i)} = 0 \right\}. \]  

(2.18)

Then \( d_z^{(i)} \) is regular, i.e., the matrix made up of \( \nabla h(z^{(i)}) \) along with the columns \( \nabla g_j(z^{(i)}) \) for \( j \in I_G(d_z^{(i)}, z^{(i)}) \), has full column rank. Furthermore, strict complementarity holds.

Note that Assumption 2.4 implies that the dual variables \( (\lambda_G^{(i)}, \nu_G^{(i)}) \) in (2.9) are bounded and unique.

This section is organized as follows: in Section 2.4.1, we derive conditions for local linear convergence (setting \( t = 1 \) in (2.14)) under additional assumptions on the Hessian at the critical point and on the gradient step size \( \alpha \). These conditions are not required to prove the global convergence of the algorithm in Section 2.4.2, albeit for \( t \leq 1 \) the convergence can be theoretically slower than linear. We finally show that the local linear convergence rate is not impeded close to the solution by the line search, i.e., \( t = 1 \) is admissible by the line search in Section 2.4.3.

2.4.1 Local convergence

According to [40, 110], we define the general recursive algorithm as a method to determine a critical point for the NLP in (2.1) via intermediate iterates \( w^{(i)} := (z^{(i)}, \lambda^{(i)}, \nu^{(i)}) \), whose update \( w^{(i+1)} \) is determined as the KKT triple of a specific optimization problem. Given the following generic optimization problem \( P(w^{(i)}) \):

\[
\begin{align*}
  z^{(i+1)} &= \arg \min_z J(z, w^{(i)}) \\
  \text{s.t. } g(z, w^{(i)}) &\leq 0 \\
  \quad h(z, w^{(i)}) &= 0,
\end{align*}
\]

(2.19)

the updates \( \lambda^{(i+1)} \) and \( \nu^{(i+1)} \) are the dual variables associated with the KKT conditions for \( P(w^{(i)}) \). As in [110], and in line with the original NLP in (2.1), we assume that the functions \( J, g \) and \( h \) are twice continuously differentiable in their first argument. Let us define
2 A projected gradient and constraint linearization method for nonlinear programs

a KKT triple as $w := (z, \lambda, \nu)$ and the function

$$U(w, w^{(i)}) := \left[ \nabla_w J(z, w^{(i)}) + \lambda^T \nabla_w g(z, w^{(i)}) + \nu^T \nabla_w \hat{h}(z, w^{(i)}); \lambda_1 g_1(z, w^{(i)}); \ldots; \lambda_m g_m(z, w^{(i)}); \hat{h}_1(z, w^{(i)}); \ldots; \hat{h}_p(z, w^{(i)}) \right].$$ (2.20)

The Sequential Quadratic Programming methods and the proposed algorithm can be recast as general recursive algorithms of the form in (2.19). Let us first consider the SQP in (1.4). A local version of this algorithm, which is not guaranteed to converge for any initialization $z^{(0)}$, takes step $t^{(i)} = 1$ for all $i \in \mathbb{N}$.

It follows from (1.4) that the update $w^{(i+1)} = (z^{(i+1)}, \lambda^{(i+1)}, \nu^{(i+1)})$ associated with (2.19) is the KKT triple of the problem $P_{QP}(w^{(i)})$:

$$z^{(i+1)} = \arg \min_z \frac{1}{2}(z - z^{(i)})^T H^{(i)}(z - z^{(i)}) + \nabla J(z^{(i)})^T(z - z^{(i)})$$

s.t. $g(z^{(i)}) + \nabla g(z^{(i)})^T(z - z^{(i)}) \leq 0$

$$h(z^{(i)}) + \nabla h(z^{(i)})^T(z - z^{(i)}) = 0.$$ (2.21)

In fact, since $t^{(i)} = 1$, by (2.16) and (2.10), the update of the dual variables is given by $(\lambda^{(i+1)}, \nu^{(i+1)}) = (\lambda_{QP}^{(i)}, \nu_{QP}^{(i)})$.

Analogously, if we fix $t^{(i)} = 1$ for all $i \in \mathbb{N}$, it follows from (2.8) that the proposed algorithm determines the update $w^{(i+1)}$ as the KKT triple of the problem $P_G(w^{(i)})$:

$$z^{(i+1)} = \arg \min_z \frac{1}{2\alpha^{(i)}}(z - z^{(i)})^T(z - z^{(i)}) + \nabla J(z^{(i)})^T(z - z^{(i)})$$

s.t. $g(z^{(i)}) + \nabla g(z^{(i)})^T(z - z^{(i)}) \leq 0$

$$h(z^{(i)}) + \nabla h(z^{(i)})^T(z - z^{(i)}) = 0.$$ (2.22)

Again, by (2.16) and (2.10), the update of the dual variables is $(\lambda^{(i+1)}, \nu^{(i+1)}) = (\lambda_G^{(i)}, \nu_G^{(i)})$.

The following result establishes some basic properties of the general recursive algorithm in (2.19) that will be necessary to establish local and global convergence of the proposed algorithm.

**Lemma 2.1** ([110] Theorem 2.1). Let $\bar{w} \in \mathbb{R}^{n+m+p}$, and suppose that $(\bar{z}, \bar{\lambda}, \bar{\nu}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$ is a KKT triple of $P(\bar{w})$ from (2.19), at
which the second-order sufficiency conditions with strict complementarity slackness hold, together with the linear independence of the gradients to the active constraints.

Then, there exist open neighborhoods $W = W(\bar{w})$ and $V = V(\bar{z}, \bar{\lambda}, \bar{\nu})$, and a continuous function $Z : W \to V$, such that: $Z(\bar{w}) = (\bar{z}, \bar{\lambda}, \bar{\nu})$, for all $w \in W$, $Z(w)$ is the unique KKT triple in $V$ of $P(w)$ and the unique zero in $V$ of the function $U(\cdot, w)$ in (2.20). Furthermore, if $Z(w) =: (z(w), \lambda(w), \nu(w))$, then for each $w \in W$, $z(w)$ is a critical point of $P(w)$ at which the second-order sufficiency conditions are satisfied with strict complementarity slackness and linear independence of the gradients to the active constraints.

Specifically, the same inequality constraints active at $z(\bar{w})$ will be active at $z(w)$, which is in accordance with [110, Proof of Theorem 2.1].

As in [18], for the analysis of local convergence we assume that the correct active set at $z^*$ is known. This is justified by Lemma 2.1, since the proposed algorithm will eventually identify the active inequality constraint for (2.1). Therefore, we define as $h_a(z) = [g_a(z); h(z)]$ the set of the active inequality and equality constraints at $z^*$, and indicate with $\xi := [\lambda_a; \nu]$ the corresponding dual variables.

The following result establishes the local convergence properties of the algorithm and apply SQP arguments to establish linear convergence to a critical point. In fact, the problem in (2.22) can be seen as an SQP program with Hessian $\frac{1}{\alpha}I \succ 0$.

**Theorem 2.1.** Assume that $(z^*, \xi^*)$ is a critical point such that $\nabla^2 L(z^*, \xi^*)$ is positive definite, and let the initialization $z^{(0)}$ be close enough to $z^*$. Then, there exist positive gradient step sizes $(\alpha^{(i)})_i$ such that the sequence $(z^{(i)})_i$ defined as in (2.22) converges to $z^*$ with linear rate.

**Proof.** The proof is similar to [19, Proof of Theorem 3.3], albeit the result obtained is different. For ease of notation, we use no superscript for the iteration $i$ and the superscript “+” for the iteration $i + 1$.

Then the proposed algorithm has the following update when $t = 1$:

$$
\xi^+ = \xi + d_\xi = \xi + (\xi_G(z) - \xi) = \xi_G(z) = \\
= (\alpha \nabla h_a(z)^T \nabla h_a(z))^{-1} (h_a(z) - \alpha \nabla h_a(z) \nabla J(z)).
$$
From (2.3), we have the optimal dual variable
\[
\xi^* = - (\nabla h_a(z^*)^\top A \nabla h_a(z^*))^{-1} (\nabla h_a(z^*) A \nabla J(z^*)) ,
\]
where \(A\) is any nonsingular matrix that is positive definite on the null space of \(\nabla h_a(z^*)^\top\). In particular, with \(A = \alpha I\) we obtain
\[
\begin{align*}
\xi^+ - \xi^* &= \xi_G(z) - \xi_G(z^*) = \nabla \xi_G(z^*)(z - z^*) + O \left( \|z - z^*\|_2^2 \right) \\
&= \left( \alpha \nabla h_a(z^*)^\top \nabla h_a(z^*) \right)^{-1} \alpha \nabla h_a(z^*)^\top \left( \frac{1}{\alpha} I - \nabla^2 L(z^*, \xi^*) \right) (z - z^*) \\
&\quad + O \left( \|z - z^*\|_2^2 \right). \\
\end{align*}
\]
(2.23)

From (2.9), we have the update \(d_z = -\alpha \nabla L(z, \xi_G(z))\), therefore
\[
\begin{align*}
z^+ - z^* &= z + d_z - z^* = z - z^* - \alpha (\nabla L(z, \xi_G(z)) - \nabla L(z^*, \xi^*)) \\
&= z - z^* - \alpha \left( \nabla^2 L(z^*, \xi^*)(z - z^*) + \nabla h_a(z^*)(\xi_G(z) - \xi^*) \right) \\
&\quad + O \left( \|z - z^*\|_2^2 \right) \\
&= \alpha \left( \left( \frac{1}{\alpha} I - \nabla^2 L(z^*, \xi^*) \right) (z - z^*) - \nabla h_a(z^*)(\xi_G(z) - \xi^*) \right) \\
&\quad + O \left( \|z - z^*\|_2^2 \right). \\
\end{align*}
\]
(2.24)

Now, by substituting (2.23) into (2.24)
\[
\begin{align*}
z^+ - z^* &= \alpha \left( \left( \frac{1}{\alpha} I - \nabla^2 L(z^*, \xi^*) \right) (z - z^*) - \nabla h_a(z^*) (\alpha \nabla h_a(z^*)^\top. \\
&\quad \nabla h_a(z^*))^{-1} \cdot \alpha \nabla h_a(z^*)^\top \left( \frac{1}{\alpha} I - \nabla^2 L(z^*, \xi^*) \right) (z - z^*) \\
&\quad + O \left( \|z - z^*\|_2^2 \right) \\
&= \alpha T(z^*) \left( \frac{1}{\alpha} I - \nabla^2 L(z^*, \xi^*) \right) (z - z^*) + O \left( \|z - z^*\|_2^2 \right) \\
&= T(z^*) (I - \alpha \nabla^2 L(z^*, \xi^*)) (z - z^*) + O \left( \|z - z^*\|_2^2 \right),
\end{align*}
\]
with \( T(z^*) := I - \nabla h_a(z^*) (\nabla h_a(z^*)^\top \nabla h_a(z^*))^{-1} \nabla h_a(z^*)^\top \) being the orthogonal projector onto the tangent space to the constraints \( h_a(z^*) \) at \( z^* \).

Then by considering the norms of the above quantities we conclude that
\[
\| z^+ - z^* \|_2 \leq \| T(z^*) (I - \alpha \nabla^2 \mathcal{L}(z^*, \xi^*)) (z - z^*) \|_2 + \gamma \| z - z^* \|_2^2
\]
\[
\leq \left( \| I - \alpha \nabla^2 \mathcal{L}(z^*, \xi^*) \|_2 + \gamma \| z - z^* \|_2 \right) \| z - z^* \|_2,
\]
for all sufficiently large iterations and some \( \gamma > 0 \), independent of the iteration. Here we have used the property of the orthogonal projector \( \| T(z^*) v \|_2 \leq \| v \|_2 \) for any vector \( v \). Since \( \nabla^2 \mathcal{L}(z^*, \xi^*) \succ 0 \), by choosing \( \alpha \leq \max_{\text{eig}} \nabla^2 \mathcal{L}(z^*, \xi^*) \), the term \( \| I - \alpha \nabla^2 \mathcal{L}(z^*, \xi^*) \|_2 \) can be made strictly smaller than 1. Thus, for a sufficiently small initialization distance \( \| z^{(0)} - z^* \|_2 \), we have \( \eta < 1 \), and the sequence \( (z^{(i)})_i \) converges at a linear rate due to the contraction mapping theorem.

### 2.4.2 Global convergence

Before showing the convergence result of the algorithm, some technical lemmas are presented that show the properties of the proposed algorithm. In particular, the following lemma ensures that the desired algorithm determines the correct active set at a critical point of (2.1).

**Lemma 2.2.** The following properties hold for Algorithm 2.1:

(i) \( \| d_z^{(i)} \|_2 = 0 \) if and only if \( z^{(i)} \) is a critical point for (2.1);

(ii) there exists \( \bar{\epsilon} \in \mathbb{R}_{>0} \) such that: if \( \| d_z \|_2 \leq \bar{\epsilon} \), then the active set \( \mathcal{I}_G \) in (2.18) of (2.8) coincides with the set of constraints that are active at a critical point \( z^* \) for (2.1).

**Proof.** (i) We first prove that, if \( \| d_z^{(i)} \|_2 = 0 \), then \( z^{(i)} \) is a KKT point for (2.1). Note that \( \| d_z^{(i)} \|_2 = 0 \) implies \( d_z^{(i)} = 0 \), i.e., that there exist \( \lambda^{(i)}_G \) and \( \nu^{(i)}_G \) such that (2.9) holds with \( d_z^{(i)} = 0 \). Therefore, by setting \( z^* = z^{(i)} \), \( \lambda^* = \lambda^{(i)}_G \) and \( \nu^* = \nu^{(i)}_G \), the KKT conditions in (2.3) are satisfied. Strict complementarity follows from Assumption 2.4.
Conversely, if \( z^{(i)} \) is a critical point for (2.1), then (2.3) holds for \( z^* = z^{(i)} \), \( \lambda^* = \lambda_G^{(i)} \) and \( \nu^* = \nu_G^{(i)} \). Now, suppose that the vector \( d_z^{(i)} \), resulting from the projection in (2.4) is nonzero, i.e., there exist \( \lambda_G^{(i)} \in \mathbb{R}^m_{\geq 0} \) and \( \nu_G^{(i)} \in \mathbb{R}^p \) such that (2.9) hold for some \( d_z^{(i)} \neq 0 \), with strict complementarity by Assumption 2.4. On the other hand, because of (2.3), the KKT conditions in (2.9) hold also for \( d_z = 0 \), with dual variables \( \bar{\lambda}_G^{(i)} \in \mathbb{R}^m_{\geq 0} \) and \( \bar{\nu}_G^{(i)} \in \mathbb{R}^p \). This generates a contradiction, because the projection (2.4) onto the convex set \( C^{(i)} \), which is nonempty by Assumption 2.1, is unique.

(ii) This result follows from Lemma 2.1 since the proposed method can be rewritten as a general recursive algorithm in the form of (2.19).

**Lemma 2.3.** For all \( i \in \mathbb{N} \), it holds that

\[
\|\lambda^{(i+1)}\|_2 \leq \max_{k \in [0,i]} \|\lambda_G^{(k)}\|_2, \quad \|\nu^{(i+1)}\|_2 \leq \max_{k \in [0,i]} \|\nu_G^{(k)}\|_2.
\]

In addition, \( \|d^{(i)}\|_2 \) and \( \|d^{(i)}_{\nu}\|_2 \) are uniformly bounded for all \( i \in \mathbb{N} \). □

**Proof.** First note that the dual variables \( \lambda_G^{(i)} \) and \( \nu_G^{(i)} \) defined in (2.9) are bounded in norm, since by Assumption 2.4 the active set is linearly independent and strong duality holds.

Then the proof follows the same argument of [60, Lemma 4.2], where the structure of (2.10) is exploited. We equivalently define the dual variable for the inequality constraints as

\[
\lambda^{(0)} := \lambda_G^{(0)} \quad \lambda^{(i+1)} := \lambda^{(i)} + t^{(i)}(\lambda_G^{(i)} - \lambda^{(i)}) \quad \forall i \in \mathbb{N}.
\]

(2.25)

We proceed by induction. The result holds for \( \lambda^{(0)} \). Now we assume that the result holds for \( \lambda^{(i)} \). Then, since \( t^{(i)} \in (0, 1] \) we have that

\[
\|\lambda^{(i+1)}\|_2 = t^{(i)} \|\lambda_G^{(i)}\|_2 + (1 - t^{(i)}) \|\lambda^{(i)}\|_2 \\
\leq t^{(i)} \|\lambda_G^{(i)}\|_2 + (1 - t^{(i)}) \max_{k \in [0, i-1]} \|\lambda_G^{(k)}\|_2 \\
\leq t^{(i)} \max_{k \in [0, i]} \|\lambda_G^{(k)}\|_2 + (1 - t^{(i)}) \max_{k \in [0, i]} \|\lambda_G^{(k)}\|_2 \\
= \max_{k \in [0, i]} \|\lambda_G^{(k)}\|_2.
\]
This proves the boundedness of $\|\lambda^{(i)}\|_2$, since both $\lambda_G^{(i)}$ and $d^{(i)}_\lambda$ are bounded by (2.10).

The proof for the boundedness of the dual variables associated to the equality constraints is analogous. \hfill \Box

The following lemma serves as a tuning rule for the parameter $\rho^{(i)}$. With a slight abuse of notation, we make explicit the dependence of the merit function $\phi(\cdot)$ on the penalty parameter $\rho$.

**Lemma 2.4.** There exists $\hat{\rho}^{(i)} \in \mathbb{R}_{\geq 0}$ such that

$$
\sup_{\rho \geq \hat{\rho}^{(i)}} \phi'(0, \rho) \leq -\frac{1}{2\alpha^{(i)}} \left\| d^{(i)}_z \right\|_2^2,
$$

where $\phi$ is as in (2.13) and $\alpha^{(i)}$ and $d^{(i)}_z$ are from (2.4). \hfill \Box

**Proof.** Using a simplified notation, for a given $\rho \in \mathbb{R}_{\geq 0}$, the gradient of $L_{\text{aug}}$ in (2.11) is

$$
\begin{bmatrix}
\nabla J(z) + \nabla g(z)\lambda + \nabla h(z)\nu + \rho \nabla g(z)(g(z) + s) + \rho \nabla h(z) h(z) \\
g(z) + s \\
h(z) \\
\lambda + \rho (g(z) + s)
\end{bmatrix}.
$$

Therefore, we have

$$
\phi'(0) = d^{\top}_z (\nabla J + \nabla g \lambda + \nabla h \nu + \rho \nabla g (g + s) + \rho \nabla h h)
$$

$$
+ d^{\top}_\lambda (g + s) + d^{\top}_\nu h + d^{\top}_s (\lambda + \rho (g + s))
$$

$$
= d^{\top}_z \nabla J + (\nabla g^{\top} d_z + d_s)^{\top} \lambda + \rho (\nabla g^{\top} d_z + d_s)^{\top} (g + s)
$$

$$
+ d^{\top}_z \nabla h \nu + d^{\top}_\lambda (g + s) + d^{\top}_\nu h + \rho (\nabla h^{\top} d_z)^{\top} h
$$

$$
= d^{\top}_z \nabla J - (g + s)^{\top} (\lambda - d_\lambda) - \rho (g + s)^{\top} (g + s)
$$

$$
- h^{\top} (\nu - d_\nu) - \rho h^{\top} h,
$$

where the last step follows from (2.17) and the last equation in (2.9), as these imply that $d_z$ satisfies

$$
\nabla g^{\top} d_z + d_s = -(g + s) \tag{2.26a}
$$

$$
\nabla h^{\top} d_z = -h. \tag{2.26b}
$$
2 A projected gradient and constraint linearization method for nonlinear programs

By rearranging the first equation in (2.9) and using the definition of \( \lambda_G \) and \( \nu_G \) we have

\[
\nabla J = \frac{-1}{\alpha} d_z - \nabla g \lambda_G - \nabla h \nu_G,
\]

that, combined with (2.26a), yields:

\[
\phi'(0) = \frac{-1}{\alpha} d_z \top d_z - d_z \top \nabla g \lambda_G - d_z \top \nabla h \nu_G - (g + s) \top (\lambda - \delta \lambda) - \rho (g + s) \top (\nu - d \nu) - \rho h \top h
\]

\[
= \frac{-1}{\alpha} d_z \top d_z + d \top \lambda_G + (g + s) \top \lambda_G - (g + s) \top (\lambda - \delta \lambda) - \rho (g + s) \top (g + s) + h \top \nu_G - h \top (\nu - d \nu) - \rho h \top h.
\]

By (2.10) and the right-hand side of (2.15), we want to prove that

\[
d \top \lambda_G + 2(g + s) \top d \lambda - \rho (g + s) \top (g + s) + 2h \top d \nu - \rho h \top h
\]

\[
= d \top \lambda_G + 2 \begin{bmatrix} (g + s) \top & h \top \end{bmatrix} \begin{bmatrix} d \lambda \\ d \nu \end{bmatrix} - \rho (g + s) \top (g + s) - \rho h \top h
\]

\[
\leq \frac{1}{2\alpha} \|d_z\|^2.
\]

for a specific choice of \( \rho \). Note that \( d \top \lambda_G \leq 0 \) because of (2.17) and the complementarity conditions in (2.9). The determination of \( \hat{\rho} \) is non-trivial only if

\[
2 \begin{bmatrix} (g + s) \top & h \top \end{bmatrix} \begin{bmatrix} d \lambda \\ d \nu \end{bmatrix} \geq \frac{1}{2\alpha} \|d_z\|^2,
\]

(2.27)

otherwise we can take \( \hat{\rho} = 0 \). Hence, if (2.27) holds, then we take \( \hat{\rho} \) such that

\[
2 \begin{bmatrix} (g + s) \top & h \top \end{bmatrix} \begin{bmatrix} d \lambda \\ d \nu \end{bmatrix} \leq 2 \begin{bmatrix} g + s \\ h \end{bmatrix} \|d \lambda\|_2 \|d \nu\|_2 \leq \hat{\rho} \begin{bmatrix} g + s \\ h \end{bmatrix} \|\begin{bmatrix} g + s \\ h \end{bmatrix}\|_2^2.
\]

This is equivalent to

\[
\hat{\rho} = 2 \frac{\|d \lambda\|_2}{\|\begin{bmatrix} g + s \\ h \end{bmatrix}\|_2}.
\]

(2.28)
Therefore, we can define the penalty parameter at the beginning of each iteration \(i\) as follows.

\[
\rho^{(i)} := \begin{cases} 
\rho^{(i-1)} & \text{if } \phi'(0, \rho^{(i-1)}) \leq -\frac{1}{2\alpha^{(i)}} \|d_z^{(i)}\|^2 \\
\max\{\hat{\rho}^{(i)}, 2\rho^{(i-1)}\} & \text{otherwise,}
\end{cases}
\]  
(2.29)

with \(\hat{\rho}^{(i)}\) as in (2.28) and \(d_\lambda, d_\nu, z\) and \(s\) evaluated at the current iteration \(i\).

**Remark 2.1.** Note that the parameter \(\rho^{(i)}\) can possibly diverge for \(i \to \infty\), if there exists an infinite set of iterations \(\{i_l\}_l\) where the parameter strictly increases. The statements given next consider this possibility and prove convergence in a general case. □

**Lemma 2.5.** Suppose \(\{i_l\}_l\) is the set of iterations in which the penalty parameter \(\rho^{(i_l)}\) increases. Then,

\[
\rho^{(i_l)} \left\|d_z^{(i_l)}\right\|^2 \leq N_\rho 
\]  
(2.30a)

\[
\rho^{(i_l)} \left\|[g(z^{(i_l)}) + s^{(i_l)}] \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix}\right\| \leq N_\rho 
\]  
(2.30b)

for some \(N_\rho \in \mathbb{R}_{>0}\).

**Proof.** The argument of the functions and the index \(i_\rho\) are dropped for ease of notation. In order for the penalty parameter to increase, the conditions in (2.27) must hold, that is,

\[
\frac{1}{2\alpha} \left\|d_z\right\|^2 \leq 2 \left\|(g + s)^\top \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix}\right\| \leq 2 \left\|[g + s] \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix}\right\|,
\]

hence

\[
\left\|[g + s] \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix}\right\| \geq \frac{1}{4\alpha} \left\|d_z\right\|^2.
\]

By substituting this last inequality into the definition of \(\hat{\rho}\), we have that

\[
\hat{\rho} = \frac{2}{\left\|[g + s] \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix}\right\|} \leq 8\alpha \frac{\left\|d_z\right\|^2}{\left\|d_z\right\|^2},
\]
and the desired result in (2.30a) hold due to Lemma 2.3. The following relation proves (2.30b):

\[
\begin{align*}
\rho \left\| \begin{bmatrix} g + s \\ h \end{bmatrix} \right\|_2^2 & \leq 2 \bar{\rho} \left\| \begin{bmatrix} g + s \\ h \end{bmatrix} \right\|_2^2 \\
& = 2 \left\| \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix} \right\|_2^2 \left\| \begin{bmatrix} g + s \\ h \end{bmatrix} \right\|_2^2 = 4 \left\| \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix} \right\|_2^2. \quad (2.31)
\end{align*}
\]

The following two lemmas, provided without proofs, give intermediate technical results that are required for the main results. Their proofs follow the same arguments in [60] with minor adjustments to reflect the update rule in Algorithm 2.1.

**Lemma 2.6 ([60, Lemma 4.6]).** Let \( \{i_l\} \) denote the set of iterations for which the parameter \( \rho^{(i)} \) increases. Then, there exists \( M \in \mathbb{R}_{>0} \) such that, for all \( l \in \mathbb{N} \),

\[
\rho^{(i_l)} \sum_{i=i_l}^{i_{l+1}-1} \left\| t^{(i)} d_z^{(i)} \right\|_2^2 < M. \quad (2.32)
\]

**Lemma 2.7 ([60, Lemma 4.9]).** The merit function step size \( t^{(i)} \) defined according to (2.14) satisfies \( \phi(t^{(i)}) - \phi(0) \leq \sigma_1 t^{(i)} \phi'(0) \), where \( \sigma_1 < \frac{1}{2} \) and \( t^{(i)} > \bar{t} \), for some \( \bar{t} > 0 \) independent of \( i \).

We are now ready to state the main result of the paper, that is, the global convergence of our proposed algorithm.

**Theorem 2.2.** Algorithm 2.1 is such that \( \lim_{i \to \infty} \left\| d_z^{(i)} \right\|_2 = 0. \)

**Proof.** The proof is similar to [60, Proof of Theorem 4.1]. If \( \left\| d_z^{(i)} \right\|_2 = 0 \) for a finite \( i \), then the algorithm terminates and the statement is true. We assume in the following that \( \left\| d_z^{(i)} \right\|_2 \neq 0 \) for all \( i \in \mathbb{N} \).
2.4 Proof of convergence of the proposed algorithm

If there is no upper bound on \( \rho^{(i)} \), then the uniform lower bound \( \tilde{t} > 0 \) from Lemma 2.7 and (2.32) implies that, for all \( \delta > 0 \), there exists \( \tilde{i} \in \mathbb{N} \) such that \( \|d_2^{(i)}\|_2 \leq \delta \) for all \( i \geq \tilde{i} \), which proves the statement.

In the bounded case, there exists a value \( \tilde{\rho} \) and an index \( \tilde{i} \) such that \( \rho^{(i)} = \tilde{\rho} \) for all \( i \geq \tilde{i} \). The proof is then by contradiction. We assume that there exist \( \epsilon > 0 \) and \( \tilde{i} \in \mathbb{N} \) such that \( \|d_2^{(i)}\|_2 > \epsilon \) for all \( i \geq \tilde{i} \). Now, every subsequent iteration must yield a decrease in the merit function in (2.11) with \( \rho^{(i)} = \tilde{\rho} \), since because of (2.14), Lemma 2.4 and Lemma 2.7 we have

\[
\phi(t^{(i)}) - \phi(0) \leq \sigma_1 t^{(i)} \phi'(0) \leq -\frac{1}{2\alpha^{(i)}} \sigma_1 \tilde{t} \epsilon^2 < 0.
\]

where the gradient step size \( \alpha^{(i)} \) is designed to be bounded. The addition of the slack variable \( s^{(i)} \) in (2.12) can only lead to a further reduction in the merit function. Therefore, since the merit function with \( \rho^{(i)} = \tilde{\rho} \) decreases by at least a fixed quantity at every iteration, it must be unbounded from below. Since by Lemma 2.3 the dual variables \( \lambda^{(i)} \) and \( \nu^{(i)} \) are bounded, the merit function in (2.11) can be unbounded from below only if the objective, or the constraints functions, are unbounded from below. This leads to a contradiction, since due to Assumptions 2.2 and 2.3 all the iterates lie in a region \( \Omega \), where the objective and constraints functions are bounded in norm. Therefore, the result follows.

Theorem 2.3. It holds that the primal and dual iterates in Algorithm 2.1 converge to the KKT triple associated to a critical point \( z^* \) of (2.1). That is:

\[
\lim_{i \to \infty} \|z^{(i)} - z^*\|_2 = \lim_{i \to \infty} \|\lambda^{(i)} - \lambda^*\|_2 = \lim_{i \to \infty} \|\nu^{(i)} - \nu^*\|_2 = 0.
\]

Proof. The proof follows the line of [60] Proofs of Corollary 4.1 and Theorem 4.2]. See Appendix C for the proof details.

2.4.3 Asymptotic linear convergence

In this section we show that merit function step sizes \( t = 1 \) are not precluded by the Wolfe conditions in (2.14) when the iterates are sufficiently close to the solution. Therefore local convergence at a linear
rate (Theorem 2.1) can be recovered. The following standard SQP assumption is considered in the analysis [60, 104].

**Assumption 2.5.** For all sufficiently large $i$, the following holds:

\[
\begin{align*}
\|z^{(i)} + d_z^{(i)} - z^*\|_2 &= o\left(\|z^{(i)} - z^*\|_2\right) \\
\|\lambda^{(i)} + d_\lambda^{(i)} - \lambda^*\|_2 &= o\left(\|\lambda^{(i)} - \lambda^*\|_2\right) \\
\|\nu^{(i)} + d_\nu^{(i)} - \nu^*\|_2 &= o\left(\|\nu^{(i)} - \nu^*\|_2\right) \\
[d_\lambda^{(i)}; d_\nu^{(i)}] &= O\left(\|d_z^{(i)}\|_2\right).
\end{align*}
\]

This assumption implies that

\[
\|d_z^{(i)}\|_2 \sim \|z^{(i)} - z^*\|_2, \quad \|d_\lambda^{(i)}\|_2 \sim \|\lambda^{(i)} - \lambda^*\|_2, \quad \|d_\nu^{(i)}\|_2 \sim \|\nu^{(i)} - \nu^*\|_2,
\]

where the notation “$\sim$” indicates that the quantities are of similar order as $i$ approaches infinity. Note that this assumption may be restrictive, as it implies a faster convergence (superlinear) than that assessed in Theorem 2.1, even though the linear convergence is a conservative bound. However, we observe that this holds true in some practical situation, in particular if the Hessian of the Lagrangian at the solution is a multiple of the identity matrix.

Next we show that the penalty parameter $\rho$ is bounded.

**Lemma 2.8.** If Assumption 2.5 holds, then there exists a finite $\bar{\rho}$ such that $\rho^{(i)} \leq \bar{\rho}$ for all $i$. \hfill $\square$

**Proof.** The proof follows the same argument as [60, Lemma 5.1]. Assume that the parameter $\rho$ is unbounded. Then, by Lemma 2.4, the condition in (2.27) must hold over an infinite subsequence of iterations. Thus, using simplified notation,

\[
\left\|\begin{bmatrix} g + s \\ h \end{bmatrix}\right\|_2 \geq \frac{1}{4\alpha} \frac{\|d_z\|_2^2}{\left\|\begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix}\right\|_2},
\]

34
and in turn, by Assumption 2.5, there exist a constant $M$ such that:

$$\left\| \begin{bmatrix} g + s \\ h \\ d_\lambda \\ d_\nu \end{bmatrix} \right\|_2 \geq \frac{1}{4\alpha} \left( \left\| d_z \right\|_2^2 \left\| d_\lambda \right\|_2^2 \right) = \frac{M}{4\alpha} > 0,$$

for all sufficiently large iterations $i$, hence the constraints are bounded from below in norm. By Lemma 2.5, the penalty parameter $\rho$ must be bounded over the infinite subsequence of iterations, contradicting the unboundedness assumption.

**Lemma 2.9.** Under Assumption 2.5, the condition in (2.14b) holds with merit function step size $t^{(i)} = 1$ for sufficiently large $i$, i.e.:

$$\phi(1) - \phi(0) \leq \sigma_1 \phi'(0),$$

where $0 < \sigma_1 < \frac{1}{2}$.

**Proof.** The proof is similar to [60, Lemma 5.2] and [104, Lemma 4.2], see Appendix C for the proof details.

**Lemma 2.10.** Under Assumption 2.5, the condition in (2.14a) holds with merit function step size $t^{(i)} = 1$ for sufficiently large $i$, i.e.:

$$|\phi'(1)| \leq \sigma_2 |\phi'(0)|,$$

where $\sigma_2 < \frac{1}{2}$.

**Proof.** The proof is similar to [60, Lemma 5.3]. See Appendix C for the proof details.
3 A non-smooth exact penalty function reformulation

The approach in Chapter 2 possibly yields an unbounded value of the penalty parameter \( \rho \). This does not affect the convergence guarantees, but it may slow down the practical convergence, by imposing a small merit function step size \( t \).

In this chapter, the global convergence of the proposed method is proved by means of a non-smooth exact penalty function, specifically an \( \ell_p \) merit function. The \( \ell_p \) merit function, with \( p \geq 1 \), has been long established as an exact penalty function and used in the SQP method. This gives two main advantages compared to the approach in Chapter 2. First, the dual variables need not be estimated, which reduces the overall number of variables in the problem and the computational burden for their evaluation. In addition, we are still able to (inexpensively) derive the dual optimum at the solution, which are obtained by the dual variables associated with the KKT conditions of the projected step. Secondly, by the considered assumptions, it is possible to prove that the penalty parameter in the \( \ell_p \) merit function remains bounded. Finally, we discuss the asymptotic behavior of the iterates close to the critical point.

The chapter is organized as follows. In Section 3.1 we review the relevant literature on non-smooth penalty functions and their use for SQP. We show in Section 3.2 how the algorithm proposed in Chapter 2 is adjusted and in Section 3.3 we prove the global convergence result.

3.1 Literature review

Non-smooth exact penalty functions have long been considered in the literature as merit functions for solving nonlinear optimization problems. In fact, mild constraint qualification conditions ensure that a strict local minimum of a smooth optimization problem is a local minimum of the exact penalty function, for a sufficiently large value of the penalty parameter. Conversely, a local minimum of the exact penalty function is a critical point of the original nonlinear problem, again provided that the penalty parameter is sufficiently large [70, 102, 13].
We consider the nonlinear optimization problem (NLP) in (2.1), here repeated for convenience:

$$\min_{z \in \mathbb{R}^n} J(z)$$

s.t. $g(z) \leq 0$

$h(z) = 0,$

and the $\ell_p$ merit function of the form

$$\gamma_p(z, \rho) := J(z) + \rho \|g(z)^+\|_p + \rho \|h(z)\|_p,$$  \hspace{1cm} (3.1)

with $\rho$ the penalty parameter, $p \geq 1$ (typically either 1, 2 or $\infty$) and where the function $g(z)^+$ is defined as follows:

$$g_j(z)^+ = \begin{cases} g_j(z) & \text{if } g_j(z) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

for every $j \in \{1, \ldots, m\}$. The results in [70] also hold for a more general class of merit functions.

The main difference with the augmented Lagrangian in (2.11) is that the merit function in (3.1) is not differentiable everywhere, because of the $\ell_p$ norm. Further, no estimate for the dual variable is needed, which results in a reduced computational burden.

The value of the penalty parameter $\rho$ that ensures equivalence of the two problems depends on the dual variables of (2.1) at the optimum. In particular, if $(z^*, \lambda^*, \nu^*)$ is a KKT triple satisfying the second-order sufficiency conditions for (2.1) and a specific constraint qualification holds, then for any $\rho > \rho^* := \max \left( \|\lambda^*\|_q, \|\nu^*\|_q \right)$ the point $z^*$ is a strict local minimum of $\gamma_p(z, \rho)$ (with $q$ such that $\frac{1}{p} + \frac{1}{q} = 1$). The Mangasarian-Fromovitz constraint qualification (MFCQ) introduced thereafter guarantees that the dual variables, hence the penalty parameter, remain bounded.

Even though this is a theoretically powerful result, the value $\rho^*$ is unknown a priori, therefore the penalty parameter $\rho$, must be updated by the optimization algorithm. This feature is indeed absent in the earliest SQP formulations, where it is assumed that the penalty parameter $\rho$ be sufficiently large. In [69], the search direction $d_z^{(i)}$ is computed by solving the problem in (1.4), along which an $\ell_1$ merit function in (3.1) is decreased. This method also presents the drawback of requiring an
exact line search to derive the next iterate, which in general amounts to solving an optimization program.

A first-order version of the approach in [69] is then proposed in [87]. Each update $d_k^{(i)}$ is a combination of a “vertical” step and an “horizontal” step. The former is a vector that satisfies the constraints up to the first-order, while the latter improves the optimality of the problem. Several possible combinations of the horizontal and the vertical components are considered, even though without theoretical convergence rate guarantees. The approach has also the merit of simplifying the line search to a standard Armijo rule and proposing an update rule for the penalty parameter, $\rho^{(i)}$, with convergence guarantees.

The main drawback of the exact penalty functions is the Maratos effect [84]. As shown in Section 2.4.3 an augmented Lagrangian approach allows one to asymptotically recover the local convergence results, since a full merit function step size $t^{(i)} = 1$ in (2.7) is allowed by the Wolfe conditions in (2.14). Conversely, non-smooth penalty functions can suffer from a very slow convergence close to the solution, since a full step size $t^{(i)} = 1$ fails the line search condition, thus preventing the SQP fast local convergence rate. One way to solve the problem is a second-order correction. The next iterate is determined on a search arc – rather than on a search line with variable $t^{(i)}$ as in (2.7) – computed by additionally considering a vector orthogonal to the decrease direction $d_k^{(i)}$. This vector is typically defined such that it satisfies a linearization of the constraints at $z^{(i)} + d_k^{(i)}$ [89, 101]. In [35, 36] an alternative approach consists of dropping a constraint based on the multiplier estimate. The resulting effect is an asymptotic 2-step superlinear convergence. An alternative to the second-order correction is the watchdog technique, where an iteration may be accepted even if not decreasing the merit function [32, 99]. If a sufficient decrease of the merit function is not obtained after a certain number of iterates, then the algorithm restores a previous iterate and performs the line search normally. Thanks to this condition, the method converges and a full step size $t^{(i)} = 1$ is allowed close to the optimum.

The $\ell_p$ merit function has also been applied to reduced Hessian methods, which are particularly efficient for large-scale problems. In this dissertation we will not investigate these approaches and we cite [14, 30, 13] for reference.

In this chapter we prove convergence of the proposed algorithm by using the $\ell_p$ merit function in (3.1). We will not consider any of the presented techniques for obtaining $t^{(i)} = 1$ asymptotically. In fact, as
it will be discussed in Chapter 4, in practice it is advantageous not to tune the gradient step size $\alpha^{(i)}$ according to Theorem 2.1. Even though this already impedes the asymptotic linear convergence, it is computationally cheaper and the convergence in practice is comparable. Therefore, the correction techniques presented for the SQP method would increase the complexity of the algorithm without a practical benefit.

3.2 Preliminaries

In contrast to the previous chapter, here we assume that the functions $J : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^p$ are continuously differentiable (not necessarily twice continuously differentiable). As before, we assume that the NLP in (2.1) has a finite number of critical points. In order to determine a critical point $z^*$, we generate a sequence $(z^{(i)})_{i=1}^\infty$ that is updated as in (2.7), that is

$$z^{(i+1)} := z^{(i)} + t^{(i)} d_z^{(i)}.$$ 

The update $d_z^{(i)}$ is determined via the projected gradient and constraint linearization step in (2.4), i.e.

$$d_z^{(i)} := \Pi_{C^{(i)}} \left( -\alpha^{(i)} \nabla J(z^{(i)}) \right),$$

with bounded gradient step size $\alpha^{(i)} \in \mathbb{R}_{>0}$ and the set $C^{(i)}$ as in (2.5):

$$C^{(i)} := \{ d_z \in \mathbb{R}^n \mid g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z \leq 0, h(z^{(i)}) + \nabla h(z^{(i)})^\top d_z = 0 \}.$$

Under the assumptions below there exist dual multipliers $\lambda_G^{(i)} \in \mathbb{R}_{\geq 0}^m$, $\nu_G^{(i)} \in \mathbb{R}^m$ such that the KKT conditions in (2.9) hold:

$$\frac{1}{\alpha^{(i)}} d_z^{(i)} + \nabla J(z^{(i)}) + \nabla g(z^{(i)}) \lambda_G^{(i)} + \nabla h(z^{(i)}) \nu_G^{(i)} = 0$$

$$\text{diag}(\lambda_G^{(i)}) \left( g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z^{(i)} \right) = 0$$

$$g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z^{(i)} \leq 0$$

$$h(z^{(i)}) + \nabla h(z^{(i)})^\top d_z^{(i)} = 0.$$ 

We consider the merit function in (3.1) and, by defining $\phi(t) := \gamma_p (z + td_z, \rho)$, we choose the merit function step size
3.3 Global convergence via a non-smooth merit function

\( t^{(i)} \in (0,1] \) that satisfies the following Armijo condition:

\[
\phi(t^{(i)}) \leq \phi(0) + \sigma t^{(i)} D\phi(0),
\]

where \( \sigma \in (0,1) \) and \( D\phi(\cdot) \) is the directional derivative of \( \phi(\cdot) \)

\[
D\phi(0) = \lim_{t \to 0^+} \frac{\phi(t) - \phi(0)}{t}.
\]

We determine the step size \( t \) via safeguarded quadratic interpolation, such that if a tentative step \( t \) fails, then a new \( t \in [\tau_1 \tilde{t}, \tau_2 \tilde{t}] \) is tested, with \( 0 < \tau_1 < \tau_2 < 1 \). The penalty parameter \( \rho^{(i)} \) is tuned such that the directional derivative \( D\phi(0) \) is sufficiently negative. This is ensured by updating \( \rho^{(i)} \) as follows:

\[
\rho^{(i)} := \max(\rho^{(i-1)}, \hat{\rho}^{(i)} + \varepsilon),
\]

with \( \varepsilon > 0 \) and \( \hat{\rho}^{(i)} \) derived next in Lemma 3.1. At start we set \( \rho^{(0)} = 0 \).

Note that this update rule makes \( \rho^{(i)} \) not decreasing with \( i \). Other approaches explored in [101] instead let the penalty parameter decrease without loss of the theoretic convergence properties. In case of premature failure due to an overly large penalty parameter, we propose the reset procedure presented in Section 4.3.2.

The method is summarized in Algorithm 3.1.

3.3 Global convergence via a non-smooth merit function

In this chapter we consider Assumptions 2.1 and 2.2, which guarantee respectively feasibility of (2.4) and the existence of a compact set containing all the iterates \( (z^{(i)})_i \) and \( (z^{(i)} + d^{(i)}_z)_i \).

Assumption 2.3 and Assumption 2.4 are instead relaxed in this chapter as follows:

**Assumption 3.1.** The function \( J \) is uniformly bounded in norm in \( \Omega \).

This assumption is guaranteed by the Weierstrass extreme value theorem, given that the function \( J \) is continuous in \( \Omega \).

**Assumption 3.2.** For all \( i \in \mathbb{N} \), and \( z^{(i)} \), let \( I_G(d^{(i)}_z, z^{(i)}) \) denote the index set of the active constraints in (2.8), i.e.,

\[
I_G(d^{(i)}_z, z^{(i)}) = \left\{ j \in \{1, \ldots, m\} \mid g_j(z^{(i)}) + \nabla g_j(z^{(i)})^T d^{(i)}_z = 0 \right\}.
\]
Algorithm 3.1 First-order algorithm with $\ell_p$ merit function

\begin{algorithm}
\begin{algorithmic}
  \STATE \textbf{Initialize} $i \leftarrow 0$, $z^{(0)} \in \mathbb{R}^n$ and $\rho^{(0)} = 0$
  \REPEAT
    \STATE COMPUTE $d_z^{(i)}$ with gradient step size $\alpha^{(i)}$ as in (2.4)
    \STATE DETERMINE $\lambda_G^{(i)}, \nu_G^{(i)}$ such that (2.9) holds
    \IF{$d_z^{(i)} = 0$}
      \STATE SET $z^* = z^{(i)}$, $\lambda^* = \lambda_G^{(i)}, \nu^* = \nu_G^{(i)}$ and STOP
    \ENDIF
    \STATE SET $\rho^{(i)} \in \mathbb{R}_{\geq 0}$ as in (3.4)
    \STATE DETERMINE the merit function step size $t^{(i)}$ that satisfies (3.3), e.g., via line search
    \STATE UPDATE $z^{(i+1)}$ as in (2.7)
    \STATE $i \leftarrow i + 1$
  \UNTIL \textit{Convergence}
  \RETURN $z^*$, $\lambda^*$ and $\nu^*$
\end{algorithmic}
\end{algorithm}

Then $d_z^{(i)}$ satisfies the Mangasarian-Fromovitz constraint qualification, i.e., the matrix $\nabla h(z^{(i)})$ has full column rank and there exists a search direction $d \in \mathbb{R}^n$ such that $\nabla g_j(z^{(i)})^\top d < 0$, $\forall j \in I_G(d_z^{(i)}, z^{(i)})$, and $\nabla h(z^{(i)})^\top d = 0$.

The MFCQ was first introduced in [8] and then extended to the case with equality constraints in [83]. This assumption is milder than LICQ and it implies that the dual variables $\lambda_G^{(i)}$ and $\nu_G^{(i)}$ are bounded in norm, even though possibly non-unique.

With the merit function in (3.1), the penalty parameter $\rho$ can be chosen according to the following tuning rule, where we define $q$ such that $\frac{1}{p} + \frac{1}{q} = 1$. With a slight abuse of notation, in the following lemma we explicit the dependence of the merit function $\phi(\cdot)$ on the penalty parameter $\rho$.

\begin{lemma}
There exists $\hat{\rho}^{(i)} \in \mathbb{R}_{\geq 0}$ such that
\begin{equation*}
\sup_{\rho \geq \hat{\rho}^{(i)}} D\phi(0, \rho) \leq -\frac{1}{\alpha^{(i)}} \left\| d_z^{(i)} \right\|_2^2,
\end{equation*}
\end{lemma}
3.3 Global convergence via a non-smooth merit function

Proof. For ease of notation we omit the iteration index $i$. The directional derivative of (3.1) is:

$$D\phi(0) \leq d_z^T \nabla J(z) - \rho \|g(z)^+\|_p - \rho \|h(z)\|_p,$$

by the differentiation shown in Appendix C.

By applying the optimality, complementarity and feasibility conditions in (2.9) and the Cauchy-Schwarz inequality $a^T b \leq \|a\|_p \|b\|_q$, we have:

$$D\phi(0) \leq -\frac{1}{\alpha} d_z^T d_z - d_z^T \nabla g(z) \lambda_G - d_z^T \nabla h(z) \nu_G - \rho \|g(z)^+\|_p - \rho \|h(z)\|_p$$

$$= -\frac{1}{\alpha} d_z^T d_z + g(z)^T \lambda_G + h(z)^T \nu_G - \rho \|g(z)^+\|_p - \rho \|h(z)\|_p$$

$$\leq -\frac{1}{\alpha} d_z^T d_z + (g(z)^+)^T \lambda_G + h(z)^T \nu_G - \rho \|g(z)^+\|_p - \rho \|h(z)\|_p$$

$$\leq -\frac{1}{\alpha} d_z^T d_z + \left(\|\lambda_G\|_q - \rho\right) \|g(z)^+\|_p + \left(\|\nu_G\|_q - \rho\right) \|h(z)\|_p,$$

(3.6)

from which the lemma follows for $\hat{\rho}(i) := \max \left(\|\lambda_G^{(i)}\|_q, \|\nu_G^{(i)}\|_q\right)$. □

The next lemma shows that the penalty parameter $\rho^{(i)}$ is bounded in the compact set $\Omega$ defined in Assumption 2.2.

**Lemma 3.2.** The parameter $\rho^{(i)}$ is bounded in $\Omega$, i.e., there exists $\bar{\rho} > 0$ such that $\rho^{(i)} \leq \bar{\rho}$ for any $z^{(i)} \in \Omega$. □

Proof. Assumption 2.2 and Assumption 3.2 imply that, for any $z^{(i)} \in \Omega$, the dual variables $\lambda_G^{(i)}$ and $\nu_G^{(i)}$ in (2.9) are bounded. The result follows from the definition of $\rho^{(i)}$ in (3.4). □

The boundedness of the penalty parameter, $\rho^{(i)}$, is used in the following lemma to guarantee the existence of a lower bound $\bar{t}$ to the step size $t^{(i)}$ in (3.3).

**Lemma 3.3.** There exists $\bar{t} = \bar{t}(\bar{\rho}) > 0$ such that the condition in (3.3) is satisfied. □
3 A non-smooth exact penalty function reformulation

Proof. For ease of notation we omit the iteration index $i$. By the continuity of the merit function in (3.1), we know that there exists a sufficiently small $t \in (0, 1]$ that satisfies the Armijo condition in (3.3). Suppose that the Armijo test failed for $\tilde{t}$, i.e.,

$$
\phi(\tilde{t}) - \phi(0) > \sigma \tilde{t} D\phi(0),
$$

and that $\tau_1 \tilde{t} \leq t$. On the other hand, by expanding $\phi(\tilde{t})$ to the second-order, we have:

$$
\phi(\tilde{t}) - \phi(0) \leq \tilde{t} D\phi(0) + (\tilde{t})^2 b \|dz\|^2,
$$

with $b = b(\rho) > 0$.

It follows from the two latter inequalities that

$$(\sigma - 1) \tilde{t} D\phi(0) < (\tilde{t})^2 b \|dz\|^2,$$

which implies, since $\sigma < 1$, that

$$
D\phi(0) > \frac{\tilde{t} b \|dz\|^2}{\sigma - 1}.
$$

By Lemma 3.1 we have that $D\phi(0) \leq -\frac{1}{\alpha} \|dz\|^2$, hence, due to the last inequality, we have that

$$
\tilde{t} > \frac{1 - \sigma}{\alpha b} > 0,
$$

with $\alpha$ designed to be bounded (e.g., $\alpha = 1 \ \forall i$). Note that from Lemma 3.2 it follows that there exists a $\bar{\rho} > 0$ such that $\rho \leq \bar{\rho}$. Therefore, since the merit function in (3.1) is monotone in $\rho$ (affine with positive coefficient for a fixed $z$), we can upper bound the term $b(\rho)$ in (3.7) with a bounded $\bar{b}(\bar{\rho})$. Hence $\tilde{t}$ is always strictly greater than zero. Finally, we can derive the lower bound

$$
\bar{t} := \frac{\tau_1 (1 - \sigma)}{\alpha b} \text{ for } t \geq \tau_1 \tilde{t}.
$$

□

Theorem 3.1. It holds that the primal iterates in Algorithm 3.1 converge to the KKT point associated to a KKT triple $(z^*, \lambda^*, \nu^*)$ of (2.1), i.e.,

$$
\lim_{i \to \infty} \left\| z^{(i)} - z^* \right\|_2 = 0.
$$

Further, it holds that:

$$
\lim_{i \to \infty} \left\| \lambda_G^{(i)} - \lambda^* \right\|_2 = \lim_{i \to \infty} \left\| \nu_G^{(i)} - \nu^* \right\|_2 = 0.
$$

□
Proof. The proof consists of showing that

$$\lim_{i \to \infty} \left\| d^{(i)}_z \right\|_2 = 0. \quad (3.8)$$

The statement of the theorem then follows by comparing (2.9) and (2.3).

We follow a similar argument to the proof of Theorem 2.2 and [60, Proof of Theorem 4.1]. For the sake of contradiction, suppose that there exists an $\varepsilon > 0$ and $\tilde{i} \in \mathbb{N}$ such that $\left\| d^{(i)}_z \right\|_2 > \varepsilon$ for all $i \geq \tilde{i}$. Note that from Lemma 3.2 $\rho^{(i)}$ is bounded.

Then, in every iteration $i \geq \tilde{i}$ the merit function decreases, due to Lemma 3.1 and in particular:

$$\phi(t^{(i)}) - \phi(0) \leq \sigma t^{(i)} D\phi(0) \leq -\frac{1}{\alpha(i)} \sigma \tilde{t} \left\| d^{(i)}_z \right\|_2^2$$

$$\leq -\frac{1}{\alpha(i)} \sigma \tilde{t} \varepsilon^2 < 0,$$

with $\alpha(i)$ is designed to be bounded (e.g., $\alpha(i) = 1 \forall i$). This implies that the merit function decreases by a finite amount at every iteration, thus is unbounded from below. However, by Assumption 3.1, this cannot hold, since the objective function is bounded in norm. Therefore, this leads to a contradiction and (3.8) must hold true.

**Remark 3.1.** If only equality constraints are present in the problem, then Assumption 3.2 reduces to the LICQ of the equality constraints, and as a result the dual variables $\nu_G$ are continuous in $z$. Hence, a stronger result can be derived in Lemma 3.2, namely the existence of a $\rho^* \in \mathbb{R}_{\geq 0}$ and $\tilde{i} \in \mathbb{N}$ such that $\rho^{(i)} = \rho^*$ for all $i \geq \tilde{i}$. This result is reported by the author in a joint paper [123].

**Remark 3.2.** The treatment in [87] (specifically for an $\ell_1$ merit function and with only equality constraints) establishes a weaker result under less restrictive assumptions. In particular, under Assumption 2.2 and LICQ of the equality constraints, the existence of a convergent subsequence of the iterates is proved. Since in this work we focus on MPC problems, the objective function is typically positive semi-definite, thus fulfilling Assumption 3.1.
4 Practical implementation for Model Predictive Control problems

In Chapters 2 and 3 we have proved the convergence of the proposed method for general constrained optimization problems. On the other hand, it is a well-known fact that the projected Gradient Method is inefficient if the feasible set is a general polytope (see e.g., [12]).

In the following, we outline a method for simplifying the computation of the projection. In Section 4.1 we transform the general nonlinear problem into an equality constrained problem via squared-slack variables, and compute the projection onto the resulting affine subspace in closed form.

In Section 4.2 we apply the method to nonlinear Model Predictive Control problems with possible box and nonlinear constraints on the input variables and a terminal or contractive constraint on the state variables. As in the standard Gradient Method for linear MPC, by writing (condensing) the state variables as an explicit function of the input, the equality constraints are directly embedded into the objective function. By the introduction of the squared-slack variables, the projection on the linearization of the constraints can be computed analytically, thus resulting in a low computational complexity. We conclude this chapter with Section 4.3 where we discuss the computational considerations for the implementation of the algorithm, such as the convergence tolerances and the tuning of the parameters.

4.1 General optimization problems

The problem in (2.1) can be reformulated as the equality constrained problem

\[
\begin{align*}
\min_{(z,y)\in \mathbb{R}^n \times \mathbb{R}^m} & \quad J(z) \\
\text{s.t.} & \quad g(z) + \frac{1}{2} \text{diag}(y) y = 0 \\
& \quad h(z) = 0,
\end{align*}
\]
hence more generally as

$$
\begin{align*}
\min_{v \in \mathbb{R}^{n+m}} & \quad J(v) \\
\text{s.t.} & \quad p(v) = 0,
\end{align*}
$$

(4.1)

with primal variable $v := [z; y]$ and $p : \mathbb{R}^{n+m} \to \mathbb{R}^{m+p}$ defined as $p([z; y]) := [g(z) + \frac{1}{2} \text{diag}(y) y; h(z)]$. Let us define $\mu := [\lambda; \nu]$ as the dual variables associated to the constraints $p(v) = 0$ ($\mu$), $g(z) + \frac{1}{2} \text{diag}(y) y = 0$ ($\lambda$), and $h(z) = 0$ ($\nu$), respectively. The equivalence between (2.1) and (4.1) is given by the following lemmas.

Lemma 4.1 ([116, Proposition 1], [12, Section 3.3.2]). The following hold:

(i) $z^*$ is a regular solution to (2.1) if and only if $[z^*; y^*]$ is a regular solution to (4.1);

(ii) if $(z^*, \lambda^*, \nu^*)$ is a KKT triple for (2.1), then $([z^*; y^*], [\lambda^*; \nu^*])$ is a KKT double for (4.1);

(iii) if $([z^*; y^*], [\lambda^*; \nu^*])$ is a KKT double for (4.1) and $\lambda^* \in \mathbb{R}^m_{\geq 0}$, then $(z^*, \lambda^*, \nu^*)$ is a KKT triple for (2.1).

Note that by the feasibility of the optimal solution to (4.1), $y^*$ is such that $y^*_j = (-2g_j(z^*))^{1/2}$ $\forall j \in \{1, \ldots, m\}$.

For the proof of part (iii), it is necessary to assume $\lambda^* \in \mathbb{R}^m_{\geq 0}$; in fact, the first-order conditions for (4.1) may in principle have negative dual variables $\lambda^*$ associated with the squared-slack equality constraints. Note that the first-order conditions, derived with respect to $y$, already guarantee complementarity slackness, i.e., $\lambda^*_j = 0$ for all indexes $j$ of the active inequality constraints at $z^*$ for (2.1).

The following lemma shows that the assumption $\lambda^* \in \mathbb{R}^m_{\geq 0}$ of part (iii) can be dropped if the solution $[z^*; y^*]$ is a local minimum.

Lemma 4.2. If $[z^*; y^*]$ is a local minimum of (4.1) with dual variables $[\lambda^*; \nu^*]$, then $(z^*, \lambda^*, \nu^*)$ is a KKT triple for (2.1).

Proof. By the second-order necessary condition for (4.1), we have

$$
\begin{bmatrix}
\tilde{z}^T & \tilde{y}^T
\end{bmatrix}
\begin{bmatrix}
\nabla_{zz}^2 \mathcal{L}(z^*, \lambda^*, \nu^*) & 0 \\
0 & \text{diag}(\lambda^*)
\end{bmatrix}
\begin{bmatrix}
\tilde{z} \\
\tilde{y}
\end{bmatrix} \geq 0
$$

(4.2)
4.1 General optimization problems

for all \( \tilde{z} \in \mathbb{R}^n, \tilde{y} \in \mathbb{R}^m \) such that:

\[
\nabla h(z^*)^\top \tilde{z} = 0, \quad \nabla g_j(z^*)^\top \tilde{z} + y_j^* \tilde{y}_j = 0 \quad \text{for all } j \in \{1, \ldots, m\}.
\]
(4.3)

Let \( j \) be the index of an arbitrary active constraint of \( z^* \). We can choose \( \tilde{z} = 0 \), with \( \tilde{y}_j \neq 0 \), and \( \tilde{y}_k = 0 \) for all \( k \neq j \). Therefore, by (4.2), we obtain \( \lambda_j^* \tilde{y}_j^2 \geq 0 \), thus \( \lambda_j^* \geq 0 \) for all active constraints. By the complementarity slackness, implied by the first-order conditions derived with respect to \( y \), we conclude that \( \lambda^* \in \mathbb{R}^m_{\geq 0} \). The result then follows from Lemma 4.1, part (iii).

The Projected Gradient and Constraint Linearization is guaranteed to converge only to critical points, and not necessarily to a local minimum. Even though there are counterexamples where the method would converge to a critical point (as the SQP would do), in practice we observe that we typically converge to a local minimum of the problem.

As shown by Part (iii) of Lemma 4.1 the condition \( \lambda^* \in \mathbb{R}^m_{\geq 0} \) ensures equivalence of the problems (4.1) and (2.1). Additional KKT points with \( \lambda^* \in \mathbb{R}^m_{< 0} \) may be introduced by adding squared-slash variables. Therefore, an a-posteriori check on the sign of the dual variables is needed in order to check convergence to a KKT point of the original problem in (2.1). Even though we cannot theoretically exclude the convergence to such points, since the Projected Gradient and Constraint Linearization method is guaranteed to converge only to a critical point, in practice we observe that, for an appropriate initialization of the slack variables, the condition on the dual variable \( \lambda^* \in \mathbb{R}^m_{\geq 0} \) is satisfied and therefore we have the desired convergence property.

The initialization required for the slack variables, \( y^{(0)} \), is derived based on the update rule. In particular, in Algorithm 2.1 the primal update \( d_v \) is computed in (2.4) as a function of the current \( v \), i.e.,

\[
d_v := \Pi_{\{d_v \in \mathbb{R}^{n+m} \mid \nabla p(v)^\top d_v = -p(v)\}} \left( -\alpha \nabla J(v) \right),
\]

where

\[
p(v) = \begin{bmatrix}
g(z) + \frac{1}{2} \text{diag}(y) y \\
h(z)
\end{bmatrix}, \quad \nabla p(v) = \begin{bmatrix}
\nabla g(z) & \nabla h(z) \\
\text{diag}(y) & 0
\end{bmatrix}.
\]

The projection admits a closed form solution. In fact, we can determine the dual variable \( \mu_G := [\lambda_G; \nu_G] \) as the solution of the dual problem
4 Practical implementation for Model Predictive Control problems

[24]:

\[ \mu_G := \begin{bmatrix} \lambda_G \\ \nu_G \end{bmatrix} = (\alpha \nabla p(v) \nabla p(v))^{-1} (p(v) - \nabla p(v) \nabla J(v)) \]

\[ = \begin{bmatrix} \nabla g(z) \nabla g(z) + \text{diag}(y)^2 & \nabla g(z) \nabla h(z) \\ \nabla h(z) \nabla g(z) & \nabla h(z) \nabla h(z) \end{bmatrix}^{-1} \]

\[ \cdot \begin{bmatrix} \frac{1}{\alpha} g(z) + \frac{1}{2\alpha} \text{diag}(y)y - \text{diag}(y) \nabla g(z) \nabla J(z) \\ \frac{1}{\alpha} h(z) - \text{diag}(y) \nabla h(z) \nabla J(z) \end{bmatrix}. \] (4.4)

Then, the primal solution is given by

\[ d_v = \begin{bmatrix} d_z \\ d_y \end{bmatrix} = -\alpha \nabla J(v) - \alpha \nabla p(v) \mu_G = \]

\[ = \begin{bmatrix} -\alpha \nabla J(z) \\ 0 \end{bmatrix} - \alpha \begin{bmatrix} \nabla g(z) & \nabla h(z) \\ \text{diag}(y) & 0 \end{bmatrix} \begin{bmatrix} \lambda_G \\ \nu_G \end{bmatrix} \] (4.5)

and dual increments \( d_\mu \) from (2.10) are

\[ d_\mu := \begin{bmatrix} d_\lambda \\ d_\nu \end{bmatrix} = \mu_G - \mu. \]

By Assumption [2.4], the matrix \( \nabla p(v) \) can be proved to be full rank, therefore the matrix \( (\alpha \nabla p(v) \nabla p(v)) \) is invertible. For this to hold, the squared-slack variable initialization, \( y^{(0)} \), has to be set different from zero, otherwise \( y^{(i)} = 0 \ \forall i \). Also note that by (4.4), the squared-slack variables do not increase the dimension of the matrix, thus the complexity of the matrix inversion does not increase.

The matrix inversion in (4.4) is commonly obtained via Cholesky factorization, which for a general dense matrix has a computational complexity of \( \mathcal{O}((m + p)^3) \) FLOPS. Therefore, for the inversion to be efficient, the sparsity of the gradients \( \nabla g \) and \( \nabla h \) should be exploited.

In the following section, we consider a general nonlinear MPC problem for which the matrix inversion can be computed symbolically, thus avoiding the Cholesky factorization.
4.2 Model Predictive Control problems

Sparsity patterns naturally arise in MPC problems, due to the causality of the dynamics and the structure of the constraints.

4.2.1 Quadratic terminal constraint

Let us consider a typical nonlinear MPC problem with input constraints and a quadratic terminal state constraint. At every time \( t_k \) starting from the current state \( x_k =: x_k|k \), the following optimization problem is solved to determine the optimal inputs \( u_{k+j}|k \), for \( j \in \{0, \ldots, N-1\} \):

\[
\min_{(x_{k+j+1}|k, u_{k+j}|k)_{j=0}^{N-1}} \sum_{j=0}^{N-1} \left\{ \frac{1}{2} u_{k+j+1}^T Q x_{k+j+1} + \frac{1}{2} u_{k+j}^T R u_{k+j} \right\} + \frac{1}{2} u_{k+N}^T P x_{k+N} \\
\text{s.t.} \quad x_{k+j+1}|k = f(x_{k+j}|k, u_{k+j}|k) \quad \forall j \in \{0, \ldots, N-1\} \\
\quad u_{k+j}|k \in [a_{k+j}|k, b_{k+j}|k] \quad \forall j \in \{0, \ldots, N-1\} \\
\quad n(u_{k+j}|k) \leq 0 \quad \forall j \in \{0, \ldots, N-1\} \\
\quad \frac{1}{2} u_{k+N}^T P x_{k+N} \leq c_k,
\]

(4.6)

where the index \( j \) spans the predicted state \( x_{k+j+1}|k \) and input \( u_{k+j}|k \) in the horizon \( N \). The bounds satisfy \( a_{k+j}|k < b_{k+j}|k \in \mathbb{R}^{n_u} \) componentwise, \( c_k \in \mathbb{R}_{>0} \) and \( Q, P, R \succeq 0 \). The discrete-time dynamics, \( f \), can be nonlinear, hence the program in (4.6) is in general nonconvex. Also the nonlinear function \( n : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x} \) can be nonconvex.

The stability constraint can also act on an index different from \( N \) or be a contractive constraint, as explained in Section 4.2.2.

The algorithm step in (2.4) can be computed in closed form by appropriately recasting the problem. For ease of presentation, we do not consider the nonlinear constraint \( n(u_{k+j}|k) \leq 0 \). An analogous treatment applies otherwise by augmenting the system. We define the vectors \( u_k := [u_k|k; \ldots; u_{k+N-1}|k] \) for the control input sequence, with bounds \( a_k := [a_k|k; \ldots; a_{k+N-1}|k] \) and \( b_k := [b_k|k; \ldots; b_{k+N-1}|k] \), and the corresponding state evolution \( x_k := [x_{k+1}|k; \ldots; x_{k+N}|k] \), and stack the state and input cost matrices \( Q = \text{blockdiag}(Q, \ldots, Q, P) \) and \( R = \text{blockdiag}(R, \ldots, R) \). We recast the dynamics in a compact form as \( x_k = \psi(u_k) \), where for a fixed initial state \( x_k \), the function \( \psi : \mathbb{R}^{n_n u} \rightarrow \mathbb{R}^{n_n x} \) maps the sequence of inputs \( u_k \) to the
predicted sequence of states $x_k$ according to the nonlinear dynamics
$\dot{x}_{k+1} = f(x_k, u_k)$ (the dependence of $\psi$ on $x_k$ is omitted for ease of notation). We also express the terminal state as $x_{k+N|k} = \psi_N(u_k)$.

Thus, by including the nonlinear dynamics within the objective and by adding the nonlinear slacks $y_{k,a}, y_{k,b} \in \mathbb{R}^{Nn_u}$ and $y_{k,c} \in \mathbb{R}$ as in (4.1), the MPC problem in (4.6) becomes

$$\min_{u_k, y_{k,a}, y_{k,b}, y_{k,c}} \quad \frac{1}{2} \psi(u_k)^T Q \psi(u_k) + \frac{1}{2} u_k^T Ru_k =: J(u)$$

$$\text{s.t.} \quad -u_k + a_k + \frac{1}{2} \text{diag}(y_{k,a}) y_{k,a} = 0$$
$$\quad u_k - b_k + \frac{1}{2} \text{diag}(y_{k,b}) y_{k,b} = 0$$
$$\quad \frac{1}{2} \psi_N(u_k)^T P \psi_N(u_k) - c_k + \frac{1}{2} y_{k,c}^2 = 0.$$  (4.7)

This formulation, albeit being unusual compared to other approaches when solving nonlinear MPC problems [45,119], leads to computational advantages for the proposed Algorithms 2.1 and 3.1.

The vector of dual variables $\mu_G$ is determined as in (4.4). Note that the matrix inversion in (4.4) can be computed analytically offline. In fact, since the gradient of the constraint is $\nabla g(u) = [-I \mid I \mid q_k]$, where $q_k \in \mathbb{R}^{Nn_u}$ is the gradient of the terminal constraints with respect to $u$, the matrix is inverted as follows:

$$\left((\nabla g(u_k)^T \nabla g(u_k) + \text{diag}([y_{k,a}; y_{k,b}; y_{k,c}]^2)^{-1} \right.$$  

$$\left. = \left[ \begin{array}{ccc}
I + \text{diag}(y_{k,a})^2 & -I & -q_k \\
-I & I + \text{diag}(y_{k,b})^2 & q_k \\
-q_k^T & q_k^T & q_k^T q_k + y_{k,c}^2 \\
\end{array} \right]^{-1} \right.$$  (4.8)

$$= \left[ \begin{array}{ccc}
D_k + B_k + r_k(B_k q_k)(B_k q_k)^T & D_k - r_k(B_k q_k)(A_k q_k)^T & r_k B_k q_k \\
D_k - r_k(A_k q_k)(B_k q_k)^T & D_k + A_k + r_k(A_k q_k)(A_k q_k)^T & -r_k A_k q_k \\
-1 & -1 & 1 \\
\end{array} \right],$$

with

$$D_k = \text{diag}_l \left( \frac{1}{y_{k,a,l}^2 + y_{k,b,l}^2 + y_{k,a,l}^2 y_{k,b,l}} \right),$$

$$A_k = \text{diag}_l \left( \frac{y_{k,a,l}^2}{y_{k,a,l}^2 + y_{k,b,l}^2 + y_{k,a,l}^2 y_{k,b,l}} \right),$$

$$B_k = \text{diag}_l \left( \frac{y_{k,b,l}^2}{y_{k,a,l}^2 + y_{k,b,l}^2 + y_{k,a,l}^2 y_{k,b,l}} \right),$$

52
for \( l \in \{1, \ldots, N_{n_u}\} \) and

\[
r_k = \left( \sum_{l=1}^{N_{n_u}} y_{k,a,l}^2 y_{k,b,l}^2 + y_{k,b,l}^2 + y_{k,a,l}^2 y_{k,b,l}^2 y_{k,l}^2 + y_{k,c}^2 \right)^{-1}.
\]

Under Assumption 2.4, we can guarantee that the denominators in \( D_k, A_k, B_k \) are always nonzero and \( r_k \) is finite. Because of the diagonal structure of \( A_k \) and \( B_k \), the vectors \( A_k q_k \) and \( B_k q_k \) are cheap to compute, and this allows one to compute the matrix multiplication in (4.4) in only \( \mathcal{O}(N_{n_u}) \) FLOPS. Moreover, since the terminal constraint in (4.6) has the same structure as the terminal cost in the objective function, the computation of \( q_k \) is inexpensive when performed together with the computation of \( \nabla J(u_k) \). The primal and slack variable updates then easily follow from (4.5).

Nonlinear input constraints in the form \( n(u) \leq 0 \) can be efficiently considered if the matrix \( \nabla n(u)^\top \nabla n(u) + \text{diag}(y^2) \) is analytically invertible for any \( u \) and \( y \). If the dimension \( n_u \) is small, this is always the case. We will show some examples in Chapter 9.

The primal variable updates \( d_{u,k} \) and the slack updates \( d_{y,k,a}, d_{y,k,b} \) and \( d_{y,k,c} \) follow from (4.5). The computation of the gradient of the objective function can also be done efficiently by exploiting the causality of the nonlinear dynamics, \( f \). From the definition of \( J(u_k) \) in (4.7), we have

\[
\nabla J(u_k) = \nabla \psi(u_k) Q x_k + R u_k \quad \text{with} \quad x_k = \psi(u_k), \quad (4.9)
\]

where the matrix \( \nabla \psi(u_k) \) contains the standard linearization matrices of the nonlinear dynamics

\[
F_{k+j|k} := \frac{\partial f}{\partial x}(x_{k+j|k}, u_{k+j|k}), \quad G_{k+j|k} := \frac{\partial f}{\partial u}(x_{k+j|k}, u_{k+j|k}),
\]

with \( j \in \{0, \ldots, N-1\} \) and it is block upper triangular. That is, \( \nabla \psi = [\nabla \psi_0, \ldots, \nabla \psi_{N-1}] \), where the column blocks \( \nabla \psi_j : \mathbb{R}^{N_{n_u}} \to \mathbb{R}^{N_{n_u} \times n_x} \)
for all $j \in \{0, \ldots, N - 1\}$ are as follows,

$$
\nabla \psi_0 = \begin{bmatrix} G_{k|k} & 0 \end{bmatrix}^T
$$

$$
\nabla \psi_1 = \begin{bmatrix} F_{k+1|k} \nabla x_{k+1|k}(u_{k|k}) & G_{k+1|k} & 0 \end{bmatrix}^T
$$

$$
\nabla \psi_1 = \begin{bmatrix} F_{k+1|k} G_{k|k} & G_{k+1|k} & 0 \end{bmatrix}^T
$$

$$
\vdots
$$

$$
\nabla \psi_{N-1} = \begin{bmatrix} F_{k+N-1|k} \nabla x_{k+N-1|k}(u_{k|k}, \ldots, u_{k+N-2|k}) & \ldots \ldots \ldots F_{k+N-1|k} \nabla x_{k+N-1|k}(u_{k|k}) & G_{k+N-1|k} \end{bmatrix}^T
$$

$$
\nabla \psi_{N-1} = \begin{bmatrix} F_{k+N-1|k} \ldots F_{k+1|k} G_{k|k} & \ldots \ldots \ldots F_{k+N-1|k} G_{k+N-2|k} & G_{k+N-1|k} \end{bmatrix}^T
$$

Note that the matrix $\nabla \psi$ is not required explicitly in (4.9), but only the product $\nabla \psi Q x_k$ is. Therefore, the block upper triangular structure of $\nabla \psi$ and the block diagonal structure of $Q$ yields,

$$
\nabla \psi Q x = \begin{bmatrix} G_{k|k}^T & \ldots & G_{k|k}^T F_{k+N-1|k}^T \ldots F_{k|k}^T \\
\vdots & \ddots & \vdots \\
G_{k+N-2|k}^T & \ldots & G_{k+N-2|k}^T F_{k+N-1|k}^T G_{k+N-1|k}^T \\
\end{bmatrix} \begin{bmatrix} Q x_{k+1|k} \\
\vdots \\
Q x_{k+N-1|k} \\
P x_{k+N|k} \end{bmatrix}
$$

$$
\nabla \psi Q x = \begin{bmatrix} G_{k+N-2|k}^T \left( Q x_{k+N-1|k} + F_{k+N-1|k}^T \left( P x_{k+N|k} \right) \right) \\
\vdots \\
G_{k+N-1|k}^T P x_{k+N|k} \end{bmatrix},
$$

which can be efficiently determined by backward substitution. For diagonal cost matrices $Q$, $R$ and full $P$, the number of FLOPS required to compute the last vector $G_{k+N-1|k}^T P x_{k+N|k}$ is upper bounded by $2(n_x^2 + n_x n_u)$ FLOPS. By not recomputing the term $P x_{k+N|k}$, the second last subvector requires a number of FLOPS upper bounded by $2(n_x^2 + n_x n_u + n_x)$ FLOPS. Since every subvector can be computed from the successive one and by also considering the term $R u_k$ and the nonlinear constraints $n(u_{k+j|k})$, the computational complexity of
computing the gradient step is $O\left(N(n_x^2 + n_x n_u + n_u n_u)\right)$. Since the subsequent steps of Algorithm 2.1 to determine $\rho^{(i)}$ from (2.13) and $t^{(i)}$ from (2.14) have lower complexity, including the computation of the merit function $\phi(t)$ and its derivative $\phi'(t)$, this is the resulting complexity of the algorithm. Note that this complexity is comparable to Nesterov’s accelerated Gradient Method for linear MPC with only box constraints \[108\], $O\left((Nn_u)^2\right)$, but the complexity depends linearly (instead of quadratically) on the prediction horizon, $N$. Further, the complexity of the SQP method depends on the complexity of the QP solver. The Active Set Method for the presented MPC problem requires $O\left((Nn_u)^2\right)$ FLOPS \[52\], while an Interior-Point Method exploiting sparsity of the MPC requires $O\left(N(n_x^3 + n_x^2 n_u)\right)$ FLOPS \[50\].

### 4.2.2 Contractive constraint

The quadratic Lyapunov constraint for stability purposes can also be considered on a state that is not necessarily the last one. In particular, the contractive MPC approach \[41\] reformulates the problem in (4.7) as follows:

$$\begin{align*}
\min_{(x_{k+j+1|k}, u_{k+j|k})} \sum_{j=0}^{N-1} \left\{ \frac{1}{2} x_{k+j|k}^T Q x_{k+j|k} + \frac{1}{2} u_{k+j|k}^T R u_{k+j|k} \right\} \\
+ \frac{1}{2} x_{k+N|k}^T P x_{k+N|k} \\
\text{s.t.} \\
x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}) \quad \forall j \in \{0, \ldots, N-1\} \\
u_{k+j|k} \in [a_{k+j|k}, b_{k+j|k}] \quad \forall j \in \{0, \ldots, N-1\} \\
n(u_{k+j|k}) \leq 0 \quad \forall j \in \{0, \ldots, N-1\} \\
\frac{1}{2} x_{k+\tilde{k}|k}^T P x_{k+\tilde{k}|k} \leq c_k,
\end{align*}$$

(4.10)

where the index $\tilde{k} \in \{1, 2, \ldots, N\}$ and the term $c_k$ depend on the current time step $k$ and are computed before solving the optimization program. By neglecting as before for the sake of presentation the nonlinear constraint $n(u_{k+j|k}) \leq 0$, the problem is rewritten in the compact form

$$\begin{align*}
\min_{u_k, y_k, a_k, y_k, b_k, y_k, c} \quad & \frac{1}{2} \psi(u_k)^T Q \psi(u_k) + \frac{1}{2} u_k^T R u_k =: J(u_k) \\
\text{s.t.} \\
& -u_k + a_k + \frac{1}{2} \text{diag}(y_k, a) y_{k, a} = 0 \\
& u_k - b_k + \frac{1}{2} \text{diag}(y_k, b) y_{k, b} = 0 \\
& \frac{1}{2} \psi_k(u_k)^T P \psi_k(u_k) - c_k + \frac{1}{2} y_{k, c}^2 = 0.
\end{align*}$$

(4.11)
Note that, by defining the vector \( q_k \in \mathbb{R}^{N_nu} \) as the derivative of the last constraint, the matrix inversion in (4.8) does not change. Further, the vector \( q_k \) will be partially sparse, since for the causality of the dynamics the last \((N - \tilde{k})n_u\) components will be identical to zero.

### 4.3 Computational considerations

The algorithms proposed in Chapters 2 and 3 introduce some design parameters to be tuned before implementing the algorithm. The line search constants in (2.14a) or (3.3) have no significant impact on the convergence of the algorithm. In this work we consider \( \sigma = \sigma_1 = 0.3 \). For the safeguarded quadratic interpolation in (3.3), a typical choice is \( \tau_1 = 0.01 \) and \( \tau_2 = 0.99 \).

The gradient step size \( \alpha \) is more important for the practical performance of the algorithm and its design is discussed in Section 4.3.1. After that, we consider the stopping criteria in Section 4.3.2. Finally, in Section 4.3.3 an automatic code generation tool for MPC problems is presented.

#### 4.3.1 Tuning of the gradient step size \( \alpha \)

The theoretical results presented in Section 2.4 show that guarantees on the convergence speed can be derived only when the iterate is close to the solution and the correct active set is determined, achieving linear rate. In fact, by Theorem 2.1 linear convergence is obtained via a specific tuning of \( \alpha \), based on the maximum eigenvalue of Hessian of the Lagrangian at the optimum. This would allow for \( t = 1 \) as shown in Section 2.4.3. On the other hand, such a value is in general unknown a priori, and an approximation based on the Hessian of the current iterate can be overly expensive to compute. From our numerical experience, we recommend instead to set an \( \alpha \) (not necessarily linked to the problem Lagrangian) and let the line search variable \( t \) become smaller than 1 in the line search. In fact, this approach achieves a comparable convergence speed to the linear rate obtained with the specific \( \alpha \) of Section 2.4.3 and \( t = 1 \). Note that setting \( \alpha \) to a value too large may overly increase the primal and dual variables and make the algorithm fail.

Besides assigning a constant value to the gradient step size \( \alpha \), another possibility is to allow it to vary in a range. In fact, from (2.4) we can see that the gradient step size \( \alpha \) specifies, before the projection,
length of the step $d_z$. In the approach shown next, we consider taking a smaller or a larger step based on the local geometry of the objective and the constraint functions.

This is typically considered by the trust-region approach, which is an alternative to line search method to obtain global convergence of SQP (see for example [99, 118] and the references therein). The method is based on a comparison of the actual and predicted reduction obtained by a specific function. In our case, we consider the merit function in (2.11) or (3.1), and the predicted decrease of the merit function is given by the following linear approximation:

$$\text{pred}(\alpha(i)) := \phi(0) - \left(\phi(0) + t(i)\phi'(0)\right) = -t(i)\phi'(0), \quad (4.12)$$

and, for the non-smooth penalty function in (3.1), the derivative is replaced by the directional derivative. From the tuning of the penalty parameter $\rho(i)$ in Lemmas 2.4 and 3.1 we know that $\text{pred}(\alpha(i))$ is always non-negative (null when converged). Note that we can write $\text{pred}$ as a function of the gradient step size $\alpha(i)$, since the (directional) derivative depends on $\alpha(i)$ via the step $d_z(i)$.

After the line search, we can measure the actual reduction of the merit function:

$$\text{ared}(\alpha(i)) := \phi(0) - \phi(t(i)), \quad (4.13)$$

where, similarly to before, the second term on the right-hand side depends on the gradient step size $\alpha(i)$. The line search conditions in (2.14a) and (3.3) guarantee that $\text{ared}(\alpha(i))$ is always non-negative (null when converged).

Then, we can compute a ratio $\text{ratio}(\alpha(i)) := \frac{\text{ared}(\alpha(i))}{\text{pred}(\alpha(i))}$. If the ratio is close to 1, then there is good agreement between the actual and predicted reduction and, if possible, we increase the next $\alpha(i+1)$. If the ratio is close to zero, then, if possible, we reduce the next $\alpha(i+1)$. Otherwise, we take $\alpha(i+1) = \alpha(i)$. Designing a range of admissible $\alpha$ (instead of letting them growing unbounded) allows us to maintain the validity of the theoretical results presented in Chapters 2 and 3.

A reasonable value of the step size $\alpha$, which can possibly work as an upper bound for the range discussed before, can be derived for non-linear MPC problems via an offline procedure. The solution of the MPC problem in (4.6) is in fact expected to converge to the origin as $k \to \infty$, where we assume that the inequality constraints be inactive.
Therefore, the associated dual variables need to be null, because of the complementarity condition in (2.3). This implies that in the origin, the Hessian of the Lagrangian in (2.2) coincides with the Hessian of the objective function. Therefore, the Hessian around the origin of the objective function of the MPC problem in (4.7) can be computed offline, together with its maximum eigenvalue considered in Theorem 2.1. The lower bound can then be taken as a fraction of this value.

4.3.2 Stopping criteria

From our numerical experience we observe that the convergence rate might be faster than linear when the solution is far from optimality, albeit theoretically the convergence rate is unknown, and then slow down once it is close to the optimal point. Analogously to standard convex optimization theory, the linear rate factor depends on second-order information and, in our case, a large condition number of the Hessian of the Lagrangian at the solution can yield a slow linear convergence in practice [12]. This means that most of the iterates might be employed to reach the desired tolerance. Thus, it is good practice to add a reasonable limit on the maximum number of iterations (e.g., 4000 iterations).

Besides a maximum number of iterations, we consider standard convergence conditions to terminate the algorithm. Clearly an exact null $d_z^{(i)}$ as in Algorithms 2.1 or 3.1 for terminating the optimization routine is too restrictive, therefore we relax the condition to a small enough value. Leveraging (2.9), if feasibility of the constraint is achieved, by bounding appropriately $\|d_z\|_2$ we can equivalently impose that the iterate $z^{(i)}$ is a KKT point for the nonlinear problem with a desired tolerance. Another condition involves the derivative of the augmented Lagrangian function, $\phi'(0)$. By (2.15), setting a negative lower bound to the derivative $\phi'(0)$ corresponds to checking that the iterate $\|d_z\|_2$ is small.

A failure of the method (e.g., if the problem is infeasible or the Jacobians are incorrect) breaks the line search procedure. In order to maximize speed we do not run any feasibility test before solving the problem and we use the failure of the line search as an indicator of well-posedness of the problem.

In the non-smooth approach of Chapter 3, in some situations we observed that a large value of penalty parameter $\rho^{(i)}$ makes the line search fail at low tolerances ($< 10^{-5}$). The problem does not seem to
be related to the Maratos effect, since a second-order correction yields the same failure. A more practically effective solution is obtained by resetting the penalty parameter $\rho^{(i)}$ based on the current lower-bound $\hat{\rho}^{(i)}$ derived in Lemma 3.1. In fact, since the rule in (3.4) implies that $\rho^{(i)} = \max_{l \leq i} \hat{\rho}^{(l)}$, this procedure reduces $\rho^{(i)}$ and consequently the merit function and, in the observed cases, it allows the method to successfully converge.

4.3.3 Automatic code generation with FalcOpt

We provide a simple-to-use tool called FalcOpt (First-order Algorithm via Linearization of Constraints for OPTimization) for code-generation of Algorithms 2.1 and 3.1 available on https://github.com/torrisig/FalcOpt together with a Wiki and examples [121]. With FalcOpt a tailored nonlinear MPC solver can be automatically generated for problems of the form:

$$\min_{(x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}} \sum_{j=0}^{N-1} \ell(x_{k+j+1|k}, u_{k+j|k}) + \ell_N(x_{k+N|k})$$

s.t.

$$x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}) \quad \forall j \in \{0, \ldots, N - 1\}$$
$$u_{k+j|k} \in [a_{k+j|k}, b_{k+j|k}] \quad \forall j \in \{0, \ldots, N - 1\}$$
$$n(u_{k+j|k}) \leq 0 \quad \forall j \in \{0, \ldots, N - 1\}$$
$$\frac{1}{2} x_{k+N|k}^T P x_{k+N|k} \leq c_k,$$

(4.14)

where the last constraint can be replaced by a contractive constraint.

FalcOpt is written in MATLAB and generates low-level, embeddable C code. The code-generation procedure is run only once and offline, during the design phase of the MPC. The generated code is library-free and heavily exploits sparsity in the problem data. Furthermore, a MEX interface can be automatically generated for testing and simulation in MATLAB.
5 A heuristic to accelerate the practical convergence speed

In Chapters 2 and 3 we have shown the convergence of the proposed projected gradient and constraint linearization method to a KKT point for any initialization $z^{(0)}$. This is obtained by a step size $t \in (0, 1]$, determined via a merit function treatment.

In this chapter we investigate a heuristic method to accelerate the practical convergence of the proposed algorithm for a specific class of MPC problems. In particular, the method applies to a general nonlinear dynamics and only box constraints on the input. Since the box constraints are linear, Algorithms 2.1 and 3.1 significantly simplify. By starting from a feasible input, any iterate generated by the algorithm stays feasible, thus the augmented Lagrangian function reduces to the objective function of the problem. For this particular class of problems, an improvement in the practical convergence is observed for an initialization of the gradient step based on the previously determined iterate. This approach is not theoretically guaranteed to converge, but it yields an interesting practical convergence speed.

The chapter is organized as follows: in Section 5.1 we present the algorithm and the heuristic, in Section 5.2 we discuss its computational efficiency and in Section 5.3 we present possible extensions to other Model Predictive Control problems. Section 5.4 contains some remarks on the computational results presented in Parts II and III.
5.1 Application to Model Predictive Control

We consider the following general nonlinear discrete-time optimal control problem without state constraints:

\[
\begin{align*}
\min_{(x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}} & \sum_{j=0}^{N-1} \left\{ \frac{1}{2} x_{k+j|k}^\top Q x_{k+j|k} + \frac{1}{2} u_{k+j|k}^\top R u_{k+j|k} \right\} \\
& + \frac{1}{2} x_{k+N|k}^\top P x_{k+N|k} \\
\text{s.t.} & \quad x_{k+j+1|k} = f \left( x_{k+j|k}, u_{k+j|k} \right) \quad \forall j \in \{0, \ldots, N - 1\} \\
& \quad u_{k+j|k} \in [a_{k+j|k}, b_{k+j|k}] \quad \forall j \in \{0, \ldots, N - 1\},
\end{align*}
\]

in which \( a_k < b_k \in \mathbb{R}^{n_u} \) and \( Q, P, R \succeq 0 \).

For ease of notation, as in the previous chapter we define the vectors
\[
\begin{align*}
& u_k := [u_k; \ldots; u_{k+N-1|k}] \\
& a_k := [a_{k|k}; \ldots; a_{k+N-1|k}] \quad \text{and} \quad b_k := [b_{k|k}; \ldots; b_{k+N-1|k}],
\end{align*}
\]
and the corresponding state evolution
\[
x_k := [x_{k+1|k}; \ldots; x_{k+N|k}],
\]
and stack the state and input cost matrices
\[
Q = \text{blockdiag} (Q, \ldots, Q, P) \quad \text{and} \quad R = \text{blockdiag} (R, \ldots, R).
\]

Then the MPC problem in (5.1) in compact form is

\[
\begin{align*}
\min_{x_k, u_k} & \quad \frac{1}{2} x_k^\top Q x_k + \frac{1}{2} u_k^\top R u_k \\
\text{s.t.} & \quad x_k = \psi (u_k) \\
& \quad u_k \in [a_k, b_k],
\end{align*}
\]

where for a fixed initial state \( x_k \) and given a sequence of inputs \( u_k \) as argument, \( \psi (u_k) \) is the predicted state sequence \( x_k \) according to the nonlinear dynamics \( f \). The dynamics is then embedded into the objective function to obtain the following:

\[
\begin{align*}
\min_{u_k} & \quad \frac{1}{2} \psi (u_k)^\top Q \psi (u_k) + \frac{1}{2} u_k^\top R u_k =: J(u_k) \\
\text{s.t.} & \quad u_k \in [a_k, b_k],
\end{align*}
\]

We notice that, as the constraints in (5.3) are linear, the definition of an augmented Lagrangian in (2.11) is not required. In fact, the augmented Lagrangian weights optimality and feasibility when the iterates are infeasible. Here the constraints are linear, therefore any iterate produced by Algorithms 2.1 and 3.1 is feasible for the original
5.1 Application to Model Predictive Control

nonlinear problem. There is no need to introduce squared-slack variables, since the feasible set is a box and the projection onto such set is already easy-to-compute. By expressing the inequality constraints in (5.3) as \( g(u_k) \leq 0 \), if the initial vector \( u_k^{(0)} \) is feasible, then the initialization \( \rho = 0 \) implies that \( g(u_k^{(0)}) + s^{(0)} = 0 \), with the slack variable \( s^{(0)} \) defined according to (2.12). As far as \( g(u_k^{(i)}) + s^{(i)} = 0 \), the test in Lemma 2.4 is passed, thus \( \rho^{(i)} \) is not increased and remains null. Therefore, by (2.12) also the successive slack variable \( s^{(i+1)} \) is such that \( g(u_k^{(i+1)}) + s^{(i+1)} = 0 \). A similar conclusion on \( \rho^{(i)} \) holds true also for the approach in Chapter 3.

Since the augmented Lagrangian coincides with the objective function, the dual iterates in (4.4) are no longer required. Thus, the primal iterates can be determined without computing the dual variables associated to the projection in (2.9). The dual variable \( \lambda^* \) can be derived, if needed, a-posteriori.

We now focus on the line search to determine the step size \( t^{(i)} \). The line search in (2.14) will apply uniquely to the objective function, similarly to the gradient step size \( \alpha^{(i)} \). To avoid the redundancy, we take \( t^{(i)} = 1 \) and consider \( \alpha^{(i)} \) as the only tunable step size. Hence:

\[
 u_k^{(i+1)} := u_k^{(i)} + d_{u,k}^{(i)}. \tag{5.4}
\]

The heuristic that we propose changes the way in which the increment \( d_{u,k}^{(i)} \) is computed. In fact, according to Algorithm 2.1 the update in (2.4) for the problem in (5.3) would be:

\[
 d_{u,k}^{(i)} := \Pi_{[a_k, b_k] - u_k^{(i)}} \left( -\alpha^{(i)} \nabla J(u_k^{(i)}) \right), \tag{5.5}
\]

with:

\[
 [a_k, b_k] - u_k^{(i)} = \left\{ d_{u,k} \in \left[ a_k - u_k^{(i)}, b_k - u_k^{(i)} \right] \subseteq \mathbb{R}^{N_{n_u}} \right\}.
\]

Note that, since the constraint in (5.3) is linear, combining (5.4) and (5.5) recovers the standard Gradient Method. In fact, because of the properties of the projection:

\[
 u_k^{(i+1)} := u_k^{(i)} + \Pi_{[a_k, b_k] - u_k^{(i)}} \left( -\alpha^{(i)} \nabla J(u_k^{(i)}) \right) = \Pi_{[a_k, b_k]} \left( u_k^{(i)} - \alpha^{(i)} \nabla J(u_k^{(i)}) \right). \tag{5.6}
\]
5 A heuristic to accelerate the practical convergence speed

As shown in (2.6), the increment \( d_{u,k}^{(i)} \) in (5.5) can be equivalently computed as a gradient step with null initialization \( d_{u,k,ini}^{(i)} = 0 \) of the following Quadratic Program:

\[
\begin{align*}
\min_{d_{u,k}} & \quad \frac{1}{2} d_{u,k}^T H \left( u_k^{(i)} \right) d_{u,k} + \nabla J \left( u_k^{(i)} \right)^T d_{u,k} =: q(d_{u,k}, u_k^{(i)}) \\
\text{s.t.} & \quad u_k^{(i)} + d_{u,k} \in [a_k, b_k],
\end{align*}
\]

whose objective function is a second order approximation of the Lagrangian of the NLP in (5.3) and the constraints are obtained by linearization of the original constraint. The matrix \( H(u_k^{(i)}) \) approximates the Hessian of the Lagrangian of (5.3) which, since the problem has only linear constraints, coincides with the Hessian of the objective function. In the following, we will consider the Gauss-Newton approximation, which is particularly suited for Model Predictive Control problems with quadratic cost [119, 45]. Thus:

\[
\begin{align*}
H(u_k^{(i)}) & := \nabla \psi(u_k^{(i)}) Q \nabla \psi(u_k^{(i)})^T + \mathcal{R} \succeq 0, \\
\nabla J(u_k^{(i)}) & = \nabla \psi(u_k^{(i)}) Q x_k^{(i)} + \mathcal{R} u_k^{(i)}. 
\end{align*}
\]

with \( x_k^{(i)} = \psi(u_k^{(i)}) \). Note that \( H(u_k^{(i)}) \succeq 0 \) follows from \( Q \succeq 0 \) and \( \mathcal{R} \succeq 0 \), and if at least one of the cost matrices is positive definite, this implies that \( H(u_k^{(i)}) \succ 0 \). The method presented in [79] consists of determining the optimum by a Newton’s type method and exploiting the structure given box constraints. Global convergence then follows from a trust-region approach. The heuristic presented here instead consists of taking an initialization \( d_{u,k,ini}^{(i)} \) not necessary null. In particular:

\[
d_{u,k}^{(i)} := \Pi_{[a_k, b_k]} - u_k^{(i)} \left( d_{u,k,ini}^{(i)} - \alpha^{(i)} \left( H(u_k^{(i)}) d_{u,k,ini}^{(i)} + \nabla J(u_k^{(i)}) \right) \right),
\]

where the initialization \( d_{u,k,ini}^{(i)} \) is defined based on the following test on the previous iterate \( d_{u,k}^{(i-1)} \):

\[
d_{u,k,ini}^{(i)} := \begin{cases} 
\Pi_{[a,b]} - u^{(i)} \left( d_{u,k}^{(i-1)} \right) & \text{if } q \left( \Pi_{[a_k,b_k]} - u^{(i)} \left( d_{u,k}^{(i-1)} \right), u_k^{(i)} \right) < 0, \\
0 & \text{otherwise.}
\end{cases}
\]

(5.10)
Algorithm 5.1 Proposed Heuristic for nonlinear MPC problem in (5.3)

\textbf{Initialize} \( i \leftarrow 0 \) and \( u_k^{(0)} \in \mathbb{R}^{N_u} \)

\textbf{repeat}
  \begin{algorithmic}
    \STATE \textbf{if} \( i = 0 \) \textbf{then}
    \STATE \textbf{Set} \( d_u^{(0)} \) from (5.10)
    \STATE \textbf{else}
    \STATE \textbf{Determine} \( d_u^{(i)} \) from (5.10)
    \STATE \textbf{end if}
    \STATE \textbf{Compute} \( d_u^{(i)} \) as in (5.9) with suitable gradient step size \( \alpha^{(i)} \in \mathbb{R}_{>0} \), e.g., with the line search in Algorithm 5.2
    \STATE \textbf{if} \( d_u^{(i)} = 0 \) \textbf{then}
    \STATE \textbf{Set} \( u_k^* = u_k^{(i)} \) and \textbf{STOP}
    \STATE \textbf{end if}
    \STATE \textbf{Update} \( u_k^{(i+1)} \) as in (5.4)
    \STATE \( i \leftarrow i + 1 \)
  \end{algorithmic}
\textbf{until} Convergence
\textbf{return} \( u_k^* \)

This approach is motivated by the following: the SQP approach in (1.4) would determine \( d_u \) as the minimum of the function \( q(d_u, u_k) \) with respect to \( d_u \), subject to the input constraint. Choosing an initialization \( d_u, ini = 0 \) clearly yields \( q(d_u, ini, u_k) = 0 \). If the previously computed \( d_u^{(i-1)} \) is such that \( q(d_u^{(i)}, u_k^{(i)}) < 0 \), then this initialization heuristic aims to accelerate the convergence.

Our proposed approach is summarized in Algorithm 5.1

5.1.1 Tuning of the gradient step size \( \alpha \)

Since the method proposed in (5.9) is an heuristic, there is no specific value of \( \alpha \) that guarantees convergence. On the other hand, some indication is given by the Gradient Method in (5.6).

One approach is to take a step size \( \alpha = 1/L(u_k) \), where \( L(\cdot) \) is the local Lipschitz constant of the gradient of the objective function \( q(\cdot, u_k) \). Computing such Lipschitz constant amounts to determining
5 A heuristic to accelerate the practical convergence speed

the maximum eigenvalue of the Hessian $H(u_k)$. As it will be shown in the next section, the Hessian is never explicitly needed (only its product with $d_{u,k}$), therefore this approach may be overly expensive.

A second approach is to determine the step size $\alpha$ via a standard Armijo rule, which requires only simple function evaluations [7, 77]. In fact, given a parameter $\sigma \in (0, \frac{1}{2})$, the Armijo rule inequality is:

$$q(d_{u,k}, u^{(i)}_k) - q(d_{u,k,ini}, u^{(i)}_k) \leq \sigma \nabla q(d_{u,k,ini}, u^{(i)}_k)^\top (d_{u,k} - d_{u,k,ini}).$$

(5.11)

If $\alpha$ were null, then $d_{u,k} = d_{u,k,ini}$. The right-hand side of (5.11) can be proven to be negative by the following lemma.

**Lemma 5.1.** For any $\alpha^{(i)} > 0$, we have:

$$\nabla q(d^{(i)}_{u,k,ini}, u^{(i)}_k)^\top (d^{(i)}_{u,k} - d^{(i)}_{u,k,ini}) < 0 \quad \text{if } \nabla q \neq 0.$$

□

**Proof.** We use a simplified notation where we omit the arguments of the function $q$ and the dependence on the index $i$. It follows from (5.9) and the properties of the projection operator that:

$$d_{u,k} - d_{u,k,ini} = \Pi_{[a,b] - u_k - d_{u,k,ini}} (-\alpha \nabla q).$$

(5.12)

Since $u_k + d_{u,k,ini} \in [a, b]$, then

$$m := a - u_k - d_{u,k,ini} \leq 0$$

$$M := b - u_k - d_{u,k,ini} \geq 0,$$

which implies the following three cases for each component $j$:

$$\Pi_{[m_j,M_j]} (-\alpha \nabla q_j) = \begin{cases} -\alpha \nabla q_j & \text{if } j \in J_1 := \{ j \mid -\alpha \nabla q_j \in (m_j, M_j) \} \\ m_j & \text{if } j \in J_2 := \{ j \mid -\alpha \nabla q_j \leq m_j \leq 0 \} \\ M_j & \text{if } j \in J_3 := \{ j \mid -\alpha \nabla q_j \geq M_j \geq 0 \}. \end{cases}$$
By \((5.12)\), we get:

\[
\nabla q(d_{u,k,ini}, u_k) = \nabla q^\top \Pi_{[m,M]} (-\alpha \nabla q) = \sum_{j=1}^{N_{nu}} \nabla q_j \Pi_{[m_j,M_j]} (-\alpha \nabla q_j)
\]

\[
= \sum_{j \in J_1} -\alpha \nabla q_j^2 + \sum_{j \in J_2} \nabla q_j m_j + \sum_{j \in J_3} \nabla q_j M_j
\]

\[
\leq \sum_{j \in J_1} -\alpha \nabla q_j^2 + \sum_{\{j \in J_2|\nabla q_j \leq 0\}} \nabla q_j m_j + \sum_{\{j \in J_3|\nabla q_j \geq 0\}} \nabla q_j M_j
\]

\[
\leq \sum_{j \in J_1} -\alpha \nabla q_j^2 + \sum_{\{j \in J_2|\nabla q_j \leq 0\}} -\alpha \nabla q_j^2 + \sum_{\{j \in J_3|\nabla q_j \geq 0\}} -\alpha \nabla q_j^2
\]

\[
< 0 \quad \text{if } \nabla q \neq 0.
\]

Note that, by \((5.10)\) and \((5.11)\), the condition \(\nabla q(d_{u,k,ini}, u_k) = 0\) implies \(d_{u,k} = d_{u,k,ini} = 0\), thus that the current iterate is a critical point for \((5.1)\).

Whenever the test on \(d_{u,k}\) in \((5.11)\) fails, the gradient step size, \(\alpha\), is reduced by a factor \(\beta < 1\) and a new \(d_{u,k}\) is computed. Based on the definition of \(q(d_{u,k}, u_k)\) in \((5.7)\), \((5.11)\) is equivalent to:

\[
\frac{1}{2} d_{u,k} H(u_k) d_{u,k} + \nabla J(u_k) \nabla d_{u,k} - \left( \frac{1}{2} d_{u,k,ini} H(u_k) d_{u,k,ini} + \nabla J(u_k) \nabla d_{u,k,ini} \right)
\]

\[
\leq \sigma \left( H(u_k) d_{u,k,ini} + \nabla J(u_k) \right)^\top (d_{u,k} - d_{u,k,ini}).
\]

Hence, besides scalar products, only evaluations of \(H(u_k) d_{u,k,ini}\), \(\nabla J(u_k)\) and \(H(u_k) d_{u,k}\) are required. The first two terms can be computed before starting the Armijo rule line search. In the following section we will show how these terms can be computed efficiently.

The steps of the line search are recapped in Algorithm 5.2.

### 5.2 Computational considerations

The algorithm step in \((5.9)\) can be computed efficiently by exploiting sparsity of the matrices involved. The term \(\nabla J(u_k)\) is computed as in Section 4.2.1.
Algorithm 5.2 Line search based on Armijo Rule \( (5.11) \)

Given constants \( \sigma \in (0, \frac{1}{2}) \), \( \beta < 1 \), current \( u^{(i)}, d_{u,k,ini} \in \mathbb{R}^{Nn_u} \) and tentative gradient step size \( \alpha^{(i)} \in \mathbb{R}_{>0} \)

**COMPUTE** \( H(u^{(i)})d_{u,k,ini} \) and \( \nabla J(u^{(i)}) \)

**repeat**
- **COMPUTE** \( d_{u,k} \) with step size \( \alpha^{(i)} \) as in \( (5.9) \)
- **DETERMINE** \( H(u^{(i)})d_{u,k} \)
  - if \( (5.11) \) holds **then**
    - Set \( d^{(i)}_{u,k} = d_{u,k} \) and Stop
  - else
    - \( \alpha^{(i)} \leftarrow \beta \alpha^{(i)} \)
  - end if
**until** \( (5.11) \) holds
**return** \( d^{(i)}_{u,k} \)

Then, we note that the Hessian \( H(u_k^{(i)}) \) is never explicitly required, only the product \( H(u_k^{(i)})d_{u,k} \) or \( H(u_k^{(i)})d_{u,k,ini} \). We will generically indicate this product as \( H(u_k^{(i)})v_k \). By the definition in \( (5.8) \), this product can be computed as follows. The matrix \( \nabla \psi(u_k^{(i)})^\top \) is block lower triangular matrix and every row group can be calculated iteratively from the one above:

\[
\mathbf{\hat{v}}_k = \begin{bmatrix}
\mathbf{\hat{v}}_{k|k} \\
\mathbf{\hat{v}}_{k+1|k} \\
\mathbf{\hat{v}}_{k+2|k} \\
\vdots 
\end{bmatrix} := \nabla \psi(u_k^{(i)})^\top \mathbf{v}_k = \begin{bmatrix}
G_{k|k} v_{k|k} \\
F_{k+1|k} \mathbf{\hat{v}}_{k|k} + G_{k+1|k} v_{k+1|k} \\
F_{k+2|k} \mathbf{\hat{v}}_{k+1|k} + G_{k+2|k} v_{k+2|k} \\
\vdots 
\end{bmatrix}.
\]

The computational complexity of this operation is upper bounded by \( 2N(n^2 + n_xn_u) \) FLOPS. Then, the product \( H(u_k^{(i)})v_k^{(i)} \) is:

\[
H(u_k^{(i)})v_k^{(i)} = \nabla \psi(u_k^{(i)}) \mathbf{Q}\mathbf{\hat{v}}_k^{(i)} + \mathbf{R}v_k^{(i)},
\]

where, again, the structure of \( \nabla \psi(u_k^{(i)}) \) and of the block diagonal matrices \( \mathbf{Q} \) and \( \mathbf{R} \) is exploited. The operation has the same structure of \( (4.9) \), therefore for diagonal cost matrices \( \mathbf{Q}, \mathbf{R} \) and full \( \mathbf{P} \), the computational complexity of the algorithm is \( \mathcal{O}(N(n^2 + n_xn_u)) \), i.e., the same of Algorithms 2.1 and 3.1.
5.3 A further extension

Algorithm [5.1] can also efficiently tackle more complicated problems than (5.1). Tracking problems can also be considered, by just modifying the quadratic cost to penalize the deviation from the desired reference. In Chapter 8 we will consider an offset-free Model Predictive Control, tracking a desired output and input reference. The problem can be written in the following form:

$$\begin{align*}
\min_{(x_{k+j|k}, y_{k+j|k})} & \sum_{j=0}^{N-1} \left\{ \frac{1}{2} (y_{k+j|k} - \bar{y}_{k+j|k})^\top Q_y (y_{k+j|k} - \bar{y}_{k+j|k}) \\
& + \frac{1}{2} (u_{k+j|k} - \bar{u}_{k+j|k})^\top R (u_{k+j|k} - \bar{u}_{k+j|k}) \right\} \\
\text{s.t.} & \quad x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}) \quad \forall j \in \{0, \ldots, N-1\} \\
& \quad y_{k+j|k} = h(x_{k+j|k}) \quad \forall j \in \{0, \ldots, N\} \\
& \quad u_{k+j|k} \in [a_{k+j|k}, b_{k+j|k}] \quad \forall j \in \{0, \ldots, N-1\},
\end{align*}$$

(5.13)

in which $a_{k+j|k} < b_{k+j|k} \in \mathbb{R}^{n_u}$ $\forall k, j$, the cost matrices $Q_y, R, P_y \succeq 0$ and $\bar{y}_{k+j|k} \in \mathbb{R}^{ny}$ and $\bar{u}_{k+j|k} \in \mathbb{R}^{n_u}$ are the desired output and input references. By using the same notation of Section 5.1 the problem in a compact form is:

$$\begin{align*}
\min_{y_k, x_k, u_k} & \frac{1}{2} (y_k - \bar{y}_k)^\top Q_y (y_k - \bar{y}_k) + \frac{1}{2} (u_k - \bar{u}_k)^\top R (u_k - \bar{u}_k) \\
\text{s.t.} & \quad x_k = \psi(u_k) \\
& \quad y_k = \eta(x_k) \\
& \quad u_k \in [a_k, b_k],
\end{align*}$$

with predicted output sequence $y_k := [y_{k+1|k}; \ldots; y_{k+N|k}]$, output cost $Q_y := \text{blockdiag}(Q_y, \ldots, Q_y, P_y)$, output and input reference sequence $\bar{y}_k := [\bar{y}_{k+1|k}; \ldots; \bar{y}_{k+N|k}]$ and $\bar{u}_k = [\bar{u}_{k|k}; \ldots; \bar{u}_{k+N-1|k}]$ and the function $\eta(x_k)$ that gives the predicted output sequence according to the output equation $h$. The equality constraints are included into the
5 A heuristic to accelerate the practical convergence speed

objective function, thus deriving the following problem:

\[
\min_{\mathbf{u}_k} \frac{1}{2} (\eta(\psi(\mathbf{u}_k)) - \bar{y}_k)^\top Q_y (\eta(\psi(\mathbf{u}_k)) - \bar{y}_k) + \frac{1}{2} (\mathbf{u}_k - \bar{u}_k)^\top \mathcal{R} (\mathbf{u}_k - \bar{u}_k)
\]

s.t. \( \mathbf{u}_k \in [\mathbf{a}_k, \mathbf{b}_k] \),

(5.14)

We indicate the cost function of (5.14) as \( J(\mathbf{u}_k) \), where for ease of notation we omit the dependence on the output and input reference \( \bar{y}_k \) and \( \bar{u}_k \). The gradient of \( J(\mathbf{u}_k) \) and the Gauss-Newton approximation of the Hessian read as follows:

\[
H(\mathbf{u}_k^{(i)}) := \nabla \psi(\mathbf{u}_k^{(i)}) \nabla \eta(\mathbf{x}_k^{(i)}) Q_y \nabla \eta(\mathbf{x}_k^{(i)})^\top \nabla \psi(\mathbf{u}_k^{(i)})^\top + \mathcal{R} \succeq 0,
\]

\[
\nabla J(\mathbf{u}_k^{(i)}) = \nabla \psi(\mathbf{u}_k^{(i)}) \nabla \eta(\mathbf{x}_k^{(i)}) Q_y (\mathbf{y}_k^{(i)} - \bar{y}_k) + \mathcal{R} (\mathbf{u}_k^{(i)} - \bar{u}_k),
\]

(5.15)

with \( \mathbf{x}_k^{(i)} = \psi(\mathbf{u}_k^{(i)}) \) and \( \mathbf{y}_k^{(i)} = \eta(\mathbf{x}_k^{(i)}) \). Note that the matrix \( \nabla \eta(\mathbf{x}_k^{(i)}) = \text{blockdiag}(H_1^\top, \ldots, H_N^\top) \) is block diagonal and this allows one to compute the vectors \( \nabla J(\mathbf{u}_k^{(i)}) \) and \( H(\mathbf{u}_k^{(i)})\mathbf{v}_k \) efficiently, with \( \mathbf{v}_k \) being either \( \mathbf{d}_{u,k}^{(i)} \) or \( \mathbf{d}_{u,k,\text{ini}}^{(i)} \). In particular:

\[
\hat{\mathbf{v}}_k = \begin{bmatrix}
\hat{v}_{k|k} \\
\hat{v}_{k+1|k} \\
\vdots \\
\hat{v}_{k+2|k}
\end{bmatrix} := \nabla \eta(\mathbf{x}_k) ^\top \nabla \psi(\mathbf{u}_k) ^\top \mathbf{v}_k
\]

\[
= \begin{bmatrix}
H_{k+1|k} G_{k|k} \mathbf{v}_{k|k} \\
H_{k+2|k} \left( F_{k+1|k} \mathbf{v}_{k|k} + G_{k+1|k} \mathbf{v}_{k+1|k} \right) \\
H_{k+3|k} \left( F_{k+2|k} \mathbf{v}_{k+1|k} + G_{k+2|k} \mathbf{v}_{k+2|k} \right) \\
\vdots
\end{bmatrix},
\]

which is computed in \( 2N(n_x^2 + n_x n_u + n_x n_y) \) FLOPS. Then:

\[
H(\mathbf{u}_k^{(i)})\mathbf{v}_k^{(i)} = \nabla \psi(\mathbf{u}_k^{(i)}) \nabla \eta(\mathbf{x}_k^{(i)}) Q_y \hat{\mathbf{v}}_k^{(i)} + \mathcal{R} \mathbf{v}_k^{(i)},
\]

is computed by exploiting the block diagonal structure of \( \nabla \eta(\mathbf{x}_k^{(i)}) \), \( Q_y \) and the block lower triangular structure of \( \nabla \psi(\mathbf{u}_k^{(i)}) \). Therefore, the resulting computational complexity of the algorithm is \( \mathcal{O} \left( N(n_x^2 + n_x n_u + n_x n_y) \right) \) FLOPS.

Algorithm 5.1 is directly written in C code for the specific examples. Future work includes integrating the algorithm in FalcOpt to enable automatic code generation.

In Parts II and III we present the computational results of the proposed algorithms on several numerical examples.
5.4 Benchmarks for comparison

In Parts II and III we present the computational results of the proposed algorithms on several numerical examples.

The proposed algorithms are run using FalcOpt, as presented in Section 4.3.3. Algorithm 5.1 is directly written in C code for the specific examples. Future work includes integrating the algorithm in FalcOpt to enable automatic code generation.

The performance of the proposed algorithms is compared with other commercial software. As a Sequential Quadratic Programming solver we use SNOPT \[58, 59\], running in Fortran and interfaced to MATLAB, and the computational time is measured internally by the solver. We note though that this solver is not specifically tailored for Model Predictive Control problems.

As an Interior-Point Method solver we use FORCES Pro \[49\]. Also this software allows the user to automatically generate C code to solve convex and nonconvex problems, together with the MEX interface to MATLAB. In addition, the generated C code can be specifically tailored for the most common embedded devices. We prefer this software to other mature solvers such as IPOPT \[1, 133\], KNITRO \[28, 29\], LOQO \[131\], for two reasons. First, because similarly to FalcOpt, it generates library-free C code, easily portable on embedded devices. Further, the internal algorithm also heavily exploits the structure of the Model Predictive Control problem, making it a fair benchmark for comparison.

We use two variants of FORCES Pro: the low-level interface is more suitable for highly parametric problems such as the ones arising in the Real-Time Iteration, and it only deals with convex problems (at most quadratically constrained). The high-level interface allows one to solve nonlinear problems (in the following it will be indicated as FORCES Pro NL).

The computational times presented in Parts II and III are relative to an off-the-shelf Windows PC with processor Intel Core i7-3740QM 2.70Ghz.
Part II

Application: control of a centrifugal compression system
6 Compression systems for industrial applications

The algorithms presented in Part I allow one to solve efficiently a variety of MPC problems. As an application of these techniques we consider industrial compression systems, typically employed for gas transportation in pipelines and in the chemical industry.

The control objective is to provide gas at desired conditions (pressure and mass flow rate) without the occurrence of unstable phenomena in the compressor. The state-of-the-art control has been based on PI regulators for decades but, recently, Model Predictive Control approaches have gained significant interest both in academic and industrial applications.

In this chapter we introduce the relevant literature on the standard and MPC control of compression systems. For the former, we present a typical PI based decoupled control. For the latter, a simple mathematical model capturing the physics is derived from the relevant thermodynamic relations.

In the next chapters we will design two MPC controllers. In Chapter 7 we consider a contractive nonlinear MPC formulation that ensures asymptotic stability of the closed-loop system by imposing the decrease of a quadratic Lyapunov function via an additional constraint. We discuss recursive feasibility and estimate the region of attraction. This approach can handle at most asymptotically vanishing disturbances, which can be too restrictive for typical industrial plants.

In Chapter 8 we consider offset-free linear and nonlinear MPC controllers to handle the effects of disturbances and unmodeled dynamics. The computational efficiency of the Real-Time Iteration, which yields a closed-loop performance comparable to the full nonlinear MPC is also discussed. Even though in this case we do not guarantee closed-loop asymptotic stability, the designed MPC controllers substantially outperform the standard PI based control in several tested scenarios.
6 Compression systems for industrial applications

6.1 Literature review

Gas compression systems are a key technology in the process industry and gas transportation systems. Compressors increase the gas pressure, and therefore its thermodynamic energy, by the action of an external actuator. This may be a gas turbine, e.g., in aviation applications or energy generation, or an electric motor, e.g., in industrial applications [75]. The latter allows one for more flexibility and ease of operation, since variable speed drives achieve high efficiency and low time constants in regulation problems [129].

Axial and centrifugal compressors have been studied extensively in the literature. The nonlinear dynamics generates an instability, called surge, leading to periodic oscillation of the system variables, i.e., mass flow rates, speed and pressures. Such oscillations can result in deep surge, which may lead to irreversible damage of the blade tips and bearings.

Surge phenomena have long been analyzed with bifurcation theory, considering the compressor speed as the parameter generating such bifurcations [90, 71, 3, 21, 6]. The phenomenon is even more complicated in axial compressors, since rotating stall can occur, and this adds additional stable but undesired equilibria to the system. Surge determines an unstable region in the variable space, which is usually displayed in compressor maps by an experimentally derived surge line. The closer the compressor operates to such a line, the higher the risk of surge occurrence. On the other hand, external conditions, e.g., the downstream process, can move the system towards the surge region, thus forcing operations close to the surge region. Alternatively, efficiency considerations can drive the choice of the desired operating point close to the surge region, where the efficiency is typically the highest. The action of external disturbances, such as an unmodeled back pressure of the upstream and downstream processes, can then result in surge occurrence in tens of milliseconds. Hence, the control objective is to track the desired operating point guaranteeing safe operations of the compression system.

Simple descriptive models have long been available in the literature [65, 66, 64]. A general compression system consists of a compressor, which takes process gas from upstream conditions and pumps it into a downstream volume (plenum) through a duct. The discharge from the plenum is via a throttle in an exit duct. Several actuators have been considered for surge avoidance, including outlet valves [74] or
6.1 Literature review

special closed coupled valves [62]. These valves introduce additional pressure drops and therefore they reduce the energy efficiency of the system. The use of variable speed drives makes it possible to consider the motor torque as an actuator for the system. In [64], a PID control of the torque is considered and in [20] a nonlinear control is designed via backstepping techniques. The structure of the model is exploited in [114] to design a Lyapunov-based controller for the closed coupled valve, which results in a control strategy analogous to sliding mode to move the operation towards the safe zone.

Industrial plants are nowadays equipped with recycle valves, which regulate bypass circuits connecting the outlet plenum to the compressor inlet. Moving the process gas through such circuits reduces the pressure ratio almost instantaneously and increases the mass flow through the compressor, and this results in moving the system away from the surge instability region. Due to their efficacy, critical industrial compression plants always have such systems installed. Standard state-of-the-art control of such devices is usually decoupled from the torque control and consists of a PI scheme with feedforward control action, based on the measured distance from the surge region. The controllers are typically integrated in higher level control systems, which interface multiple compressors for load-sharing and other turbomachinery [115, 25].

The improvements in computational power of embedded controllers have recently opened the way to advanced controllers such as Model Predictive Control (MPC) for time critical applications such as compressor control. The benefits to the industrial implementation are shown experimentally in [51], where it is observed that tuning the standard PI control for surge avoidance becomes extremely difficult in the presence of external disturbances, since a specific tuning for one scenario can be ineffective for all others. In [10] a linearized MPC is employed for a compression system, driven by a gas turbine, which controls the rotor speed and the inlet guide vane positioning. A configuration with a recycle valve is used in [11]. In [27], additional temperature limitations are considered as output constraints in the linearized MPC formulation. Dealing with nonlinear dynamical models complicates the MPC optimization problem, especially in the presence of small sampling times. Linearization at every time step is the strategy employed in [39, 38, 37], which is validated experimentally, addressing technical feasibility and implementability in actual industrial controllers. In [72], an explicit multi-parametric nonlinear programming method is applied to the control of the outlet valve of a compression system. In
6 Compression systems for industrial applications

A controller for the recycle valve and the drive torque is designed, based on backstepping techniques combined with MPC. Additionally, the system is robust to possible model mismatches on the output valve characteristics.

The nonlinearity of the compression system limits the achievable performance of linearized MPC approaches. In [124], such a strategy is compared to nonlinear MPC that considers the full nonlinear model [41]. The divergence between linear and nonlinear model predictions is considerably reduced by the Real-Time Iteration approach [45, 46, 48], that has similar closed-loop performance to the nonlinear MPC.

6.2 The control problem

The goal of the control of a compression system is to deliver a mass flow rate of gas, \( m \), at an increased outlet pressure, \( p_o \), compared to the inlet pressure, \( p_i \). The system has two actuators, namely the torque provided by the external motor, \( \tau_d \), and the throttle of a recycle valve. The latter acts as an emergency actuator in transient to avoid the occurrence of instabilities. Clearly, the actuators are respectively constrained by the maximum power of the motor (limiting the maximum / minimum torque) and by the throttle position of the recycle valve. A typical state-of-the-art decoupled MIMO control of the two actuators is presented in Section 6.2.3. Apart from the fact that such control cannot optimize a desired performance objective, this may require a very tedious tuning procedure. In fact, the desired closed-loop performance is an aggressive control close to the instability and a smooth control close to the target point. In the MPC formulation, instead, we can consider at the same time the complex dynamics, and therefore coupled control, performance indicators and actuator constraints. A suitable performance indicator is a standard quadratic objective, with the weights designed such that the use of the recycle valve is limited.

6.2.1 Mathematical model

A general compression system with a recycle valve is modeled by the scheme in Figure 6.1 [38]. We consider two tanks, of volumes \( V_i \) and \( V_o \), which represent the plenums before and after the compressor. The inlet tank stores uncompressed gas flowing from an upstream source through the inlet valve. The outlet tank receives gas from the compressor, which is then released to a downstream sink through the outlet valve. The
6.2 The control problem

The control problem involves modeling the pressure drops due to the upstream process and the downstream sink. These are typically distributed pressure losses in long pipes or devices that require gas at pressure. The compressor is connected to the inlet tank and the compressed gas flows via a duct, of length $L$ and section area $A$, before entering the outlet tank. We assume that there is no intermediate duct between the inlet of the compressor and the inlet tank. A recycle valve, whose throttle opening can be manipulated, operates from the outlet tank to the inlet tank to provide surge protection when necessary.

Since the simulation results are related to a real air compression system, in this work we will consider the gas to be air and the atmospheric conditions for upstream and downstream. We remark that the validity of our results does not depend on the gas considered, but only on the availability of the mathematical model in Section 6.2.1 describing its physical properties.

The inlet tank pressure, $p_i$, the outlet tank pressure, $p_o$, and the mass flow rate through the compressor $m$, are related by the model (6.1) to the tank volumes and the sonic velocity at the inlet and outlet tank conditions, $a_i$ and $a_o$, typically assumed to be the sonic velocity at ambient conditions, $a_s$. The compressor speed, $\omega$, is related to the compressor torque, $\tau_c$, by the classic mechanical shaft equation, where the drive torque, $\tau_d$, is assumed to be the manipulated input, $v$ is a friction coefficient and $J$ is the mechanical inertia. A first-order equation models the actuation of the recycle valve with a time constant $T_r$, filtering the steady-state recycle mass flow $m_{r,ss}$ with the state variable $m_r$. The mass flow rates through the inlet valve $m_i$, the outlet valve $m_o$ and the recycle valve in steady-state $m_{r,ss}$, are specified in (6.2). The compressor characteristics $\pi(\omega, m)$ and the torque profile $\tau_c(\omega, m)$ are generally experimentally measured and approximated with polynomial equations [63].
The nonlinear system dynamics are given by:

\[
\begin{align*}
\dot{p}_i &= \frac{a_i^2}{V_i} (m_i(p_i) - m + m_r) \\
\dot{p}_o &= \frac{a_o^2}{V_o} (-m_o(p_o) + m - m_r) \\
\dot{m} &= \frac{4}{L} (\pi (\omega, m) p_i - p_o) \\
\dot{\omega} &= \frac{1}{J} (\tau_d - \tau_c (\omega, m) - \nu \omega) \\
\dot{m}_r &= \frac{1}{T_r} (m_{r, ss}(p_i, p_o, k_r) - m_r),
\end{align*}
\]

(6.1)

where

\[
\begin{align*}
m_i(p_i) &:= k_i \sqrt{p_{atm} - p_i} \\
m_o(p_o) &:= k_o \sqrt{p_o - p_{atm}} \\
m_{r, ss}(p_i, p_o, k_r) &:= k_r \sqrt{p_o - p_i}.
\end{align*}
\]

(6.2)

Note that by the physics and the operation of the compressor, we have that \( p_i < p_{atm} < p_o \) and therefore the square roots in (6.2) are always well-defined.

The valve constants \( k_i, k_o \) and \( k_r \) are determined by the valve throttle positions, hence they are positive for open valves and zero when closed. The recycle valve constant, \( k_r \), is considered to be a manipulated input for surge control, while the \( k_i \) and \( k_o \) are set by other system controllers to manage overall flow rates. The actuator of the system is the valve throttle opening, that will affect the valve constant, \( k_r \), with known characteristics. Non-return valves not displayed in the picture ensure that the mass flow rates \( m_i, m_o \) and \( m_r \) cannot reverse direction.

Historically, compression systems have been extensively analyzed via non-dimensional models \[65, 71\]. This is particularly convenient when the assumption of constant speed \( \omega \) is made. In our particular case, however, we assume that the speed is a state of the system, variable and influenced by the action of the actuators. Hence, we prefer the control oriented state space representation to the classical non-dimensional formulation. Similar considerations can be found in the recent literature \[64, 62, 124, 39, 38, 37\].

In the following, we deal with constrained control techniques in the time domain, thus we consider a discrete-time model for (6.1). Specifically, we generate an Euler discretization with zero-order hold assumption and sampling time \( T_s = 50 \text{ ms} \). From our numerical experience, a higher order discretization is not required. This is evident also in the simulation results shown in Chapter 7 and 8.
6.2 The control problem

Dynamical system is input affine, and the model can be written in a compact form as

\[ x^+ = f(x) + g(x)u, \]  

where the state \( x \) and \( u \) are expressed as the deviation from a desired operating condition:

\[ x := \begin{bmatrix} p_i - \bar{p}_i, & p_o - \bar{p}_o, & m - \bar{m}, & \omega - \bar{\omega}, & m_r - \bar{m}_r \end{bmatrix}^\top, \]
\[ u := \begin{bmatrix} \tau_d - \bar{\tau}_d, & k_r - \bar{k}_r \end{bmatrix}^\top, \]  

where the bar superscripts indicate desired steady-state values computed according to the procedure in Section 6.2.2. The inputs are subject to box constraints, due to the maximum / minimum provicable drive torque on the rotor shaft and the maximum / null recycle valve characteristics, corresponding to fully open / closed throttle.

In Figure 6.2, a typical compressor map is given. The pressure ratio \( p_o/p_i \) is displayed as a function of the mass flow rate \( m \). The Surge Line (SL) is a physical limit that must not be exceeded and it is in general experimentally determined. The SL is typically shifted by an experimentally determined safety margin to obtain the Surge Control Line (SCL), shown as a dashed line. The distance of the operating point, displayed as a red dot, from the SL and the SCL defines respectively the Surge Distance (SD) and Surge Control Distance (SCD) \([21, 10, 38]\). The Choke Line (CL) is a maximum mass flow rate limit that, in contrast to the surge related lines, cannot be physically reached.
by the machine and hence is not considered in the control problem. Finally, the grey curves represent speed isolines in steady-state conditions. Theoretically, the SL connects all maxima of the speed lines. In industrial practice, though, the SL is determined experimentally by pushing the system towards surge at different speeds and recording the SL points before experiencing significant oscillations. Usually, simple straight lines are considered by interpolation of the experimentally determined points. The controlled output of the system, $y$, is defined following typical industrial practice as the pressure ratio and the Surge Control Distance (SCD). By assuming that the SCL has equation $p_o/p_i = \beta_1 m + \beta_2$, with $\beta_1$ and $\beta_2$ determined experimentally, we have that

$$y = h(x) := \begin{bmatrix} \frac{p_o}{p_i} \\ m - \frac{1}{\beta_1} \left( \frac{p_o}{p_i} - \beta_2 \right) \end{bmatrix}. \tag{6.5}$$

### 6.2.2 Steady-state conditions

The steady-state conditions for the system are easily obtained by imposing zero state derivatives in (6.1).

In the following, we derive the steady-state conditions corresponding to a desired output $\bar{y}$, which then become set-points for reference tracking and are included in the stage costs. According to (6.5), the first component gives the pressure ratio. Combining the first and second equation in (6.1) we have

$$m_i(\bar{p}_i) = m_o(\bar{p}_o),$$

which, combined with the first equation in (6.5), allows us to determine the steady-state inlet tank pressure as follows:

$$\bar{p}_i = P_{atm} \frac{k_i^2 + k_o^2}{k_i^2 + k_o^2 \bar{y}_1},$$

$$\bar{p}_o = \bar{y}_1 \bar{p}_i.$$

Then the steady-state mass flow rate $\bar{m}$ is readily obtained from (6.5) as

$$\bar{m} = \frac{1}{\beta_1} (\bar{y}_1 - \beta_2) + \bar{y}_2,$$
while the steady-state recycle mass flow rate $\bar{m}_r$ and recycle valve characteristic $\bar{k}_r$ satisfy the following:

$$\bar{m}_r = \bar{m} - m_i(\bar{p}_i) = \bar{k}_r \sqrt{\bar{p}_o - \bar{p}_i}.$$

The steady-state compressor speed $\bar{\omega}$ comes from the solution of the third equation of (6.1). The compressor characteristics $\pi(\bar{\omega}, \bar{m})$ with fixed $\bar{m}$ is a second order equation in $\bar{\omega}$ and it is solved analytically. For the structure of the equation, the two zeros are both real and only one is positive. Finally, the steady-state torque input, $\bar{\tau}_d$, is readily obtained from the fourth equation in (6.1).

### 6.2.3 Standard industrial control

The state-of-the-art solution consists of independent process and anti-surge controllers realized using PI or PID controllers. The process controller maintains a given set point by manipulating the drive torque or speed, whereas the anti-surge controller is responsible for keeping the operation in the stable region of the compressor map by manipulating one or several recycle valves. The process controller typically has several modes of operation, e.g., discharge pressure control, discharge flow control or pressure ratio control. Anti-surge control is realized by computing the Surge Control Distance (SCD) illustrated in Figure 6.2 and using it as an error to drive the operating point to the SCL.

Typical execution rates for process controllers are $0.5 - 1$ seconds, while anti-surge control is much faster, with sampling time in the range of $50 - 200$ ms depending on the system dynamics. In the present case study the process controller operates in pressure ratio control mode with a sampling time of $T_{s,\tau} = 500$ ms, whereas the anti-surge controller is executed with a sampling time of $T_s = 50$ ms. The conventional controller consists of an anti-windup PI scheme as follows

$$e_k = \bar{y} - y_k$$

$$e_{\text{int},k} = e_{\text{int},k-1} + T_s e_k - k_{\text{awu}} (u_{k-1} - v_{k-1})$$

$$v_k = k_p e_k + k_i e_{\text{int},k}$$

$$u_k = \Pi_{[a,b]}(v_k).$$

(6.6)

To improve the system responses, a simple gain scheduling logic is introduced for both controllers. For the anti-surge controller, the gains are multiplied by a factor of 1.5 if the SCD is smaller than 25%, whereas for the process controller the same factor is used in an error band of ±5%.
7 Fast contractive MPC with stability and recursive feasibility guarantees

Model predictive control has been shown to be an effective approach for the regulation of compression systems [105, 39, 38, 37], where quadratic cost functions are typically used to trade off between performance and actuator usage.

Maintaining the process within safe limits is a critical requirement, for instance to avoid surge phenomena. In this chapter we review and apply the contractive MPC approach [41] to ensure closed-loop stability of the compression system. This consists of adding an additional constraint to the MPC formulation, imposing the decrease (contraction) of a quadratic control Lyapunov function [15, 16]. Then in Section 7.3 we introduce a numerical procedure for guaranteeing recursive feasibility of such a constraint and in Section 7.4 we show a numerical example on the compression system, where Algorithms 2.1 and 3.1 are employed with computational times competitive with commercial solvers.

7.1 Literature review

Well-known methods in the literature address closed-loop stability of nonlinear Model Predictive Control schemes by a terminal cost and/or a terminal state constraint. We refer to [86] for a comprehensive survey of these techniques introduced in the last century.

A popular approach developed later to ensure closed-loop stability for nonlinear systems via a specific terminal cost is in [82]. It consists of considering a sufficiently long prediction horizon, while keeping the control horizon and the number of decision variables small. After the control horizon, the inputs are computed via an auxiliary control law that is locally stabilizing. Closed-loop stability is obtained via a penalty on the terminal state which is an upper bound of the infinite-horizon cost associated with the auxiliary controller.

A recent approach is the so-called unconstrained optimization [67, 107]. Closed-loop stability is determined via relaxed dynamic programming results by selecting a sufficiently large prediction horizon. Further, a bound on the performance of the finite-horizon control is derived.
with respect to the infinite-horizon costs. Under mild assumptions on the controllability of the considered system, the minimal prediction horizon for closed-loop stability is derived by solving iteratively linear programs.

The approach in [73] consists of imposing an additional constraint that the terminal state equals exactly zero. This approach leads to computational difficulties, as for a nonlinear system an infinite number of iterates is in general required to reach the origin. On the contrary, an approximate satisfaction leads to a loss of stability in the considered region. This is overcome by the formulation in [88], where a linear locally stabilizing controller is used in such a region. The resulting control is then shown to be globally asymptotically stabilizing.

A terminal cost combined with a quadratic terminal constraint is considered in [33]. The latter ensures that the terminal state is contained in an ellipsoid, where the system can be stabilized by a linear state feedback control. Such control is not implemented in practice, but it only determines the terminal region and the terminal cost via an offline procedure. Further, the terminal cost is shown to bound the infinite horizon cost of the nonlinear system controlled by the local linear state feedback.

Closed-loop stability can be ensured even when soft constraints are considered in the optimization program. In [137, 135] slack variables are introduced in the state constraints and penalized in the cost function. This clearly has the effect of enlarging the set of recursively feasible points for which closed-loop stability can be guaranteed. A formal proof is presented for the case of linear systems, as well as the input-to-state stability in presence of additive process noise. Furthermore, stability can be guaranteed even if the optimization program is interrupted before convergence [138, 136, 134]. A feasible start Interior-Point Method is considered, initialized by a real-time procedure that constructs a feasible warm-start solution.

The approach that we consider in this chapter is the Contractive MPC [41]. Given a suitable quadratic control Lyapunov function (CLF) for the nonlinear system, a decrease (contraction) of a factor $\gamma \in [0, 1)$ is imposed every $N$ time steps, with $N$ the prediction horizon. It follows that such constraint in the MPC problem is time-varying and it can apply also to intermediate states, depending on the current time step. This approach yields exponential stability with factor $\gamma$ of the resulting nonlinear system. Moreover, the time-varying quadratic constraint introduces sparsity in the optimization problem that can be efficiently
exploited by Algorithms 2.1 and 3.1 to obtain fast computational times, as shown in Section 7.4. The method is reviewed in more detail in the next section.

### 7.2 Formulation of the control problem

The theory here follows the approach in [41]. For simplicity, the case without disturbance is presented, but the same method also applies to an asymptotically decaying disturbance.

Figure 7.1 illustrates the concept, where a quadratic Lyapunov function \( V(x) := \|x\|_P^2 := x^TPx \) contracts by a factor \( \gamma \in [0,1) \) every \( N \) time steps, with \( N \) fixed and equal to the prediction horizon. Since such a constraint is time-varying, it appears in the nonlinear MPC problem at time steps that are multiples of \( N \). By denoting the current time step as \( t_k \), we define \( t_{k+N|k} \) as the predicted time step in which the contractive constraint is enforced. For example, at the first time step \( t_0 \), the contractive constraint is \( N \) samples in the future, hence \( \tilde{k} = N \). In the following time step, \( t_1 \), we have \( \tilde{k} = N - 1 \), and then recursively up to the time step \( t_N \), where we again have \( \tilde{k} = N \). Hence, given the current state \( x_k \), the contractive MPC problem for the control of the compressor is:

\[
\begin{align*}
\min_{(x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}} & \quad \sum_{j=0}^{N-1} \left\{ \frac{1}{2} x_{k+j|k}^T Q x_{k+j|k} + \frac{1}{2} u_{k+j|k}^T R u_{k+j|k} \right\} + \frac{1}{2} x_{k+N|k}^T P x_{k+N|k} \\
\text{s.t.} & \quad x_{k+j+1|k} = f(x_{k+j|k}) + g(x_{k+j|k}) u_{k+j|k} \\
& \quad u_{k+j|k} \in [a, b] \quad \forall j \in \{0, \ldots, N - 1\} \\
& \quad x_{k+\tilde{k}|k}^T P x_{k+\tilde{k}|k} \leq \gamma x_{k+N-N|k}^T P x_{k+N-N|k},
\end{align*}
\]

(7.1)

where \( \gamma \in [0,1) \) is the contraction parameter, \( x_{k+j|k} \) and \( u_{k+j|k} \) respectively denote the state and the input predicted \( j \) time steps ahead, and the state \( x_{k+\tilde{k}|k} \) denotes the predicted state at the time \( t_{k+\tilde{k}|k} \).

#### 7.2.1 Theoretical guarantees

The desired stability results are obtained by proving the decrease of a quadratic control Lyapunov function of the form \( V(\cdot) := \|\cdot\|_P^2 \).

Before stating the local closed-loop stability result, we introduce the following assumptions.
7 Fast contractive MPC with stability and recursive feasibility guarantees

Figure 7.1: Contraction of the Lyapunov function, with factor $\gamma \in [0, 1)$.

Assumption 7.1. $(x,u) = (0,0)$ is an equilibrium point of (6.3).

Assumption 7.2. The linearization of the model dynamics in (6.3) around the origin is stabilizable.

Assumption 7.3. There exists a constant $\gamma_1 \in (0, \infty)$ such that for any initial condition $x_k \in \mathcal{E}_{\gamma_1} := \{ x \in \mathbb{R}^n \mid V(x) \leq \gamma_1 \}$ at time $t_k$, the optimization problem in (7.1) is feasible. Further, the problem is feasible at any future time step $t_{k+jN}$, for any $j \in \mathbb{N}$.

Assumption 7.4. There exists a constant $\beta \in (0, \infty)$ so that for any $x_k \in \mathcal{E}_{\gamma_1}$, the predicted $N$ states by the model in (6.3) remain inside the set $\mathcal{E}_{\beta V(x_k)} := \{ x \in \mathbb{R}^n \mid V(x) \leq \beta V(x_k) \}$.

Assumption 7.1 is guaranteed without loss of generality by an appropriate change of variables. Stabilizability of the linearized system around the origin is a typical assumption in the analysis of closed-loop stability of nonlinear systems. Further, since the input is constrained, Assumption 7.4 is always satisfied except for systems with finite escape time. Determining $\beta$ is not necessary in practice, but we only need to know its existence. The feasibility of the problem in (7.1) required in Assumption 7.3 needs to be verified by a separate test, presented in the next section, which ensures that the function $V(\cdot) := \|\cdot\|_P^2$ is a control Lyapunov function.

Based on these assumptions, local exponential stability is guaranteed by the following theorem, given without the proof.
Theorem 7.1 ([41], Theorem 1). The MPC in (7.1) renders the dynamical system in (6.3) closed-loop exponentially stable on the set $\mathcal{E}_{\gamma_1}$, i.e. for any $x_0 \in \mathcal{E}_{\gamma_1}$, the resulting trajectory satisfies the inequality:

$$
\|x_k\|_P \leq a \|x_0\|_P e^{-\frac{(1-\sqrt{\gamma})(t_k-t_0)}{N_s}},
$$

with $a \geq \beta e^{1-\sqrt{\gamma}}$.

When an asymptotically decaying disturbance is included, an equivalent theorem establishes the local asymptotic stability. We also refer to [41] for a further result about the objective function being an exponentially decaying Lyapunov function for the closed-loop system.

### 7.3 Numerical test of recursive feasibility

In this section we present a method to check Assumption 7.3 for an input affine dynamics. This method is numerical, since it requires to run an offline procedure where we grid the state-space and, for every data point, we check specific conditions. Our goal is to derive the ellipsoid $\mathcal{E}_{\gamma_1} := \{ x \in \mathbb{R}^n \mid V(x) \leq \gamma_1 \}$ in Assumption 7.3, where we can guarantee satisfaction of the contractive constraint in (7.1). Ensuring satisfaction of this constraint amounts to showing that the quadratic function $V(x) := \|x\|_P^2 := x^\top Px$ is a Control Lyapunov Function (CLF) with contraction factor $\gamma$ for every point in $\mathcal{E}_{\gamma_1}$.

In our numerical simulations, the problem of finding $P > 0$ and $\gamma \in [0,1)$ is solved numerically. Let us start from the knowledge of a CLF $V(x) := \|x\|_P^2 := x^\top Px$ for the system linearized around the origin, i.e., the desired equilibrium. For instance, the matrix $P$ can be the solution of the Riccati equation, with weight matrices $Q$ and $R$. $\gamma$ is set to a desired value, e.g., $\gamma = 0.999$.

To estimate the region in which $V$ is a CLF for the nonlinear system, we grid the $n$-dimensional state space around the origin. Given a desired contraction factor $\gamma \in [0,1)$, we check the discrete-time Lyapunov-decrease condition for every grid point $x$, i.e.:

$$V(f(x) + g(x)u) \leq \gamma V(x),$$

for some $u \in [a,b] \subset \mathbb{R}^m$, which is equivalent to:

$$(f(x) + g(x)u)^\top P (f(x) + g(x)u) \leq \gamma x^\top Px.$$
7 Fast contractive MPC with stability and recursive feasibility guarantees

Now, for a given state \( x \in \mathbb{R}^n \), the existence of a feasible control input \( u \in [a, b] \subset \mathbb{R}^m \) such that (7.3) holds can be determined by solving the QP

\[
 u^*(x) := \arg\min_{u \in \mathbb{R}^m} u^\top g(x)^\top P g(x) u + 2 f(x)^\top P g(x) u \\
\text{s.t. } u \in [a, b],
\]

(7.4)

and then checking the inequality

\[
 J^*(x) := J(u^*(x)) \leq \gamma x^\top P x - f(x)^\top P f(x).
\]

(7.5)

The numerical approach that we propose is to check the inequality in (7.5) over a finite number of state vectors on a sufficiently fine grid. Specifically, we take \( x \in \mathbb{R}^n \) as a generic grid vector and \( u^* = u^*(x) \). Then we consider the Lyapunov-decrease condition in an \( \varepsilon \)-neighborhood of \( x \), that is,

\[
\max_{\delta \in \varepsilon B} V \left( f(x + \delta) + g(x + \delta) u^* \right) - \gamma V(x + \delta) \leq 0.
\]

(7.6)

By defining \( L(x) \) as the local Lipschitz constant of the function \( V \left( f(\cdot) + g(\cdot) u^* \right) - \gamma V(\cdot) \) relative to the neighborhood \( x + \varepsilon B \), (7.6) can be obtained as follows:

\[
\max_{\delta \in \varepsilon B} V \left( f(x + \delta) + g(x + \delta) u^* \right) - \gamma V(x + \delta) \\
= \max_{\delta \in \varepsilon B} V \left( f(x) + g(x) u^* \right) - \gamma V(x) + V \left( f(x + \delta) + g(x + \delta) u^* \right) - \gamma V(x) \\
- \gamma V(x + \delta) - (V \left( f(x) + g(x) u^* \right) - \gamma V(x)) \\
\leq V \left( f(x) + g(x) u^* \right) - \gamma V(x) + \varepsilon \max_{\delta \in \varepsilon B} \frac{1}{\varepsilon} V \left( f(x + \delta) + g(x + \delta) u^* \right) \\
- \gamma V(x + \delta) - (V \left( f(x) + g(x) u^* \right) - \gamma V(x)) \left| V \left( f(x) + g(x) u^* \right) - \gamma V(x) \right| \\
= V \left( f(x) + g(x) u^* \right) - \gamma V(x) + \varepsilon L(x) \leq 0.
\]

(7.7)

The last inequality in (7.7) may not hold in a neighbourhood of the origin. In fact, the desired steady-state point \( x = 0 \) is such that \( f(0) = 0 \), hence \( u^* = 0 \) from (7.4). As \( V(0) = 0 \), (7.7) becomes \( \varepsilon L(0) \leq 0 \), with \( L(0) = \max_{\delta \in \varepsilon B} \frac{1}{\varepsilon} V \left( f(\delta) \right) - \gamma V(\delta) \) becomes \( \varepsilon L(0) \leq 0 \). Therefore, by computing an upper bound to \( L(x) \), we derive the Lyapunov-decrease condition in (7.7) to be checked outside a neighborhood of the origin.

Note that this method introduces conservativeness since we are considering a 1-step contraction instead of an \( N \)-step contraction. If in 1
7.3 Numerical test of recursive feasibility

Algorithm 7.1 Find $\bar{\gamma}$, CLF values $(V_k)_k$ and flags $(r_k)_k$

Set $\bar{\gamma}$ to an initial estimate.

$k = 0$

for all $x_i \in \mathcal{X}$ do

if $x_i^\top Px_i < \bar{\gamma}$ then

CHECK (7.5) for $x_i$

if (7.5) holds for $x_i$ then

$k \leftarrow k + 1$

$V_k \leftarrow x_i^\top Px_i$

CHECK (7.7) for $x_i$

if (7.7) holds for $x_i$ then

$r_k \leftarrow 2$

else

$r_k \leftarrow 1$

end if

else

$\bar{\gamma} \leftarrow x_i^\top Px_i$

end if

end if

end for

return $\bar{\gamma}, V, r$

We now propose a procedure to numerically determine the set of states that, if reachable in at most $N$ steps, provides a certificate of asymptotic stability and recursive feasibility for the MPC algorithm.

Given the CLF $V(x)$ and the level set value $\gamma_1 \in \mathbb{R}_{>0}$, we define the contractive set:

$$\mathcal{E}_{\gamma_1} = \{ x \in \mathbb{R}^n \mid V(x) \leq \gamma_1 \},$$

in the sense that every point in its boundary contracts by a factor $\sqrt{\gamma}$. The numerical test aims at deriving the largest region $\mathcal{E}_{\gamma_1}$ in which the constraint in (7.2) is satisfied for any point $x \in \mathcal{E}_{\gamma_1}$, for some feasible input $u(x) \in [a, b]$. Note that Algorithm 7.1 does not test any
7 Fast contractive MPC with stability and recursive feasibility guarantees

\[ \mathcal{E}_{\bar{\gamma}} = \{ x \in \mathbb{R}^n \mid \|x\|_P^2 < \bar{\gamma} \} \]

\[ \mathcal{E}_{\gamma_1} = \{ x \in \mathbb{R}^n \mid \|x\|_P^2 \leq \gamma_1 \} \]

\[ \mathcal{E}_{\gamma_2} = \{ x \in \mathbb{R}^n \mid \|x\|_P^2 \leq \gamma_2 \} \]

**Figure 7.2:** Numerical test of Lyapunov decrease.

grid point larger than \( \bar{\gamma} \), and this value is reset whenever a grid point not satisfying (7.4) is found. This makes the number of grid points significantly small, compared to the cardinality of the grid.

Since the condition in (7.6) cannot be satisfied for \( x = 0 \), we denote the small neighborhood of the origin in which the condition does not hold as

\[ \mathcal{E}_{\gamma_2} = \{ x \in \mathbb{R}^n \mid V(x) \leq \gamma_2 \} , \quad (7.9) \]

for some constant \( \gamma_2 \in \mathbb{R}_{>0} \). We consider \( \mathcal{E}_{\gamma_2} \) sufficiently small, such that any state in this set is practically stabilizable via MPC.

We determine the numerical values \( \gamma_1 \) and \( \gamma_2 \) through Algorithm 7.1.

The algorithm tests all points \( \{x_i\}_i \) in a grid \( \mathcal{X} \) and returns the smallest level set value \( \bar{\gamma} := V(x_i) \) such that \( x_i \) is infeasible, i.e., not satisfying the Lyapunov constraint (7.5). Therefore, the algorithm ensures that there exists no point in the region \( \mathcal{E}_{\bar{\gamma}} = \{ x \in \mathbb{R}^n \mid V(x) < \bar{\gamma} \} \) that does not satisfy (7.5).

In addition, the algorithm returns the CLF values \( (V_k)_k \) and flags \( (r_k)_k \) that are used to compute \( \gamma_1 \) and \( \gamma_2 \) as follows.

\[
\gamma_2 := \min_k V_k \tag{7.10}
\]

\[
s.t. \quad r_k = 2; \quad (7.10)
\]

\[
\gamma_1 := \max_k V_k \tag{7.11}
\]

\[
s.t. \quad V_k < \bar{\gamma} \quad (7.11)
\]

\[
r_k = 2 \quad \forall k \in \{ l \mid \gamma_2 \leq V_l \leq V_k \} . \tag{7.11}
\]
The optimization problems in (7.10) and (7.11) can be solved efficiently, e.g., by sorting and inspecting the values $(V_k)_k$ and $(r_k)_k$.

Figure 7.2 shows the concept: Algorithm 7.1 determines an upper bound $\bar{\gamma}$ to $\gamma_1$, meaning that there exists an infeasible point $\bar{x}$ such that $V(\bar{x}) = \|\bar{x}\|_P^2 = \bar{\gamma}$. The values of $\gamma_1$ and $\gamma_2$ satisfying respectively (7.10) and (7.11) are found by considering the CLF values $(V_k)_k$ and flags $(r_k)_k$ generated by Algorithm 7.1. All the points in $\mathcal{E}_{\gamma_1} \setminus \mathcal{E}_{\gamma_2} := \{ x \in \mathbb{R}^n \mid \|x\|_P^2 \in [\gamma_2, \gamma_1] \}$ are such that $r_k = 2$, therefore the desired contractive constraint in (7.1) holds. The points in $\mathcal{E}_{\bar{\gamma}} \setminus \mathcal{E}_{\gamma_1} := \{ x \in \mathbb{R}^n \mid \|x\|_P^2 \in [\gamma_1, \bar{\gamma}] \}$ instead are such that $r_k$ is either 1 or 2. The points strictly inside region $\mathcal{E}_2$ satisfy condition (7.5) but not (7.7) and they are considered close enough to the equilibrium, indicated by the plus, to achieve practical stabilization.

Note that the considered test also applies to continuous-time CLFs. In fact, if the system is input-affine, the test in (7.2) becomes a linear program (LP), while for the test in (7.7) is easily modified by changing the condition in (7.6) and the function used to compute the local Lipschitz constant $L(x)$.

7.4 Simulation results

The desired reference point affects the resulting level sets $\gamma_1$, $\gamma_2$ and $\bar{\gamma}$. In fact, in the compressor control problem, choosing a steady-state null recycle valve actuation, $\bar{k}_r = 0$, corresponding to a null recycle mass flow rate, $\bar{m}_r = 0$, shrinks the level set $\gamma_1$ to zero, because the system is operated on its boundary. Thus, in this section we consider the equilibrium with specifications $\bar{p}_o = 1.2$ bar and recycle valve opening $\bar{k}_r = 0.2 k_{r,max}$. We choose $P$ from the solution of the Riccati equation and $\gamma = 0.999$. Algorithm 7.1 is applied to determine $\gamma = 7.714e-4$. We sort in increasing order the Lyapunov function evaluations and the corresponding indexes derived by Algorithm 7.1 and obtain $\gamma_2 := 7.368e-5$ and $\gamma_1 := 2.322e-4$. 

93
7 Fast contractive MPC with stability and recursive feasibility guarantees

Next, we consider the requirement that the state can reach the ellipsoid \( E_{\gamma_1} \) \((7.8)\) in \( N \) prediction steps, i.e.,

\[
\min_{(x_{k+j+1|k}, u_{k+j|k})} \sum_{j=0}^{N-1} \left\{ \frac{1}{2} x_{k+j|k}^T Q x_{k+j|k} + \frac{1}{2} u_{k+j|k}^T R u_{k+j|k} \right\} \\
+ \frac{1}{2} x_{k+N|k}^T P x_{k+N|k} \\
\text{s.t.} \quad x_{k+j+1|k} = f (x_{k+j|k}) + g (x_{k+j|k}) u_{k+j|k} \\
u_{k+j|k} \in [a, b] \quad \forall j \in \{0, \ldots, N-1\} \\
x_{k+N|k}^T P x_{k+N|k} \leq \gamma_1.
\]

(7.12)

The feasibility of (7.12), together with the feasibility of (7.1), ensures recursive feasibility of the MPC for every successive time step.

Figure 7.3 shows a state trajectory on the compressor map under the contractive MPC, along with the trajectory obtained using the state-of-the-art industrial controller presented in Section 6.2.3. The simulation is based on the realistic test bench whose specifications are given in Appendix D. The prediction horizon is tuned to \( N = 20 \) steps, with sampling time \( T_s = 50 \) ms, and the simulation time is \( t = 50 \) s. The system modeling, including the Surge Line (SL), Surge Control Line (SCL) and Choke Line (CL) definitions, follows from [37]. Note that the state-of-the-art controller, tuned according to the procedure in Section 6.2.3, has a significant overshoot at reaching the desired operating point. From a process control point of view, this overshoot corresponds to the closest point to the SCL in the whole simulation. This risk is avoided by the MPC scheme, in which the target point is tracked more naturally.

An estimate of the level of conservativeness introduced by the approach can be obtained by checking the contractive inequality in (7.2) after computing the optimal solution of (7.1). For this numerical example we observe that the contractive constraint never becomes active, and in particular the distance from the boundary of the constraint shrinks as the operating point reaches the desired target. The conservativeness introduced allows for possible asymptotically decaying disturbances in the dynamics [41].
7.4 Simulation results

Figure 7.3: Compressor map: the dash-dot blue line corresponds to the contractive nonlinear MPC and the red line to the standard PID control. The contractive nonlinear MPC ensures asymptotic stability and recursive feasibility: the closed-loop trajectory converges to the desired equilibrium (+) starting from an initial condition (×) which is feasible for both (7.12) and (7.1).

7.4.1 Computational times

The structure of the problem in (7.1) makes it possible to apply the projected gradient and constraint linearization method in Algorithms 2.1 and 3.1 by following the procedure explained in Section 4.2.2.

The computational results of the MPC problem in (7.1) are given in Tables 7.1 – 7.4. The best and nearly best performances are highlighted in bold font. The KKT optimality tolerance for the solution of the optimization problem has been first set to $\varepsilon = 10^{-6}$ (Tables 7.1 and 7.2), then to $\varepsilon = 10^{-3}$ (Tables 7.3 and 7.4).

As discussed in Section 5.4, the performance of the proposed algorithm is compared to other commercial solvers, by solving either the full nonlinear problem or some suitable approximation. As approximations to the nonlinear program in (7.1), we consider the Real-Time Iteration, presented in Appendix B. Further, since the contractive constraint in (7.1) would be linearized in the RTI as well as the other constraints, we additionally consider a RTI with a quadratic constraint (QC), i.e., where the contractive constraint is kept in its quadratic form, thus solving a Quadratically Constrained Quadratic Program (QCQP). This form is allowed by the IPM implemented in the solver FORCES Pro.
Both the approaches are warm-started by shifting the input sequence obtained at the previous time instance and then setting the last input equal to the second last. The two RTI-based methods achieve a very similar closed-loop performance to the full nonlinear solution (they appear coincident in Figure 7.3), but the RTI iteration is not guaranteed to be recursively feasible by the test presented in Section 7.3 since the states are predicted via approximated dynamics.

The computational time required by SNOPT is significantly higher, due to the fact that the solver is general-purpose and does not exploit the sparsity introduced by the MPC problem as the proposed algorithm or FORCES Pro. In this particular example, the proposed algorithm also outperforms by one order of magnitude FORCES Pro. Further, increasing the tolerance to $\varepsilon = 10^{-3}$ results in halving the computational times for the proposed method, while the benefits for IPM are limited (the barrier methods typically require at least tens of iterations for the convergence of the barrier parameter). Algorithm 3.1 with $p = 2$ only slightly outperforms Algorithm 2.1.

In practice we observe that the number of iterations required in this case by the proposed algorithms to solve the MPC problem is comparable to those of the other considered methods, as shown in Tables 7.2 and 7.4. Since each iteration of the former is significantly cheaper than SQP (it has to solve a QP at every iteration) and IPM (at least one linear system needs to be solved at every iteration, based on the specific method), the computational advantage achieved is substantial.

## 7.5 Conclusion

In this chapter we have applied the contractive MPC for the control of a compression system. We have introduced a procedure that allows one to determine the region where the contractive MPC is recursive feasible. Such a procedure is numeric and, even though computationally expensive, it can run offline.

Even though tuned for the same purpose, i.e., reference tracking and surge avoidance, the simulation results on the compression system show a better closed-loop performance of the MPC compared to the state-of-the-art controller, as observed from the pressure ratio overshoot in Figure 7.3. The theoretically guaranteed stability is also observed in practice. Furthermore, the algorithms proposed in Chapters 2 and 3, implemented in the software FalcOpt, can efficiently exploit the sparsity
7.5 Conclusion

Table 7.1: Computational times for the contractive MPC of the compressor, tol. $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method (Solver)</th>
<th>Avg. (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>0.0892</td>
<td>0.0724</td>
<td>0.169</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>0.101</td>
<td>0.0797</td>
<td>0.307</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>$\mathbf{0.0835}$</td>
<td>$\mathbf{0.0677}$</td>
<td>$\mathbf{0.167}$</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>0.147</td>
<td>0.111</td>
<td>0.418</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>256</td>
<td>109</td>
<td>889</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>1.06</td>
<td>0.616</td>
<td>2.70</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>0.727</td>
<td>0.605</td>
<td>1.27</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>0.783</td>
<td>0.656</td>
<td>4.25</td>
</tr>
</tbody>
</table>

Table 7.2: Number of iterations to convergence for the contractive MPC of the compressor, tol. $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>$\mathbf{24.8}$</td>
<td>$\mathbf{21}$</td>
<td>$\mathbf{27}$</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>25.0</td>
<td>20</td>
<td>36</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>$\mathbf{24.7}$</td>
<td>$\mathbf{21}$</td>
<td>$\mathbf{27}$</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>25.1</td>
<td>20</td>
<td>39</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>8.55</td>
<td>4</td>
<td>22</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>23.2</td>
<td>13</td>
<td>40</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>15.2</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>15.9</td>
<td>14</td>
<td>22</td>
</tr>
</tbody>
</table>
7 Fast contractive MPC with stability and recursive feasibility guarantees

Table 7.3: Computational times for the contractive MPC of the compressor, tol. $\varepsilon = 10^{-3}$

<table>
<thead>
<tr>
<th>Method (Solver)</th>
<th>Avg. (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>0.0488</td>
<td>0.0371</td>
<td>0.125</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>0.0533</td>
<td>0.0377</td>
<td>0.110</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>0.0460</td>
<td>0.0330</td>
<td>0.112</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>0.0772</td>
<td>0.0515</td>
<td>0.139</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>114</td>
<td>46.8</td>
<td>593</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>0.982</td>
<td>0.502</td>
<td>2.43</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>0.599</td>
<td>0.515</td>
<td>1.03</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>0.684</td>
<td>0.539</td>
<td>1.18</td>
</tr>
</tbody>
</table>

Table 7.4: Number of iterations to convergence for the contractive MPC of the compressor, tol. $\varepsilon = 10^{-3}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>12.8</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>12.8</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>12.9</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>13.3</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>1.65</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>19.7</td>
<td>10</td>
<td>35</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>12.7</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>14.6</td>
<td>12</td>
<td>21</td>
</tr>
</tbody>
</table>
introduced by the contractive constraint and outperform the Interior-Point solver considered by one order of magnitude.
When operating in practice a compression plant, with external unmod-eled disturbances and model mismatch, we may be more interested in being able to track a desired reference, rather than having stability guarantees.

In this section, we discuss the offset-free MPC formulation to ensure that the closed-loop dynamics converge to the desired operating point, even in presence of model mismatch and non-asymptotically decaying disturbance inputs. We consider both a linearized MPC approach (described in Section 8.1) and the nonlinear MPC problem (Section 8.2). Finally, in Section 8.3, we present the simulation results, including the effect of possible typical model mismatch and external disturbances. Algorithm 5.1 is then employed to derive computational times competitive with commercial solvers.

8.1 Linearized Model Predictive Control

The first MPC approach that we consider relies on the linearization of the nonlinear system dynamics in (6.3) around the target steady-state point \((\bar{x}, \bar{u})\), computed from the desired output \(\bar{y}\) according to the procedure in Section 6.2.2. If the operating points are close to the target steady-state point, the linearized model will be a good approximation of the nonlinear dynamics. The linear time-invariant model can be rewritten as

\[
\Delta x^+ = \bar{A} \Delta x + \bar{B} \Delta u \\
\Delta y = \bar{C} \Delta x,
\]

where, given the input affine plant in (6.3), \(\Delta x := x - \bar{x}, \Delta u := u - \bar{u}, \Delta y := y - \bar{C} \bar{x}\) and \(\bar{A} := (\partial f/\partial x)(\bar{x}) + (\partial g/\partial x)(\bar{x}) \bar{u}, \bar{B} := g(\bar{x}), \bar{C} := (\partial h/\partial x)(\bar{x})\).

Offset-free MPC approaches consider an augmented model with trivial disturbance dynamics, where an estimate of the augmented state is obtained via a filter (e.g., Kalman Filter) [100, 81]. The goal is to
obtain offset-free tracking of the desired outputs. In our case study, we augment the linearized model as

\[
\begin{bmatrix}
\Delta x \\
d
\end{bmatrix}^+ = \begin{bmatrix}
\bar{A} & B_d \\
0 & I
\end{bmatrix} \begin{bmatrix}
\Delta x \\
d
\end{bmatrix} + \begin{bmatrix}
\bar{B} \\
0
\end{bmatrix} \Delta u
\]

\[
\Delta y = \begin{bmatrix}
\bar{C} & C_d
\end{bmatrix} \begin{bmatrix}
\Delta x \\
d
\end{bmatrix},
\]

(8.1)

where \(d \in \mathbb{R}^2\), thus \(B_d \in \mathbb{R}^{5 \times 2}\) and \(C_d \in \mathbb{R}^{2 \times 2}\). We indicate with \(\nabla \hat{x}_{k|k}\) and \(\hat{d}_{k|k}\) the filter estimates. From our numerical experience, a good choice for the design of the disturbance model is a pure input disturbance, that is, \(B_d = B\) and \(C_d = 0\), since with this choice the disturbance is filtered by the dynamics. We have experienced poorer performance with a pure output disturbance model, i.e., \(B_d = 0\) and \(C_d = I\), because of the presence of poles of the linearized model characteristic polynomial close to \((1, 0)\) [93].

For the design of the steady-state Kalman filter, we follow the standard procedure: an additive zero-mean process and measurement white-noise is considered in the equations, then the discrete algebraic Riccati equation is solved to obtain the update gain \(L = [L_x; L_d]\). This design yields a full rank square matrix \(L_d\), that automatically satisfies one of the offset-free tracking conditions [100]. Since in our application the state is easily measured, we assume the measurement noise covariance to be significantly smaller than the process covariance.

Given the current estimate of the disturbance \(\hat{d}_{k|k}\), the following model inversion gives the updated state and input set-points \(\bar{x}_{k|k}\), \(\bar{u}_{k|k}\) at time \(k\):

\[
\begin{bmatrix}
I - \bar{A} & -\bar{B} \\
\bar{C} & 0
\end{bmatrix} \begin{bmatrix}
\bar{x}_{k|k} \\
\bar{u}_{k|k}
\end{bmatrix} = \begin{bmatrix}
B_d \hat{d}_{k|k} \\
-C_d \hat{d}_{k|k} + \Delta \bar{y}
\end{bmatrix},
\]

(8.2)

where, due to the expression of our function \(h\), the matrix on the left-hand side is square and invertible, and the reference \(\Delta \bar{y} := \bar{y} - \bar{C} \bar{x}\).

Note that the resulting input \(\bar{u}_{k|k}\) could be infeasible for the actuator limit. In this case, the control would not be offset-free and only a point close to the desired output \(\bar{y}\) would be reached. The existing input constraint must not be active at the steady-state point in order to obtain offset-free tracking [100]. In industrial practice, it is often the case that the desired operating point corresponds to null recycle valve actuation, thus resulting in one active input constraint. The constraint
is active in this configuration and there is no guarantee that the system is offset-free. On the other hand, from our simulation experience the offset observed is negligibly small and completely satisfactory for practical purposes.

The linearized MPC problem is then expressed as

\[
\min_{\begin{array}{l}(x_{k+j|k}, y_{k+j|k}) \in \mathbb{R}^n \\
(u_{k+j|k}) \in \mathbb{R}^m
\end{array}} \sum_{j=0}^{N-1} \left\{ \frac{1}{2} (\Delta y_{k+j|k} - \bar{y}) Q_y (\Delta y_{k+j|k} - \bar{y}) + \frac{1}{2} (\Delta u_{k+j|k} - \bar{u}_{k|k}) R (\Delta u_{k+j|k} - \bar{u}_{k|k}) \right\} \\
\text{s.t.} \quad \Delta x_{k|k} = \Delta \hat{x}_{k|k} \\
\Delta x_{k+j+1|k} - \bar{x}_{k|k} = \bar{A} (\Delta x_{k+j|k} - \bar{x}_{k|k}) + \bar{B} (\Delta u_{k+j|k} - \bar{u}_{k|k}) \\
\Delta u_{k+j|k} + \bar{u}_{k|k} \in [a, b] \quad \forall j \in \{0, \ldots, N-1\} \\
\Delta y_{k+j|k} = \bar{C} \Delta x_{k+j|k} \quad \forall j \in \{0, \ldots, N\},
\]

where \(Q_y, R \succeq 0\). The cost function trades off the distance from the SL with the usage of the actuators, and in particular the emergency recycle valve.

### 8.2 Nonlinear Model Predictive Control

The offset-free MPC approach also applies to the nonlinear case, via the internal model principle. Based on the current disturbance estimate and the desired reference, the target state and input steady-state conditions are computed. Finally, the nonlinear MPC tracks the computed target state and input, subject to the augmented dynamics and the original constraints. Analogously to the linear case and in accordance to the model in (6.3), we consider a pure input disturbance model:

\[
x^+ = f(x) + g(x)(u + d) \\
d^+ = d \\
y = h(x).
\]

We consider an extended Kalman Filter due to the nonlinear dynamics. Specifically, given the current estimate of the disturbance \(\hat{d}_{k|k}\) and the
desired output reference $\bar{y}$, the state and input set-point at time $k$, $(\bar{x}_{k|k}, \bar{u}_{k|k})$, are given by

\[
\bar{x}_{k|k} = f(\bar{x}_{k|k}) + g(\bar{x}_{k|k})(\bar{u}_{k|k} + \hat{d}_{k|k}) \\
\bar{y} = h(\bar{x}_{k|k}).
\]

(8.5)

As in the linear case, the derived set-point needs to satisfy the original constraint for offset-free control and uniqueness of the set-point, which holds true in our application. The nonlinear MPC problem is then

\[
\begin{align*}
\min_{(x_{k+j|k}, y_{k+j|k})_{j=0}}^{(u_{k+j|k})_{j=0}} & \sum_{j=0}^{N-1} \left\{ \frac{1}{2} (y_{k+j|k} - \bar{y})^\top Q_y (y_{k+j|k} - \bar{y}) \\
& + \frac{1}{2} (u_{k+j|k} - \bar{u}_{k|k})^\top R (u_{k+j|k} - \bar{u}_{k|k}) \right\} \\
\text{s.t.} & \quad x_{k|k} = \hat{x}_{k|k} \\
\quad & x_{k+j+1|k} = f( x_{k+j|k} ) + g( x_{k+j|k} ) ( u_{k+j|k} + \hat{d}_{k|k} ) \\
& \quad u_{k+j|k} \in [a, b] \quad \forall j \in \{0, \ldots, N-1\} \\
& \quad y_{k+j|k} = h( x_{k+j|k} ) \quad \forall j \in \{0, \ldots, N\}.
\end{align*}
\]

(8.6)

As in Chapter 7, we also consider the Real-Time Iteration as an approximation of (8.6), see Appendix B. More detail about the offset-free MPC implementation for the compression system can be found in [109].

### 8.3 Simulation Results

The performance of the offset-free controllers is compared in closed-loop simulations to the industrial control presented in Section 6.2.3. The torque and valve coupled MPC controllers that we propose are designed to see with $N = 20$ prediction steps ahead. The output and input cost matrices, $Q_y = \text{diag}([50; 10])$ and $R = \text{diag}([1; 2])$, are designed such that the control aggressiveness is similar to the state-of-the-art control and in order to use mostly the torque input.

The simulations are performed starting from a steady-state point on the compressor map, corresponding to an outlet pressure of 1.3 bar,
Figure 8.1: Nominal case (first scenario) - Compressor map: the red line corresponds to the standard PID control, the dashed light green line corresponds to the linearized MPC, dash-dot blue to the nonlinear MPC and the yellow thick to the RTI. All of the controllers start from the same initial condition (×) and have the same target equilibrium (+).

Section 8.3.1 considers the nominal case, i.e., assuming exact knowledge of the model parameters in Section 6.2. In Section 8.3.2 typical disturbances and model mismatch are considered. Details about the required computational times are reported in Section 8.3.3.

8.3.1 Nominal case

As in the previous simulation of Section 7.4 the identified compressor map for the simulation test bench is given in Figure 8.1. Speed isolines computed from the compressor characteristics, \( \pi(\omega, m) \), are displayed for visualization purposes, as well as the Surge Line (SL) and the Choke Line (CL). The Surge Control Line (SCL) is drawn at a safety distance from the SL by a shift of an experimentally determined offset.

As a first scenario, we consider a target point located in a safe zone on the right side of the SCL, due to a large opening of the inlet and outlet valves. From Figure 8.1 we notice that the MPC based controllers have a similar closed-loop response as the conventional controller. Moreover,
the RTI gets a closed-loop response essentially equal to the nonlinear MPC in (8.6).

The industrial anti-surge control does not employ the recycle valve actuation at all, since the SCL is never crossed, and neither does the MPC, as a result of a cost matrix tuning that penalizes the usage of the emergency recycle valve. Therefore, the recycle mass flow rate \( m_r = 0 \).

A second scenario is displayed in Figure 8.2 where the target equilibrium is located on the SCL, due to different external conditions, i.e., throttle opening of the input and output valves. This desired steady-state operating point requires a positive recycle valve actuation, according to Section 6.2.2. This is due to smaller openings of the inlet and outlet valve compared to the previous case, to simulate a more extreme external process requirement. The corresponding closed-loop response is displayed in a compressor map in Figure 8.2 and the torque and recycle valve actuation in Figure 8.3.

In this case the conventional control and the linearized MPC move closer to the surge line, while the RTI and nonlinear MPC stay on the right side of the SCL by operating the recycle valve, as shown in Figure 8.3b. The RTI tracks the nonlinear MPC closely in closed loop. The differences between the linearized and nonlinear MPC are due to

**Figure 8.2:** Nominal case (second scenario) - Compressor map: the red line corresponds to the standard PID control, the dashed light green line corresponds to the linearized MPC, dash-dot blue to the nonlinear MPC and the yellow thick to the RTI. All of the controllers start from the same initial condition (×) and have the same target equilibrium (+).
8.3 Simulation Results

Figure 8.3: Nominal case (second scenario) - Torque and recycle valve actuation, details of the first seconds of the simulation: the red line corresponds to the standard PID control, the dashed light green line corresponds to the linearized MPC, dash-dot blue to the nonlinear MPC and the yellow thick to the RTI.

the more accurate dynamics employed in the prediction, and become more evident the further the current state is from the desired target. In fact, the linearization matrices in (8.1) are computed offline around the desired target.

Additional (soft) output constraints to keep away from the SL can be introduced in the problem. The closed loop dynamics does not change compared to the results in Figure 8.2 (the SL is never violated). Note that output constraints cannot be handled by the proposed algorithms in FalcOpt, but they can be easily taken into account by the SQP and Interior-Point Method solvers.

8.3.2 Sensitivity and robustness analysis

In industrial practice, the correct operation of the compression system is jeopardized by disturbances and possible model mismatch. The most critical disturbances are caused by unmodeled back pressure of the upstream and downstream processes. In fact, an increase of the outlet pressure, or equivalently a decrease of the inlet pressure, has the effect of increasing the pressure ratio, and hence acts to move the system closer to the surge line.

Some assumptions and approximations used in the modeling phase
lead typically to plant-model mismatches. For example, the nonlinear behavior of valves and the extrapolation of the compressor map to the left of the SL, which is typically not fully explored in the identification phase. Moreover, degradation and maintenance introduce slow time-varying effects that are usually neglected in the modeling. In order to investigate the influence of disturbances and plant-model mismatch, a sensitivity analysis is carried out and the results are compared to the nominal simulation case.

Back pressure of the upstream/downstream process is considered by adding in the second scenario of Figure 8.2 an unmodeled pressure in the outlet tank, $d_o \in \mathbb{R}_{>0}$ and in the inlet tank, $d_i \in \mathbb{R}_{<0}$. This disturbance is introduced as a state and output bias and it affects only the simulation plant model, as it is not considered in the control plant model. This has the effect of increasing the pressure ratio and therefore it brings the system closer to surge.

The results, relative to the second scenario, are displayed graphically in Figure 8.4 for $d_i = -d_o = -0.007$ bar. By comparing Figures 8.2 and 8.4 it is possible to see that, because of the introduced bias, the initial condition is now on the left-hand side of the SCL (the pressure ratio has increased to 1.412). This small initialization difference is quite critical, since the conventional control almost enters the surge zone, while all of the MPC controllers keep closer to the safe zone.

For a quantitative analysis of the results, the following performance metrics are introduced:

$$AIEu = \sum_k |u_k|$$

$$RIEy = \sum_k |y_k - \bar{y}|$$

$$SD_{\text{min}} = \min_k SD$$

with $k$ spanning the simulation time and where the Absolute Integral Error on each input, $AIEu$, according to (6.4) represents each actuator displacement, the Relative Integral Error on each output, $RIEy$, integrates the distance in transient from the desired reference $\bar{y}$, and the minimum Surge Distance, $SD_{\text{min}}$, is the minimum safety margin from the surge zone during the transient. Negative values of $SD_{\text{min}}$ indicate violation of the SL constraint.

Note that more positive $SD_{\text{min}}$ values indicate more effective surge protection compared to smaller or negative values, as can be also seen in
8.3 Simulation Results

Table 8.1: Performance metrics in the nominal case (benchmark)

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Conventional</th>
<th>Lin MPC</th>
<th>RTI</th>
<th>NL MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIEu(τ_d)</td>
<td>100.00</td>
<td>100.20</td>
<td>106.87</td>
<td><strong>105.23</strong></td>
</tr>
<tr>
<td>AIEu(k_r)</td>
<td><strong>100.00</strong></td>
<td>101.86</td>
<td>107.94</td>
<td>107.45</td>
</tr>
<tr>
<td>RIEy(p_o/p_i)</td>
<td>100.00</td>
<td>108.93</td>
<td><strong>91.51</strong></td>
<td>94.96</td>
</tr>
<tr>
<td>RIEy(SCD)</td>
<td>100.00</td>
<td>148.46</td>
<td><strong>36.74</strong></td>
<td>47.41</td>
</tr>
<tr>
<td>SD_min</td>
<td>100.00</td>
<td>161.87</td>
<td><strong>273.42</strong></td>
<td><strong>273.43</strong></td>
</tr>
</tbody>
</table>

the compressors maps. Table 8.1 shows the metrics in the nominal case (second scenario) and it serves as a reference for the sensitivity analysis. Each metric is reported as a percentage of the respective metric for the conventional controller, considered as the benchmark. For instance, the minimum surge distance achieved by RTI and nonlinear MPC is more than double (hence safer) that obtained by the conventional controller, as can also be seen in the compressor map of Figure 8.2. The best performance for every metric is highlighted in bold font.

The results for the constant pressure disturbance simulation are presented in Table 8.2. As before, each metric is expressed as a percentage of the corresponding metric in the nominal case for the conventional controller, set to 100% in Table 8.1. Although the controllers use the actuators to a similar extent (see AIEu(τ_d) and AIEu(k_r)), they yield different performance for the error on the Surge Control Distance (see RIEy(SCD)), and minimum surge distance, SD_min. The RTI and nonlinear MPC have Surge Control Distance values (RIEy(SCD)) more than 90% smaller, compared to the conventional controller in the same conditions. However, the minimum surge distance is obtained by the linearized MPC (even though the nonlinear MPC has a similar SD_min value), whereas from this indicator it is evident that the conventional controller is close to enter into the surge region.

A model mismatch analysis is formulated by introducing a deterministic error on the process model parameters. The distinction is made between dynamic parameters Θ_d, defining time constants of the differential equation in (6.1), e.g., V_i, V_o, J, and static parameters Θ_s, com-
prisining both the valve characteristics $\Theta_v$ and coefficients of maps $\Theta_m$. Thus, the sets of parameters are defined as follows:

$$
\begin{align*}
\Theta_d & := \begin{bmatrix} a_i & a_o & V_i & V_o & A & L & J & T_r \end{bmatrix}^T \\
\Theta_s & := \left[ \Theta_v ; \Theta_m \right] \\
\Theta_v & := \begin{bmatrix} k_i & k_o \end{bmatrix}^T \\
\Theta_m & := \begin{bmatrix} \theta_\tau & \theta_\pi \end{bmatrix}^T,
\end{align*}
$$

(8.8)

where $\theta_\tau$ are the coefficients for the torque map, $\tau_c(\omega, m)$, and $\theta_\pi$ the coefficients of the compressor map, $\pi(\omega, m)$. The representative perturbed element $\tilde{\theta}_j$ is then defined as

$$
\tilde{\theta}_j = \theta_j (1 \pm b),
$$

(8.9)

where $\theta_j$ is the corresponding parameter used for control design, and the sign of $\pm$ is always taken in the most critical direction, e.g., it is a minus for the outlet valve characteristics, $k_o$, since this moves the system closer to surge. The compressor map in $\Theta_m$ is offset and rotated,
Table 8.2: Performance metrics for constant pressure disturbance, in percentage with respect to Table 8.1, conventional control

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Conventional</th>
<th>Lin MPC</th>
<th>RTI</th>
<th>NL MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>disturbance ( d_i = -d_o = -0.007 ) bar</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIE( \text{Eu}(\tau_d) )</td>
<td>95.89</td>
<td>101.85</td>
<td>107.17</td>
<td>106.67</td>
</tr>
<tr>
<td>AIE( \text{Eu}(k_r) )</td>
<td>98.30</td>
<td>102.21</td>
<td>107.59</td>
<td>107.82</td>
</tr>
<tr>
<td>RIE( \text{Y}(p_o/p_i) )</td>
<td>111.99</td>
<td>108.40</td>
<td>93.40</td>
<td>92.53</td>
</tr>
<tr>
<td>RIE( \text{Y} )SCD)</td>
<td>144.41</td>
<td>158.40</td>
<td>52.04</td>
<td>49.23</td>
</tr>
<tr>
<td>( \text{SD}_{\text{min}} )</td>
<td>19.72</td>
<td>156.74</td>
<td>130.52</td>
<td>146.60</td>
</tr>
</tbody>
</table>

compared to the nominal case, in order to model typical interpolation errors.

Table 8.3 summarizes the results of the analysis, along with the corresponding perturbation magnitude, \( b \). Three separate case studies are considered:

- mismatch on dynamic parameters \( \Theta_d \),
- mismatch on valve characteristics \( \Theta_v \),
- mismatch on all the parameters \( \Theta_d, \Theta_v, \Theta_m \).

The results are again reported in percentage of the respective metric in the nominal case for the conventional controller.

As expected, the system is not particularly sensitive to variations of the dynamic parameters, especially for the MPC controllers. A factor \( b \) of 40% on every dynamic parameter is considered, with the sign in (8.9) arranged in the most critical way, i.e., such that it drives the system to instability. The benefits of the RTI and nonlinear MPC in terms of minimum Surge Distance, \( \text{SD}_{\text{min}} \), are maintained, as it is still at least 60% larger than the benchmark case. The conventional control enters surge, affecting the error on the SCD, i.e., high RIE\( \text{Y}(\text{SCD}) \) and negative \( \text{SD}_{\text{min}} \), while the linearized MPC yields a poor surge distance margin. It is also interesting to note that the RTI and nonlinear MPC use the torque actuation comparably to the linearized MPC, as can be seen from the metric AIE\( \text{Eu}(\tau_d) \), while achieving a smaller error RIE\( \text{Y}(p_o/p_i) \).
For variations of the valve characteristics, the changes are more noticeable. For a factor $b = 10\%$, all the controllers reach the desired operating point. The controllers use comparably the recycle valve actuation (see $\text{AIEu}(k_r)$), but the error on the Surge Control Distance, $\text{RIEY}(\text{SCD})$, is significantly larger for the conventional control and the linear MPC than for the RTI and nonlinear MPC. The latter controllers achieve a minimum SD, $\text{SD}_{\text{min}}$, that is only slightly deteriorated, compared to the nominal case, in contrast to the conventional controller and the linearized MPC, for which the corresponding reduction in margin are around $40\%$.

Finally, for a combined mismatch of all the parameters, the considered deviation factor is $b = 5\%$. For a fixed speed and mass flow rate, the actual compressor map has a lower pressure ratio than the nominal case, and similarly for the torque map. With this setting, all the MPC techniques achieve a sufficient minimum SD with relatively small recycle valve actuation, $\text{AIEu}(k_r)$. The conventional control does not enter surge, but it comes dangerously close to the surge line. As in the other case, the RTI and nonlinear MPC obtain a similar error on the SCD, $(\text{RIEY}(\text{SCD}))$ compared to the other techniques. They track the SCL very accurately during the transient. Apart from this metric, in this configuration the linearized MPC performs similarly as the RTI and nonlinear MPC.

Also in this sensitivity analysis, we remark that the RTI closely approximates the nonlinear MPC in term of the computed metrics.

### 8.3.3 Computational times

The nonlinear MPC has been solved by using Algorithm 5.1 and, as discussed in Section 5.4, the results are compared with the commercial software FORCES Pro and SNOPT. The computation times and number of iterations for the linearized MPC, RTI and nonlinear MPC are given in Tables 8.4 and 8.5. The best performance is highlighted in bold font. The KKT optimality tolerance for the solution of the optimization problem has been set to $\varepsilon = 10^{-6}$.

The proposed algorithm requires a very short time to solve the nonlinear MPC problem, that makes it one order of magnitude faster and perfectly suitable for real-time implementation. Remarkably, in this case we observe that the number of iterations required by Algorithm 5.1 to solve the MPC problem is comparable to SNOPT and even smaller than IPM. Note that in the formulation of Algorithm 5.1 neither slack
Table 8.3: Performance metrics for model parameter mismatch, in percentage with respect to Table 8.1, conventional control

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Conventional</th>
<th>Lin MPC</th>
<th>RTI</th>
<th>NL MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>mismatch on dynamic parameters, $\Theta_d$, $b = 40%$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIEu($\tau_d$)</td>
<td>180.31</td>
<td><strong>123.08</strong></td>
<td>131.91</td>
<td>128.97</td>
</tr>
<tr>
<td>AIEu($k_r$)</td>
<td>134.75</td>
<td><strong>133.25</strong></td>
<td>138.74</td>
<td>138.01</td>
</tr>
<tr>
<td>RIEy($p_o/p_i$)</td>
<td>111.74</td>
<td>114.58</td>
<td><strong>97.64</strong></td>
<td>100.99</td>
</tr>
<tr>
<td>RIEy(SCD)</td>
<td>3952.44</td>
<td>349.10</td>
<td>142.78</td>
<td><strong>111.50</strong></td>
</tr>
<tr>
<td>SD$\text{_{min}}$</td>
<td>$-182.73$</td>
<td>7.85</td>
<td>168.44</td>
<td><strong>187.90</strong></td>
</tr>
<tr>
<td>mismatch on valve characteristics, $\Theta_v$, $b = 10%$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIEu($\tau_d$)</td>
<td>100.15</td>
<td><strong>99.36</strong></td>
<td>106.26</td>
<td>104.93</td>
</tr>
<tr>
<td>AIEu($k_r$)</td>
<td><strong>164.62</strong></td>
<td>166.51</td>
<td>170.94</td>
<td>170.95</td>
</tr>
<tr>
<td>RIEy($p_o/p_i$)</td>
<td>100.01</td>
<td>111.91</td>
<td><strong>95.70</strong></td>
<td>98.56</td>
</tr>
<tr>
<td>RIEy(SCD)</td>
<td>144.06</td>
<td>205.98</td>
<td>30.63</td>
<td><strong>21.34</strong></td>
</tr>
<tr>
<td>SD$\text{_{min}}$</td>
<td>62.79</td>
<td>114.04</td>
<td>254.08</td>
<td><strong>266.06</strong></td>
</tr>
<tr>
<td>mismatch on all parameters, $\Theta_d$ and $\Theta_s$, $b = 5%$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIEu($\tau_d$)</td>
<td><strong>95.68</strong></td>
<td>95.72</td>
<td>103.28</td>
<td>101.84</td>
</tr>
<tr>
<td>AIEu($k_r$)</td>
<td><strong>131.42</strong></td>
<td>133.31</td>
<td>138.40</td>
<td>138.27</td>
</tr>
<tr>
<td>RIEy($p_o/p_i$)</td>
<td>100.18</td>
<td>110.10</td>
<td><strong>90.46</strong></td>
<td>93.33</td>
</tr>
<tr>
<td>RIEy(SCD)</td>
<td>137.57</td>
<td>183.87</td>
<td><strong>32.26</strong></td>
<td>41.65</td>
</tr>
<tr>
<td>SD$\text{_{min}}$</td>
<td>36.89</td>
<td>154.96</td>
<td>146.23</td>
<td><strong>148.21</strong></td>
</tr>
</tbody>
</table>
variables nor dual variable sequences are introduced for the problem, so less iterations are required for convergence.

It is not surprising that FORCES Pro may solve the nonlinear problem in a time comparable to the Linearized MPC and RTI, which are convex problems. In fact, the NL and convex optimization solvers implemented use different Interior-Point Methods and compute the Jacobian matrices differently [49].

The measured time for the solver SNOPT is overly long, more than a second in the worst-case, and clearly not suitable for real-time implementation. All of the other controllers are implementable in real-time (sampling time $T_s = 50\, \text{ms}$). We refer to [122] for an assessment of the computational times on an industrial PLC specifically suited for the control of the compression system.

8.4 Conclusion

In this chapter we have analyzed offset-free MPC schemes to control the compression system. Specifically, the goal is that the outputs track desired static references without offset. This is possible both in a linear and nonlinear MPC formulation. Further, we also analyze the RTI as an approximation of the nonlinear MPC.

The closed-loop performance of the MPC schemes introduced for the compression system is assessed in sensitivity tests, in which typical disturbances and a model mismatch are introduced. Here we see the benefits of MPC techniques, which guarantee safer conditions compared to the state-of-the-art control, according to the metrics considered. From the simulations we also observe that the nonlinear MPC and the RTI are by far superior in terms of performance to the linearized MPC.

The nonlinear MPC problem can be efficiently solved by the heuristic Algorithm [5.1]. The computational times obtained are about one order of magnitude smaller than commercial alternatives. However, the computational times obtained with the Interior-Point solver FORCES Pro are nevertheless suitable for a practical implementation.
8.4 Conclusion

Table 8.4: Computational times for the offset-free MPC of the compressor, tol. $\epsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method (Solver)</th>
<th>Avg. (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 5.1</td>
<td>0.0385</td>
<td>0.0141</td>
<td>0.144</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>353</td>
<td>31.2</td>
<td>2.01e3*</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>0.924</td>
<td>0.726</td>
<td>1.63</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>0.962</td>
<td>0.747</td>
<td>1.57</td>
</tr>
<tr>
<td>Lin MPC (FORCES Pro)</td>
<td>0.788</td>
<td>0.579</td>
<td>1.03</td>
</tr>
</tbody>
</table>

Table 8.5: Number of iterations to convergence for the offset-free MPC of the compressor, tol. $\epsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method (Solver)</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 5.1</td>
<td>5.49</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>6.38</td>
<td>2</td>
<td>30*</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>17.6</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>17.8</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>Lin MPC (FORCES Pro)</td>
<td>17.3</td>
<td>13</td>
<td>19</td>
</tr>
</tbody>
</table>

* The maximum number of iterations in SNOPT is set to 30 and it is reached in some MPC instances.
Part III

Further applications and conclusion
9 Further applications

In this chapter we present more Model Predictive Control applications and use the proposed algorithms to solve their optimization problems.

We start with the control of a power inverter, where very small sampling times and single precision computations are required. Model Predictive Control is used to actuate the switches of the inverter, based on a desired voltage and current reference. We note that few steps of the proposed algorithm yield the desired closed-loop performance, while the other considered commercial software cannot compute a meaningful solution in time. This is presented in Section 9.1.

Next, in Section 9.2 we consider another power electronics application: a trajectory planner for an induction motor. This is meant to be an online high-level control that determines the electric current set-points that minimize the power consumption of the induction motor, in order to reach a desired speed. This is particularly suited for traction applications and it can be realized via an MPC controller. A slower frequency can be considered and the computational times of the proposed algorithms are comparable with IPM.

Finally, our last example is the swing-up of an inverted pendulum. In Section 9.3 we consider the MPC with and without a terminal constraint. In the first case we apply the algorithms proposed in Chapters 2 and 3. In the second case we show the performance of the heuristic approach presented in Chapter 5. This example shows the limitations of the proposed algorithm (the method is outperformed by IPM) and highlights the differences between Algorithms 2.1 and 3.1.

9.1 Model Predictive Control at high frequency: Pulse-Width Modulation control of a power inverter

The control of the switches of a power inverter is an application where the proposed algorithms allow one to push Model Predictive Control to a high frequency. Power electronics applications typically impose very short sampling times, which make the MPC based approaches not always computationally affordable. Typical sampling frequencies are of
the order of kHz and the embedded hardware controlling these devices has limited computational power.

We consider a Pulse-Width Modulation (PWM) control of a voltage source inverter, connected via a resonant (LC) filter to a load. These systems are typically employed for conversion of energy from Direct Current to Alternating Current (DC/AC), such as in Uninterruptible Power Supplies (UPS) [91].

In this section we will only present the computational results obtained by the proposed algorithms. For the technical discussion of the modeling part and more detailed simulation results we refer to [5, 4].

The control problem of designing the PWM signal to track desired three-phase sinusoidal voltage and current references is recast as a convex MPC. Given an initial state \( x_{k+1} | k \in \mathbb{R}^{12} \) and input \( u := [u_1; u_2; u_3] \in \mathbb{R}^6 \):

\[
\begin{align*}
\min_{(x_{k+1} | k, u_{k+1} | k)} & \sum_{j=0}^{N-1} \frac{1}{2} (x_{k+j | k} - \bar{x}_{k+j | k})^\top Q (x_{k+j | k} - \bar{x}_{k+j | k}) \\
\text{s.t.} & x_{k+j+1 | k} = F x_{k+j | k} + G_{k+j | k} u_{k+j | k} \\
& u_{k+j | k} \in \mathcal{U} \quad \forall j \in \{0, \ldots, N-1\},
\end{align*}
\] (9.1)

with \( \mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \mathcal{U}_3 \) and

\[
\mathcal{U}_l = \left\{ u_l \in \mathbb{R}^2 \mid \|u_l - r_1\|_2^2 \leq R_1^2, \|u_l - r_2\|_2^2 \leq R_2^2 \right\},
\]

for all \( l \in \{1, 2, 3\} \). The state \( x \) contains the three-phase inverter currents and voltages, the input \( u \) is a nonlinear function of the pulse-width and pulse position, from which the PWM signal can be uniquely reconstructed. The dynamics are linear time-varying, with constant state matrix \( F \) and variable input matrix \( \{G_{k+j | k}\}_{j=0}^{N-1} \). Note that the set \( \mathcal{U} \) can be seen as the intersection of six circles, where each circle involves only two components of the 6-dimensional input vector. The reference \( \{\bar{x}_{k+j | k}\}_{j=0}^{N} \in \mathbb{R}^{12} \) is set such that the state tracks a desired sinusoidal. No contribution to the cost is considered for the bounded input \( u \in \mathcal{U} \). The prediction horizon employed for the problem is \( N = 2 \), which is clearly not sufficiently large to guarantee closed-loop stability. Since the computational requirements are very strict (see Section 9.1.2), we cannot afford to include stability constraints. This is typically the case in power electronics applications, see e.g., [85, 22].
9.1 Model Predictive Control at high frequency: Pulse-Width Modulation control of a power inverter

9.1.1 Handling the input constraints

In this section we show how to apply Algorithms 2.1 and 3.1 to solve the problem in (9.1). In particular, the projection onto the linearized constraints in (2.4) can be analytically derived, thus avoiding expensive numerical computations.

Let us define the input constraints in (9.1) in a compact form. First, we consider the predicted input vector $u_k := [u_{k|k}; \ldots; u_{k+N-1|k}]$ and define the input constraints in the form $g(u_k) := [g_0(u_{k|k}); \ldots; g_{N-1}(u_{k+N-1|k})] \leq 0$, where each element $g_j : \mathbb{R}^6 \rightarrow \mathbb{R}^6$ is defined as:

$$
g_j(u) = g_j \left( \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \right) := \frac{1}{2} \begin{bmatrix} \|u_1 - r_1\|_2^2 - R_1^2 \\ \|u_1 - r_2\|_2^2 - R_2^2 \\ \|u_2 - r_1\|_2^2 - R_1^2 \\ \|u_2 - r_2\|_2^2 - R_2^2 \\ \|u_3 - r_1\|_2^2 - R_1^2 \\ \|u_3 - r_2\|_2^2 - R_2^2 \end{bmatrix} \right),$$

$\forall j \in \{0, \ldots, N - 1\}$. Thus, the gradient is

$$\nabla g(u_k) = \text{blockdiag} \left( \nabla g_0(u_{k|k}), \ldots, \nabla g_{N-1}(u_{k+N-1|k}) \right),$$

with:

$$\nabla g_j(u) = \nabla g_j \left( \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \right) = \text{blockdiag}_l \left( \begin{bmatrix} u_1 - r_1 \\ u_2 - r_1 \\ u_3 - r_1 \end{bmatrix} \right),$$

for all $l \in \{1, 2, 3\}$. Note that the projection onto the linearization of the constraints (a polyhedron) could be in principle computable by partitioning the state-space. This procedure is not easily automatized and therefore we prefer to use the squared-slack variable formulation and sacrifice some computational performance.

As the slack variables, $y_k := [y_{k|k}; \ldots; y_{k+N-1|k}]$, are introduced such that $g(u_k) + \frac{1}{2} \text{diag}(y_k)y_k = 0$, according to the procedure in (4.1) the matrix that needs to be inverted at every step of the algorithm is of the form

$$\nabla g(u_k)^{\top} \nabla g(u_k) + \text{diag}(y_k)^2$$

$$= \text{blockdiag}_j \left( \nabla g_j(u_{k+j|k})^{\top} \nabla g_j(u_{k+j|k}) + \text{diag}(y_{k+j|k})^2 \right),$$
9 Further applications

for all $j \in \{0, \ldots, N-1\}$. Each of these block is a $6 \times 6$ matrix, which can be again decomposed in diagonal blocks:

$$
\nabla g_j(u)^{\top} \nabla g_j(u) + \text{diag}(y)^2 \\
= \text{blockdiag}_l \begin{pmatrix}
(u_l - r_1)^{\top}(u_l - r_1) + y_{l,1}^2 & (u_l - r_1)^{\top}(u_l - r_2) \\
(u_l - r_2)^{\top}(u_l - r_1) & (u_l - r_2)^{\top}(u_l - r_2) + y_{l,2}^2
\end{pmatrix},
$$

for all $l \in \{1, 2, 3\}$. Therefore, the matrix inversion reduces to the inversion of $2 \times 2$ matrices, which can be computed analytically. Finally, the projection onto the linearized constraints is derived by the equations in (4.4) and (4.5).

9.1.2 Simulation and computational results

The performance of the MPC control is analyzed in the numerical simulation in [5, Section V]. A transient from zero initial condition is considered, along with a sudden change in voltage reference that makes a 180 deg phase shift at the zero crossing. The numerical results are shown in Figure 9.1. We observe that the controller is able to track the desired voltage and current references and the effect of the disturbances is well mitigated.

The computational results are presented in Tables 9.1 and 9.2. The best and nearly best performances are highlighted in bold font. For real-time implementation, first the convergence tolerance is dropped to $10^{-3}$. The performance of Algorithms 2.1 and 3.1 is compared to FORCES Pro for the same problem. The times in Table 9.1 have been measured on the Windows PC specified in Section 5.4. Future work includes the testing of the algorithms on embedded devices.

The considered sampling and control frequency is $3.2$ kHz ($T_s \approx 300\mu s$) and the proposed algorithms require around one tenth of this time to solve the problem. For our practical purposes, we can limit the maximum number of iterations of the proposed algorithms to 50 iterations. After this number of iterations, in fact, the feasibility conditions are fulfilled and the closed-loop performance is acceptable for practical purposes, as it can be seen in the last column of Table 9.2 (also an harmonic analysis reveals a practically acceptable difference). The IPM implemented in FORCES Pro, instead, requires up to 14 interior-point iterations, and there is no possibility of substantially reducing the computational times by setting a maximum number of iterations. Note that embedded devices are typically about one order of magnitude slower than an off-the-shelf computer.
Further, despite the fact that both the proposed method and IPM in FORCES Pro \[49\] are implementable on embedded devices (they are written in low-level library-free C code and use static memory allocation), only the proposed algorithms work well with single precision. This typically yields a speed-up by around a factor two on embedded devices. In fact, the IPM typically suffers from the loss of precision introduced by float variables, in particular when computing the solution of linear systems, and this leads to a solver failure. Each iteration of the proposed algorithms, instead, requires only matrix products and vector sums, on which single precision has small impact.

9.2 Trajectory planner for an induction motor

In this section we discuss one further power electronics application of the proposed algorithm. We consider an induction motor, which we model with the following nonlinear system:

\[
\begin{align*}
\dot{x}_1 &= -ax_1 + b_1u_1 \\
\dot{x}_2 &= -\frac{k}{J}x_2 + \frac{1}{J}b_2x_1 u_2 - \frac{1}{J}T_{\text{load}},
\end{align*}
\]

Figure 9.1: Three-phase inverter actual (blue) and reference (red with dot markers) states: (a) voltages, (b) currents. At time \(t = 0.0078\) s a disturbance in terms of a sudden change in the voltage reference is introduced.
9 Further applications

Table 9.1: Computational times for the MPC of the converter, tol. \( \varepsilon = 10^{-3} \)

<table>
<thead>
<tr>
<th>Method (Solver)</th>
<th>Avg. (( \mu s ))</th>
<th>Best (( \mu s ))</th>
<th>Worst (( \mu s ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>22.4</td>
<td>17.1</td>
<td>32.3*</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 1 ) (FalcOpt)</td>
<td>27.2</td>
<td>20.9</td>
<td>49.0*</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 2 ) (FalcOpt)</td>
<td>23.0</td>
<td>18.2</td>
<td>31.1*</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = \infty ) (FalcOpt)</td>
<td>36.2</td>
<td>27.7</td>
<td>48.7*</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>344</td>
<td>278</td>
<td>503</td>
</tr>
</tbody>
</table>

Table 9.2: Number of iterations to convergence and closed-loop cost for the MPC of the converter, tol. \( \varepsilon = 10^{-3} \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>44.0</td>
<td>33</td>
<td>50*</td>
<td>0.681</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 1 ) (FalcOpt)</td>
<td>44.0</td>
<td>33</td>
<td>50*</td>
<td>0.681</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 2 ) (FalcOpt)</td>
<td>44.0</td>
<td>33</td>
<td>50*</td>
<td>0.681</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = \infty ) (FalcOpt)</td>
<td>43.9</td>
<td>33</td>
<td>50*</td>
<td>0.682</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>11.6</td>
<td>11</td>
<td>14</td>
<td>0.680</td>
</tr>
</tbody>
</table>

* The maximum number of iterations has been designed a posteriori after running the simulation to guarantee a closed-loop performance sufficiently close to the other NL solvers. The maximum error in the optimality conditions is \( \varepsilon_o = 4.1 \cdot 10^{-3} \) and the maximum error in the feasibility conditions is \( \varepsilon_f = 3.1 \cdot 10^{-6} \).
where \( a, b_1, b_2 \in \mathbb{R}_{>0} \) are the known parameters depending on the rotor resistance and on the impedances. The term \( k \in \mathbb{R}_{>0} \) is the friction coefficient of the motor and \( J \in \mathbb{R}_{>0} \) is the inertia of the rotor. We assume that the load torque, \( T_{\text{load}} \in \mathbb{R} \) is known and constant. A similar treatment would apply for a time-varying load torque.

The simplified model in (9.2) is obtained by expressing the quantities in the synchronous reference frame. We refer to [91] for a detailed explanation of the standard reference frames and the models of the induction motor. The state \( x_1 \) is the rotor flux, while \( x_2 \) is the absolute rotor speed. The input \( u \in \mathbb{R}^2 \) is the stator current in the direct and quadrature components.

Applying real-time Model Predictive Control directly to control the induction motor is considered prohibitive for the small sampling times, which in turn would impose a short prediction horizon. In the literature, Explicit Model Predictive Controllers have been considered [22, 85]. From the energy efficiency point of view we observe that a prediction horizon of around 100 ms gives the desired performance [113, 78]. On the other hand, the control frequency is 1 kHz \( (T_s = 1 \text{ ms}) \), therefore an online MPC with such a prediction horizon would not be feasible.

Therefore, we are interested in building an input trajectory planner for the induction motor, in order to minimize a cost function \( J(\cdot) \) that trades off actuator displacement (hence energy consumption) and tracking accuracy. The input trajectory will then be fed to low-level state-of-the-art controllers that track the desired reference [17].

The trajectory planner is cast as the following Model Predictive Control problem:

\[
\min_{(x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}} \sum_{j=0}^{N-1} \frac{1}{2} (x_{k+j|k} - \bar{x}_{k+j|k})^\top Q (x_{k+j|k} - \bar{x}_{k+j|k}) + \sum_{j=0}^{N-1} \frac{1}{2} u_{k+j|k}^\top R u_{k+j|k}
\]

s.t.
\[
x_{k+j+1|k} = f \left( x_{k+j|k}, u_{k+j|k} \right) \quad \forall j \in \{0, \ldots, N - 1\}
\]
\[
u_{k+j|k} \in \left[ a_{k+j|k}, b_{k+j|k} \right] \quad \forall j \in \{0, \ldots, N - 1\}
\]
\[
\frac{1}{2} \left\| u_{k+j|k} \right\|_2^2 \leq c_k \quad \forall j \in \{0, \ldots, N - 1\},
\]

where the dynamics \( f \) are obtained by discretization of the model in (9.2) and \( a_{k+j|k} < b_{k+j|k} \) for all \( j \); \( c_k \in \mathbb{R}_{>0} \) are parameters. Similar
to the power inverter example, we do not consider any constraint to
guarantee closed-loop stability.

The last quadratic constraints, bounding the magnitude of the sta-
tor currents, can be easily handled by Algorithms 2.1 and 3.1. By
using the standard compact notation, after including the dynam-
ics in the cost function as in (4.7), we express all the constraints
in the form

$$g(u_k) := \left[ g_0(u_{k|k}); \ldots; g_{N-1}(u_{k+N-1|k}) \right] \leq 0,$$

with $u_k := [u_{k|k}; \ldots; u_{k+N-1|k}]$. Each constraint $g_j : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{2n_u+1}$
is defined as:

$$g_j(u) = \begin{bmatrix} -u + a_{k+j|k} \\ u - b_{k+j|k} \\ \frac{1}{2} \|u\|^2 - c_k \end{bmatrix},$$

and, by introducing the slack variables $y = [y_a; y_b; y_c]$, we observe that
the matrix inverse required in the procedure in Section 4.1 is obtained
by inverting the following block matrices:

$$(\nabla g_j(u)^\top \nabla g_j(u) + \text{diag}(y)^2)^{-1}$$

$$= \begin{bmatrix} I + \text{diag}(y_a)^2 & -I & -u \\ -I & I + \text{diag}(y_b)^2 & u \\ -u & u & u^\top u + y_c^2 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} D_j + B_j + r_j(B_ju)(B_ju)^\top & D_j - r_j(B_ju)(A_ju)^\top & r_jB_ju \\ D_j - r_j(A_ju)(B_ju)^\top & D_j + A_j + r_j(A_ju)(A_ju)^\top & -r_jA_ju \\ r_j(B_ju)^\top & -r_j(A_ju)^\top & r_j \end{bmatrix},$$

where, similarly to (4.8),

$$D_j = \text{diag}_l \left( \frac{1}{y_{a,l}^2 + y_{b,l}^2 + y_{a,l}^2 y_{b,l}} \right),$$

$$A_j = \text{diag}_l \left( \frac{y_{a,l}^2}{y_{a,l}^2 + y_{b,l}^2 + y_{a,l}^2 y_{b,l}} \right),$$

$$B_j = \text{diag}_l \left( \frac{y_{b,l}^2}{y_{a,l}^2 + y_{b,l}^2 + y_{a,l}^2 y_{b,l}} \right),$$

for all $l \in \{1, \ldots, n_u\}$, and

$$r_j = \left( \sum_{l=1}^{n_u} \frac{y_{a,l}^2 y_{b,l}}{y_{a,l}^2 + y_{b,l}^2 + y_{a,l}^2 y_{b,l}} u_l^2 + y_c^2 \right)^{-1}. \quad (9.3)$$
9.2 Trajectory planner for an induction motor

9.2.1 Simulation and computational results

The numerical simulations in this section are based on a realistic induction motor with the parameters given in Appendix E. The goal of the control is to track a desired speed $\bar{x}_2 = 80 \text{ rad/s}$, with a load torque $T_{load} = 0.3 \text{ Nm}$. The sampling time is set to $T_s = 10 \text{ ms}$ and the prediction horizon of the MPC is $N = 10$ steps. We do not track a specific rotor flux, $\bar{x}_1$, thus the state matrix $Q = \text{diag}([0; 1])$, while the input matrix is $R = \text{diag}([0.1; 0.1])$. The input constraint parameters are set to $a_{k+j|k} = [0.1; 0.1]$, $b_{k+j|k} = [3.89; 3.89]$ for all $k,j$, while $c_k = 7.57$ for all $k$. In this simulation we assume the ideal condition where the optimal inputs, resulting from the MPC, are directly applied to the motor, i.e., the low-level control is able to perfectly track the desired reference.

As before, we consider both full nonlinear solutions of the MPC problems and approximate solutions. Along with the classic Real-Time Iteration (see Appendix B) we consider a modified version that accounts for the quadratic constraints (similar to Section 7.4). In Figure 9.2 the state trajectory is shown. What we notice is that the RTI is not appropriate to solve this problem. In fact, only a linearization of the quadratic input constraint is considered in the resulting approximated MPC problem, thus to ensure feasibility of the solution, a correction of the computed input is required. The input vector is scaled in order to keep it inside the feasible circle. Thus, the control is not able to track the desired speed and it chatters as it can be seen from the flux dynamics $x_1$ (this is the reason for the different steady-state point that we observe in Figure 9.2). The RTI with Quadratic Constraint, instead, accounts efficiently for this and is a good approximation to the nonlinear solution.

We now present the computational aspects of the considered control problems. In particular, we show the computational times and number of iterations for two different tolerances on the KKT conditions: $\epsilon = 10^{-3}$ in Table 9.3 and Table 9.4 and $\epsilon = 10^{-6}$ in Table 9.5 and Table 9.6. The best and nearly best performances are highlighted in bold font. We compare the performance with the commercial solvers SNOPT and FORCES Pro, as discussed in Section 5.4.

When the desired tolerance is set to $10^{-3}$, the proposed algorithms outperform the other exact and approximate solutions (Table 9.3). As in the application presented in Section 9.1 the solve times for the proposed algorithms are in the order of tens of $\mu$s. The worst-case was
The proposed algorithms it is at least three times slower than the Interior-Point Method in FORCES Pro NL (Table 9.5). The RTI schemes perform even faster and, for the results shown in Figure 9.2, the RTI with QC seems the best strategy for this tolerance. The result is due to the large number of iterations required by the proposed algorithms to reach the desired tolerance, which increase by one order of magnitude. Note that the strategy $p = \infty$ requires in the worst-case double the number of iterations compared to the other choices of $p$, while the best variant among the proposed algorithms on average and in the worst-case for this example is Algorithm 2.1. This is driven not only by a smaller number of iterations required, but also by a smaller average of line search iterates.
per iteration \( i \) (1.66 versus 1.85 for \( p = 1 \) or \( p = 2 \) and 2.08 for \( p = \infty \)). By comparing the times and the number of iterates we also observe that the computational time per iterate of Algorithms 2.1 and 3.1 with \( p = 2 \) is slightly faster than what obtained by Algorithm 3.1 with \( p = 1 \) and \( p = \infty \).

9.3 Control of an inverted pendulum

We consider the inverted pendulum shown in Figure 9.3 consisting of a rod of length \( l = 0.3 \) m with mass \( m = 0.2 \) kg concentrated at the tip and no friction acting on the cart and swing. The mass of the cart is \( M = 0.5 \) kg and the gravitational acceleration \( g = 10 \) m/s\(^2\). The states \( x_1 \) and \( x_2 \) are respectively the cart position and velocity and \( x_3 \) and \( x_4 \), the pendulum angle and angular velocity. The input \( u \) is the applied force on the cart and it is subject to box constraints, \([-10, 10]\) N. We discretize the continuous-time dynamics

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{mg \sin x_3 \cos x_3 - mlx_4^2 \sin x_3 + u}{M + m \sin^2(x_3)} \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{g}{l} \sin x_3 + \frac{mg \sin x_3 \cos^2 x_3 + u \cos x_3 - mlx_4^2 \sin x_3 \cos x_3}{l (M + m \sin^2 x_3)},
\end{align*}
\]

with the explicit Euler method with sampling time \( T_s = 0.077 \) s. The system has two sets of unforced equilibria, the unstable ones \([p; 0; 2k\pi; 0]\), and the stable ones \( x = [p; 0; \pi + 2k\pi; 0] \), with \( p \in \mathbb{R} \) and \( k \in \mathbb{Z} \). Physically, the former correspond to the pendulum in the upright position, while the latter in the natural upside-down configuration. The goal of the controller is to stabilize the system around the origin, that is, to the unstable equilibrium, starting from the stable one at \( x_0 := [0; 0; \pi; 0] \).
Further applications

Table 9.3: Computational times for the trajectory planner for an induction motor. Tolerance \( \varepsilon = 10^{-3} \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>0.0579</td>
<td>0.0232</td>
<td>0.222</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 1 ) (FalcOpt)</td>
<td>0.0715</td>
<td>0.0319</td>
<td>0.293</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 2 ) (FalcOpt)</td>
<td>0.0497</td>
<td>0.0220</td>
<td>0.147</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = \infty ) (FalcOpt)</td>
<td>0.124</td>
<td>0.0532</td>
<td>0.255</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>26.8</td>
<td>15.6</td>
<td>46.8</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>0.154</td>
<td>0.138</td>
<td>0.323</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>0.117</td>
<td>0.103</td>
<td>0.188</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>0.115</td>
<td>0.101</td>
<td>0.294</td>
</tr>
</tbody>
</table>

Table 9.4: Number of iterations to convergence for the trajectory planner for an induction motor. Tolerance \( \varepsilon = 10^{-3} \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>49.8</td>
<td>23</td>
<td>141</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 1 ) (FalcOpt)</td>
<td>49.4</td>
<td>23</td>
<td>182</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = 2 ) (FalcOpt)</td>
<td>46.1</td>
<td>23</td>
<td>87</td>
</tr>
<tr>
<td>Algorithm 3.1, ( p = \infty ) (FalcOpt)</td>
<td>48.6</td>
<td>25</td>
<td>90</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>6.86</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>14.8</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>19.0</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>16.6</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>
9.3 Control of an inverted pendulum

Table 9.5: Computational times for the trajectory planner for an induction motor. Tolerance $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>0.721</td>
<td>0.160</td>
<td>1.59</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>1.10</td>
<td>0.156</td>
<td>2.45</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>0.798</td>
<td>0.0734</td>
<td>1.85</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>2.21</td>
<td>0.333</td>
<td>7.00</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>42.8</td>
<td>15.6</td>
<td>78.0</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>0.205</td>
<td>0.139</td>
<td>0.606</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>0.131</td>
<td>0.114</td>
<td>0.214</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>0.117</td>
<td>0.0988</td>
<td>0.459</td>
</tr>
</tbody>
</table>

Table 9.6: Number of iterations to convergence for the trajectory planner for an induction motor. Tolerance $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>625</td>
<td>114</td>
<td>1056</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>769</td>
<td>110</td>
<td>1108</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>775</td>
<td>67</td>
<td>1109</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>793</td>
<td>119</td>
<td>2371</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>15.4</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>14.8</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>16.8</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>16.6</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>
9 Further applications

Figure 9.3: Scheme of the inverted pendulum (source: Wikipedia).

9.3.1 Case with terminal constraint

We consider an MPC of the form (4.6), i.e.

\[
\min \left\{ \sum_{j=0}^{N-1} \left( \frac{1}{2} x_{k+j|k}^T Q x_{k+j|k} + \frac{1}{2} u_{k+j|k}^T R u_{k+j|k} \right) + \frac{1}{2} x_{k+N|k}^T P x_{k+N|k} \right\}
\]

s.t.

\[
x_{k+j+1|k} = f \left( x_{k+j|k}, u_{k+j|k} \right) \quad \forall j \in \{0, \ldots, N-1\}
\]

\[
u_{k+j|k} \in \left[ a_{k+j|k}, b_{k+j|k} \right] \quad \forall j \in \{0, \ldots, N-1\}
\]

\[
\frac{1}{2} x_{k+N|k}^T P x_{k+N|k} \leq c_k,
\]

where the model \( f \) indicates the discrete-time model obtained by Euler discretization of (9.4). The desired closed-loop performance is achieved with a prediction horizon of \( N = 8 \) and the cost matrices are \( Q = \text{diag}([10; 0.1; 100; 0.1]) \) and \( R = 1 \). The terminal cost matrix, \( P \), is determined by the Algebraic Riccati Equation using linearized dynamics around the desired equilibrium.

In this example the constant \( c_k \) in the terminal constraint is set to \( c_k = 1.5 \forall k \), so that such constraint is active at the first MPC instances. A rule with variable \( c_k \) can be found in [33].

For the specific problem considered here, the RTI effectively stabilizes the pendulum, but its closed-loop cost is larger than that obtained by a full nonlinear solution, as it can be seen from Figure 9.4, and it has the advantage of requiring low computational times. A similar con-
sideration holds for the RTI with QC, where considering the terminal constraint has limited impact on the closed-loop performance. Further, the linearization of the terminal constraint may result to be infeasible, even though the terminal constraint is feasible for the original nonlinear problem. Thus, a reformulation with soft constraints is required.

The performance of the proposed algorithms is compared to SNOPT and FORCES Pro, as well as the RTI and the RTI with Quadratic Constraint, as discussed in Section 5.4. The computational results are reported in Tables 9.7 and 9.8. The best and nearly best performances are highlighted in bold font.

In the implementation we have noticed that the solver SNOPT requires an overly long time to solve the first optimization problem. As the level of optimality does not reach the desired tolerance, the solver exceeded the maximum number of major iterations (150) in the first MPC instance, thus requiring more than 1 second for the solution of the problem. The average and worst-case times in Table 9.7 for the SQP solver do not account for this first MPC instance. The problem does not occur in the proposed algorithms with the same initialization.

In this case, the dynamics are highly nonlinear and contain sinusoids, which are expensive to compute. As a result, Algorithms 2.1 and 3.1 require a number of iterations to converge that is larger than IPM by 1 or even 2 orders of magnitude. Note that because of the slow final convergence, we need to set a maximum number of iterations. The $\ell_p$ merit functions (Algorithm 3.1) here allow one to reduce the maximum number of iterations to yield the desired closed-loop performance. Because of the fewer iterations required for convergence, we observe that Algorithm 3.1 is significantly faster than Algorithm 2.1. In particular, for $p = 1$, the former is more than 4 times faster than the latter. Even though in this case the proposed algorithms are not as fast as the Interior-Point Method and RTI, they are close in terms of computational time.

Finally, as discussed before we observe that the convergence rate can be fast when far from the solution and become slower once it is close to the KKT point. Figure 9.5 shows, as an example, the error on the optimality conditions, which using the notation in Section 4.1, is defined as:

$$\varepsilon_\alpha(v) := \|\nabla J(v) + \nabla p(v)\mu\|_2.$$  

This is computed via the iterate norm $d_v$, as a function of the iteration number for a particular MPC instance. Note that the gradient step size $\alpha$ in (2.4) is not tuned according to Theorem 2.1 as discussed in
9 Further applications

Table 9.7: Computational times for control of the inverted pendulum, with terminal constraint, tol. $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method (Solver)</th>
<th>Avg. (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>3.05</td>
<td>0.411</td>
<td>4.74**</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>0.768</td>
<td>0.228</td>
<td>1.20**</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>1.189</td>
<td>0.176</td>
<td>2.10**</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>1.36</td>
<td>0.233</td>
<td>1.72**</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>9.66*</td>
<td>4.00</td>
<td>370*</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>0.127</td>
<td>0.0779</td>
<td>0.451</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>0.160</td>
<td>0.107</td>
<td>0.244</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>0.169</td>
<td>0.120</td>
<td>0.221</td>
</tr>
</tbody>
</table>

Table 9.8: Number of iterations to convergence for control of the inverted pendulum, with terminal constraint, tol. $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.1 (FalcOpt)</td>
<td>2.55e3</td>
<td>370</td>
<td>3600**</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 1$ (FalcOpt)</td>
<td>565</td>
<td>143</td>
<td>750**</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = 2$ (FalcOpt)</td>
<td>978</td>
<td>162</td>
<td>1200**</td>
</tr>
<tr>
<td>Algorithm 3.1, $p = \infty$ (FalcOpt)</td>
<td>731</td>
<td>118</td>
<td>750**</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>2.89*</td>
<td>1</td>
<td>15*</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>11.9</td>
<td>7</td>
<td>46</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>19.9</td>
<td>14</td>
<td>21</td>
</tr>
<tr>
<td>RTI with QC (FORCES Pro)</td>
<td>19.4</td>
<td>13</td>
<td>20</td>
</tr>
</tbody>
</table>

* The first MPC optimization in SNOPT exceeds the maximum number of major iterations and is not considered in the average and worst-case times.

** The maximum number of iterations has been designed a posteriori after running the simulation to guarantee a closed-loop performance sufficiently close to the other NL solvers. The maximum error in the optimality conditions is $\varepsilon_o = 8.1 \cdot 10^{-2}$ and the maximum error in the feasibility conditions is $\varepsilon_f = 2.2 \cdot 10^{-5}$. 
9.3 Control of an inverted pendulum

![Diagram](image)

**Figure 9.4:** Swing-up simulation of an inverted pendulum, case with terminal constraint: (a) state dynamics, where the angle is in blue and the cart position in light green; (b) input: force applied on the cart. The solid lines show the proposed gradient algorithm solution and they overlap with those of the Sequential Quadratic Programming and the Interior-Point Method solution. The dash-dot lines show the Real-Time Iteration solution, the dotted lines show the Real-Time Iteration with Quadratic Constraint solution.

Chapter 4 As in standard convex optimization theory, the linear rate factor depends on second-order information and, in our case, a large condition number of the Hessian of the Lagrangian at the solution can yield a slow linear convergence in practice. This does not seem to be related to the Maratos effect, as it also takes place when using the Augmented Lagrangian as merit function.

One possible way to overcome this effect is to combine this method with Sequential Quadratic Programming. In fact, the proposed method can be seen as a warm start for superlinearly convergent methods. By running the proposed algorithm up to a larger than desired tolerance (e.g., $\varepsilon = 10^{-3}$), we have a good estimate of the active set at the solution (see Lemma 2.2), thus we can evaluate the Hessian of the Lagrangian, or a suitable approximation. This second-order information makes the convergence rate superlinear and would allow us to get to a critical point in tens of iterations (instead of thousands).
9 Further applications

![Graph showing error as a function of iteration number]

Figure 9.5: Error on the optimality conditions as a function of the iteration number for a particular MPC instance. The desired tolerance is set to $10^{-6}$.

9.3.2 Case without terminal constraint

Without terminal constraint, the MPC problem is:

$$
\min_{(x_{k+j+1|k}, u_{k+j|k})} \sum_{j=0}^{N-1} \left\{ \frac{1}{2} x_{k+j|k}^T Q x_{k+j|k} + \frac{1}{2} u_{k+j|k}^T R u_{k+j|k} \right\} + \frac{1}{2} x_{k+N|k}^T P x_{k+N|k}
$$

subject to:

$$
x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}) \quad \forall j \in \{0, \ldots, N-1\}
$$

$$
u_{k+j|k} \in [a_{k+j|k}, b_{k+j|k}] \quad \forall j \in \{0, \ldots, N-1\},
$$

with the same prediction horizon and objective function as before.

In Figure 9.4 it is shown the closed-loop dynamics of the full nonlinear MPC solution and the Real-Time Iteration. For the structure of the constraints in this case we use Algorithm 5.1 to solve nonlinear problem. The closed-loop performance is very similar to the case with the terminal constraint.

The computational times obtained in the simulations of Figure 9.4 are given in Table 9.9 for a tolerance on the KKT optimality $\varepsilon = 10^{-6}$. The best performance is highlighted in bold font. For the case without terminal constraint, the proposed algorithm instead is superior both to SNOPT and comparable to FORCES Pro NL. We remind that Algorithm 5.1 does not introduce either slack variables or a sequence of...
9.4 Conclusion

In this chapter we have considered more applications for the theoretical algorithms presented in the thesis. In particular, the algorithm is very suited for time-critical power electronics applications. Here we consider two examples, in particular the control of a power converter and a trajectory planner for an induction motor. The proposed method is fast at getting close to the desired optimal point and, in case of tolerances of the order of $10^{-3}$, a critical point is determined much faster than other (even approximate) solvers. Additional advantages for a practical implementation on microcontrollers is the possibility to work with single precision, which is in general not an option with more complex methods.

Finally, we present the limitations of the algorithm in the simulations of an inverted pendulum. The example is particular instructive.
### Table 9.9: Computational times for control of the inverted pendulum, without terminal constraint, tol. $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 5.1</td>
<td>0.0600</td>
<td>0.00532</td>
<td>0.234</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>27.0</td>
<td>15.6</td>
<td>109</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>0.152</td>
<td>0.0726</td>
<td>0.382</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>0.0696</td>
<td>0.0479</td>
<td>0.133</td>
</tr>
</tbody>
</table>

### Table 9.10: Number of iterations to convergence for control of the inverted pendulum, without terminal constraint, tol. $\varepsilon = 10^{-6}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg.</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 5.1</td>
<td>38.6</td>
<td>3</td>
<td>171</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>9.45</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>IPM (FORCES Pro NL)</td>
<td>9.36</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>RTI (FORCES Pro)</td>
<td>8.50</td>
<td>7</td>
<td>14</td>
</tr>
</tbody>
</table>
because the dynamics are highly nonlinear and fairly computationally expensive to evaluate, because of the trigonometric operations. Here we make a distinction between the methods presented in Chapters 2 and 3 and the heuristic presented in Chapter 5. For the reasons presented above, the former at a strict tolerance require a large number of iterations to converge, and therefore the methods are slower compared to the Interior-Point solver considered. Some differences in terms of computationally performance arise based on the $p$-norm considered in Algorithm 3.1 but all of the proposed methods anyway converge within the required sampling time. For the heuristic algorithm, instead, even at a strict tolerance the convergence is anyway at least one order of magnitude faster than the other solvers.
10 Conclusion

In this thesis we have introduced an optimization method for the Nonlinear Programs arising in Model Predictive Control problems. The method consists of taking a gradient step from the current iterate and projecting it onto a linearization of the constraints. This first-order approach does not fall into any of the existing optimization methods, since it is a combination of Steepest Descent Method and Sequential Quadratic Programming.

This thesis presents results in three directions: (i) we establish the convergence theory of the method and show its convergence properties, (ii) we develop algorithms exploiting the structure of a class of nonlinear Model Predictive Control problems, for which the projected gradient and constraint linearization method has low computational complexity, (iii) we show in several control applications the performance of the method in terms of computational time.

Theory

The simplest implementation of the projected gradient and constraint linearization method with unit merit function step size is only locally convergent, as shown in Section 2.4.1. This result is derived by showing that close to the optimum the method behaves as a contractive mapping for a sufficiently small gradient step size, hence it converges with linear rate. Interestingly, we can find a parallelism between the Steepest Descent and the proposed method. In fact, a bound on the gradient step size guaranteeing convergence can be derived based on the Lipschitz constant of the gradient of the Lagrangian.

The approach can be modified to guarantee global convergence via the introduction of merit functions, as presented in Sections 2.4.2 and 3.3. Any step of the optimization method is then designed to decrease the merit function. In fact, a local minimum of the merit function can be related to a critical point of the problem, satisfying optimality conditions. We have established global convergence with two different choices of the merit functions: the augmented Lagrangian and the $\ell_p$ penalty function. The differences are mostly in the properties
of the function and the technical assumptions required for the convergence result. Based on the choice of the merit function, we then derive two algorithms to practically solve optimization problems.

### Computation

The proposed algorithms are particularly suited for a specific class of MPC problems. In particular we require that the projection onto the linearization of the constraints can be computed analytically. This is the case for a larger class of problems than what is typically considered in the Steepest Descent Method. We can consider nonlinear dynamics, box and low-dimensional nonlinear input constraints and a terminal or contractive stability constraint (but, for instance, similarly to Gradient Method, stage-wise state constraints are computationally expensive to embed).

We also devise a third algorithm that, for only box constrained MPC problems, does not require the introduction of a merit function. This algorithm yields a good practical performance, even though there are no theoretical guarantees about its convergence properties.

The automatic code generation tool FalcOpt (First-order Algorithm via Linearization of Constraints for OPTimization) allows one to implement the proposed algorithms to solve an MPC problem. By using this tool, in this thesis we have considered several applications to validate the method.

### Applications

We have analyzed the problem of controlling compression systems for chemical industry or gas transportation. We have designed a contractive MPC to ensure stability guarantees and observed that the proposed optimization method is numerically efficient. The proposed algorithms are about one order of magnitude faster than the other software using Interior-Point Method or Sequential Quadratic Programming. As an auxiliary result, we have also presented an offline numerical procedure to ensure recursive feasibility of the contractive MPC. We have also designed offset-free linear and nonlinear MPC controllers for compression systems, which have the practical benefit of being robust to typical disturbances and parameter mismatch. We observe that the heuristic algorithm solves the nonlinear problem more than one order of magnitude faster than the other commercial software considered.
The proposed method has the interesting feature of getting close to the solution of the problem in a small number of iterations. The final convergence to the desired tolerance however may take many iterations, since as a first-order method it is prone to zigzagging effects. This feature makes the proposed algorithms particularly efficient in time-critical applications, such as the ones arising in power electronics. We can afford to control the power inverter switches or the induction motor currents with low computation times, reaching sampling frequencies that are not achieved by the other considered solvers. Further, working in single precision has a limited practical impact, compared e.g., to the Interior-Point Method, where numerical problems make the solver fail to deliver any usable solution.

Finally, we observe the behavior of the algorithms in a typical control application such as the inverted pendulum. If the nonlinearity of the problem is strong, it may happen that many iterations may be required for convergence. Even though the dynamics are quite expensive to compute and consequently the proposed algorithms perform worse compared to the commercial software considered, we note that the algorithms still converge and the computational times are anyways below the required sampling times.

From our numerical assessment we conclude that the proposed algorithms outperform the other optimization methods if the desired tolerance is large (e.g., $\varepsilon = 10^{-3}$) and the dynamics are cheap to evaluate. In fact, a small tolerance may require many iterations for convergence, due to the linear convergence rate. The dynamics are evaluated at every iteration, therefore if they are computationally expensive, this will affect the computational speed.

Among the proposed merit functions, there is not a universal best. The augmented Lagrangian and the $\ell_2$ non-smooth exact penalty function yield a better performance on average, while the $\ell_1$ penalty function is a safer choice in strongly nonlinear problems. The $\ell_\infty$ merit function, instead, performs consistently worse than the other variants in our numerical tests.

There are three possible directions for future research. From the theoretical perspective, it would be interesting to drop the assumption of feasibility of each iterate in Section A. This would require a relaxation of the mathematical formulation to determine a new iterate that still reduces the merit function. The main challenge, though, is to keep the
projection easy-to-compute, in order to keep the numerical complexity low.

The tool FalcOpt can be enriched with an automatic problem detection, that can choose for the user the algorithm most suited for the specific problem, e.g., the heuristic algorithm when only box constraints are present. In order to improve the performance in presence of small desired tolerance or highly nonlinear dynamics, the proposed method could be used as a warm-start for a superlinearly convergent method such as Sequential Quadratic Programming.

Finally, the performance achieved in the power electronics applications suggests that the method has most potential with time critical applications. Besides analyzing control problems of other electronic circuits, the implementation of the proposed algorithms on high-performance integrated circuits, such as Field Programmable Logic Arrays (FPGAs), could open new frontiers in high frequency nonlinear Model Predictive Control.
A Review of the Sequential Quadratic Programming method

The Sequential Quadratic Programming (SQP) method updates the sequence \( \{z^{(i)}\}_i \) via the solution of a sequence of Quadratic Programs (QPs). In particular, given the current iterate \( z^{(i)}, i \in \mathbb{N} \), the method generates the QP

\[
d_{z}^{(i)} := \arg \min_{dz} \frac{1}{2} d_z^\top H^{(i)} d_z + \nabla J(z^{(i)})^\top d_z
\]

s.t. \( g(z^{(i)}) + \nabla g(z^{(i)})^\top d_z \leq 0 \)

\[
h(z^{(i)}) + \nabla h(z^{(i)})^\top d_z = 0,
\]

where \( H^{(i)} \) is either the exact Hessian of the Lagrangian in (2.2) or an appropriate approximation. The dual variables \( \lambda^{(i)}_{QP} \in \mathbb{R}_m^p \) and \( \nu^{(i)}_{QP} \in \mathbb{R}_p^p \) are associated with the inequality and equality constraints, respectively. The resulting KKT conditions for the QP in (A.1) are:

\[
H^{(i)} d_{z}^{(i)} + \nabla J(z^{(i)}) + \nabla g(z^{(i)}) \lambda^{(i)}_{QP} + \nabla h(z^{(i)}) \nu^{(i)}_{QP} = 0
\]

\[
\text{diag}(\lambda^{(i)}_{QP}) \left( g(z^{(i)}) + \nabla g(z^{(i)})^\top d_{z}^{(i)} \right) = 0
\]

\[
g(z^{(i)}) + \nabla g(z^{(i)})^\top d_{z}^{(i)} = 0
\]

\[
h(z^{(i)}) + \nabla h(z^{(i)})^\top d_{z}^{(i)} = 0.
\]

Based on the solution \( d_{z}^{(i)} \) to (A.1), the sequence is updated as

\[
 z^{(i+1)} := z^{(i)} + t^{(i)} d_{z}^{(i)},
\]

where \( t^{(i)} \in (0, 1] \) is a merit function step size to be determined.

To prove convergence results for the SQP methods, some regularity and boundedness assumptions are typically considered [60, Assumptions (i)-(iii)].

Assumption A.1. The matrices \( \{H^{(i)}\}_{i=1}^\infty \) are positive definite, with bounded condition number, and smallest eigenvalue uniformly bounded away from zero, i.e., \( \exists \gamma \in \mathbb{R}_{>0} \) such that, for all \( i \in \mathbb{N} \),

\[
d^\top H^{(i)} d \geq \gamma \|d\|_2^2 \quad \forall d \in \mathbb{R}^n.
\]
This assumption may be relaxed in reduced Hessian methods (see e.g., [14]). In the projected gradient and constrained linearization method this assumption is not required, since only first-order information is used.

**Assumption A.2.** For all $i \in \mathbb{N}$, the QP in (A.1) is feasible.

This assumption is typically satisfied in the practical problem that we aim to solve and it allows us to simplify the mathematical treatment (see Chapter 4). If the assumption does not hold true, then standard SQP arguments can be used to derive an iterate $d_z$ that solves a relaxed problem [13].

**Assumption A.3.** For all $i \in \mathbb{N}$, let $\mathcal{I}_{QP}(d_z^{(i)}, z^{(i)})$ denote the index set of the active inequality constraints in (A.1) parametric in $z^{(i)}$, i.e.,

$$
\mathcal{I}_{QP}(d_z^{(i)}, z^{(i)}) := \left\{ j \in \{1, \ldots, m\} \mid g_j(z^{(i)}) + \nabla g_j(z^{(i)})^\top d_z^{(i)} = 0 \right\}.
$$

Then $d_z^{(i)}$ is regular, i.e., the matrix made up of $\nabla h(z^{(i)})$ along with the columns $\nabla g_j(z^{(i)})$, $j \in \mathcal{I}_{QP}(d_z^{(i)}, z^{(i)})$, has full column rank. Further, strict complementarity holds.

Note that this linear independence constraint qualification (LICQ) assumption implies that the dual variables $\lambda_{QP}^{(i)}, \nu_{QP}^{(i)}$ in (A.2) are bounded and unique.

**Assumption A.4.** There exists a compact set $\Omega \subset \mathbb{R}^n$, such that for all $i \in \mathbb{N}$, $z^{(i)}, z^{(i)} + d_z^{(i)} \in \Omega$.

This is a standard assumption in Sequential Quadratic Programming methods. It guarantees a number of properties, such as the existence of at least an accumulation point of the sequence $(z^{(i)})_i$.

**Assumption A.5.** The functions $J$, $g$, $h$, and their first and second derivatives are uniformly bounded in norm in $\Omega$.

Several possible choices for the Hessian $H^{(i)}$ have been considered in the literature. By setting $H^{(i)}$ as the Hessian of the Lagrangian of (2.1) and unit merit function step size, local convergence to the desired $z^*$ is achieved with a quadratic rate [61, 116, 117].
Other choices make the computation of $H^{(i)}$ less expensive, but deteriorate the convergence speed. See [34] for a general overview of superlinear convergence theorems for SQP methods.

To ensure global convergence to a critical point, a merit function in the form of an augmented Lagrangian is considered:

$$L_{\text{aug}}(z, \lambda, \nu, s, \rho) := J(z) + (g(z) + s)^\top \lambda + h(z)^\top \nu + \frac{\rho}{2} \|g(z) + s\|_2^2 + \frac{\rho}{2} \|h(z)\|_2^2, \quad (A.4)$$

where $\rho \in \mathbb{R}_{\geq 0}$ is a penalty parameter to be determined and $s \in \mathbb{R}^m_{\geq 0}$ is a vector of slack variables, defined at the beginning of each iteration $i$ such that its $j$-th component satisfies the following equation [60, Equation (2.8)]:

$$s_j^{(i)} := \begin{cases} \max(-g_j(z^{(i)}), 0) & \text{if } \rho = 0 \\ \max(-g_j(z^{(i)}) - \frac{\lambda_j}{\rho}, 0) & \text{otherwise.} \end{cases} \quad (A.5)$$

When the parameter $\rho$ is nonzero, the vector $s$ as defined in (A.5) yields the value of $L_{\text{aug}}$ minimized with respect to the slack variables alone, i.e., $\partial L_{\text{aug}}/\partial s = 0$, subject to the non-negativity constraint $s \in \mathbb{R}^m_{\geq 0}$.

Several possibilities for the design of the dual variables $\lambda$ and $\nu$ have been considered in the literature. In [18, 104] a least square estimate, function of $z$ and dependent on the Jacobian matrices of the objective and constraints, is proposed and global convergence properties are proved, although this approach is computationally expensive. In fact, the derivative of the merit function will then depend on the second-order derivatives of the objective and the constraints, which are typically expensive to evaluate.

This is partially alleviated in [68, 103], where the multipliers $\lambda_{\text{QP}}$ and $\nu_{\text{QP}}$ are used as constant estimates of the dual variables, thereby reducing the computational burden. On the other hand, this has the effect of redefining the merit function and leads to theoretical difficulties when proving global convergence.

The approach we follow in this appendix builds upon that in [60]. Therein, $\lambda$ and $\nu$ are considered as additional variables, updated with merit function step size $t^{(i)}$ along with the primal sequence $(z^{(i)})_i$. 
Specifically, in view of [60], we consider the iterative update
\[
\begin{bmatrix}
z^{(i+1)} \\
\lambda^{(i+1)} \\
\nu^{(i+1)} \\
s^{(i+1)}
\end{bmatrix} := \begin{bmatrix}
z^{(i)} \\
\lambda^{(i)} \\
\nu^{(i)} \\
s^{(i)}
\end{bmatrix} + t^{(i)} \begin{bmatrix}
d^{(i)}_z \\
d^{(i)}_\lambda \\
d^{(i)}_\nu \\
d^{(i)}_s
\end{bmatrix},
\] (A.6)
where \(d^{(i+1)}_z\) is from (A.1), \(d^{(i)}_\lambda := \lambda^{(i)}_{QP} - \lambda^{(i)}\), \(d^{(i)}_\nu := \nu^{(i)}_{QP} - \nu^{(i)}\) and the slack variation \(d^{(i)}_s\) satisfies
\[
g(z^{(i)}) + \nabla g(z^{(i)})^\top d^{(i)}_z + s^{(i)} + d^{(i)}_s = 0.
\] (A.7)

Then we define the function \(\phi : \mathbb{R} \to \mathbb{R}\) as
\[
\phi(t) := \mathcal{L}_{\text{aug}}(z + td, \lambda + td\lambda, \nu + td\nu, s + td, \rho)
\] (A.8)
to determine the merit function step size \(t\), e.g., via a backtracking line search starting from \(t = 1\), that satisfies the Wolfe conditions [44, 98]:
\[
\begin{align*}
\phi(t) - \phi(0) &\leq \sigma_1 t \phi'(0) \quad \text{(A.9a)} \\
|\phi'(t)| &\leq -\sigma_2 \phi'(0) \quad \text{or} \quad (t = 1 \text{ and } \phi'(1) \leq -\sigma_2 \phi'(0)), \quad \text{(A.9b)}
\end{align*}
\]
for some \(0 < \sigma_1 \leq \sigma_2 < \frac{1}{2}\). For ease of notation, we avoid making explicit the dependence of \(\phi\) on the arguments \(z, \lambda, \nu, s\) of the augmented Lagrangian function \(\mathcal{L}_{\text{aug}}\).

Note that if the derivative \(\phi'(0)\) is negative, then there exists a merit function step size \(t^{(i)} \in (0, 1]\) such that the conditions in (A.9) hold. The condition on the derivative \(\phi'(0)\) is checked numerically at every iteration \(i \in \mathbb{N}\) via the inequality condition
\[
\phi'(0) \leq -\frac{1}{2}(d^{(i)}_z)^\top H^{(i)} d^{(i)}_z.
\] (A.10)
If this latter inequality does not hold true, then the parameter \(\rho\) is adjusted. In particular, there exists a lower bound \(\hat{\rho} \in \mathbb{R}_{\geq 0}\) such that the inequality in (A.10) holds for all \(\rho \geq \hat{\rho}\) [60, Lemma 4.3].

For the practical implementation of the algorithm, the line search in (A.9) is typically simplified in order to check only the first condition in (A.9a) [43, 99]. This has the effect of reducing the computational burden required to compute the derivative \(\phi'(t)\) and it does not impede convergence of the algorithm in practice. To derive the merit function step size \(t\), a backtracking line search is employed with safeguarded polynomial interpolation [77].

The SQP steps for the NLP in (2.1) are summarized in Algorithm A.1.
Algorithm A.1 Sequential Quadratic Programming

Initialize $i \leftarrow 0$, $z^{(0)} \in \mathbb{R}^n$ and $\rho^{(0)} = 0$

repeat

Compute $d_z^{(i)}$ in (A.1) and $\lambda_{QP}^{(i)}$, $\nu_{QP}^{(i)}$ such that (A.2) holds

if $d_z^{(i)} = 0$ then

Set $z^* = z^{(i)}$, $\lambda^* = \lambda_{QP}^{(i)}$, $\nu^* = \nu_{QP}^{(i)}$ and STOP

else

if $i = 0$ then

Set $\lambda^{(0)} = \lambda_{QP}^{(0)}$, $\nu^{(0)} = \nu_{QP}^{(0)}$

end if

Set $d_{\lambda}^{(i)} = \lambda_{QP}^{(i)} - \lambda^{(i)}$, $d_{\nu}^{(i)} = \nu_{QP}^{(i)} - \nu^{(i)}$

end if

Determine $s^{(i)}$ and $d_s^{(i)}$ from (A.5) and (A.7)

Set $\rho^{(i)} \in \mathbb{R}_{\geq 0}$ such that (A.10) holds

Determine the merit function step size $t^{(i)}$ that satisfies (A.9), e.g., via line search

Update $z^{(i+1)}$, $\lambda^{(i+1)}$, $\nu^{(i+1)}$, $s^{(i+1)}$ in (A.6)

$i \leftarrow i + 1$

until Convergence

return $z^*$, $\lambda^*$ and $\nu^*$
B Review of the Real-Time Iteration

In this appendix we review the Real-Time Iteration, which is an approximate method to solve nonlinear MPC problems. Here we mainly follow the approach in [45, 46, 47, 48], which is specifically tailored for nonlinear MPC. The general idea of the method is to stop prematurely the Sequential Quadratic Programming algorithm before it converges to a critical point of the nonlinear problem. Usually, only one iteration is enough to derive a satisfactory approximation of the critical point and it requires significantly less computational time. Therefore, in this appendix we will consider the case in which we perform only one iteration, but the same treatment applies otherwise.

Let us consider the following MPC problem:

$$\begin{align*}
\min_{(x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}} & \sum_{j=0}^{N-1} \ell(x_{k+j|k}, u_{k+j|k}) + \ell_N(x_{k+N|k}) \\
\text{s.t.} & \quad x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}) \forall j \in \{0, \ldots, N-1\} \\
& \quad n(x_{k+j|k}, u_{k+j|k}) \leq 0 \forall j \in \{0, \ldots, N-1\},
\end{align*}$$

(B.1)

where $\ell$ is a nonlinear objective, $n : \mathbb{R}^{nx} \rightarrow \mathbb{R}^n$ are stage-wise constraints and $f$ is a discretized dynamics. Alternatively, multiple-shooting methods can be employed to integrate the continuous-time dynamics.

At time $t_k$, instead of solving the full nonlinear problem, the RTI considers another optimization problem, obtained as an approximation of (B.1). Specifically, the objective function is a second-order approximation of the Lagrangian of the problem and the constraints are obtained via linearization of the constraints of (B.1). In particular, starting from a sequence of states and inputs $\left(x_{k+j+1|k}, u_{k+j|k}\right)_{j=0}^{N-1}$, we look for the update $\left(\Delta x_{k+j+1|k}, \Delta u_{k+j|k}\right)_{j=0}^{N-1}$ that solves the following optimiza-
B Review of the Real-Time Iteration

tion problem:

\[
\begin{align*}
\min_{\Delta x_{k+j+1|k}, \Delta u_{k+j|k}} & \sum_{j=0}^{N-1} \left\{ \frac{1}{2} \begin{bmatrix} \Delta x_{k+j|k} \\ \Delta u_{k+j|k} \end{bmatrix}^{\top} H_{k+j|k} \begin{bmatrix} \Delta x_{k+j|k} \\ \Delta u_{k+j|k} \end{bmatrix} \\
+ \begin{bmatrix} \frac{\partial \ell}{\partial x} (x_{k+j|k}, u_{k+j|k}) \\ \frac{\partial \ell}{\partial u} (x_{k+j|k}, u_{k+j|k}) \end{bmatrix}^{\top} \begin{bmatrix} \Delta x_{k+j|k} \\ \Delta u_{k+j|k} \end{bmatrix} \right\} \\
+ \frac{1}{2} \Delta x_{k+N|k}^{\top} H_{k+N|k} \Delta x_{k+N|k} + \frac{\partial \ell}{\partial x} (x_{k+N|k})^{\top} \Delta x_{k+N|k} & \text{s.t.} \\
\Delta x_{k|k} = 0 \\
\Delta x_{k+j+1|k} & = \left[ \begin{bmatrix} \frac{\partial f}{\partial x} (x_{k+j|k}, u_{k+j|k}) \\ \frac{\partial f}{\partial u} (x_{k+j|k}, u_{k+j|k}) \end{bmatrix} \begin{bmatrix} \Delta x_{k+j|k} \\ \Delta u_{k+j|k} \end{bmatrix} + f (x_{k+j|k}, u_{k+j|k}) - x_{k+j+1|k} \forall j \in \{0, \ldots, N-1\} \\
\begin{bmatrix} \frac{\partial n}{\partial x} (x_{k+j|k}, u_{k+j|k}) \\ \frac{\partial n}{\partial u} (x_{k+j|k}, u_{k+j|k}) \end{bmatrix}^{\top} \begin{bmatrix} \Delta x_{k+j|k} \\ \Delta u_{k+j|k} \end{bmatrix} + n (x_{k+j|k}, u_{k+j|k}) & \leq 0 \forall j \in \{0, \ldots, N-1\}, \right. \\
\end{align*}
\]

(B.2)

where \( H_{k+j|k} \) is the matrix approximating the Hessian of the La-grangian of (B.1) around \((x_{k+j+1|k}, u_{k+j|k})\) and \( H_{k+N|k} \) around \((x_{k+N|k}, u_{k+N|k})\). Once that the solution to (B.2) is determined, the sequence of iterates \((x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}\) is updated as follows:

\[
\begin{bmatrix} x_{k+j+1|k} \\ u_{k+j|k} \end{bmatrix} \leftarrow \begin{bmatrix} x_{k+j+1|k} \\ u_{k+j|k} \end{bmatrix} + t \begin{bmatrix} \Delta x_{k+j+1|k} \\ \Delta u_{k+j|k} \end{bmatrix} \forall j \in \{0, \ldots, N-1\}.
\]

The step size \( t \in (0, 1] \) is derived via a line search on a merit function, see (A.9) for SQP. Alternatively, a trust-region approach can be considered, where the step size \( t \) is replaced by an additional trust-region constraint in (B.2) [119]. The initial sequence of \((x_{k+j+1|k}, u_{k+j|k})_{j=0}^{N-1}\) is typically derived based on the solution of the MPC problem computed at the previous time step \(k-1\). At the very first iterate, \(k = 1\), the sequence can be either initialized with all zeros, or with a pre-computed solution of the nonlinear MPC problem.

Bounds on the loss of optimality of (B.1) compared to (B.2) are derived for optimal control problems in [46]. However, this proof is
obtained for the case without stage-wise constraints. Additional details can be considered, such as using in (B.2) the linearization matrices computed at the time step $k - 1$, in order to reduce the preparation time before solving the optimization problem. In this work though we do not introduce any error due to the pre-computation of the linearization matrices, i.e., the matrices are computed at time step $k$ just before solving the MPC problem in (B.2).
C Technical proofs

Proof of Theorem 2.3. It follows from Theorem 2.2 and Lemma 2.2 that if \( \|d_z^{(i)}\|_2 = 0 \), then, by (2.3), (2.16) and (2.9), \( \lambda_G^{(i)} = \lambda^* \) and \( \nu_G^{(i)} = \nu^* \), and the algorithm terminates. Thus, we will assume henceforth that \( \|d_z^{(i)}\|_2 \neq 0 \) for all \( i \in \mathbb{N} \). By defining, for all \( k \leq i \in \mathbb{N} \),

\[
\gamma_{k,i} := \begin{cases} \bar{t}(i) & \text{if } k = i \\ \bar{t}(k) \prod_{m=k+1}^i (1 - \bar{t}(m)) & \text{otherwise,} \end{cases}
\]

with \( \bar{t}(0) := 1 \) and \( \bar{t}(k) = t(k) \), the definition in (2.25) implies that

\[
\lambda^{(i+1)} = \sum_{k=0}^i \gamma_{k,i} \lambda^{(k)}_G, \quad (C.1)
\]

for all \( i \in \mathbb{N} \), because of the initial condition \( \lambda^{(0)} = \lambda^{(0)}_G \). Then, by Lemma 2.7 we have

\[
0 < \bar{t} \leq \bar{t}(i) \leq 1 \quad \forall i \in \mathbb{N} \quad (C.2a)
\]

\[
\sum_{k=0}^i \gamma_{k,i} = 1 \quad (C.2b)
\]

\[
\gamma_{k,i} \leq (1 - \bar{t})^{i-k} \quad \forall k < i. \quad (C.2c)
\]

Since \( z^{(i)} \rightarrow z^* \), the iterates will reach a neighborhood of \( z^* \) where the problem in (2.8) identifies the correct active set (Lemma 2.2) and the active constraint will have full column rank. Assume that the property holds for \( i > \tilde{i} \). From (2.9), the definition of \( \lambda_G^{(i)} \) and Assumption 2.2 for \( i \geq \tilde{i} \) there exists \( M \in \mathbb{R}_{>0} \) such that:

\[
\lambda_G^{(i)} = \lambda^* + M^{(i)} d^{(i)} v^{(i)}, \quad (C.3)
\]

with \( |M^{(i)}| \leq M, d^{(i)} = \max \left\{ \|d_z^{(i)}\|_2, \|z^* - z^{(i)}\|_2 \right\} \) and \( \|v^{(i)}\|_2 = 1 \).

For any given \( \varepsilon \in \mathbb{R}_{>0} \), Theorem 2.2 implies that \( i_1 \) can be chosen such that for all \( i \geq i_1 \), we have

\[
|M^{(i)} d^{(i)}| \leq \varepsilon/2. \quad (C.4)
\]
Then, we define the iteration index $i_2$ such that for all $i \geq i_2$, we have

\[(1 - \bar{t})^i \leq \frac{\varepsilon}{2(i + 1)(1 + \hat{\lambda}_G + \|\lambda^*\|_2)}, \quad (C.5)\]

with $\hat{\lambda}_G$ being an upper bound to $\|\lambda^{(i)}_G\|_2$ for all $i$. Now let $\bar{i} := \max\{i_1, i_2\}$. Then, from (C.1) and (C.3), for all $i \geq \bar{i}$ we have

$$\lambda^{(i+1)} = \sum_{k=0}^{i} \gamma_{k,i} \lambda_G^{(k)} + \sum_{k=i+1}^{i} \gamma_{k,i} \left( \lambda^* + M^{(k)} d^{(k)} v^{(k)} \right).$$

Hence it follows from (C.2b) that

$$\lambda^{(i+1)} - \lambda^* = \sum_{k=0}^{i} \gamma_{k,i} \lambda_G^{(k)} - \lambda^* + \sum_{k=i+1}^{i} \gamma_{k,i} M^{(k)} d^{(k)} v^{(k)}.$$

Since the dual variables $\lambda_G^{(k)}$ and $v^{(k)}$ are bounded in norm, it follows that

$$\left\| \lambda^{(i+1)} - \lambda^* \right\|_2 \leq (\hat{\lambda}_G + \|\lambda^*\|_2) \sum_{k=0}^{\bar{i}} \gamma_{k,i} + \sum_{k=i+1}^{i} \gamma_{k,i} \left| M^{(k)} d^{(k)} \right|.$$  (C.6)

For all iterations $i \geq 2\bar{i}$, it follows from (C.2a) and (C.2c) that

$$\sum_{k=0}^{\bar{i}} \gamma_{k,i} \leq \sum_{k=0}^{\bar{i}} (1 - \bar{t})^{\bar{i} - k} \leq \sum_{k=0}^{\bar{i}} (1 - \bar{t})^{\bar{i} - k} \leq (\bar{i} + 1)(1 - \bar{t})^{\bar{i}}.$$

By using (C.5), we derive the bound $\left( \hat{\lambda}_G + \|\lambda^*\|_2 \right) \sum_{k=0}^{\bar{i}} \gamma_{k,i} \leq \frac{1}{2} \varepsilon$. on the first term on the right-hand side of (C.6), and to bound the second term in (C.6), we use (C.2b) and (C.4):

$$\sum_{k=i+1}^{i} \gamma_{k,i} \left| M^{(k)} d^{(k)} \right| \leq \frac{1}{2} \varepsilon \sum_{k=i+1}^{i} \gamma_{k,i} \leq \frac{1}{2} \varepsilon.$$  (C.7)

Finally, by combining (C.6) and (C.7), we obtain that for all $\varepsilon$, there exists $\tilde{i}$ such that

$$\left\| \lambda^{(i)} - \lambda^* \right\|_2 \leq \varepsilon \text{ for all } i \geq 2\tilde{i} + 1,$$

which implies that $\lim_{i \to \infty} \left\| \lambda^{(i)} - \lambda^* \right\|_2 = 0$. The convergence of the dual variables for the equality constraint is analogous. \Box
Proof of Lemma 2.9. By continuity of the second derivative we can derive the following relation between the objective function evaluations:

\[
J(z + d_z) = J(z) + \frac{1}{2} \nabla J(z)^\top d_z + \frac{1}{2} \nabla J(z)^\top d_z + \frac{1}{2} d_z^\top \nabla^2 J(z) d_z \\
+ o \left( \|d_z\|^2 \right)
\]

\[
= J(z) + \frac{1}{2} \nabla J(z)^\top d_z + \frac{1}{2} \left( \nabla J(z + d_z) - \nabla^2 J(z) d_z \right)^\top d_z + \frac{1}{2} d_z^\top \nabla^2 J(z) d_z \\
+ o \left( \|d_z\|^2 \right)
\]

\[
= J(z) + \frac{1}{2} \left( \nabla J(z) + \nabla J(z + d_z) \right)^\top d_z + o \left( \|d_z\|^2 \right),
\]

(C.8)

and analogously for the constraint functions \(g\) and \(h\):

\[
g(z + d_z) = g(z) + \frac{1}{2} \left( \nabla g(z) + \nabla g(z + d_z) \right)^\top d_z + o \left( \|d_z\|^2 \right) \quad \text{(C.9)}
\]

\[
h(z + d_z) = h(z) + \frac{1}{2} \left( \nabla h(z) + \nabla h(z + d_z) \right)^\top d_z + o \left( \|d_z\|^2 \right) \quad \text{(C.10)}
\]

By Assumption 2.5 we have that

\[
\nabla J(z + d_z) = \nabla J(z^*) + \nabla^2 J(z^*) (z + d_z - z^*) + o \left( \|z + d_z - z^*\|_2 \right)
\]

\[
= \nabla J(z^*) + o \left( \|d_z\|_2 \right)
\]

\[
\nabla g(z + d_z) = \nabla g(z^*) + o \left( \|d_z\|_2 \right)
\]

\[
\nabla h(z + d_z) = \nabla h(z^*) + o \left( \|d_z\|_2 \right),
\]

(C.11)

hence substituting into (C.8), (C.9), (C.10) we obtain:

\[
J(z + d_z) = J(z) + \frac{1}{2} \left( \nabla J(z) + \nabla J(z^*) \right)^\top d_z + o \left( \|d_z\|^2 \right)
\]

\[
g(z + d_z) = g(z) + \frac{1}{2} \left( \nabla g(z) + \nabla g(z^*) \right)^\top d_z + o \left( \|d_z\|^2 \right) \quad \text{(C.12)}
\]

\[
h(z + d_z) = h(z) + \frac{1}{2} \left( \nabla h(z) + \nabla h(z^*) \right)^\top d_z + o \left( \|d_z\|^2 \right).
\]

Let us use the simplified notation and denote \(J(z)\) as \(J\), \(J(z + d_z)\) as \(J^+\) and \(J(z^*)\) as \(J^*\) (and similarly for \(g\) and \(h\) and their gradients). By
By Lemma 2.4, the derivative \( \phi'(0) \) can be recast as
\[
\phi'(0) = d_z^T \nabla J - 2(g + s)^T \lambda - (\nabla g^T d_z + d_s^T) \lambda_G - \rho \|g + s\|^2 - 2h^T \nu \\
+ h^T \nu_G - \rho \|h\|^2 ,
\]
which when substituted in (C.13) yields
\[
\phi(1) - \phi(0) = \frac{1}{2} \phi'(0) + \frac{1}{2} (\nabla J^* + \nabla g^* \lambda_G + \nabla h^* \nu_G) d_z + \frac{1}{2} \lambda_G d_s \\
+ o \left( \|d_z\|^2 \right) .
\]
Next we note that Assumption 2.5 implies specific conditions on the convergence of the variables \( \lambda_G \) and \( \nu_G \). That is,
\[
\lambda_G - \lambda^* = d_\lambda + \lambda - \lambda^* = o \left( \|\lambda - \lambda^*\|_2 \right) = o \left( \|d_\lambda\|_2 \right) = o \left( \|d_z\|_2 \right) \\
\nu_G - \nu^* = o \left( \|d_z\|_2 \right) .
\]
Therefore, by using the KKT conditions in (2.3) we have

$$
\nabla J(z^\star) + \nabla g(z^\star)\lambda_G + \nabla h(z^\star)\nu_G = \nabla J(z^\star) + \nabla g(z^\star)\lambda^* + \nabla h(z^\star)\nu^* \\
+ \nabla g(z^\star) (\lambda_G - \lambda^*) + \nabla h(z^\star) (\nu_G - \nu^*) = o(\|dz\|_2).
$$

Since $\lambda_G^T d_s \leq 0$ (where equality holds when the correct active set is identified, see the proof of Lemma 2.10), then

$$
\phi(1) - \phi(0) \leq \frac{1}{2} \phi'(0) + \frac{1}{2} \lambda^T_G d_s + o\left(\|dz\|_2^2\right) \leq \frac{1}{2} \phi'(0) + o\left(\|dz\|_2^2\right).
$$

To prove that eventually $\phi(1) - \phi(0) - \sigma_1 \phi'(0) \leq 0$ with $\sigma_1 \in (0, 1/2)$, from the previous inequality we have that $\phi(1) - \phi(0) - \sigma_1 \phi'(0) \leq \left(\frac{1}{2} - \sigma_1\right) \phi'(0) + o\left(\|dz\|_2^2\right) \leq 0$ for all sufficiently large $i$, since by Lemma 2.4 the term $\phi'(0)$ is upper bounded by a negative term proportional to $\|dz\|_2^2$ and the term $o\left(\|dz\|_2^2\right)$ vanishes at least quadratically fast as $i$ approaches infinity.

Proof of Lemma 2.10. Let us use the simplified notation and denote $J(z)$ as $J$, $J(z + dz)$ as $J^+$ and $J(z^\star)$ as $J^\star$ (and similarly for $g$ and $h$ and their gradients).

By definition, the derivative $\phi'(1)$ is:

$$
\phi'(1) = d_z^T (\nabla J^+ + \nabla g^+ \lambda_G + \nabla h^+ \nu_G + \rho \nabla g^+ (g^+ - g - \nabla g^T dz)) \\
+ \rho \nabla h^+ h^+) + d_A^T (g^+ - g - \nabla g^T dz) + d_H^T h^+ + d_s^T (\lambda_G + \rho (g^+ - g - \nabla g^T dz)),
$$

and by (C.11) we have

$$
\phi'(1) = d_z^T (\nabla J^* + \nabla g^* \lambda_G + \nabla h^* \nu_G) + \rho d_z^T \nabla g^+ (g^+ - g - \nabla g^T dz) \\
+ d_s^T (\lambda_G + \rho (g^+ - g - \nabla g^T dz)) + o\left(\|dz\|_2^2\right),
$$

where we have used the fourth condition in Assumption 2.5 and the following relations resulting from (2.4) and Taylor expansions:

$$
g^+ - g - \nabla g^T dz = o(\|dz\|_2), \\
h^+ = h + \nabla h^T dz + o(\|dz\|_2) = o(\|dz\|_2).
$$
Moreover, by (2.17) $d_s = -\nabla g^\top d_z - (g + s)$, hence,
\[
\phi(1) = d_z^\top (\nabla J^* + \nabla g^* \lambda_G + \nabla h^* \nu_G) + d_s^\top \lambda_G + \\
\rho (d_z^\top (\nabla g^+ - \nabla g) - (g + s)^\top) (g^+ - g - \nabla g^\top d_z) + o \left( \|d_z\|_2^2 \right).
\]

From (2.12) and Assumption 2.5, $\|g + s\|_2 = O(\|d_z\|_2)$ and $d_s^\top \lambda_G = 0$ when the correct active set is determined, see Lemma 2.2. In fact, if $g_j < 0$ for some $j$ at $z^*$ then $\lambda_{G,j} = 0$. Then, if $\rho = 0$, $s_j = -g_j$. Otherwise, if $\rho \in \mathbb{R}_{>0}$, as $\lambda_j$ converges to $\lambda_j^* = 0$, for sufficiently large iterates $i$ we have that $0 \leq \lambda_j < -\rho g_j$. Then, by (2.12) we have
\[
s_j = -g_j - \frac{\lambda_j}{\rho} \quad \Rightarrow \quad g_j + s_j = -\frac{\lambda_j}{\rho} = -\frac{\lambda_j - \lambda_j^*}{\rho} = O(\|d_z\|_2).
\]

If $g_j = 0$, then $s_j = 0$ and $\lambda_{G,j} > 0$ by Assumption 2.4. By the complementarity conditions in (2.9) and (2.12) it results that $\lambda_{G,j}(s_j + d_{s,j}) = 0$, thus $d_{s,j} = 0$. Therefore, in any case we conclude that
\[
|\rho (d_z^\top (\nabla g^+ - \nabla g) - (g + s)) (g^+ - g - \nabla g^\top d_z) | = o \left( \|d_z\|_2^2 \right). \quad (C.15)
\]

Since by Assumption 2.5 the quantity $\nabla J^* + \nabla g^* \lambda_G + \nabla h^* \nu_G$ is $o(\|d_z\|_2)$, then $\phi'(1) = o(\|d_z\|_2^2)$. By Lemma 2.4, $|\phi'(0)| \geq \frac{1}{2\alpha} \|d_z\|_2^2$, thus (A.9b) will eventually be satisfied at every iteration.

**Proof of Lemma 3.1.** First, we note that the directional derivative of $\|g(z)^+\|_p$ along $d_z$, with $p \geq 1$ is:
\[
\lim_{t \to 0^+} \frac{\|g(z + td_z)^+\|_p - \|g(z)^+\|_p}{t}
\]
\[
= \lim_{t \to 0^+} \frac{\left( \sum_j \max \left( g_j(z + td_z), 0 \right)^p \right)^{\frac{1}{p}}}{t} - \frac{\left( \sum_j \max \left( g_j(z), 0 \right)^p \right)^{\frac{1}{p}}}{t}
\]
\[
= \lim_{t \to 0^+} \frac{\left( \sum_j \max \left( g_j(z) + t\nabla g_j(z)^\top d_z + O(t^2 \|d_z\|_2^2), 0 \right)^p \right)^{\frac{1}{p}}}{t}
\]
\[
- \frac{\left( \sum_j \max \left( g_j(z), 0 \right)^p \right)^{\frac{1}{p}}}{t},
\]
\[
(C.16)
\]
where we have respectively applied the definition of $\|\cdot\|_p$ and then expanded the function $g$ to the second-order. Now, based on the sign of $g_j(z)$ and by considering the feasibility condition $\nabla g_j(z)^\top d_z \leq -g_j(z)$, there are three possibilities:

- $g_j(z) < 0$: then, $\max \left( g_j(z), 0 \right) = 0$. By definition of big $O$, there exist $b_j \in \mathbb{R}_{>0}$ $\forall j \in \{1, \ldots, m\}$ such that

$$
g_j(z) + t\nabla g_j(z)^\top d_z + O(t^2 \|d_z\|_2^2) \leq g_j(z) - tg_j(z) + b_j t^2 \|d_z\|_2^2 \leq g_j(z) - t \left( g_j(z) - b_j t \|d_z\|_2^2 \right).$$

Note that the quantity in the brackets is negative for any value of $t$. Therefore, for a sufficiently small $t$:

$$\max \left( g_j(z) + t\nabla g_j(z)^\top d_z + O(t^2 \|d_z\|_2^2), 0 \right) = 0;
$$

- $g_j(z) = 0$: then, $\max \left( g_j(z), 0 \right) = 0$ and there exist $b_j \in \mathbb{R}_{>0}$ $\forall j \in \{1, \ldots, m\}$ such that:

$$
\max \left( g_j(z) + t\nabla g_j(z)^\top d_z + O(t^2 \|d_z\|_2^2), 0 \right) \\
\leq \max \left( (1-t)g_j(z) + b_j t^2 \|d_z\|_2^2, 0 \right) \\
= b_j t^2 \|d_z\|_2^2;
$$

- $g_j(z) > 0$: then, $\max \left( g_j(z), 0 \right) = g_j(z)$ and there exist $b_j \in \mathbb{R}_{>0}$ $\forall j \in \{1, \ldots, m\}$ such that for a sufficiently small $t$:

$$
\max \left( g_j(z) + t\nabla g_j(z)^\top d_z + O(t^2 \|d_z\|_2^2), 0 \right) \\
\leq \max \left( (1-t)g_j(z) + b_j t^2 \|d_z\|_2^2, 0 \right) \\
= (1-t)g_j(z) + b_j t^2 \|d_z\|_2^2.$$

Thus, by taking out the higher order terms, by (C.16) we have:

\[
\lim_{t \to 0^+} \frac{\|g(z + td_z)^+\|_p - \|g(z)^+\|_p}{t} \leq \lim_{t \to 0^+} \frac{\left(\sum_{j | g_j(z) > 0} (1 - t)^p g_j(z)^p\right)^{1/p} - \left(\sum_{j | g_j(z) > 0} g_j(z)^p\right)^{1/p}}{t} = \left(\sum_{j | g_j(z) > 0} g_j(z)^p\right)^{1/p}.
\]

The directional derivative of the equality constraints is obtained by following the approach in [13, Section 5.6.1]. In particular, we observe that:

\[
\|h(z + td_z)\|_p - \|h(z)\|_p \leq \|h(z) + t\nabla h(z) \top d_z\|_p - \|h(z)\|_p + b_1 t^2 \|d_z\|_2^2 = -t \|h(z)\|_p + b_1 t^2 \|d_z\|_2^2,
\]

where \(b_1 \in \mathbb{R}_{>0}\) and the last equation follows from the feasibility condition \(\nabla h(z) \top d_z = -h(z)\). On the other hand, by the Taylor expansion we can also derive the following lower bound:

\[
\|h(z + td_z)\|_p - \|h(z)\|_p \geq -t \|h(z)\|_p - b_1 t^2 \|d_z\|_2^2.
\]

Therefore, by the squeeze theorem:

\[
\lim_{t \to 0^+} \frac{\|h(z + td_z)\|_p - \|h(z)\|_p}{t} = -\|h(z)\|_p.
\]

By (C.17) and (C.18), the directional derivative \(D\phi(0)\) is then:

\[
D\phi(0) = d_z \top \nabla J(z) + \rho \lim_{t \to 0^+} \frac{\|g(z + td_z)^+\|_p - \|g(z)^+\|_p}{t} + \rho \lim_{t \to 0^+} \frac{\|h(z + td_z)\|_p - \|h(z)\|_p}{t} \leq d_z \top \nabla J(z) - \rho \|g(z)^+\|_p - \rho \|h(z)\|_p.
\]
D Compressor specifications

The centrifugal compressor used for the simulation is a Continental 020, whose specifications are given in Table D.1, together with the compression system details.
## D Compressor specifications

### Table D.1: Compression system specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical Drive Power</td>
<td>15 kW</td>
</tr>
<tr>
<td>Torque Limitation</td>
<td>$[-100, 100]$ Nm</td>
</tr>
<tr>
<td>Inlet Tank Volume</td>
<td>$V_i = 1.172$ m$^3$</td>
</tr>
<tr>
<td>Outlet Tank Volume</td>
<td>$V_o = 0.595$ m$^3$</td>
</tr>
<tr>
<td>Duct Ratio</td>
<td>$\frac{A}{L} = 1.676e-4$ m</td>
</tr>
<tr>
<td>Rotational Inertia</td>
<td>$J = 2$ kg m$^2$</td>
</tr>
<tr>
<td>Recycle Valve Constant</td>
<td>$T_r = 0.5$ s</td>
</tr>
</tbody>
</table>
E Induction motor specifications

The induction motor used for the simulation is a Baldor EM3546, whose specifications are given in Table E.1.
### Table E.1: Induction motor specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drive Power</td>
<td>750 W</td>
</tr>
<tr>
<td>Rated Voltage</td>
<td>115 V</td>
</tr>
<tr>
<td>Rated Frequency</td>
<td>60 Hz</td>
</tr>
<tr>
<td>Rated Torque</td>
<td>2.07 Nm</td>
</tr>
<tr>
<td>Rotor Constant ( a )</td>
<td>5.59 s(^{-1})</td>
</tr>
<tr>
<td>Scaled Rotor Resistance ( b_1 )</td>
<td>1.12 Ω</td>
</tr>
<tr>
<td>Mutual Rotor Inductance Ratio ( b_2 )</td>
<td>0.957</td>
</tr>
<tr>
<td>Rotational Inertia</td>
<td>0.0053 kg m(^2)</td>
</tr>
<tr>
<td>Friction Coefficient</td>
<td>1.0e−2 s(^{-1})</td>
</tr>
</tbody>
</table>
Bibliography


[34] T. F. Coleman, On characterizations of superlinear convergence for constrained optimization, Computational Solution of Nonlinear Systems of Equations, 26 (1990), pp. 113–133.


[66] ——, Surge and rotating stall in axial flow compressors, part II: experimental results and comparison with theory, Transactions of the ASME, 98 (1976), pp. 199–211.


[128] G. Torrisi, S. Mariéthoz, and R. Smith, *Identification of magnetic characteristics of induction motors based on the Jiles-


Curriculum Vitae

Giampaolo Torrisi

Born Dec. 10 1988 in Rome, Italy

2013 − 2017 Doctorate at the Automatic Control Laboratory, ETH Zürich, Switzerland

2012 − 2013 Process Engineer at Eli Lilly, Florence, Italy

Spring 2012 Visiting period and Master Thesis at the Automatic Control Laboratory, ETH Zürich, Switzerland

2010 − 2012 M.Sc. in Electrical and Automation Engineering, University of Florence

2007 − 2010 B.Sc. in Mechanical Engineering, University of Florence

2002 − 2007 Secondary School “Liceo Scientifico Guido Castelnuovo”, Florence, Italy